Yao Jiang

Cryptographic Tools for Cloud Security

Thesis for the degree of Philosophiae Doctor

Trondheim, April 2021

Norwegian University of Science and Technology
Faculty of Information Technology
and Electrical Engineering
Department of Mathematical Sciences

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Information Technology
and Electrical Engineering
Department of Mathematical Sciences

ISBN 978-82-326-6671-3 (printed ver.)
ISBN 978-82-326-6597-6 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (electronic ver.)

Doctoral theses at NTNU, 2021:126

PRINTED IN
NORWAY
NO - 1598

NORDIC SWAN ECOLABEL
Printed matter
2041 0731

Printed by Skipnes Kommunikasjon AS

Acknowledgments

Throughout my PhD study, I have received a great deal of support and assistance. I would like to express my deepest gratitude to everyone that has helped me along the way.

Firstly, I would like to thank my supervisor Kristian Gjøsteen for continuous support of my PhD study and research, encouragement and patience. He guide me to choose the right direction and become a mature PhD, without him I would not have been able to complete this research.

I would like to thank my co-authors: Colin, Gareth, Herman and Kristian for writing papers, discussing creative ideas and working with me. I have enjoyed and benefited from this collaborative work, the discussions improved our work. Many thanks to my co-supervisor Colin, your insightful feedback pushed me to sharpen my understanding and elevated my knowledge of cryptography. I would also like to give an extra thanks to the post-doctor, Gareth, for showing me good research habits and patiently answering all my questions.

Furthermore, many thanks to all members in the NaCl group for good feedback and interesting discussions.

Finally, I would particularly like to express my greatest appreciation to my husband and research partner Herman for helping my work and life, discussing and sharing cool ideas, and taking care of our babies while I need to focus on my research.

Yao Jiang
Trondheim, December 2020

Introduction

A cloud storage provider, such as Amazon, Facebook, Dropbox, Microsoft and Google, allows clients to store and maintain a large volume of data online. It also ensures that clients can easily access the data anywhere and at any time, and clients can conveniently share documents to those who have been granted access. These attractive features makes cloud technology an important part of our daily lives.

However, with the increase of popularity in using cloud services, there is an increasing probability of losing sensitive data and personal information, such as medical records and financial records. The storage provider might be attacked by hackers or malware infections. In addition, employees that maintain the cloud service might manipulate or leak a client's data. All of these motivate the need to securely store data in the cloud without relying on trusting the cloud provider. Cloud cryptography uses encryption techniques to protect private data stored in the cloud and the expected fundamental security goals are confidentiality and integrity, where *Confidentiality* means that data is protected from disclosure to any adversary and *Integrity* means that data is protected from modification by any adversary.

On the other hand, *efficiency* is another important element to consider for cloud cryptography, as each cloud user might store large files like videos and images, and the cloud server maintains data for millions of users. It is crucial that the encryption scheme used by a cloud service provider is efficient in computation and storage.

The data stored in the cloud might stay there for many years, and the data is growing rapidly due to millions of clients continuously storing more and more data. Switching encryption schemes in this setting is expensive, the amount of work needed to download, re-encypt, and re-upload all the data stored in the cloud would be immense. Using classical schemes (Diffie-Hellman key exchange, RSA, ElGamal, etc.) would suffice for now, however, an attacker might steal and store a client's encrypted data and wait for quantum computers to be available. It is well-known that a sufficiently powerful quantum computer running Shor's algorithm could solve the integer factorization problem or the discrete logarithm problem, which are problems that the current popular encryption algorithms relies on. It is therefore better to use post-quantum secure schemes now.

This thesis discusses two practical problems for using the cloud: sharing and updating keys. We will analyze the security requirements and develop new cryptographic tools to solve these two problems. We will provide constructions aiming to maximize the security and efficiency. There are two main research topics in this thesis.

- Offline assisted group key exchange with forward secrecy. We study how to securely share data with some desired cloud security properties.
- Updatable encryption. We study how to rotate keys and ciphertexts. In particular, we will discuss the security expectations for updatable encryption schemes and, from our findings, construct UE schemes with these expected properties.

Additionally, we solve the problem of constructing post-quantum secure protocols/schemes. We construct protocols and schemes that are post-quantum secure and fit with the desired sharing or updating properties described above.

Organization. In this introduction we first provide general knowledge of homomorphic encryption, then we discuss offline assisted group key exchange and updatable encryption

In the sections of the latter two topics, we will also analyze how to create post-quantum secure constructions that solves the corresponding sharing or updating problem. In the end of the introduction, we combine the techniques discussed in this thesis to show how a client could both share and update data.

Homomorphic Encryption

Homomorphic encryption allows anyone to evaluate functions on ciphertexts, where the resulting output is a valid ciphertext. This ciphertext can be decrypted and its underlying plaintext matches the output of the same function applied to the underlying plaintexts of the input ciphertexts. That is,

$$\mathsf{Dec}(\mathsf{c}_1 \oplus_{\mathcal{C}} \mathsf{c}_2) = \mathsf{Dec}(\mathsf{c}_1) \oplus_{\mathcal{M}} \mathsf{Dec}(\mathsf{c}_2),$$

where $\oplus_{\mathcal{C}}$ and $\oplus_{\mathcal{M}}$ are operations on the ciphertext space and plaintext space, respectively.

The offline group key exchange protocols and the updatable encryption schemes we construct in this thesis all have some homomorphic property. We will show how to build these desired constructions using schemes with homomorphic properties.

Well-known lattice-based constructions, such as NewHope [1] and NTRU [5], have homomorphic properties. Lattice-based constructions are also important candidates for providing post-quantum secure schemes and protocols. We will show how to use lattice-based schemes to construct post-quantum secure cloud schemes.

Offline Assisted Group Key Exchange

A cloud user Alice would like to share files with her collaborators, Bob and Carol, by using a cloud server. They don't want any attacker or the cloud server to see the content of the shared files. Hence, the shared files will be encrypted under a file encryption key before they are uploaded to the cloud. In other words, this file-sharing problem results in a key-sharing problem.

Alice wishes to establish a file encryption key and securely share this key with Bob and Carol. *Forward secrecy* is desired in this sharing scenario. That is, if a user loses her long-term key then any previously completed session keys will not be compromised. In practice, users are not online all the time, hence, it is desirable to establish this file encryption key without any interaction among the sharing parties. The main benefit of having this *non-interactive* property is that Alice could share files whenever she wishes to and her collaborators could immediately get the shared key and the shared data when they come online. We will discuss how to construct a group key exchange protocol that has forward secrecy and is non-interactive.

There are many possible solutions for sharing keying materials with partners. The simplest way is to use a public key encryption (PKE) scheme: Alice encrypts the shared key material under Bob or Carol's public key and sends the encrypted key to Bob and Carol, respectively. When Bob or Carol comes online, they can decrypt the received ciphertext to get the file encryption key and use this key to open the files Alice has shared. However, this solution does not provide forward secrecy. Whenever a sharing partner loses their long-term secret key the file encryption key will be revealed as well.

Another way is to use a group key exchange (GKE) protocol, which can provide forward secrecy. However, GKE requires all sharing partners to be online when they establish the agreed key. This is not practical in the real world, as no client will be online all the time and it is not reasonable to ask all partners waiting online until the last partner arrives to establish a key (the waiting time is non-trivial as well). As a consequence, no sharing member can look at the shared files before the key has been agreed.

In the first paper of this thesis, we provide a new protocol, offline assisted group key exchange (OAGKE) protocol, that allows any user to use a modified key encapsulation primitive, a *blinded key encapsulation mechanism* (BKEM), to transport a file encryption key to potentially many sharing partners via the (untrusted) cloud server. A conceptual overview of our OAGKE construction is given in Fig. 1. This protocol achieves the two desired properties: forward secrecy and non-interactivity.

The benefit of this new idealized primitive (BKEM) is that it can be used to hide file encryption keys by using a blinding algorithm Blind. Sharing parties can safely obtain the file encryption key by creating a blinded encapsulation and asking the cloud server to help decapsulating this blinded encapsulation. The first paper also presents two instantiations of BKEMs based on well-known hardness assumptions, namely the DDH and the RSA problems.

In this OAGKE protocol, a session is when a user shares a key, and the shared file encryption key is the session key. Initially, Alice runs half of the group key exchange protocol by interacting with the cloud server: she generates a file encryption key and some "key information", which is an encapsulation of the file encryption key under the server's ephemeral encapsulation key. When her partner Bob comes online, Bob runs his half part of the group key exchange protocol by blinding the value of the received encapsulation and sending the blinded encapsulation to the cloud server. The cloud server decapsulates the blinded encapsulation and returns the output (the blinded key) to Bob. Using this blinded key Bob can recover the file encryption key.

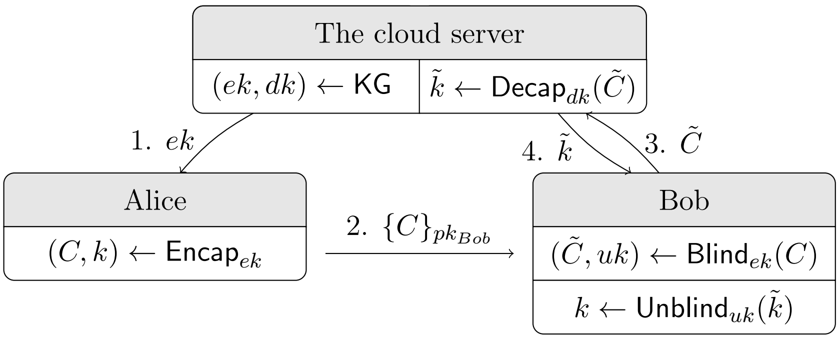
The cloud server
$(ek, dk) \leftarrow \mathsf{KG}$
$\tilde{k} \leftarrow \mathsf{Decap}_{dk}(\tilde{C})$
1. ek
4. $\tilde{k}$
3. $\tilde{C}$
Alice
$(C, k) \leftarrow \mathsf{Encap}_{ek}$
2. $\{C\}_{pk_{Bob}}$
Bob
$(\tilde{C}, uk) \leftarrow \mathsf{Blind}_{ek}(C)$
$k \leftarrow \mathsf{Unblind}_{uk}(\tilde{k})$

Figure 1: Diagram describing how the OAGKE protocol uses the BKEM to do key exchange between Alice, the cloud server and a sharing partner Bob. The file encryption key k is used by Alice to encrypt one or more files. C is an encapsulation of the file encryption key, $\tilde{C}$ is a blinded encapsulation of the encapsulation C, uk is the unblind key which can be used to recover the blinded key $\tilde{k}$ of the file encryption key k. The numbered arrows indicate the order in which operations occur.

Notice that, in Figure 1, the encapsulation is encrypted under Bob's public key before sending out, so the server cannot obtain the file encryption key through the encrypted encapsulation. On the other hand, the server cannot get any information of the file encryption key from the blinded encapsulation if the BKEM scheme is properly constructed. Therefore, the file encryption key is unknown to the cloud server. Furthermore, if the server honestly deletes all ephemeral data (ephemeral encapsulation and decapsulation keys),[1] then even if an adversary later corrupts Bob's long-term secret key she is not able to decapsulate the encapsulation to gain the file encryption key. Hence, our protocol has forward secrecy.[2]

Paper I *Offline Assisted Group Key Exchange* The paper designs a non-interactive group key exchange protocol with forward secrecy, which uses a new primitve called BKEM for key exchange. The paper also provides a security analysis of this group key exchange protocol.

It is natural to employ homomorphic encryption schemes to construct BKEMs. When the sharing partner Bob blinds an encapsulated file encryption key, he is applying a function on, what can be seen as, ciphertexts. When using a homomorphic encryption scheme, the encapsulation algorithm of BKEM is realized by the encryption algorithm of the homomorphic scheme, which is used to generate a ciphertext $C_1 = \mathsf{Enc}_{ek}(k_1)$ as the encapsulation and the underlying plaintext k_1 as the file encryption key. To blind the encapsulation, each sharing partner use the scheme's homomorphic property to add a random value k_2 to the encapsulated file encryption key. This random value is encrypted to create the ciphertext $C_2 = \mathsf{Enc}_{ek}(k_2)$, and the operation $\oplus_{\mathcal{C}}$ is applied to C_1 and C_2 to blind the encapsulation, that is $\tilde{C} = C_1 \oplus_{\mathcal{C}} C_2$. The decapsulation algorithm is realized by the homomorphic encryption scheme's decryption algorithm, which can decrypt the blinded encapsulation since it is a valid ciphertext. Hence, the blinded key is the sum of the file encryption key and the random value, that is

$$\tilde{k} = \mathsf{Dec}_{dk}(C_1 \oplus_{\mathcal{C}} C_2) = \mathsf{Dec}_{dk}(C_1) \oplus_{\mathcal{M}} \mathsf{Dec}_{dk}(C_2) = k_1 \oplus_{\mathcal{M}} k_2,$$

which is known to the corresponding sharing partner. Therefore, after receiving the decrypted key (the blinded key) from the server, the

[1]This assumption allows us to have the best possible level of forward secrecy in our collaboration scenario.

[2]The OAGKE protocol does not provide forward secrecy if the cloud server is corrupted before all ephemeral data are deleted.

sharing partner can recover the file encryption key by adding the inverse of the previously generated random value to the blinded key, that is $k_1 = \tilde{k} \oplus_{\mathcal{M}} (\oplus_{\mathcal{M}} k_2^{-1})$. Since we only need one operation in the blinding algorithm: a somewhat group homomorphic encryption scheme is sufficient.

The second paper of this thesis furthers the understanding of BKEMs and generically builds BKEMs from homomorphic encryption schemes. The second paper also provides two instantiations of BKEM built from primitives with post-quantum security.

Paper II *Cloud-assisted Asynchronous Key Transport with Post-Quantum Security* The paper provides a generic homomorphic-based BKEM construction. Furthermore, the paper constructs the first post-quantum secure BKEMs with a security proof.

Updatable Encryption

Again, to prevent any attacker or the cloud server seeing the content of the data, a cloud user encrypts documents locally before sending them to the cloud. Hence, anyone with this encryption key has access to the encrypted cloud data.

What constitutes a cloud user varies. A cloud user can be a user who may want to change her encryption key to a fresh key in the event she loses a device that has her file encryption keys stored. A cloud user can be a company who may want to rotate the data access key to a fresh key in some cases where an employee leaves the company. A cloud user can be a media-services provider who may want to alter the data access key to a fresh key in a situation where a customer's membership period expires. After an encryption key is updated to a new one, it is reasonable to expect all old ciphertexts stored in the cloud will be updated to be ciphertexts encrypted under the new key. The old key will no longer valid and only the key holder of the new key can retrieve the underlying information of all updated ciphertexts. During this ciphertext updating process, it is also reasonable to expect that no information of the outsourced data is leaked.

Key rotation is the process of generating a new key and altering ciphertexts from the old key to the new key without changing the underlying message. The main benefit of key rotation is that it can be used to protect the data and reduce the risk of key compromise over time.

Cloud clients can do key rotation by downloading, decrypting, re-encrypting and re-uploading ciphertexts. This, however, is a very expensive approach. *Updatable encryption* (UE) provides a solution such that the cloud assists in updating ciphertexts from an old key to a new key with the help of a client-provided *update token*. Furthermore, *ciphertext-independent* updatable encryption schemes make it possible for the cloud to update all ciphertexts a client owns using only a single token, where ciphertext-independent means the token generation is independent of the old ciphertext, which enables the client to generate an update token without downloading any ciphertexts from the cloud. Hence, the major advantage of ciphertext-independent updatable encryption schemes is its efficiency in terms of bandwidth. We stress that in this thesis we focus on ciphertext-independent updatable encryption schemes.

The use cases of updatable encryption appear promising, and for these use cases we also want to provide a secure UE scheme. The first thing to consider is what an attacker can possibly do. Secondly, what kind of security we wish to achieve.

Users might lose their keys from time to time, and tokens are very likely to become lost during transmission. It is reasonable to consider that the adversary has the ability to adaptively corrupt keys and tokens. Lehmann and Tackmann [4] introduced this *corruption model* for UE schemes.

The first security property one would expect from updatable encryption is *confidentiality*. Consider a motivating example, a journalist who stores a contact list with a cloud storage provider, where each entry in the list is a ciphertext. At some point, the storage is compromised and an adversary recovers the ciphertexts. Apart from the importance of keeping the underlying plaintexts unreadable, it may also be important that the cryptography does not reveal which of the contacts are recent, and which are old. That is, it must be hard to decide if some ciphertext was recently created, or if it has been updated from a ciphertext stored in an earlier epoch.

Lehmann and Tackmann [4] have studied confidentiality notions to achieve the following properties. The first property is the adversary should not be able to determine anything about the underlying plaintext of a given ciphertext. The second property is, given a ciphertext in the current epoch, the adversary should not be able to tell which ciphertext (that existed in the previous epoch) the current given ciphertext was updated from. However, none of the security properties studied before could capture our motivating example. The third paper of this thesis introduces a new security notion that guarantees that the adversary is unable to distinguish between a fresh encrypted ciphertext and an updated ciphertext. Additionally, this new notion implies earlier notions [4].

Another security property one would expect from updatable encryption is *integrity*. In the cloud scenario, ciphertexts might become altered when communicated over an unsecure channel or sent to an untrusted cloud server. For example, the adversary might be able to modify the ciphertext to some other valid ciphertext such that the (shared) data owners will see an invalid value after retrieving the altered data. Integrity can ensure that the data has not been changed by any unauthorized parties. Klooß et al. [3] formally defined integrity for updatable encryption schemes, however, a composition result of the style given by Bellare and Namprempre for symmetric encryption [2] – the combination of CPA security and integrity of ciphertexts (CTXT) gives CCA security – has been missing. The third paper provides this composition result for updatable encryption.

Note that corrupted tokens could help the adversary learn more information such as keys and ciphertexts of different epochs. The adversary can use the inferred information to decrypt ciphertexts to get the underlying plaintexts. These corruption powers allow the adversary to trivially win a security game, and we should exclude these trivial win conditions in the security analysis. A detailed analysis of trivial win conditions for updatable encryption schemes is provided in the third paper.

Except for security, we also want *efficient* UE schemes, especially for the encryption and update algorithms. A modern database may contain large numbers of files, hence, efficiency is critical both for users, who will have to encrypt plaintexts initially, and for servers, who will have to update ciphertexts for all of their users. The third paper creates a highly efficiently UE scheme, SHINE, which also achieves CPA, CTXT and CCA security.

Paper III *Fast and Secure Updatable Encryption* The paper presents a new confidentiality notion for updatable encryption schemes that implies prior notions. The paper also constructs a secure and highly efficient updatable encryption scheme called SHINE.

Recall that revealed tokens can help the adversary gain more information about ciphertexts and keys. The basic requirement of UE is that an update token can move ciphertexts from the old key to the new key. If the update token can only upgrade ciphertexts, it is *uni-directional.* If the update token can both upgrade and downgrade ciphertexts, it is *bi-directional.* On the other hand, the update token can potentially be used to derive keys from other keys. Similarly, in the *uni-directional key update setting*, an update token can only derive the new key from the old key. In the *bi-directional key update setting*, an update token can additionally derive the old key from the new key.

Intuitively, UE schemes with uni-directional updates are desirable, such schemes leak less ciphertext/key information to an adversary compared to schemes with bi-directional updates. In the fourth paper, we define security notions in terms of different update setting and analyze the relationship between security notions with uni- and bi-directional updates. We show that the (confidentiality and integrity) security of UE schemes are not influenced by uni- or bi-directional updates. This is a surprising result.

On the other hand, we wish to have a UE scheme that is post-quantum secure, where we use an LWE-based homomorphic encryption scheme to construct an LWE-based UE scheme To make sure updated ciphertexts look random, our LWE-based UE scheme uses the re-randomization idea from the RISE scheme in the work by Lehmann and Tackmann [4]. The abstract approach of using the update token to re-randomizing the ciphertext is as follows, the update algorithm

(1) uses the update token to move a ciphertext, $C_{old} = \mathsf{Enc}_{pk_{old}}(m)$, from the old key to the new key, $C_{mid} = \mathsf{Enc}_{pk_{new}}(m)$;

(2) generates a random ciphertext $C_{rand} = \mathsf{Enc}_{pk_{new}}(u)$ using the new public key and the plaintext $u = 0$ if $\oplus_{\mathcal{M}}$ is addition and $u = 1$ if $\oplus_{\mathcal{M}}$ is multiplication;

(3) outputs the sum of the above two ciphertexts $C_{new} = C_{mid} \oplus_{\mathcal{C}} C_{rand}$ as the updated ciphertext.

The first step might be enough for constructing a secure UE scheme like SHINE. However, for schemes like RISE and our LWE-based UE scheme, only using the first step is not enough to construct a secure UE scheme. Such constructed schemes cannot even achieve IND-UPD, because one entry of the updated ciphertext remains the same as the old ciphertext. That is why we need the re-randomization procedure (Step (2) and (3)).

Using the homomorphic property, the output of the update algorithm is a random ciphertext with the same plaintext of the old ciphertext under the new public key. Using this method, we can possibly construct a secure updatable encryption scheme. In particular, the security of these UE schemes follows from the security of the underlying homomorphic encryption schemes. There are fruitful homomorphic encryption constructions, specifically, we use a post-quantum secure lattice-based homomorphic encryption scheme to construct a UE scheme in the fourth paper, which ensures our UE scheme achieves post-quantum security.

Note that when the update algorithm performs one re-randomization, the error terms grows. In the updatable encryption setting, the total number of epochs will be a comparatively small integer in practice. We stress that the updatable encryption has an upper bound of how many times a ciphertext can be updated. By some parameter setting, the error terms will not grow too big and the updatable encryption scheme is correct with overwhelming probability.

Paper IV *The Direction of Updatable Encryption does not Matter Much* This paper compares security notions based on the uni- and bi-directional updates, and constructs a UE scheme with post-quantum security.

Key Management Application

To conclude this introduction, we show how we can use the above discussed sharing and updating techniques to solve a problem: A user wishes to share files securely with her collaborators, these collaborators can be her own devices. Additionally, she wants to rotate keys and ciphertexts periodically to reduce the risk of key compromise. In the meanwhile, she might remove some collaborators to stop them from accessing the shared documents.

We sketch a method combining OAGKE and UE to solve this problem. A user Alice uses a UE scheme to generate a shared key and then applies the OAGKE protocol to share this key with her partners, Bob and Carol. When Alice wishes to move to the next period, she will use the UE scheme to generate a new key and an update token. The update token will be sent to the cloud server and the collaborators she wishes to keep, e.g. Bob, by running the OAGKE protocol. Note that now the server is also a collaborator of Alice while she runs the OAGKE protocol and the update token is the shared value in the OAGKE protocol. The cloud server will perform an update to move (encrypted) shared files from the old key to the new key using the update token provided by Alice, and Bob will update his old shared key to the new key using the update token provided by Alice. Bob will now use the new key to access the shared data. Note that the UE scheme we use here should have at least uni-directional key updates, since an update token can upgrade keys. Now only Bob has access to the files shared by Alice. In Fig 2, we show how Alice shares and updates files to her collaborators by running the OAGKE protocol and using a UE scheme.

In our key management approach, each session secret of the OAGKE protocol is either a key or a token. The advantage of using OAGKE protocol is that we can protect the shared data while it is in transit, which is either the initial shared key or an update token. The OAGKE protocol helps UE to reduce the adversary power of corrupting keys and tokens. Which means the adversary has to corrupt a user (or a device), to get some key information.

However, if we only use OAGKE, when the adversary corrupts a user it can see all keys of that user. As a result, everything in the

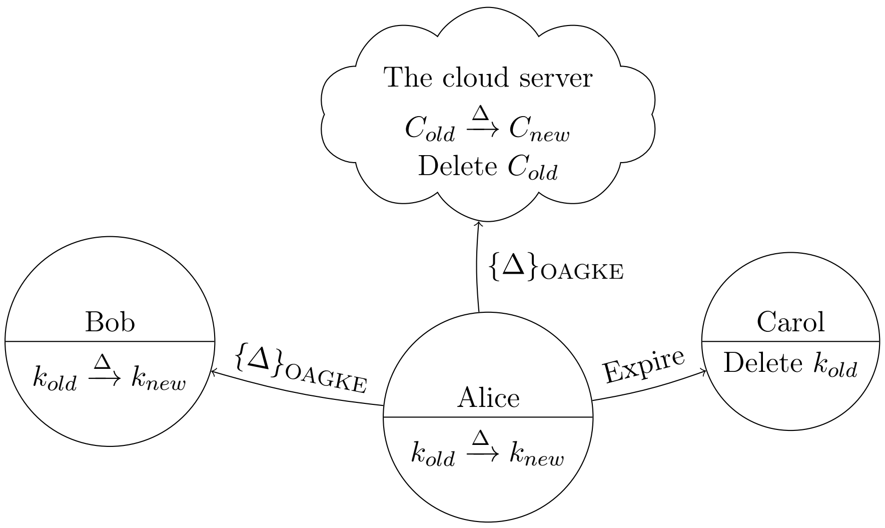
The cloud server
$C_{old} \xrightarrow{\Delta} C_{new}$
Delete C_{old}
$\{\Delta\}_{\text{OAGKE}}$
Bob
$k_{old} \xrightarrow{\Delta} k_{new}$
$\{\Delta\}_{\text{OAGKE}}$
Alice
$k_{old} \xrightarrow{\Delta} k_{new}$
Expire
Carol
Delete k_{old}

Figure 2: Initially, Alice runs the OAGKE protocol to share a key k_{old} to Bob and Carol. C_{old} are files encrypted under key k_{old} stored in the cloud. At some time, Alice wants to update the shared key to stop Carol having access to the shared files. She shares the update token Δ by running the OAGKE protocol to the cloud server and Bob to help them update ciphertexts and key, respectively. Hence, Bob still can see the content of the shared documents. In the meanwhile, she sends “Expire” to Carol, then Carol will delete the shared key and ephemeral values.

cloud encrypted by these shared session keys will lose. UE ensures that an adversary corrupting a user will not know the previous and future keys. By the help of UE in this key management scenario, the user can rotate keys when her partner is corrupted. As long as the user notices this corruption in time and everything is updated before the adversary tries to retrieve information from the cloud, nothing will be lost.

References

[1] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - a new hope. Cryptol-

ogy ePrint Archive, Report 2015/1092, 2015. `https://eprint.iacr.org/2015/1092`.

[2] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *J. Cryptol.*, 21(4):469–491, 2008.

[3] Michael Klooß, Anja Lehmann, and Andy Rupp. (R)CCA secure updatable encryption with integrity protection. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 68–99. Springer, 2019.

[4] Anja Lehmann and Björn Tackmann. Updatable Encryption with Post-Compromise Security. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 685–716. Springer, 2018.

[5] Damien Stehlé and Ron Steinfeld. Making NTRU As Secure As Worst-case Problems over Ideal Lattices. In Kenneth G. Paterson, editor, *EUROCRYPT*, Lecture Notes In Computer Science, pages 27–47. Springer-Verlag, 2011.

Paper I

Offline Assisted Group Key Exchange

Colin Boyd, Gareth T. Davies, Kristian Gjøsteen and Yao Jiang

Published in the 21st Information Security Conference, ISC2018.

Offline Assisted Group Key Exchange

Colin Boyd, Gareth T. Davies, Kristian Gjøsteen, and Yao Jiang

NTNU – Norwegian University of Science and Technology
{colin.boyd,gareth.davies,kristian.gjosteen,
yao.jiang}@ntnu.no

Abstract

We design a group key exchange protocol with forward secrecy where most of the participants remain offline until they wish to compute the key. This is well suited to a cloud storage environment where users are often offline, but have online access to the server which can assist in key exchange. We define and instantiate a new primitive, a blinded KEM, which we show can be used in a natural way as part of our generic protocol construction. Our new protocol has a security proof based on a well-known model for group key exchange. Our protocol is efficient, requiring Diffie–Hellman with a handful of standard public key operations per user in our concrete instantiation.

Keywords: Authenticated Key Exchange, Group Key Exchange, Forward Secrecy, Cloud Storage, Blinded Key Encapsulation

1 Introduction

We consider the following collaboration scenario. Isabel would like to use a cloud storage provider to share some files with her collaborators Robin and Rolf. While Isabel and her collaborators have some level of trust in the cloud storage provider, they do not want the provider to be able to see the contents of their files. In other words, Isabel needs to share some secret key material with Robin and Rolf. This paper addresses the problem of sharing this secret key material.

There are a number of possible solutions. The simplest is for Isabel to encrypt the key material using public key encryption and send the ciphertexts to Robin and Rolf, who can then decrypt. However, this solution does not provide *forward secrecy.* If either Robin or Rolf's decryption keys are compromised at any point in the future, the confidentiality of the key material is also compromised.

Group key exchange (GKE) can give us forward secrecy. However, Isabel and her collaborators will not be online all the time, and the time spent offline is non-trivial. If Isabel and her collaborators want to use a traditional GKE, then Isabel cannot share her files until every collaborator has been online. Likewise, the individual collaborators cannot look at the shared files until every other collaborator has been online. This is impractical, and no system that has interactions between the initiator and the responders can be practical in this setting.

In this paper, we propose a GKE protocol that provides forward secrecy and is non-interactive with respect to the sharing parties, hence suitable for our collaboration scenario: Isabel comes online, runs her part of the GKE protocol, receives the key material and shares the files. As the individual collaborators come online, they run their part of the GKE protocol, receive the key material and get access to the shared files.

1.1 Secure Sharing and Forward Secrecy

The users in our collaboration scenario will be content to trust their cloud storage provider (CSP) to make their data available. Some users will be content to trust their CSP to use simple access control to prevent unauthorized access or modification. However, for many users such a convenient trust assumption regarding confidentiality or integrity is either unreasonable, legally impossible or otherwise undesirable. For this reason, many CSPs support (in addition to access control) the obvious solution of user-side encryption of data, where the CSP does not know the key material used for encryption and decryption[1].

The use of encryption means that groups of users must establish shared key material in order to share data. This suggests group key exchange. However, group key exchange protocols are usually interactive, while in our collaboration scenario, Isabel's collaborators may not all be online at the same time, so completing the group key exchange would take too long, and until the key material was agreed upon, no work could be done.

We therefore desire *non-interactive* solutions that allow the initiator to complete their actions before any recipients come online, and do not require any interaction between the recipients. This rules out traditional group key exchange protocols [5, 2, 18, 3, 13].

The natural non-interactive solution is to use public key encryption (or perhaps other similar primitives, such as broadcast encryption). However, in the outsourced storage scenario, *forward secrecy* – compromise of long-term keys does not compromise previously completed sessions – is important. Forward secrecy is typically achieved through the use of interaction with Diffie–Hellman or other ephemeral keys. Using ephemeral keys for confidentiality and long-term keys only for authentication ensures that later release of long-term secrets does not reveal the session key.

Forward secrecy presents an inherent conflict with our requirement to have a non-interactive solution. Indeed, a simple generic argument implies that forward secrecy without interaction is impossible: without interaction the recipient cannot provide an ephemeral input and therefore the recipient's long-term key alone must be sufficient to recover the session key. Recent proposals have attempted to work around this argument in different ways. The first line of work, including the X3DH [23] and ART [8] protocols, insists that recipients upload some pre-keys to the CSP at some point before the initiator begins their activity. These pre-keys are then used as if they were ephemeral, however if one recipient never comes online then they could sit on the server indefinitely: this is a re-definition of ephemeral and long-term keys, as used by standard key exchange security models. Another approach, taken by Green and Miers [14] and further

[1]This practice is confusingly often called *zero knowledge* in commercial circles.

developed by Günther et al. [16] and Derler et al. [12], concerns so-called zero-round-trip-time (0RTT) key exchange. In this model, the long-term decryption key is updated (punctured) once the recipient comes online, in such a way that the crucial ciphertexts can no longer be decrypted by that (long-term) key. Thus the long-term key is no longer static but evolves over time. Forward secrecy with puncturable encryption relies crucially on the assumption that the protocol (single) message arrives at the receiver. Until that happens the receiver private key is not updated and so the encrypted data is vulnerable to receiver compromise. In addition we note that these works rely on less efficient cryptographic primitives and require increased storage and secure deletion properties at the receiver. Fig. 1 summarizes selected existing literature on file-sharing protocols.

Protocol	Forward Secrecy	Non-Interactive	Security Proof	Efficient	Parties
X3DH [23]	✓[a]	✗[d]	✗	✓	2
ART [8]	✓[a]	✗[d]	✓	✓	N
0RTT KE [14, 16, 12]	✓[b]	✓	✓	✗	2
GKE [5, 2, 18, 3, 13]	✓	✗	✓	✓	N
Mona, Tresorit [22] [20, 21]	✗	✓	✓	✓	N
Chu et al. [7]	✗	✓	✓	✓	N
This work	✓[c]	✓	✓	✓	N

Figure 1: Comparison of secure sharing protocols. [a]Re-defined 'ephemeral keys'; [b]Re-defined 'long-term keys'; [c]If the server honestly deletes all ephemeral data; [d]Users must upload pre-keys.

1.2 Contributions

In this paper, we run into two major obstacles. We need a group key exchange protocol that is non-interactive with respect to the initiator and the responders, and that at the same time provides forward secrecy.

We overcome these obstacles by noting that the cloud server is online at all times, and use ephemeral values provided by the cloud server to give us forward secrecy. This allows us to achieve the best possible level of forward secrecy in our collaboration scenario, without trusting the cloud server. Our protocol is simple and relies only on standard assumptions.

We regard the following as the main contributions of this paper.

- We propose a novel practical group key exchange protocol suitable for use in cloud storage. Our protocol is described in Section 5.
- We include a formal security analysis of our protocol in a strong security model with trust assumptions suited to the cloud scenario. The proof is in a security model which is detailed in Section 3.
- We introduce definitions and constructions for a new cryptographic primitive, blinded KEMs, which may find other applications. We describe this primitive and provide two secure constructions in Section 4.

2 Preliminaries

For a set S, denote $xgetsrS$ to mean choosing x uniformly at random from S. We write **return** $b' \overset{?}{=} b$ as shorthand for **if** $b' = b$ **then return** 1; **else return** 0, with an output of 1 indicating successful adversarial behavior.

2.1 Public-key encryption

A public-key encryption scheme $\mathsf{PKE} = (\mathsf{KG}_{\mathsf{pke}}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M}$ is defined as follows. $\mathsf{KG}_{\mathsf{pke}}$ takes as input some security parameter(s), if any, and outputs a public encryption key pk and a secret decryption key sk. Enc takes a message m and produces a ciphertext c using pk: $c \leftarrow \mathsf{Enc}_{pk}(m)$. Dec decrypts a ciphertext c using

sk to recover m or in the case of failure a symbol $\perp$: $m/\perp \leftarrow \mathsf{Dec}_{sk}(c)$. Correctness requires that $m \leftarrow \mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m))$ for all $m \in \mathcal{M}$.

We denote the usual advantage of an adaptive chosen ciphertext adversary $\mathcal{A}$ against real-or-random security for the public-key encryption scheme by $\mathbf{Adv}^{\mathsf{ror\text{-}cca2}}_{\mathsf{PKE}}(\mathcal{A})$. In our protocol's security proof, it is actually convenient to use a generalization of this notion, which we discuss in Appendix A.

2.2 Digital signatures

A signature scheme $\mathsf{DS} = (\mathsf{KG}_{\mathsf{sig}}, \mathsf{Sign}, \mathsf{Verify})$ with message space $\mathcal{M}$ is defined as follows. $\mathsf{KG}_{\mathsf{sig}}$ takes as input some security parameter(s), if any, and outputs a signing key sk and a public verification key vk. Sign creates a signature σ on a message m: $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$. Verify verifies that the signature on the message is in fact valid: $0/1 \leftarrow \mathsf{Verify}_{vk}(m, \sigma)$, with 1 indicating successful verification. Correctness requires that $\mathsf{Verify}_{vk}(m, \mathsf{Sign}_{sk}(m)) = 1$ for all $m \in \mathcal{M}$.

Definition 1. Let $\mathsf{DS} = (\mathsf{KG}_{\mathsf{sig}}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme. Then the $\mathsf{suf\text{-}cma}$ advantage of an adversary $\mathcal{A}$ against DS is defined as

$$\mathbf{Adv}^{\mathsf{suf\text{-}cma}}_{\mathsf{DS}}(\mathcal{A}) = \mathbf{Pr}[\mathbf{Exp}^{\mathsf{suf\text{-}cma}}_{\mathsf{DS}}(\mathcal{A}) = 1].$$

where the experiment $\mathbf{Exp}^{\mathsf{suf\text{-}cma}}_{\mathsf{DS}}(\mathcal{A})$ is given in Fig. 2.

```
$\underline{\mathbf{Exp}^{\mathsf{suf\text{-}cma}}_{\mathsf{DS}}(\mathcal{A}):}$
  $\mathsf{S_{LIST}} \leftarrow \emptyset$
  $sk, vk \leftarrow \mathsf{KG}_{\mathsf{sig}}$
  $(m, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Sign}}(vk)$
  if $\mathsf{Verify}_{vk}(m, \sigma)$ and $(m, \sigma) \notin \mathsf{S_{LIST}}$
    return 1
  else
    return 0

$\mathcal{O}.\mathsf{Sign}(m)$ :
  if $m \notin \mathcal{M}$ then
    return $\perp$
  $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$
  $\mathsf{S_{LIST}} \leftarrow \mathsf{S_{LIST}} \cup (m, \sigma)$
  return $\sigma$
```

Figure 2: The experiment defining $\mathsf{suf\text{-}cma}$ security for signature schemes.

```
$\underline{\mathbf{Exp}^{\mathsf{DDH}}_{\mathbb{G}}(\mathcal{A}):}$
    $b \xleftarrow{\$} \{0,1\}$
    $x, y, z \xleftarrow{\$} \mathbb{Z}_q$
    **if** $b = 1$
        $c \leftarrow g^{xy}$
    **else**
        $c \leftarrow g^{z}$
    $b' \leftarrow \mathcal{A}(g^x, g^y, c)$
    **return** $b' \overset{?}{=} b$
```

Figure 3: DDH experiment.

```
$\underline{\mathbf{Exp}^{\mathsf{CR}}_{\mathcal{F}}(\mathcal{A}):}$
    $\mathsf{f} \xleftarrow{\$} \mathcal{F}$
    $x, y \leftarrow \mathcal{A}(f)$
    **if** $x \neq y \wedge \mathsf{f}(x) = \mathsf{f}(y)$ **then**
        **return** 1
    **else**
        **return** 0
```

Figure 4: Collision resistance experiment.

Note that in the existential unforgeability under chosen message attack ($\mathsf{euf\text{-}cma}$) game the list $\mathsf{S}_{\mathsf{LIST}}$ only keeps track of the messages queried by the adversary during the Sign queries phase, so $\mathcal{A}$ is not allowed to output (m, σ_2) if she sent m to $\mathcal{O}.\mathsf{Sign}$ and received σ_1.

2.3 Hardness assumptions

Definition 2. Fix a cyclic group $\mathbb{G}$ of prime order q with generator g. The advantage of an algorithm $\mathcal{A}$ solving the *Decision Diffie-Hellman (DDH)* problem for $\mathbb{G}$ and g is

$$\mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}}(\mathcal{A}) = 2\left|\Pr[\mathbf{Exp}^{\mathsf{DDH}}_{\mathbb{G}}(\mathcal{A}) = 1] - \frac{1}{2}\right|$$

where the experiment $\mathbf{Exp}^{\mathsf{DDH}}_{\mathbb{G}}(\mathcal{A})$ is given in Fig. 3.

Definition 3. Let $\mathcal{F}$ be a family of functions. The *collision resistance* advantage of an adversary $\mathcal{A}$ running in time t is

$$\mathbf{Adv}^{\mathsf{CR}}_{\mathcal{F}}(\mathcal{A}) = \left|\Pr[\mathbf{Exp}^{\mathsf{CR}}_{\mathcal{F}}(\mathcal{A}) = 1]\right|$$

where the experiment $\mathbf{Exp}^{\mathsf{CR}}_{\mathcal{F}}(\mathcal{A})$ is given in Fig. 4.

Note that in an abuse of notation, we sometimes write $\mathbf{Adv}^{\mathsf{CR}}_{\mathsf{f}}(\mathcal{A})$, with the understanding that the function family $\mathcal{F}$ exists and that the choice of a function f is done at some point.

3 GKE protocol model

The model described in this section is based on previous models for group key exchange such as those of Katz and Yung [18] and Bresson and Manulis [4]. This includes game-based security definitions.

3.1 Communication Model

A GKE protocol P is a collection of probabilistic algorithms that determines how oracles of the principals behave in response to signals (messages) from their environment.

Protocol participants and long-lived keys. Each principal V in the protocol is either a user U or a server S. In every session, each user may act as either an initiator I or a responder R. Each principal V holds long-term secret keys, and corresponding public keys of all principals are known to all.

Session identifiers and partner identifiers. Protocol principals maintain multiple instances, or sessions, that may be run simultaneously and we denote a session of principal V by the oracle $\prod_{\mathsf{V}}^{\alpha}$ with $\alpha \in \mathbb{N}$.

Each oracle $\prod_{\mathsf{V}}^{\alpha}$ is associated with the variables $\mathsf{status}_{\mathsf{V}}^{\alpha}$, $\mathsf{role}_{\mathsf{V}}^{\alpha}$, $pid_{\mathsf{V}}^{\alpha}$, $sid_{\mathsf{V}}^{\alpha}$, k_{V}^{α} as follows:

- $\mathsf{status}_{\mathsf{V}}^{\alpha}$ takes a value from $\{unused, ready, accepted, rejected\}$.
- $\mathsf{role}_{\mathsf{V}}^{\alpha}$ takes a value from: S, I, R.
- $pid_{\mathsf{V}}^{\alpha}$ contains a set of principals.
- $sid_{\mathsf{V}}^{\alpha}$ contains a string defined by the protocol.
- k_{V}^{α} the agreed session key (if any).

A session identifier, denoted *sid*, is a protocol-defined value stored at a principal intended to provide a link to other sessions in the same protocol run. A set of partner identifiers, denoted *pid*, contains the identities of all intended users in a session.

Each oracle $\prod_{\mathsf{V}}^{\alpha}$ is *unused* until initialization, by which it is told to act as a server or a user together with the long term secret keys. During initialization all oracles begin with $\mathsf{status}_{\mathsf{V}}^{\alpha} = \textit{ready}$ and $\mathsf{role}_{\mathsf{V}}^{\alpha}$, $\textit{pid}_{\mathsf{V}}^{\alpha}$, $\textit{sid}_{\mathsf{V}}^{\alpha}$ and k_{V}^{α} all equal to $\perp$.

Executing the protocol. After the protocol starts, each oracle $\prod_{\mathsf{V}}^{\alpha}$ learns its partner identifier $\textit{pid}_{\mathsf{V}}^{\alpha}$ and sends, receives and processes messages.

If the protocol at oracle $\prod_{\mathsf{V}}^{\alpha}$ *fails*, for example if signature verification or key confirmation fails, then the oracle changes its state to *rejected* and no longer responds to protocol messages. Otherwise, if V is a user, after computing k_{V}^{α} oracle $\prod_{\mathsf{V}}^{\alpha}$ changes its state to *accepted* and no longer responds to protocol messages, and if V is the server, oracle $\prod_{\mathsf{V}}^{\alpha}$ accepts after all responder oracles get their messages or expiration.

3.2 Security Notions

Adversarial model. An efficient adversary $\mathcal{A}$ interacts with sessions by using the set of queries defined below. This models the ability of $\mathcal{A}$ to completely control the network, deciding which instances run and obtaining access to other useful information. The Test query can only be asked once by $\mathcal{A}$ and is only used to measure adversary's success; it does not correspond to any actual adversary's ability.

- $\mathsf{Execute}(\mathcal{S})$: Input a set of unused oracles $\mathcal{S}$ which execute an honest run of the protocol. The oracles compute what the protocol specifies and returns the output messages.

- $\mathsf{Send}(\prod_{\mathsf{V}}^{\alpha}, m)$: Sends message m to oracle $\prod_{\mathsf{V}}^{\alpha}$. The oracle computes what the protocol defines, and sends back the output message (if any), together with the status of $\prod_{\mathsf{V}}^{\alpha}$.

- $\mathsf{Corrupt}(\mathsf{V})$: Outputs principal V's long-term secret key.

- $\mathsf{Reveal}(\prod_{\mathsf{V}}^{\alpha})$: Outputs session key k_{V}^{α} if oracle $\prod_{\mathsf{V}}^{\alpha}$ has accepted and holds some session key k_{V}^{α}.

- $\mathsf{Test}(\prod_{\mathsf{V}}^{\alpha})$: If oracle $\prod_{\mathsf{V}}^{\alpha}$ has status *accepted*, holding a session key k_{V}^{α}, then a bit b is randomly chosen and this query outputs the session key k_{V}^{α} if $b = 1$, or a random string from the session key space if $b = 0$.

Partnering. A secure GKE protocol should ensure that the session key established in an oracle $\prod_{\mathsf{V}}^{\alpha}$ is independent of session keys established in other sessions, except for the partners of $\prod_{\mathsf{V}}^{\alpha}$. This is modeled by allowing the adversary to reveal any session key except the one in the Test session and its partners. Informally, partnering is defined in such a way that oracles who are supposed to agree on the shared session key are partners.

Definition 4. Two oracles $\prod_{\mathsf{V}}^{\alpha}$ and $\prod_{\mathsf{W}}^{\beta}$ are *partners* if $pid_{\mathsf{V}}^{\alpha} = pid_{\mathsf{W}}^{\beta}$ and $sid_{\mathsf{V}}^{\alpha} = sid_{\mathsf{W}}^{\beta}$.

Freshness. The notion of freshness models the conditions on the adversary's behaviour that are required to prevent trivial wins.

Definition 5. An oracle $\prod_{\mathsf{V}}^{\alpha}$ is *fresh* if neither this oracle nor any of its partnered oracles have been asked a Reveal query, and either

- no server player nor any player in $pid_{\mathsf{V}}^{\alpha}$ was corrupted before every partnered oracle reached status *accepted*; or
- no player in $pid_{\mathsf{V}}^{\alpha}$ is ever corrupted.

Security Game. Bringing together everything we have introduced so far, we can describe the game that allows us to measure the advantage of an adversary against a GKE protocol.

Definition 6. Let P be a GKE protocol. The game $\mathbf{Exp}_{\mathsf{P}}^{\mathsf{ake}}(\mathcal{A})$ consists of the following three phases:

- *Initialization.* Each principal V runs the key generation algorithm to generate long-term key pairs. The secret keys are only known to the principal, while public keys are revealed to every principal and the adversary.

- *Queries.* The adversary $\mathcal{A}$ is allowed to make Execute, Send, Reveal, Corrupt, and Test queries. During this phase, $\mathcal{A}$ is only allowed to ask only one Test query to a fresh oracle, which should remain fresh until the end of this phase.

- *Guessing.* $\mathcal{A}$ outputs its guess b'.

The output of the game is 1 if $b = b'$, otherwise 0.

The *advantage* of the adversary $\mathcal{A}$ against the ake-security of P is

$$\mathbf{Adv}_{\mathsf{P}}^{\mathsf{ake}}(\mathcal{A}) = 2\left|\Pr[\mathbf{Exp}_{\mathsf{P}}^{\mathsf{ake}}(\mathcal{A}) = 1] - 1/2\right|.$$

4 Blinded KEM

The concept of using public-key encryption to transport keys for use in symmetric encryption is by now well studied [9, 10, 11, 19, 1, 17]. This primitive is known as a *key encapsulation mechanism* (KEM) and is used in conjunction with a *data encapsulation mechanism* (DEM) that models some symmetric encryption scheme. This KEM-DEM framework is widely deployed in internet protocols, however – as we mentioned earlier – it does not provide any forward secrecy. The cloud scenario allows the initiator to store the encapsulated key and the DEM ciphertext in some repository for the recipient to later retreive, but we ask: can the (untrusted) cloud give us some notion of forward secrecy of the key that the initator wishes to transport?

It is well known how to turn a KEM into a key exchange protocol. We shall introduce a new primitive, which we call *blinded KEM*, and in the next section we will explain how to turn such a primitive into a group key exchange protocol suitable for our purposes.

Compared to a traditional KEM, a blinded KEM has two additional algorithms: a blinding algorithm takes some encapsulation[2] and adds a blinding value, and an unblinding algorithm (that requires an unblinding key created by the blinding algorithm) removes this blinding value from the blinded key. Note that this construction does not

[2] We abuse nomenclature throughout the rest of the paper and use 'encapsulation' to refer to a key encapsulation that is yet to be blinded.

generalize existing KEMs since our decapsulation procedure works on blinded encapsulations rather than encapsulations.

The point of this new idealized primitive is to allow parties to safely outsource decapsulation by creating a blinded encapsulation, having someone else decapsulate and then unblinding the result. With careful key management, this idea will give us forward secrecy in our cloud scenario. We will develop this idea into a group key exchange protocol in the next section.

The concept of blinding is best known in the context of blind signatures, but have been used extensively in many areas of cryptography. It has also been used in the context of blind decryption [15, 24], and some of the schemes are quite similar to our constructions, even though they have very different applications in mind and also different security requirements.

After providing a definition of this primitive's algorithms, we give two natural constructions (based on DH and RSA).

Definition 7. A *blinded key encapsulation mechanism (blinded KEM)* BKEM consists of five algorithms

$$\mathsf{BKEM} = (\mathsf{KG}_{\mathsf{BKEM}}, \mathsf{Encap}, \mathsf{Blind}, \mathsf{Decap}, \mathsf{Unblind}).$$

The *key generation* algorithm $\mathsf{KG}_{\mathsf{BKEM}}$ outputs an encapsulation key ek and a decapsulation key dk. The *encapsulation* algorithm Encap takes as input an encapsulation key and outputs an encapsulation C and a key $k \in \mathcal{G}$. The *blinding* algorithm takes as input an encapsulation key and an encapsulation and outputs a blinded encapsulation $\tilde{C}$ and an unblinding key uk. The *decapsulation* algorithm Decap takes a decapsulation key and a (blinded) encapsulation as input and outputs a (blinded) key $\tilde{k}$. The *unblinding* algorithm takes as input an unblinding key and a blinded key and outputs a key.

The algorithms satisfy the correct decapsulation requirement: When $(ek, dk) \leftarrow \mathsf{KG}_{\mathsf{BKEM}}$, $(C, k) \leftarrow \mathsf{Encap}_{ek}$, $(\tilde{C}, uk) \leftarrow \mathsf{Blind}_{ek}(C)$ and $\tilde{k} \leftarrow \mathsf{Decap}_{dk}(\tilde{C})$, then

$$\mathsf{Unblind}_{uk}(\tilde{k}) = k.$$

Definition 8. Let $\mathsf{BKEM} = (\mathsf{KG}_{\mathsf{BKEM}}, \mathsf{Encap}, \mathsf{Blind}, \mathsf{Decap}, \mathsf{Unblind})$ be a blinded KEM. The *distinguishing advantage* of any adversary $\mathcal{A}$

against BKEM getting r blinded decapsulation samples is

$$\mathbf{Adv}^{\mathsf{ind}}_{\mathsf{BKEM}}(\mathcal{A}, r) = 2\Big|\Pr[\mathbf{Exp}^{\mathsf{ind}}_{\mathsf{BKEM}}(\mathcal{A}, r) = 1] - 1/2\Big|,$$

where the experiment $\mathbf{Exp}^{\mathsf{ind}}_{\mathsf{BKEM}}(\mathcal{A}, r)$ is given in Fig. 5.

```
$\underline{\mathbf{Exp}^{\mathsf{ind}}_{\mathsf{BKEM}}(\mathcal{A}, r):}$
  $b \xleftarrow{\$} \{0, 1\}$
  $(ek, dk) \leftarrow \mathsf{KG}_{\mathsf{BKEM}}$
  $(C, k_1) \leftarrow \mathsf{Encap}_{ek}$
  $k_0 \xleftarrow{\$} \mathcal{G}$
  $\mathbf{for}\ j \in \{1, \ldots, r\}\ \mathbf{do}$
    $(\tilde{C}_j, uk_j) \leftarrow \mathsf{Blind}_{ek}(C)$
    $\tilde{k}_j \leftarrow \mathsf{Decap}_{dk}(\tilde{C}_j)$
  $b' \leftarrow \mathcal{A}(ek, C, k_b, \{(\tilde{C}_j, \tilde{k}_j)\}_{1 \le j \le r})$
  $\mathbf{return}\ b' \stackrel{?}{=} b$
```

Figure 5: Indistinguishability experiment $\mathbf{Exp}^{\mathsf{ind}}_{\mathsf{BKEM}}(\mathcal{A}, r)$ for a blinded KEM.

Definition 9. Let ek be any public key and let C_0 and C_1 be two encapsulations. Define X_0 and X_1 to be the statistical distribution of the blinded encapsulation output by $\mathsf{Blind}_{ek}(C_0)$ and $\mathsf{Blind}_{ek}(C_1)$, respectively. We say that the blinded KEM is *ϵ-blind* if the statistical distance of X_0 and X_1 is at most ϵ.

Definition 10. Let ek be any public key and let C be an encapsulation of the key k. Let $\tilde{C}$ be a blinded encapsulation of C with corresponding unblinding key uk. We say that the blinded KEM is *rigid* if there is exactly one $\tilde{k}$ such that $\mathsf{Unblind}_{uk}(\tilde{k}) = k$.

We now present two instantiations of blinded KEMs based on well-known hardness assumptions, namely DDH and the RSA problem.

4.1 Construction I: DH-based

We consider the following DH-based blinded KEM (DH-BKEM). Let $\mathbb{G}$ be a group of prime order q with generator g and define DH-BKEM in Fig. 6.

```
$\mathsf{KG}_{\mathsf{BKEM}}()$ :
    $s \xleftarrow{\$} \mathbb{Z}_q^*$
    $ek \leftarrow g^s$
    $dk \leftarrow s$
    return $ek, dk$

$\mathsf{Encap}_{ek}$ :
    $i \xleftarrow{\$} \mathbb{Z}_q^*$
    $C \leftarrow g^i$
    $k \leftarrow ek^i$
    return $C, k$

$\mathsf{Blind}_{ek}(C)$ :
    $t \xleftarrow{\$} \mathbb{Z}_q^*$
    $\tilde{C} \leftarrow C^t$
    $uk \leftarrow t^{-1} \bmod q$
    return $\tilde{C}, uk$

$\mathsf{Decap}_{dk}(\tilde{C})$ :
    $\tilde{k} \leftarrow \tilde{C}^{dk}$
    return $\tilde{k}$

$\mathsf{Unblind}_{uk}(\tilde{k})$ :
    $k \leftarrow \tilde{k}^{uk}$
    return $k$
```

Figure 6: Diffie-Hellman-based blinded KEM (DH-BKEM).

Theorem 1. DH-BKEM *is a* 0*-blind* BKEM *and is rigid. Furthermore, let* $\mathcal{A}$ *be any adversary against the above construction getting* r *blinded decapsulation samples. Then there exists an adversary* $\mathcal{B}_r$ *against* DDH *such that*

$$\mathbf{Adv}^{\mathsf{ind}}_{\mathsf{DH\text{-}BKEM}}(\mathcal{A}, r) \leq \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}}(\mathcal{B}_r).$$

The running time of $\mathcal{B}_r$ *is essentially the same as the running time of* $\mathcal{A}$*.*

Proof. For any encapsulation, since t is a random number, the blinded encapsulation $\tilde{C}$ output by Blind is uniformly distributed on $\mathbb{G}$. It follows that the construction is 0-blind. In a similar vein, the unblinding procedure is a permutation on the keyspace so the construction is rigid.

```
Reduction $\mathcal{B}_r$.
    for $j \in \{1, \ldots, r\}$ do
        $t_j \xleftarrow{\$} \mathbb{Z}_q^*; \tilde{C}_j \leftarrow g^{t_j}; \tilde{k}_j \leftarrow ek^{t_j}$
    $b' \leftarrow \mathcal{A}(ek, C, k, \{(\tilde{C}_j, \tilde{k}_j)\}_{1 \le j \le r})$
    return $b'$
```

Figure 7: DDH adversary $\mathcal{B}_r$ playing $\mathbf{Exp}_{\mathbb{G}}^{\mathsf{DDH}}(\mathcal{B}_r)$, used in the proof of Theorem 1.

Next, consider a tuple (ek, C, k). The reduction $\mathcal{B}_r$ is given in Fig. 7. In the event that (ek, C, k) is a DDH tuple, then $\mathcal{B}_r$ perfectly simulates the input of $\mathcal{A}$ in $\mathbf{Exp}^{\mathsf{ind}}_{\mathsf{DH\text{-}BKEM}}(\mathcal{A}, r)$ when $b = 1$. Otherwise, $\mathcal{B}_r$ perfectly simulates the input of $\mathcal{A}$ in $\mathbf{Exp}^{\mathsf{ind}}_{\mathsf{DH\text{-}BKEM}}(\mathcal{A}, r)$ when $b = 0$. The claim follows. □

4.2 Construction II: RSA-based

We consider the following RSA-based blinded KEM ($\mathsf{RSA\text{-}BKEM}$). Unlike the above DH-based blinded KEM, this is less suitable for use in key exchange, since generating RSA keys is quite expensive. The scheme needs a hash function $\mathsf{H}_{\mathsf{RSA\text{-}BKEM}}$, and is detailed in Fig. 8.

Just like for the DH-based construction, this scheme is a blinded KEM, it is 0-blind and any adversary against indistinguishability in the random oracle model can be turned into an adversary against the RSA problem, in a straight-forward way. We omit the proof. Note that this construction is not rigid since any hash collision provides two different values that map to the same k. (Dealing with this would complicate the security proof for little gain.)

5 Offline Assisted Group Key Exchange Protocol

We now describe a generic protocol for cloud-assisted group key exchange using a blinded KEM, and then give a concrete instantiation

```
$\underline{\mathsf{KG}_{\mathsf{BKEM}}():}$
    $p, q, n, e, d \leftarrow \mathsf{RSA.KG}$
    $ek \leftarrow (n, e)$
    $dk \leftarrow (n, d)$
    $\mathbf{return}\ ek, dk$

$\underline{\mathsf{Encap}_{ek}:}$
    $i \xleftarrow{\$} \{1, \ldots, n-1\}$
    $C \leftarrow i^e \bmod n$
    $k \leftarrow \mathsf{H}_{\mathsf{RSA\text{-}BKEM}}(i)$
    $\mathbf{return}\ C, k$

$\underline{\mathsf{Blind}_{ek}(C):}$
    $t \xleftarrow{\$} \{1, \ldots, n-1\}$
    $\tilde{C} \leftarrow (t^e C) \bmod n$
    $uk \leftarrow t^{-1} \bmod n$
    $\mathbf{return}\ \tilde{C}, uk$

$\underline{\mathsf{Decap}_{dk}(\tilde{C}):}$
    $\tilde{k} \leftarrow \tilde{C}^d \bmod n$
    $\mathbf{return}\ \tilde{k}$

$\underline{\mathsf{Unblind}_{uk}(\tilde{k}):}$
    $k' \leftarrow (\tilde{k}uk) \bmod n$
    $k \leftarrow \mathsf{H}_{\mathsf{RSA\text{-}BKEM}}(k')$
    $\mathbf{return}\ k$
```

Figure 8: RSA-based blinded KEM (RSA-BKEM).

using our DH-based blinded KEM from Section 4.1. Our scenario consists of the following participants:

- The *initiator* wants to establish a shared key k with a set of responders. First, the initiator I interacts with the server, then the initiator generates a key and "invitation messages" for the responders $\mathsf{R}_1, ..., \mathsf{R}_n$.

- Each *responder* wants to allow the initiator to establish a shared key with him. When responder R_i gets their "invitation message" from the initiator, they will interact with the server to decrypt the shared key.

- The *server* temporarily stores information assisting in the computation of the shared secret key k, until every responder has gotten the key.

A conceptual overview of our construction is given in Fig. 9: the numbering indicates the order in which the phases of the protocol

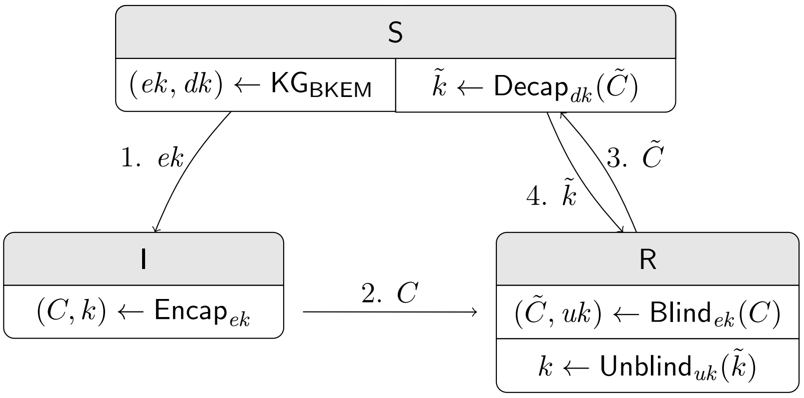
S
$(ek, dk) \leftarrow \mathsf{KG}_{\mathsf{BKEM}}$
$\tilde{k} \leftarrow \mathsf{Decap}_{dk}(\tilde{C})$
1. ek
4. $\tilde{k}$
3. $\tilde{C}$
I
$(C, k) \leftarrow \mathsf{Encap}_{ek}$
2. C
R
$(\tilde{C}, uk) \leftarrow \mathsf{Blind}_{ek}(C)$
$k \leftarrow \mathsf{Unblind}_{uk}(\tilde{k})$

Figure 9: Diagram describing how the group key exchange protocol uses the blinded KEM to do key exchange in the single responder case. For clarity, identities, nonces, session identifiers, key confirmation, public key encryption and digital signatures are omitted. Fig. 10 contains a more detailed message sequence chart for the single responder case.

are done. A more diagrammatic overview is provided for the single-responder case in Fig. 10, and the general case is presented in Fig. 11. In these figures and for the rest of this section we will reduce notational overload by writing $\mathsf{Sign}_{\mathsf{R}_j}$ instead of $\mathsf{Sign}_{sk_{\mathsf{R}_j}}$ (and $\mathsf{Enc}_{\mathsf{R}_j}$ instead of $\mathsf{Enc}_{pk_{\mathsf{R}_j}}$ etc.), and allow the reader to infer which type of key is being used from the algorithm in use.

Definition 11. An Offline Assisted Group Key Exchange Protocol (OAGK) is defined in Fig. 11 and is parameterized by the following components. Let

- $\mathsf{BKEM} = (\mathsf{KG}_{\mathsf{BKEM}}, \mathsf{Encap}, \mathsf{Blind}, \mathsf{Decap}, \mathsf{Unblind})$ be a *blinded KEM*,
- $\mathsf{DS} = (\mathsf{KG}_{\mathsf{sig}}, \mathsf{Sign}, \mathsf{Verify})$ be a *signature scheme*,
- $\mathsf{PKE} = (\mathsf{KG}_{\mathsf{pke}}, \mathsf{Enc}, \mathsf{Dec})$ be a *public-key encryption scheme*,
- H be a hash function,

I		S		R
		Stage 1:		
Choose nonce $\mathsf{N_I}$				
$\sigma_1 \leftarrow \mathsf{Sign_I}(\mathsf{N_I}, \mathit{pid})$				
	$\xrightarrow{\mathsf{N_I}, \mathit{pid}, \sigma_1}$			
		Verify σ_1		
		$(ek, dk) \leftarrow \mathsf{KG_{BKEM}}$		
		$\sigma_2 \leftarrow \mathsf{Sign_S}(\mathsf{N_I}, \mathit{pid}, ek)$		
		$\mathit{sid} \leftarrow \mathsf{H}(\mathsf{I}, \mathsf{N_I}, \mathit{pid}, ek)$		
Verify σ_2	$\xleftarrow{(ek, \sigma_2)}$			
$\mathit{sid} \leftarrow \mathsf{H}(\mathsf{I}, \mathsf{N_I}, \mathit{pid}, ek)$				
		Stage 2:		
$(C, k) \leftarrow \mathsf{Encap}_{ek}$				
$k_\mathsf{I} \leftarrow \mathsf{KDF}(''1'', k, \mathit{sid})$				
$\tau_\mathsf{I} \leftarrow \mathsf{KDF}(''2'', k, \mathit{sid})$				
$c \leftarrow \mathsf{Enc_R}(C, ek, \tau_\mathsf{I}, \mathit{sid}, \mathit{pid})$				
$\sigma_3 \leftarrow \mathsf{Sign_I}(c)$				
	$\xrightarrow{c, \sigma_3}$			Verify σ_3
Accept k_I				$(C, ek, \tau_\mathsf{I}, \mathit{sid}, \mathit{pid}) \leftarrow \mathsf{Dec_R}(c)$
		Stage 3:		
				$(\tilde{C}, uk) \leftarrow \mathsf{Blind}_{ek}(C)$
				$\sigma_4 \leftarrow \mathsf{Sign_R}(\mathit{sid}, ek, \tilde{C})$
		Verify σ_4	$\xleftarrow{(\mathit{sid}, \tilde{C}, \sigma_4)}$	
		Verify $\mathsf{R} \in \mathit{pid}$		
		$\tilde{k} \leftarrow \mathsf{Decap}_{dk}(\tilde{C})$		
		$\sigma_5 \leftarrow \mathsf{Sign_S}(\mathit{sid}, \tilde{k})$		
			$\xrightarrow{(\mathit{sid}, \tilde{k}, \sigma_5)}$	Verify σ_5
				$k \leftarrow \mathsf{Unblind}_{uk}(\tilde{k})$
				$k_\mathsf{R} \leftarrow \mathsf{KDF}(''1'', k, \mathit{sid})$
				$\tau_\mathsf{R} \leftarrow \mathsf{KDF}(''2'', k, \mathit{sid})$
				$\tau_\mathsf{R} \overset{?}{=} \tau_\mathsf{I}$
				Accept k_R

Figure 10: Message sequence chart for the OAGK protocol with a single responder R. Fig. 11 contains a complete protocol description for the multi-responder case.

- KDF be a key derivation function.

Note that in our model, we do not have a reveal state query, so there is no need to explicitly erase state information. In a real implementation, making sure that ephemeral and medium-term key material is erased at appropriate times is vital.

In order to break our protocol an adversary must compromise both the server and one of the users. The server stores a medium-term key which is deleted after the protocol run is complete (or after a time-out) after which compromise of the server is allowed. We note that it would not be difficult to enhance our protocol with *forward secure encryption* [6] if receiver compromise is deemed a likely risk.

5.1 Efficiency

There are different ways to measure the efficiency of group key exchange protocols, including the number of protocol messages, the number of rounds of parallel messages, and the (average) computation per user. There exist theoretically efficient examples [2, 3] but most practical protocols employ a generalisation of the Diffie–Hellman protocol. One such generalisation is the well-known scheme of Burmester and Desmedt [5] which requires 2 rounds of communication and 3 exponentiations per user in its unauthenticated version.

An example of a modern optimised protocol is that of Gao et al. [13] which adds signatures to all messages and requires users to verify the signature on broadcast messages from all other users. In comparison our requirements are relatively modest. We require 3 rounds but do not use broadcast messages at all. The protocol participants perform 5 public key operations each, consisting of signature generation/verification, public key encryption/decryption and key encapsulation/decapsulation. As mentioned, the non-interactive nature of our scenario means that we wish for the initiator to be able to do all of their interaction during some initial phase.

5.2 Protocol Security

An adversary against the GKE protocol OAGK plays the game defined in Section 3.2. We need to give a useful bound for its advantage.

```
$\mathsf{I}$ running oracle $\prod_{\mathsf{I}}^{\alpha}$ as initiator on input $pid$:
  1. Choose random $\mathsf{N_I}$.
  2. $\sigma_1 \leftarrow \mathsf{Sign_I}(\mathsf{N_I}, pid)$.
  3. Send $(\mathsf{N_I}, pid, \sigma_1)$ to $\mathsf{S}$.
  10. Get $(ek, \sigma_2)$ from $\mathsf{S}$.
  11. Verify that $\sigma_2$ is $\mathsf{S}$'s signature on $(\mathsf{N_I}, pid, ek)$.
  12. $sid \leftarrow \mathsf{H}(\mathsf{I}, \mathsf{N_I}, pid, ek)$.
  13. $(C, k) \leftarrow \mathsf{Encap}_{ek}$.
  14. Session key $k_{\mathsf{I}}^{\alpha} \leftarrow \mathsf{KDF}(''1'', k, sid)$
  15. Key confirmation: $\tau_{\mathsf{I}}^{\alpha} \leftarrow \mathsf{KDF}(''2'', k, sid)$
  16. For every responder $\mathsf{R}_j$ in $pid$, do:
      (a) $c_j \leftarrow \mathsf{Enc}_{\mathsf{R}_j}(C, ek, \tau_{\mathsf{I}}^{\alpha}, sid, pid)$.
      (b) $\sigma_{3,j} \leftarrow \mathsf{Sign_I}(c_j)$.
      (c) Send $(c_j, \sigma_{3,j})$ to $\mathsf{R}_j$.
  17. Output $k_{\mathsf{I}}^{\alpha}$.

Phase I of $\mathsf{S}$ running oracle $\prod_{\mathsf{S}}^{\beta}$ as server on message $(\mathsf{N_I}, pid, \sigma_1)$ from $\mathsf{I}$:
  4. Verify that $\sigma_1$ is $\mathsf{I}$'s signature on $(\mathsf{N_I}, pid)$.
  5. $(ek, dk) \leftarrow \mathsf{KG_{BKEM}}$.
  6. $\sigma_2 \leftarrow \mathsf{Sign_S}(\mathsf{N_I}, pid, ek)$.
  7. $sid \leftarrow \mathsf{H}(\mathsf{I}, \mathsf{N_I}, pid, ek)$.
  8. Store $(sid, \mathsf{I}, pid, dk, \emptyset)$.
  9. Send $(ek, \sigma_2)$ to $\mathsf{I}$.
```

Figure 11: Part 1. The three roles of the group key exchange protocol. Suppose $\{\mathsf{R}_j\}_{j \in J}$ are the identities of users that I wishes to share a common session key with ($pid = \mathsf{I} | \{\mathsf{R}_j\}_{j \in J}$). Note that the line numbering indicates the order in which the lines of the various roles are reached during a protocol execution.

R_j running oracle $\prod_{\mathsf{R}_j}^{\nu}$ as responder on message $(c_j, \sigma_{3,j})$ from I:

```
18. Verify that $c_j$ is $\mathsf{I}$'s signature on $\sigma_{3,j}$.
19. $(C, ek, \tau_{\mathsf{I}}^{\alpha}, sid, pid) \leftarrow \mathsf{Dec}_{\mathsf{R}_j}(c_j)$.
20. $(\tilde{C}_j, uk_j) \leftarrow \mathsf{Blind}_{ek}(C)$.
21. $\sigma_4 \leftarrow \mathsf{Sign}_{\mathsf{R}_j}(sid, ek, \tilde{C}_j)$.
22. Send $(sid, \tilde{C}_j, \sigma_4)$ to $\mathsf{S}$.
32. Get $(sid, \tilde{k}_j, \sigma_5)$ from $\mathsf{S}$.
33. Verify that $\sigma_5$ is $\mathsf{S}$'s signature on $(sid, \tilde{k}_j)$.
34. $k_j \leftarrow \mathsf{Unblind}_{uk_j}(\tilde{k}_j)$.
35. Session key: $k_{\mathsf{R}_j}^{\nu} \leftarrow \mathsf{KDF}(''1'', k_j, sid)$
36. Key confirmation: $\tau_{\mathsf{R}_j}^{\nu} \leftarrow \mathsf{KDF}(''2'', k_j, sid)$
    If $\tau_{\mathsf{R}_j}^{\nu} = \tau_{\mathsf{I}}^{\alpha}$ then
        Accept and output $k_{\mathsf{R}_j}^{\nu}$.
    else
        Reject.
```

Phase II of S running oracle $\prod_{\mathsf{S}}^{\beta}$ as server on message $(sid, \tilde{C}_j, \sigma_4)$ from R_j, with stored state $(sid, \mathsf{I}, pid, dk, T)$:

```
23. Lock the state $(sid, \dots)$ until done.
24. Verify that $\sigma_4$ is $\mathsf{R}_j$'s signature on $(sid, ek, \tilde{C}_j)$.
25. Verify that $\mathsf{R}_j \in pid$.
26. Verify that $\mathsf{R}_j \notin T$.
27. $\tilde{k}_j \leftarrow \mathsf{Decap}_{dk}(\tilde{C}_j)$.
28. $\sigma_5 \leftarrow \mathsf{Sign}_{\mathsf{S}}(sid, \tilde{k}_j)$.
29. Send $(sid, \tilde{k}_j, \sigma_5)$ to $\mathsf{R}_j$.
30. Let $T' = T \cup \{\mathsf{R}_j\}$.
31. Update the state $(sid, \dots, T)$ to $(sid, \dots, T')$.
```

Figure 11: Part 2. The three roles of the group key exchange protocol. Suppose $\{\mathsf{R}_j\}_{j \in J}$ are the identities of users that I wishes to share a common session key with ($pid = \mathsf{I}|\{\mathsf{R}_j\}_{j \in J}$). Note that the line numbering indicates the order in which the lines of the various roles are reached during a protocol execution.

Theorem 2. *Consider an adversary $\mathcal{A}$ against the GKE protocol* OAGK *running with n users, having at most s sessions, each involving at most r responders. Then adversaries $\mathcal{B}_0$, $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$ and $\mathcal{B}_4$ exist, running in essentially the same time as $\mathcal{A}$, such that*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{ake}}_{\mathsf{OAGK}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{CR}}_{\mathsf{H}}(\mathcal{B}_0) &+ (n+1)\mathbf{Adv}^{\mathsf{suf\text{-}cma}}_{\mathsf{DS}}(\mathcal{B}_1) \\ &+ snr\mathbf{Adv}^{\mathsf{ror\text{-}cca2}}_{\mathsf{PKE}}(\mathcal{B}_2) + s\mathbf{Adv}^{\mathsf{CR}}_{\mathsf{KDF}}(\mathcal{B}_3) + sr\epsilon \\ &+ s\mathbf{Adv}^{\mathsf{ind}}_{\mathsf{BKEM}}(\mathcal{B}_4, r) \\ &+ \textit{negligible terms.}\end{aligned}$$

We sketch the ideas used in the proof. We need to guess which session the adversary is going to issue the Test query for. If we guess correctly, the game proceeds unchanged. If we guess incorrectly, the game immediately stops, we flip a coin b' and pretend that the adversary output b'. It is clear that the adversary's advantage in this game is now $1/s$ times the original advantage.

We must also handle the situation where the adversary issues a corruption query that would render our chosen session non-fresh. In this case, the game immediately stops, we flip a coin b' and pretend that the adversary output b'. Observe that if we stop for this reason, the adversary could not issue a test query (and our chosen session is now the only session a test query could be issued for), so the adversary would have no information about b. The probability that the adversary guesses b correctly is therefore unchanged.

Depending on when the server is corrupted (if it is corrupted at all), we need to bound the adversary's advantage in slightly different ways. An upper bound on the adversary's advantage will then be the sum of the two different bounds.

If we suppose that every partnered oracle in our session reached status *accepted* before the server or any player running a partnered oracle is corrupted. In this case, thanks to the signatures and the nonces, the adversary sees at most a blinded KEM public encapsulation key, an encapsulation of a session key, at most r blinded encapsulations of the same session key with corresponding blinded decapsulations. By indistinguishability for the blinded KEM, it follows that the adversary cannot distinguish between the actual encapsulated key and a randomly chosen key, so the adversary has no information about b.

Next, suppose no responder player is ever corrupted. In this case, the adversary (in the worst case) chooses the keys for the blinded KEM, but the public key encryption ensures that the adversary cannot see the actual encapsulation of the key. In other words, the adversary only sees blinded encapsulations of an unknown encapsulation, which reveals little information about the encapsulated key by ϵ-blindness of the blinded KEM. Furthermore, the rigidity of the blinded KEM ensures that every responder can detect an incorrect server response, unless a collision in the key derivation function occurs.

5.3 Proof of Theorem 2

The proof of the theorem consists of a sequence of games.

Game 0

The first game is the game from Def. 6, defining security for our protocol. Let E_0 be the event that the adversary's guess b' equals b from the Test oracle (and let E_i be the corresponding event for Game i). Then

$$\mathbf{Adv}^{\mathsf{ake}}_{\mathsf{OAGK}}(\mathcal{A}) = \Big|\mathbf{Pr}[E_0] - 1/2\Big|. \tag{1}$$

Game 1

We modify the game so that if two server oracles or two initiator oracles ever arrive at the same sid, the game stops.

For this to happen, either two different sessions at sid computing algorithm must choose the same input values for hash function, or we have found a collision in H. The former event is included inside the event that two server oracles choose the same values for ek and two initiator oracles choose the same values for $\mathsf{N_I}$. Since there are at most s initiator oracles and server oracles, and s^2 must be small compared to the number of possible nonces and KEM encapsulation keys, the only possible non-negligible term[3] is the possibility of finding

[3]To be secure, the KEM key generation algorithm must provide sufficient min-entropy to allow us to ignore the possibility that the KEM encapsulation keys collide.

a collision in H. We can easily construct a collision-finding algorithm $\mathcal{B}_0$ from $\mathcal{A}$, which shows that

$$\Big|\mathbf{Pr}[E_1] - \mathbf{Pr}[E_0]\Big| \leq \mathbf{Adv}_{\mathsf{H}}^{\mathsf{CR}}(\mathcal{B}_0) + \text{negligible terms.} \tag{2}$$

Game 2

We modify the game so that if any oracle ever verifies a signature from an uncorrupted principal that was not created by another oracle, the game stops.

If this happens, our adversary has produced a forgery for DS. We can trivially produce a forger $\mathcal{B}_1$ for the signature scheme using a standard hybrid argument. Since our n users and the server all have a signing key, we get that

$$\Big|\mathbf{Pr}[E_2] - \mathbf{Pr}[E_1]\Big| \leq (n+1)\mathbf{Adv}_{\mathsf{DS}}^{\mathsf{suf\text{-}cma}}(\mathcal{B}_1). \tag{3}$$

By inspection of the protocol, it is now apparent that in Game 2, the partnering relation on oracles from Def. 4 is an equivalence relation on accepting oracles for which

- every equivalence class whose *pid* contains an uncorrupted initiator contains an initiator oracle; and
- if the server is uncorrupted, every equivalence class contains exactly one server oracle.

Furthermore, this equivalence relation can be extended to an equivalence relation on all oracles, where oracles are related if and only if they have the same *sid*.

Game 3

The next modification we make is to guess which session the adversary will query with the Test query, by choosing a number uniformly at random from $\{1, 2, \ldots, s\}$, identifying the corresponding initiator oracle and guessing that session. If the adversary sends the Test query to this session, we proceed as usual. Otherwise, we stop when the adversary issues the Test query, flip a coin b' and pretend that the adversary output b'.

Since we choose the session randomly, the adversary cannot know anything about which session we choose. It follows that

$$\left|\mathbf{Pr}[E_2] - 1/2\right| = s\left|\mathbf{Pr}[E_3] - 1/2\right|. \tag{4}$$

Game 4

The next modification we make is that if the adversary every issues a Corrupt query such that our chosen session becomes unfresh, we stop the game, flip a coin b' and pretend that the adversary output b'.

If we never stop the game, this game proceeds exactly as Game 3.

If the adversary corrupts players so that our chosen session becomes unfresh, the adversary cannot ask a Test query of our session. This means that in Game 3, the eventual Test query would go to some other session, which would cause the game to stop and a coin b' to be flipped.

We get that

$$\mathbf{Pr}[E_4] = \mathbf{Pr}[E_3]. \tag{5}$$

Let F be the event that the server is corrupted before our chosen session has completed. Referring to the two clauses in Def. 5, if F is false, the first clause applies, otherwise the second clause applies.

It is easy to show that

$$|\mathbf{Pr}[E_4] - 1/2| \leq |\mathbf{Pr}[E_4|F] - 1/2| + |\mathbf{Pr}[E_4|\neg F] - 1/2|. \tag{6}$$

We can therefore analyse the two cases separately, which we shall proceed to do, using two sequences of games, each beginning with Game 4.

Game 5

We begin by assuming that the server is corrupted, which by the freshness requirements means that the adversary will never get to corrupt the players in our chosen session. We modify the game by having our initiator oracle encrypt random messages instead the real messages. Any responder oracle that receives this exact ciphertext will use values directly from our initiator oracle, instead of decrypting the (nonsense) ciphertext.

We shall now use $\mathcal{A}$ and any difference in $\mathbf{Pr}[E_4|F]$ and $\mathbf{Pr}[E_5]$ to construct an adversary $\mathcal{B}_2$ against multi-user security of public key encryption, as defined in Appendix A.

Our adversary $\mathcal{B}_2$ works as follows:

- It gets encryption keys for the users as input.
- When $\mathcal{B}_2$ must simulate a responder oracle that gets input from a corrupted initiator, it uses its decryption oracle to get the decryption of the ciphertexts.
- When $\mathcal{B}_2$ simulates responders that get input from an uncorrupted initiator, then because we have forbidden signature forgeries, the ciphertext was created by an initiator oracle, so $\mathcal{B}_2$ knows what is inside the ciphertext and does not need to decrypt that ciphertext.
- When the adversary corrupts a principal, $\mathcal{B}_2$ gets the decryption key from its oracle.
- When simulating the initiator oracle of our chosen session, $\mathcal{B}_2$ uses its encryption oracle to encrypt the messages.

We see that if $\mathcal{B}_2$'s encryption oracle encrypts the real messages, $\mathcal{B}_2$ perfectly simulates the situation in Game 4 given F. If $\mathcal{B}_2$'s encryption oracle encrypts random messages, $\mathcal{B}_2$ perfectly simulates the situation in Game 5 given F.

We get that

$$\Big|\mathbf{Pr}[E_5|F] - \mathbf{Pr}[E_4|F]\Big| \leq nr\mathbf{Adv}^{\mathsf{ror\text{-}cca2}}_{\mathsf{PKE}}(\mathcal{B}_2). \tag{7}$$

Game 6

Next, we modify the responder oracles in our chosen session so that they reject if the unblinded decapsulated key k_j computed in Step 34 does not match the key k computed by the initiator oracle in Step 13.

If a responder oracle rejects in this game, but would not have rejected in the previous game, it has found a collision in KDF. We can therefore construct a collision finder $\mathcal{B}_3$ such that

$$\left|\mathbf{Pr}[E_6|F] - \mathbf{Pr}[E_5|F]\right| \leq \mathbf{Adv}^{\mathsf{CR}}_{\mathsf{KDF}}(\mathcal{B}_3). \tag{8}$$

Game 7

In this game, we modify the responder oracles of our chosen session so that instead of using the encapsulation sent by the initiatior oracle, they create their own encapsulation of a random, independent key using the corrupt server's encapsulation key, blind it and compare the unblinded decapsulation with this key. Instead of computing the key to be output, they simply output the one output by the initiator oracle.

By rigidity, there is exactly one server response that a responder oracle will accept, and this answer depends only on the blinding sent by the responder, not on which encapsulation was used to create the blinding.

It follows by ϵ-blindness that

$$\left|\mathbf{Pr}[E_7|F] - \mathbf{Pr}[E_6|F]\right| \leq r\epsilon. \tag{9}$$

Furthermore, we see that in this game, the adversary has no information about the key chosen by the initiator oracle and later output by the responder oracles. This means that if the adversary asks a Test query for this session the response will be a random key, regardless of the value of b. It follows that

$$\mathbf{Pr}[E_7|F] = 1/2. \tag{10}$$

By equations (7)–(10) we get that

$$\left|\mathbf{Pr}[E_4|F] - 1/2\right| \leq nr\mathbf{Adv}^{\mathsf{ror\text{-}cca2}}_{\mathsf{PKE}}(\mathcal{B}_2) + \mathbf{Adv}^{\mathsf{CR}}_{\mathsf{KDF}}(\mathcal{B}_3) + r\epsilon. \tag{11}$$

Game 5'

Now we assume that the server is not corrupted until every responder has accepted. We modify the game so that in our chosen session, the initiator oracle ignores the encapsulated key and instead outputs a randomly chosen key. The responder oracles also ignore the key they compute and instead output the key chosen by the initiator oracle.

We can now construct an adversary $\mathcal{B}_4$ against indistinguishability for our blinded KEM. The adversary $\mathcal{B}_4$ gets an encapsulation key, an encapsulation, a key and r pairs of blindings and blinded decapsulations as input. It uses the encapsulation key to simulate the server message to the initiator oracle. It uses the encapsulation to simulate the messages to the responders. And it uses the blindings and blinded decapsulations to simulate the conversations between the responders and the server. Finally, it has the oracles of our chosen session output its input key.

We see that if the key input to $\mathcal{B}_4$ is the real encapsulated key, then $\mathcal{B}_4$ perfectly simulates the situation in Game 4 given $\neg F$. If the key input to $\mathcal{B}_4$ is a random key, then $\mathcal{B}_4$ perfectly simulates the situation in this game given $\neg F$.

We get that

$$\left|\mathbf{Pr}[E_{5'}|\neg F] - \mathbf{Pr}[E_4|\neg F]\right| = \mathbf{Adv}^{\mathsf{ind}}_{\mathsf{BKEM}}(\mathcal{B}_4, r). \tag{12}$$

Furthermore, if the adversary asks a Test query for our chosen session in this game, the response will be a random key regardless of the value of b. It follows that

$$\mathbf{Pr}[E_{5'}|\neg F] = 1/2. \tag{13}$$

By equations (12) and (13) we get that

$$\left|\mathbf{Pr}[E_4|\neg F] - 1/2\right| \leq \mathbf{Adv}^{\mathsf{ind}}_{\mathsf{BKEM}}(\mathcal{B}_4, r). \tag{14}$$

The claim now follows by equations (1)–(6), (11) and (14).

5.4 Instantiating the protocol with the DH-BKEM

We instantiate the above offline assisted group key exchange protocol OAGK with the DH-based blinded KEM from Section 4.1, the protocol denoted by $\mathsf{DH\text{-}OAGK}$. In this instantiation, we choose the nonce $\mathsf{N_I}$ from the group $\mathbb{G}$.

In Fig. 12, we present the core of the resulting protocol (without identities, nonces, session identifiers, key confirmation, authentication and encryption) similar to Fig. 9. We only show one responder.

Thm. 1 and Thm. 2 show that this instantiation is secure.

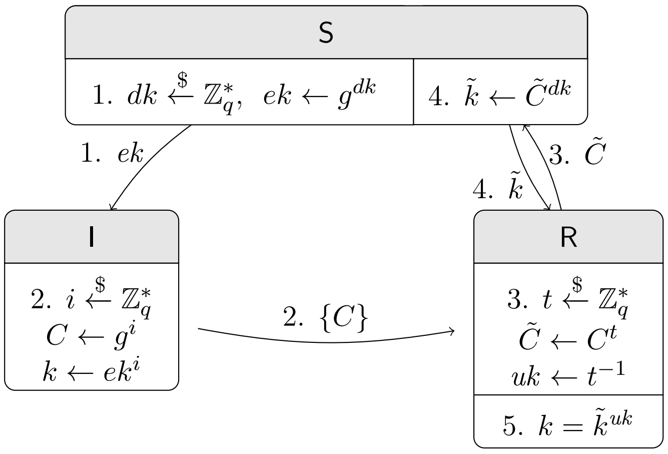
S
1. $dk \xleftarrow{\$} \mathbb{Z}_q^*$, $ek \leftarrow g^{dk}$
4. $\tilde{k} \leftarrow \tilde{C}^{dk}$
1. ek
3. $\tilde{C}$
4. $\tilde{k}$
I
2. $i \xleftarrow{\$} \mathbb{Z}_q^*$
$C \leftarrow g^i$
$k \leftarrow ek^i$
2. $\{C\}$
R
3. $t \xleftarrow{\$} \mathbb{Z}_q^*$
$\tilde{C} \leftarrow C^t$
$uk \leftarrow t^{-1}$
5. $k = \tilde{k}^{uk}$

Figure 12: Running protocol DH-OAGK with one responder, where $\{C\} = \mathsf{Enc}_{\mathsf{R}}(C, \cdots)$.

References

[1] Masayuki Abe, Rosario Gennaro, Kaoru Kurosawa, and Victor Shoup. Tag-KEM/DEM: A new framework for hybrid encryption and a new analysis of Kurosawa–Desmedt KEM. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 3494 of *Lecture Notes in Computer Science*, pages 128–146. Springer, 2005.

[2] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *IACR Cryptology ePrint Archive*, 2002:80, 2002.

[3] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfusca-

tion. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014: 34th Annual Cryptology Conference*, pages 480–499. Springer Berlin Heidelberg, 2014.

[4] Emmanuel Bresson and Mark Manulis. Securing group key exchange against strong corruptions. In Masayuki Abe and Virgil D. Gligor, editors, *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2008*, pages 249–260. ACM, 2008.

[5] Mike Burmester and Yvo Desmedt. A secure and efficient conference key distribution system. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 275–286. Springer, 1995.

[6] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. *J. Cryptology*, 20(3):265–294, 2007.

[7] Cheng-Kang Chu, Sherman S. M. Chow, Wen-Guey Tzeng, Jianying Zhou, and Robert H. Deng. Key-aggregate cryptosystem for scalable data sharing in cloud storage. *IEEE Trans. Parallel Distrib. Syst.*, 25(2):468–477, 2014.

[8] Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. On ends-to-ends encryption: Asynchronous group messaging with strong security guarantees. Cryptology ePrint Archive, Report 2017/666, 2017. https://eprint.iacr.org/2017/666.

[9] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *IACR Cryptology ePrint Archive*, 2001:108, 2001.

[10] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1):167–226, 2003.

[11] Alexander W. Dent. A designer's guide to KEMs. In Kenneth G. Paterson, editor, *Cryptography and Coding, 9th IMA International Conference, Cirencester, UK, December 16-18, 2003, Proceedings*, volume 2898 of *Lecture Notes in Computer Science*, pages 133–151. Springer, 2003.

[12] David Derler, Tibor Jager, Daniel Slamanig, and Christoph Striecks. Bloom filter encryption and applications to efficient forward-secret 0-rtt key exchange. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 425–455. Springer, 2018.

[13] Weizheng Gao, Kashi Neupane, and Rainer Steinwandt. Tuning a two-round group key agreement. *Int. J. Inf. Sec.*, 13(5):467–476, 2014.

[14] M. D. Green and I. Miers. Forward secure asynchronous messaging from puncturable encryption. In *2015 IEEE Symposium on Security and Privacy*, pages 305–320, May 2015.

[15] Matthew Green. Secure blind decryption. In *Proceedings of the 14th International Conference on Practice and Theory in Public Key Cryptography Conference on Public Key Cryptography*, PKC'11, pages 265–282, Berlin, Heidelberg, 2011. Springer-Verlag.

[16] Felix Günther, Britta Hale, Tibor Jager, and Sebastian Lauer. 0-RTT key exchange with full forward secrecy. In *EUROCRYPT (3)*, volume 10212 of *Lecture Notes in Computer Science*, pages 519–548, 2017.

[17] Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference*, volume 4622 of *Lecture Notes in Computer Science*, pages 553–571. Springer, 2007.

[18] Jonathan Katz and Moti Yung. Scalable protocols for authenticated group key exchange. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, pages 110–125. Springer Berlin Heidelberg, 2003.

[19] Kaoru Kurosawa and Yvo Desmedt. A new paradigm of hybrid encryption scheme. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2004.

[20] István Lám, Szilveszter Szebeni, and Levente Buttyán. Invitation-oriented TGDH: key management for dynamic groups in an asynchronous communication model. In *41st International Conference on Parallel Processing Workshops, ICPPW 2012*, pages 269–276. IEEE Computer Society, 2012.

[21] István Lám, Szilveszter Szebeni, and Levente Buttyán. Tresorium: Cryptographic file system for dynamic groups over untrusted cloud storage. In *41st International Conference on Parallel Processing Workshops, ICPPW 2012*, pages 296–303. IEEE Computer Society, 2012.

[22] Xuefeng Liu, Yuqing Zhang, Boyang Wang, and Jingbo Yan. Mona: Secure multi-owner data sharing for dynamic groups in the cloud. *IEEE Trans. Parallel Distrib. Syst.*, 24(6):1182–1191, 2013.

[23] Moxie Marlinspike and Trevor Perrin. The X3DH key agreement protocol. `https://signal.org/docs/specifications/x3dh/`, November 2016.

[24] Kouichi Sakurai and Yoshinori Yamane. Blind decoding, blind undeniable signatures, and their applications to privacy protection. In *Proceedings of the First International Workshop on Information Hiding*, pages 257–264, London, UK, UK, 1996. Springer-Verlag.

A Multi-user Public-Key Encryption Security

In the proof of the main theorems, it is convenient to consider a multi-user variant of public key encryption. The security notion we consider is equivalent to the usual real-or-random security notion for public key encryption. We first explain and define the notion and then prove the relevant theorem.

We consider a *multi-user setting* with n users. All users use PKE, each user U_i keeps their own secret decryption key sk_i and all public encryption keys are assumed to be known to the public (and thus all algorithms).

For our security analysis we define the *adversary's capacity.* The adversary is given all public keys and can ask for challenge encryptions of any (valid) message under different public keys. In a chosen-ciphertext attack the adversary is allowed to ask for decryptions of arbitrary ciphertexts, except for those that would allow a trivial win.

We also give the adversary the ability to corrupt a user, that is, obtain the secret key of the corrupted user. In order to prevent trivial wins, we must restrict this capability to users for which the adversary has not yet asked for challenge encryptions. (This is a fundamental restriction for ordinary public-key encryption. For other notions such as puncturable encryption or non-committing encryption, this restriction could be somewhat relaxed.)

We now define *real-or-random indistinguishability* for a multi-user public-key encryption scheme under chosen-ciphertext attack and corruption attack ($\mathsf{mu\text{-}ror\text{-}cca2}$): an adversary cannot distinguish encryptions of chosen plaintexts, possibly encrypted under different public keys, from the encryptions of equal-length random strings, encrypted under the same public keys.

In the definition of the security experiment we employ a list $\mathsf{F}_{\mathsf{LIST}}$ of of forbidden ciphertext, a list $\mathsf{U}_{\mathsf{LIST}}$, and a corrupted user $\mathsf{C}_{\mathsf{LIST}}$ to prevent trivial wins.

Remark 1. *We now describe the restrictions on our adversaries that we enforce by using* $\mathsf{F}_{\mathsf{LIST}}$*,* $\mathsf{U}_{\mathsf{LIST}}$ *and* $\mathsf{C}_{\mathsf{LIST}}$*. If the adversary asks its real-or-random (*$\mathcal{O}.\mathsf{RoR}$*) challenge oracle for some corrupted user (that belongs to* $\mathsf{C}_{\mathsf{LIST}}$*), the oracle will return encryptions of real messages.*

If the adversary asks for decryptions of some ciphertext that it received from its $\mathcal{O}.\mathsf{RoR}$ *oracle, the adversary will obtain nothing (to stop trivial wins). In a* $\mathsf{Corrupt}$ *query, the adversary cannot reveal the secret key of some user in* $\mathsf{U_{LIST}}$*, since the challenge oracle has returned encryptions under their key (which means that revealing the key would allow the adversary to win trivially by decrypting the challenge ciphertext).*

Definition 12. Let $\mathsf{PKE} = (\mathsf{KG_{pke}}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme. Then the $\mathsf{mu\text{-}ror\text{-}cca2}$ advantage of an adversary $\mathcal{A}$ against PKE is defined as

$$\mathbf{Adv}_{\mathsf{PKE}}^{(t,\,n,\,c)\text{-}\mathsf{mu\text{-}ror\text{-}cca2}}(\mathcal{A}) = 2\left|\mathbf{Pr}[\mathbf{Exp}_{\mathsf{PKE}}^{(t,\,n,\,c)\text{-}\mathsf{mu\text{-}ror\text{-}cca2}}(\mathcal{A}) = 1] - \frac{1}{2}\right|.$$

where n is the number of users, c the maximal number of corrupted users and t the maximal number of challenge ciphertexts the adversary can receive, respectively. The experiment $\mathbf{Exp}_{\mathsf{PKE}}^{(t,\,n,\,c)\text{-}\mathsf{mu\text{-}ror\text{-}cca2}}(\mathcal{A})$ is given in Fig. 13.

```
$\mathbf{Exp}_{\mathsf{PKE}}^{(t,\,n,\,c)\text{-}\mathsf{mu\text{-}ror\text{-}cca2}}(\mathcal{A})$ :
    $b \xleftarrow{\$} \{0,1\}$
    $\mathsf{F_{LIST}}, \mathsf{U_{LIST}}, \mathsf{C_{LIST}} \leftarrow \emptyset$
    for $j \in \{1,\ldots,r\}$ do
        $(sk_j, pk_j) \leftarrow \mathsf{KG_{pke}}$
        $\overrightarrow{pk} \leftarrow \overrightarrow{pk} \cup pk_j$
    $b' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{RoR}_b, \mathcal{O}.\mathsf{Dec}, \mathcal{O}.\mathsf{Corrupt}}(\overrightarrow{pk})$
    return $b' \overset{?}{=} b$

$\mathcal{O}.\mathsf{Corrupt}(pk)$
    if $pk \in \mathsf{U_{LIST}}$ then
        return $\perp$
    $\mathsf{C_{LIST}} \leftarrow \mathsf{C_{LIST}} \cup \{pk\}$
    return $sk$

$\mathcal{O}.\mathsf{RoR}_b(pk, m)$ :
    if $pk \in \mathsf{C_{LIST}}$ then
        return $c \leftarrow \mathsf{Enc}_{pk}(m)$
    $m_1 \leftarrow m$
    $m_0 \xleftarrow{\$} \mathcal{M}_{pk}$
    $c \leftarrow \mathsf{Enc}_{pk}(m_b)$
    $\mathsf{U_{LIST}} \leftarrow \mathsf{U_{LIST}} \cup \{pk\}$
    $\mathsf{F_{LIST}} \leftarrow \mathsf{F_{LIST}} \cup \{c\}$
    return $c$

$\mathcal{O}.\mathsf{Dec}(pk, c)$
    if $c \in \mathsf{F_{LIST}}$ then
        return $\perp$
    $m \leftarrow \mathsf{Dec}_{sk}(c)$
    return $m$
```

Figure 13: The experiment defining (t, n, c)-$\mathsf{mu\text{-}ror\text{-}cca2}$ security for a public-key encryption scheme $\mathsf{PKE} = (\mathsf{KG_{pke}}, \mathsf{Enc}, \mathsf{Dec})$.

The following result describes the relationship between the usual ror-cca2 notion and the mu-ror-cca2 notion.

Theorem 3. *Let* $\mathsf{PKE} = (\mathsf{KG}_{\mathsf{pke}}, \mathsf{Enc}, \mathsf{Dec})$ *be a public-key encryption scheme. Let* $\mathcal{A}$ *be an adversary against* PKE *under adaptive chosen ciphertext attack and corruption attack in the multi user setting, running with* n *users. Suppose* c *is the maximal number of corrupted users,* t *is the maximal number of challenge ciphertexts the adversary can receive. Then there exists an adversary* $\mathcal{B}$ *against* PKE *under adaptive chosen ciphertext attack in the single user setting, such that*

$$\mathbf{Adv}_{\mathsf{PKE}}^{(t,\,n,\,c)\text{-}\mathsf{mu\text{-}ror\text{-}cca2}}(\mathcal{A}) \leq nt\mathbf{Adv}_{\mathsf{PKE}}^{\mathsf{ror\text{-}cca2}}(\mathcal{B}).$$

A.1 Proof of Theorem 3

The proof is in three parts. The first part is a straight-forward hybrid argument, reducing the number of key pairs to one. The second part shows that when we only consider a single key pair, we can disregard the corruption oracle. And finally, the third part is again a straight-forward hybrid argument reducing the number of challenge encryptions to one. This completes the argument, since $(1, 1, 0)$-mu-ror-cca2 is the same as ror-cca2.

Part 1. We first prove that there exists an adversary $\mathcal{A}_1$ against PKE under adaptive chosen ciphertext attack and corruption attack in the single user setting, such that

$$\mathbf{Adv}_{\mathsf{PKE}}^{(t,\,n,\,c)\text{-}\mathsf{mu\text{-}ror\text{-}cca2}}(\mathcal{A}) \leq n\mathbf{Adv}_{\mathsf{PKE}}^{(t,\,1,\,1)\text{-}\mathsf{mu\text{-}ror\text{-}cca2}}(\mathcal{A}_1). \tag{15}$$

Proof. We use a hybrid argument with $n + 1$ hybrid games, counting from 0. For corrupted users, $\mathcal{O}.\mathsf{RoR}$ will always encrypt real messages. In the ith hybrid game the challenge oracle $\mathcal{O}.\mathsf{RoR}$ will encrypt real messages for the ith first public keys. For the remaining $n - i$ public keys, the challenge oracle will encrypt random messages.

An adversary's advantage is bounded by n times the average distinguishing advantage for the same adversary against two consecutive hybrid games.

Now we use a (t, n, c)-adversary $\mathcal{A}$ to create a $(t, 1, 1)$-adversary $\mathcal{A}_1$ against the scheme, and prove that this new adversary has the same advantage as the average distinguishing advantage for $\mathcal{A}$ against two consecutive hybrid games. The adversary $\mathcal{A}_1$ is given in Fig. 14.

If $\mathcal{A}_1$'s challenge oracle always encrypts the real message, then $\mathcal{A}_1$ perfectly simulates the ith hybrid game for $\mathcal{A}$. Likewise, if $\mathcal{A}_1$'s challenge oracle always encrypts random messages, then $\mathcal{A}_1$ perfectly simulates the $i-1$th hybrid game for $\mathcal{A}$.

When $\mathcal{A}_1$ has chosen i, and thereby the two hybrid games to potentially simulate, its advantage is exactly equal to the distinguishing advantage of $\mathcal{A}$ for the two consecutive hybrid games chosen. Since $\mathcal{A}_1$ chooses i uniformly at random, the advantage of $\mathcal{A}_1$ is exactly equal to the average distinguishing advantage of $\mathcal{A}$ against two consecutive hybrid games.

The claim follows. □

Part 2. We now prove that there exists an $(t, 1, 0)$-mu-ror-cca2 adversary $\mathcal{A}_2$ against PKE such that

$$\mathbf{Adv}_{\mathsf{PKE}}^{(t,1,1)\text{-}\mathsf{mu\text{-}ror\text{-}cca2}}(\mathcal{A}_1) = \mathbf{Adv}_{\mathsf{PKE}}^{(t,1,0)\text{-}\mathsf{mu\text{-}ror\text{-}cca2}}(\mathcal{A}_2). \tag{16}$$

Proof. We first note that if $\mathcal{A}_1$ calls its corruption oracle on its single public key, it has no way to get any information about b, so its advantage is 0.

The adversary $\mathcal{A}_2$ runs $\mathcal{A}_1$. It forwards any $\mathcal{O}.\mathsf{RoR}$ and $\mathcal{O}.\mathsf{Dec}$ queries from $\mathcal{A}_1$ to its own oracles. If $\mathcal{A}_1$ queries its corruption oracle, $\mathcal{A}_2$ stops, flips a fair coin b' and outputs b'.

If $\mathcal{A}_1$ does not query its corruption oracle, $\mathcal{A}_2$ proceeds exactly as $\mathcal{A}_1$ and wins with exactly the same probability. Furthermore, if $\mathcal{A}_1$ does query its corruption oracle, $\mathcal{A}_2$ does not proceed exactly as $\mathcal{A}_1$, but it wins with exactly the same probability.

Let E be the event that $\mathcal{A}_1$ wins, E' the event that $\mathcal{A}_2$ wins, and let F be the event that $\mathcal{A}_1$ queries its corruption oracle, while F' is the probability that $\mathcal{A}_2$ flips a fair coin to determine its result. Note that $\mathbf{Pr}[F] = \mathbf{Pr}[F']$ by definition, and $\mathbf{Pr}[E|F] = \mathbf{Pr}[E'|F']$ and

```
Reduction $\mathcal{A}_1$.
    $i \xleftarrow{\$} \{1, 2, \ldots, n\}$
    $\mathsf{F_{LIST}}, \mathsf{U_{LIST}}, \mathsf{C_{LIST}} \leftarrow \emptyset$
    receive $pk_i$
    for $j \in \{1, \ldots, n\} \setminus \{i\}$ do
        $(sk_j, pk_j) \leftarrow \mathsf{KG_{pke}}$
    $\overrightarrow{pk} \leftarrow (pk_1, pk_2, \ldots, pk_n)$
    $b' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{RoR}_b, \mathcal{O}.\mathsf{Dec}, \mathcal{O}.\mathsf{Corrupt}}(\overrightarrow{pk})$
    return $b'$

$\mathcal{O}.\mathsf{Corrupt}(pk_j)$
    if $pk_j \in \mathsf{U_{LIST}}$ then
        return $\perp$
    if $pk_j = pk_i$ then
        $\underline{sk \leftarrow \mathcal{O}.\mathsf{Corrupt}(pk_i)}$
    else
        $sk \leftarrow sk_j$
    $\mathsf{C_{LIST}} \leftarrow \mathsf{C_{LIST}} \cup \{pk_j\}$
    return $sk$
```

```
$\mathcal{O}.\mathsf{RoR}_b(pk_j, m)$ :
    if $pk_j \in \mathsf{C_{LIST}}$ then
        return $c \leftarrow \mathsf{Enc}_{pk_j}(m)$
    $m^{\$} \xleftarrow{\$} \mathcal{M}_{pk}$
    if $pk_j = pk_i$ then
        $\underline{c \leftarrow \mathcal{O}.\mathsf{RoR}(pk_i, m)}$
    if $j < i$ then
        $c \leftarrow \mathsf{Enc}_{pk_j}(m)$
    if $j > i$ then
        $c \leftarrow \mathsf{Enc}_{pk_j}(m^{\$})$
    $\mathsf{U_{LIST}} \leftarrow \mathsf{U_{LIST}} \cup \{pk_j\}$
    $\mathsf{F_{LIST}} \leftarrow \mathsf{F_{LIST}} \cup \{c\}$
    return $c$

$\mathcal{O}.\mathsf{Dec}(pk_j, c)$
    if $c \in \mathsf{F_{LIST}}$ then
        return $\perp$
    if $pk_j = pk_i$ then
        $\underline{m \leftarrow \mathcal{O}.\mathsf{Dec}(pk_i, c)}$
    if $j \neq i$ then
        $m \leftarrow \mathsf{Dec}_{sk_j}(c)$
    return $m$
```

Figure 14: Reduction $\mathcal{A}_1$ playing $\mathbf{Exp}_{\mathsf{PKE}}^{(t,1,1)\text{-}\mathsf{mu\text{-}ror\text{-}cca2}}(\mathcal{A}_1)$, used in proof of (15).

$\mathbf{Pr}[E|\neg F] = \mathbf{Pr}[E'|\neg F]$ by the above paragraphs. Then we have

$$\begin{aligned}\mathbf{Pr}[E] &= \mathbf{Pr}[E|F]\mathbf{Pr}[F] + \mathbf{Pr}[E|\neg F]\mathbf{Pr}[\neg F]\\ &= \mathbf{Pr}[E'|F']\mathbf{Pr}[F'] + \mathbf{Pr}[E'|\neg F']\mathbf{Pr}[\neg F']\\ &= \mathbf{Pr}[E'].\end{aligned}$$

The claim follows. □

Part 3. We now prove, again using a standard hybrid argument, that there exists an $(1,1,0)$-mu-ror-cca2 adversary $\mathcal{A}_3$ such that

$$\mathbf{Adv}_{\mathsf{PKE}}^{(t,1,0)\text{-mu-ror-cca2}}(\mathcal{A}_2) \leq t\mathbf{Adv}_{\mathsf{PKE}}^{(1,1,0)\text{-mu-ror-cca2}}(\mathcal{A}_3). \qquad (17)$$

Proof. Again, we have a hybrid argument with $t+1$ hybrid games, counting from 0. In the ith hybrid game, the challenge oracle $\mathcal{O}.\mathsf{RoR}$ will encrypt the real message for the first i queries, and then encrypt random messages for the remaining $t-i$ queries.

An adversary's advantage is bounded by t times the average distinguishing advantage for the same adversary against two consecutive hybrid games.

Now we use a $(t,1,0)$-mu-ror-cca2 adversary $\mathcal{A}_2$ to create a $(1,1,0)$-mu-ror-cca2 adversary $\mathcal{A}_3$ against the scheme, and prove that this new adversary has the same advantage as the average distinguishing advantage for $\mathcal{A}_2$ against two consecutive hybrid games. The adversary $\mathcal{A}_3$ is given in Fig. 15

```
Reduction $\mathcal{A}_3$.
    receive $pk$
    $\mathsf{F_{LIST}} \leftarrow \emptyset$
    $b' \leftarrow \mathcal{A}_2^{\mathcal{O}.\mathsf{RoR}_b,\mathcal{O}.\mathsf{Dec},\mathcal{O}.\mathsf{Corrupt}}(pk)$
    return $b'$

$\mathcal{O}.\mathsf{Dec}(pk, c)$
    if $c \in \mathsf{F_{LIST}}$ then
        return $\perp$
    $\underline{m \leftarrow \mathcal{O}.\mathsf{Dec}(c)}$
    return $m$

$\mathcal{O}.\mathsf{RoR}_b(pk, m_j)$ :
    $m^\$ \xleftarrow{\$} \mathcal{M}_{pk}$
    if $j = i$ then
        $\underline{c \leftarrow \mathcal{O}.\mathsf{RoR}(m_i)}$
    if $j < i$ then
        $c \leftarrow \mathsf{Enc}_{pk}(m_j)$
    if $j > i$ then
        $c \leftarrow \mathsf{Enc}_{pk}(m^\$)$
    $\mathsf{F_{LIST}} \leftarrow \mathsf{F_{LIST}} \cup \{c\}$
    return $c$
```

Figure 15: The reduction $\mathcal{A}_3$ from $(t,1,0)$-mu-ror-cca2 to $(1,1,0)$-mu-ror-cca2 used to prove (17).

If $\mathcal{A}_3$'s challenge oracle encrypts the real message, then $\mathcal{A}_3$ perfectly simulates the ith hybrid game for $\mathcal{A}_2$. Likewise, if $\mathcal{A}_3$'s challenge oracle encrypts a random message, then $\mathcal{A}_3$ perfectly simulates the $i-1$th hybrid game for $\mathcal{A}_2$.

When $\mathcal{A}_3$ has chosen i, and thereby two hybrid games to potentially simulate, its advantage is exactly equal to the distinguishing advantage of $\mathcal{A}_2$ for the two consecutive hybrid games chosen. Since $\mathcal{A}_3$ chooses i uniformly at random, the advantage of $\mathcal{A}_3$ is exactly equal to the average distinguishing advantage of $\mathcal{A}_2$ against two consecutive hybrid games.

The claim follows. $\square$

Now we observe that a $(1, 1, 0)$-mu-ror-cca2 adversary against the scheme is simply an ror-cca2 adversary, and the theorem follows from equations (15)–(17).

Paper II

Cloud-assisted Asynchronous Key Transport with Post-Quantum Security

Gareth T. Davies, Herman Galteland, Kristian Gjøsteen and Yao Jiang

Published in the 25th Australasian Conference on Information Security and Privacy, ACISP2020.

Cloud-assisted Asynchronous Key Transport with Post-Quantum Security

Gareth T. Davies[1], Herman Galteland[2], Kristian Gjøsteen[2], and Yao Jiang[2]
[1]Bergische Universität Wuppertal, Germany.
davies@uni-wuppertal.de
[2]Norwegian University of Science and Technology, NTNU, Norway.
{herman.galteland,kristian.gjosteen,yao.jiang}@ntnu.no

Abstract

In cloud-based outsourced storage systems, many users wish to securely store their files for later retrieval, and additionally to share them with other users. These retrieving users may not be online at the point of the file upload, and in fact they may never come online at all. In this asynchoronous environment, key transport appears to be at odds with any demands for forward secrecy. Recently, Boyd et al. (ISC 2018) presented a protocol that allows an initiator to use a modified key encapsulation primitive, denoted a blinded KEM (BKEM), to transport a file encryption key to potentially many recipients via the (untrusted) storage server, in a way that gives some guarantees of forward secrecy. Until now all known constructions of BKEMs are built using RSA and DDH, and thus are only secure in the classical setting.

We further the understanding of secure key transport protocols in two aspects. First, we show how to generically build blinded KEMs from homomorphic encryption schemes with certain properties. Second, we construct the first post-quantum secure blinded KEMs, and the security of our constructions are based on hard lattice problems.

Keywords: Lattice-based cryptography, NTRU, Group Key Exchange, Blinded Key Encapsulation, Forward Secrecy, Cloud Storage, Post-quantum cryptography

1 Introduction

Consider the following scenario: a user of a cloud storage service wishes to encypt and share a file with a number of recipients, who may come online to retrieve the file at some future time. In modern cloud storage environments, access control for files is normally done via the storage provider's interface, and the user is usually tasked with performing any encryption and managing the resulting keys. However the users do not trust the server, and in particular may be concerned that key compromise may occur to any of the involved parties at some point in the future – they thus desire some forward secrecy guarantees. A number of approaches can be taken for transporting a (randomly chosen) file encryption key from the initiator to the recipients. The first option is public-key encryption – simply encrypting under each recipient's public key. This approach does not provide any forward secrecy, however if the initiator were to use puncturable encryption then this would provide a (currently inefficient) solution for achieving forward secrecy. The users could also perform a (necessarily interactive) group key exchange protocol, however this requires all recipients to be online: a disqualifying criterion for many usage scenarios. The challenge of providing efficient key transport that allows asynchronous fetching by the recipients and simultaneously gives some forward secrecy guarantees appears to invoke trade-offs.

Recent work by Boyd et al. [10] (hereafter BDGJ) provided a solution that utilized the high availability of the storage provider. The initiator essentially performs key encapsulation, using an (public) encapsulation key belonging to the server, and sends an encapsulated value (out-of-band) to each recipient. Then, each recipient blinds this value in such a way that when it asks the server to decapsulate, the server does not learn anything about the underlying file encryption key, and the homomorphic properties of the scheme enable successful unblinding by the recipient. This encapsulation-and-blinding procedure was named by the authors as a *blinded KEM* (BKEM), and the complete protocol built from this was called offline assisted group key exchange (OAGKE). Forward secrecy is achieved if the recipients delete their ephemeral values after recovering the file encryption key, and if the server deletes its decapsulation key after all recipients have been online and recovered the file.

A conceptual overview of the construction, which can achieve all these security properties, is described in Figure 1, and we refer to BDGJ [10] for full details. In the protocol, the server runs the key generation algorithm KG and the decapsulation algorithm Decap to help the initiator share file encryption key k. The blinding algorithm Blind, executed by the responder, should prohibit the server from learning any information about the file encryption key. After the server has decapsulated a blinded encapsulation, the responder can use the unblinding algorithm Unblind to retrieve the file encryption key.

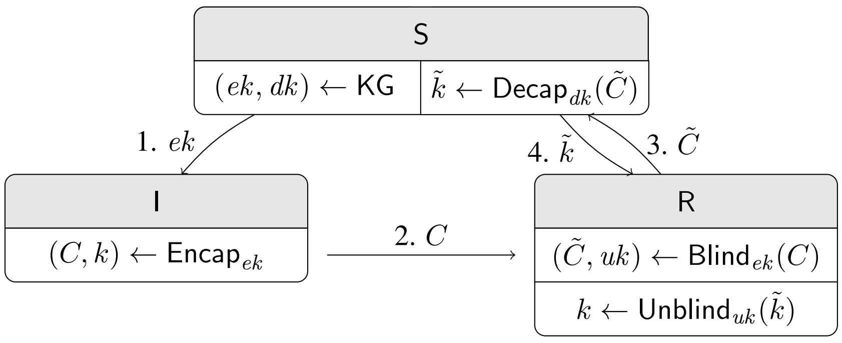
S
$(ek, dk) \leftarrow \mathsf{KG}$
$\tilde{k} \leftarrow \mathsf{Decap}_{dk}(\tilde{C})$
1. ek
4. $\tilde{k}$
3. $\tilde{C}$
I
$(C, k) \leftarrow \mathsf{Encap}_{ek}$
2. C
R
$(\tilde{C}, uk) \leftarrow \mathsf{Blind}_{ek}(C)$
$k \leftarrow \mathsf{Unblind}_{uk}(\tilde{k})$

Figure 1: A simplified overview of an OAGKE protocol [10] between an initiator I, server S and potentially many recipients R (one is given here for ease of exposition), built using a BKEM. File encryption key k is used by I to encrypt one or more files. The numbered arrows indicate the order in which operations occur.

While the approach appears promising, their two BKEM constructions built from DDH and RSA, are somewhat ad hoc, and further do not resist attacks in the presence of quantum computers. In this work we focus on one of the components of the OAGKE protocol, namely the BKEM scheme. Our wish is to achieve a post-quantum secure OAGKE protocol, where we need the individual components – a blinded KEM (parameterized by a homomorphic encryption scheme), a collision resistant hash function, a digital signature scheme, and a key derivation function – to all be post-quantum secure. Achieving post-quantum security of all components except for the BKEM has been covered extensively in prior work, and thus we focus on finding post-quantum constructions of BKEMs.

Much work has been done in the past on constructing regular key encapsulation mechanisms (KEMs) [1,18,19,21,30,32] that are post-quantum secure [8,14,31,35,37] (the ongoing NIST standardization effort [40] specifically asks for KEMs), however BKEMs do not generalize KEMs, since decapsulation operates on blinded ciphertexts.

Providing post-quantum-secure BKEMs invokes a number of technical challenges. The Blind algorithm must modify the file encryption key by incorporating some randomness r, in such a way that after decapsulation (by the server) the recipient can strip off r to recover the file encryption key. In the DDH setting this is straightforward since the recipient can simply exponentiate the encapsulation, and apply the inverse on the received value from the server (the RSA setting is similarly straightforward), and, importantly, the encapsulation (with the underlying file encryption key) *and* multiple blinded samples (each with a value that is derived from the file encryption key) will all look like random group elements. In the security game for BKEMs (as provided by BDGJ), the adversary receives: an encapsulation of a 'real' key, a number of blinded versions of this encapsulation (blinded encapsulations), a number of blinded versions of the 'real' key (blinded keys), and either this 'real' key or a random key, and must decide which it has been given. If the blinded key samples (the $\tilde{k}$s) leak information about the file encryption key then the adversary's task in this game becomes much easier. For example, if the blinding algorithm alters the file encryption key such that the blinded keys are located close to it then exhaustive search becomes possible. We overcome this hurdle by using a large blinding value to hide the file encryption key. Similarly the blinded encapsulation samples (the $\tilde{C}$s) can leak information about the blinding value used to hide the file encryption key, which can be used to recover the file encryption key. For example, if the blinded encapsulation is a linear combination of the original encapsulation, the blinding value, and some small error then the distance between the blinded encapsulation and the original encapsulation could reveal the blinding value, or a small interval containing it, and therefore the file encryption key. By making sure blinded encapsulations look fresh then all blinded encapsulation samples and the encapsulation looks independent of each other. We use these techniques to provide secure BKEMs built from (a variant of) NTRU [29,41] and ideas from Gentry's FHE scheme [24].

The second shortfall of the work of BDGJ lies in the non-generic nature of their constructions. The two provided schemes appear to have similar properties, yet do not immediately indicate how any further BKEM schemes could be constructed. We show how to generically build BKEMs from homomorphic encryption schemes with minimal properties. This allows us to more precisely cast the desirable properties of schemes used to build BKEMs, generalizing the way that the responder alters the content of an encapsulation (ciphertext) by adding an encrypted random value. Essentially, the resulting blinded ciphertext is an encryption of the sum of a file encryption key and the random value. The server can decrypt the blinded ciphertext to retrieve the blinded key, and then the responder can unblind by removing (subtracting) the random value.

1.1 Related work

Boyd et al. [10] formalized OAGKE and BKEMs, and provided BKEM constructions based on Diffie-Hellman and RSA. To our knowledge these are the only BKEM constructions in the literature.

Recent works on secure messaging have shown how to perform secure key transport in the presence of pre-keys of the recipients [17,38,42]: we wish to avoid this assumption in our system architecture. Puncturable encryption has developed rapidly [6,22,27,28], however current constructions are still impractical or unsuitable for the cloud-based key transport scenario that we consider.

Gentry introduced the first fully homomorphic encryption (FHE) scheme, based on lattice problems, and gave a generic framework [24]. Soon after, several FHE schemes followed this framework [11, 16, 23, 26]: all of these schemes rely on the learning with errors (LWE) problem. Two FHE schemes based their security on an overstretched variant of the NTRU problem [9,33], however, subfield lattice attacks against this variant was subsequently found [2,15], and consequently these schemes are no longer secure. As a side note, our NTRU based BKEM construction relies on the hardness of the LWE problem.

To make a BKEM from existing post-quantum secure KEM schemes we need, for each individual scheme, a method for altering the encapsulations in a predictable way. Most of the post-quantum secure KEM schemes submitted to NIST are built from a PKE scheme, where we can use our techniques to make a BKEM if the PKE scheme supports one homomorphic operation. FrodoKEM is the only submission that advertises its additive homomorphic properties of its FrodoPKE scheme [3]. Other submissions based on lattices [34], LWE [4, 5, 20], or NTRU [7, 13] are potential candidates for a BKEM construction. Note that the NTRU submission of Chen et al. [13] does not use the Gaussian distribution to sample their polynomials, and NTRU Prime of Bernstein et al. [7] uses a large Galois group to construct their polynomial field, instead of a cyclotomic polynomial. Furthermore, the NTRU contruction of Stehlé and Steinfeld [41] chooses the distribution of the secret keys such that the public key looks uniformly random and they provide a security proof which relies on this.

1.2 Our contribution

Our aim in this work is to further the understanding of the use of blinding in cryptography attaining post-quantum security. In particular, we focus on blinded KEMs and their possible instantiations, in order to deliver secure key transport protocols in cloud storage environments. Specifically, we provide:

- a generic homomorphic-based BKEM construction, and show that it meets the expected indistinguishability-based security property for BKEMs, under feasible requirements.
- two instantiations of our homomorphic-based BKEM, built from primitives with post-quantum security. The proof chain is as follows.

$$\text{Hard problems} \xrightarrow[\text{or Lyubashevsky et al. [36]}]{\text{Quantum, Gentry [24]}} \mathsf{IND\text{-}CPA\ HE} \xrightarrow{\text{This work}} \begin{array}{c}\mathsf{IND\text{-}secure}\\ \mathsf{HE\text{-}BKEM}\end{array}$$

As long as the underlying schemes HE (which rely on hard lattice problems) are post-quantum secure, then our $\mathsf{HE\text{-}BKEM}$ schemes are post-quantum secure.

1.3 Organization

In Section 2 we provide the necessary background of ideal lattices and the discrete Gaussian Distribution. In Section 3 we formally define BKEMs and their security. In Section 4 we construct a generic homomorphic BKEM schemes and analyze its security requirements. In Section 5 we provide two homomorphic-based BKEM constructions and prove that they are secure.

L	a lattice
$\mathbf{B}$	a basis of ideal lattice I
$\mathcal{P}(\mathbf{B})$	the half-open parallelepiped associated to the basis $\mathbf{B}$, where $\mathcal{P}(\mathbf{B}) = \{\sum_{i=1}^{n} x_i \mathbf{b}_i \mid x_i \in [-1/2, 1/2)\,,$ $\mathbf{b}_i \in \mathbf{B}\}$
$R = \mathbb{Z}[x]/(f(x))$	a polynomial ring, where $f(x)$ is a monic polynomial of degree n
$\|\mathbf{v}\|$	Euclidean norm of a vector $\mathbf{v}$
$\|\mathbf{B}\|$	norm of a basis $\mathbf{B}$, where $\|\mathbf{B}\| = \max_{1\leq i\leq n}(\|\mathbf{b}_i\|)$
L^*	dual lattice of L
$\gamma_\times(R)$	multiplicative expansion factor
$\mathbf{r} \mod \mathbf{B}$	the distinguished representative of the coset $\mathbf{r} + I$
$R \mod \mathbf{B}$	set of all distinguished representatives in R
$\mathcal{B}_{\mathbf{c}}(r)$	closed Euclidean ball centered at $\mathbf{c}$ with radius r
$\mathcal{B}(r)$	closed Euclidean ball centered at $\mathbf{0}$ with radius r
$\lambda_i(L)$	the ith successive minimum
$\Delta(D_1, D_2)$	statistical distance between two discrete distributions D_1 and D_2
$D_{L,s,\mathbf{c}}$	discrete Gaussian distribution over L centered at $\mathbf{c}$ with standard deviation s
$\eta_\epsilon(L)$	smoothing parameter for lattice L

Figure 2: Summary of notation used in this paper.

2 Preliminaries

This section introduces terminology and results from [24, 25, 39], and provides an introduction to our notation and building blocks for constructing post-quantum secure homomorphic encryption schemes. In Fig. 2 we given an overview of the notation used in the paper. Towards the end of this section we detail two specific constructions of post-quantum secure homomorphic encryption schemes [24, 41].

2.1 Notation

Given n linearly independent vectors $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$, $\mathbf{b}_i \in \mathbb{R}^m$, the m-dimensional *lattice* L generated by the vectors is $L = \{\sum_{i=1}^{n} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z}\}$. If $n = m$ then L is a *full-rank n-dimensional lattice*, we always use full-rank lattices in this paper.

Suppose $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n\}$ is a basis of I, let $\mathcal{P}(\mathbf{B}) = \{\sum_{i=1}^{n} x_i \mathbf{b}_i \mid x_i \in [-1/2, 1/2), \mathbf{b}_i \in \mathbf{B}\}$ be the half-open parallelepiped associated to the basis $\mathbf{B}$. Let $R = \mathbb{Z}[x]/(f(x))$ be a polynomial ring, where $f(x)$ is a monic polynomial of degree n. Any ideal $I \subseteq R$ yields a corresponding integer sublattice called *ideal lattice* of the polynomial ring. For convenience, we identify all ideals of R with its ideal lattice. Let $\|\mathbf{v}\|$ be the Euclidean norm of a vector $\mathbf{v}$. Define the *norm of a basis* $\mathbf{B}$ to be the Euclidean norm of its longest column vector, that is, $\|\mathbf{B}\| = \max_{1 \leq i \leq n}(\|\mathbf{b}_i\|)$.

For a full-rank n-dimensional lattice L, let $L^* = \{\mathbf{x} \in \mathbb{R}^n \mid \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}, \forall \mathbf{y} \in L\}$ denote its *dual lattice*. If $\mathbf{B}$ is a basis for the full-rank lattice L, then $(\mathbf{B}^{-1})^T$ is a basis of L^*. Let $\gamma_\times(R) = \max_{\mathbf{x},\mathbf{y} \in R} \frac{\|\mathbf{x} \cdot \mathbf{y}\|}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|}$ be the *multiplicative expansion factor*. For $\mathbf{r} \in R$, define $\mathbf{r} \mod \mathbf{B}$ to be the unique vector $\mathbf{r}' \in \mathcal{P}(\mathbf{B})$ such that $\mathbf{r} - \mathbf{r}' \in I$. We call $\mathbf{r} \mod \mathbf{B}$ to be the *distinguished representative* of the coset $\mathbf{r} + I$. Denote $R \mod \mathbf{B} = \{\mathbf{r} \mod \mathbf{B} \mid \mathbf{r} \in R\}$ to be the set of all distinguished representatives in R, this set can be chosen to be the same as the half-open parallelepiped $\mathcal{P}(\mathbf{B})$ associated to the basis $\mathbf{B}$. For convinience we treat $R \mod \mathbf{B}$ and $\mathcal{P}(\mathbf{B})$ as the same set. Let $\mathcal{B}_\mathbf{c}(r)$ denote the closed Euclidean ball centered at $\mathbf{c}$ with radius r, for $\mathbf{c} = \mathbf{0}$ we write $\mathcal{B}(r)$. For any n-dimensional lattice L and $i = 1, \ldots, n$, let the *ith successive minimum* $\lambda_i(L)$ be the smallest radius r such that $\mathcal{B}(r)$ contains i linearly independent lattice vectors.

The *statistical distance* between two discrete distributions D_1 and D_2 over a set S is $\Delta(D_1, D_2) = \frac{1}{2} \sum_{s \in S} |\mathbf{Pr}[D_1 = s] - \mathbf{Pr}[D_2 = s]|$.

2.2 Discrete Gaussian Distributions over Lattices

Definition 1 (Discrete Gaussian Distribution)**.** Let $L \subseteq \mathbb{R}^n$ be a lattice, $s \in \mathbb{R}^+$, $\mathbf{c} \in \mathbb{R}^n$. For all $\mathbf{x} \in L$, let $\rho_{s,\mathbf{c}}(\mathbf{x}) = \exp(-\pi \|\mathbf{x} - \mathbf{c}\|^2 / s^2)$. For a set S let $\rho_{s,\mathbf{c}}(S) = \sum_{\mathbf{x} \in S} \exp(-\pi \|\mathbf{x} - \mathbf{c}\|^2 / s^2)$. Define the *discrete Gaussian distribution* over L centered at $\mathbf{c}$ with standard deviation s to be the probability distribution

$$D_{L,s,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{s,\mathbf{c}}(\mathbf{x})}{\rho_{s,\mathbf{c}}(L)}.$$

If the standard deviation of a discrete Gaussian distribution is larger than the smoothing parameter, defined below,then there are known, useful, results of discrete Gaussian distributions that we will use in this paper.

Definition 2 (Smoothing parameter)**.** For lattice L and real value $\epsilon > 0$, let the *smoothing parameter* $\eta_\epsilon(L)$ denote the smallest s: $\rho_{1/s}(L^* \setminus \{\mathbf{0}\}) \leq \epsilon$. We say that "$s$ exceeds the smoothing parameter" if $s \geq \eta_\epsilon(L)$ for negligible ϵ.

Below we show that the discrete Gaussian distribution is spherical if its standard deviation is larger than the smoothing parameter.

Lemma 1 (Micciancio and Regev [39])**.** *Let L be any full-rank n-dimensional lattice. For any $\mathbf{c} \in \mathbb{R}^n$, real $\epsilon \in (0,1)$, and $s \geq \eta_\epsilon(L)$, we have*

$$\mathbf{Pr}[\|\mathbf{x} - \mathbf{c}\| > s \cdot \sqrt{n} \mid \mathbf{x} \leftarrow D_{L,s,\mathbf{c}}] \leq \frac{1+\epsilon}{1-\epsilon} \cdot 2^{-n}.$$

For a discrete Gaussian distribution over L centered at $\mathbf{0}$, with standard deviation s, $D_{L,s,\mathbf{0}}$ we let the translated discrete Gaussian distribution over L centered at any $\mathbf{c}$, with standard deviation s, be $D_{L,s,\mathbf{c}}$. Below we show that the statistical distance between the original discrete Gaussian distribution and its translated discrete Gaussian distribution is negligible when $\|\mathbf{c}\|$ is small.

Lemma 2 (Brakerski and Vaikuntanathan [12])**.** *Let L be any full-rank n-dimensional lattice. For any $s \geq \eta_\epsilon(L)$, and any $\mathbf{c} \in \mathbb{R}^n$, we have then the statistical distance between $D_{L,s,\mathbf{0}}$ and $D_{L,s,\mathbf{c}}$ is at most $\|\mathbf{c}\|/s$.*

2.3 Gentry's homomorphic encryption scheme

Let $\mathsf{GHE} = (\mathsf{KG_{GHE}}, \mathsf{Enc_{GHE}}, \mathsf{Dec_{GHE}}, \mathsf{Add_{GHE}})$ be an (additively) Homomorphic encryption scheme derived from ideal lattices, with algorithms as defined in Figure 3. The scheme is similar to Gentry's somewhat-homomorphic scheme [24]. The parameters of the GHE scheme are chosen as follows:

- Polynomial ring $R = \mathbb{Z}[x]/(f(x))$,
- Basis $\mathbf{B}_I$ of the ideal $I \subseteq R$,
- $\mathsf{IdealGen}$ takes $(R, \mathbf{B}_I)$ as input and outputs public and secret bases $\mathbf{B}_J^{pk}$ and $\mathbf{B}_J^{sk}$ of some ideal J, where I and J are relatively prime,
- Samp takes $(\mathbf{B}_I, \mathbf{x} \in R, s)$ as input and outputs a sample from the coset $\mathbf{x} + I$ according to a discrete Gaussian distribution with standard deviation s. In our construction we use the following two distributions.
  - $\mathsf{Samp}_1(\mathbf{B}_I, \mathbf{x}, s) = \mathbf{x} + D_{I,s,-\mathbf{x}}$,
  - $\mathsf{Samp}_2(\mathbf{B}_I, \mathbf{x}, s) = \mathbf{x} + D_{I,s,\mathbf{0}}$.
- Plaintext space $\mathcal{P} = R \mod \mathbf{B}_I$ is the set of distinguished representatives of cosets of I with respect to the basis $\mathbf{B}_I$.

```
$\mathsf{KG}_{\mathsf{GHE}}(R, \mathbf{B}_I)$ :
    $(\mathbf{B}_J^{pk}, \mathbf{B}_J^{sk}) \xleftarrow{\$} \mathsf{IdealGen}(R, \mathbf{B}_I)$
    $\mathsf{pk} = (R, \mathbf{B}_I, \mathbf{B}_J^{pk}, \mathsf{Samp}), \mathsf{sk} = \mathbf{B}_J^{sk}$
    return $\mathsf{pk}, \mathsf{sk}$
$\mathsf{Dec}_{\mathsf{GHE}}(\mathsf{sk}, \psi)$ :
    $\pi \leftarrow (\psi \mod \mathbf{B}_J^{sk}) \mod \mathbf{B}_I$
    return $\pi$

$\mathsf{Enc}_{\mathsf{GHE}}(\mathsf{pk}, s, \pi \in \mathcal{P})$ :
    $\psi' \leftarrow \mathsf{Samp}(\mathbf{B}_I, \pi, s)$
    $\psi \leftarrow \psi' \mod \mathbf{B}_J^{pk}$
    return $\psi$
$\mathsf{Add}_{\mathsf{GHE}}(\mathsf{pk}, \psi_1, \psi_2)$ :
    $\psi \leftarrow \psi_1 + \psi_2 \mod \mathbf{B}_J^{pk}$
    return $\psi$
```

Figure 3: The algorithms of the GHE homomorphic encryption scheme, which is similar to Gentry's somewhat homomorphic encryption scheme [24].

Correctness. Let X_{Enc} denote the image of Samp and X_{Dec} denote $R \mod \mathbf{B}_J^{sk} = \mathcal{P}(\mathbf{B}_J^{sk})$. Notice that all ciphertexts are in $X_{\mathsf{Enc}} + J$, because X_{Dec} is the set of distinguished representatives with respect to $\mathbf{B}_J^{sk}$. The correctness requirement of this encryption scheme is $X_{\mathsf{Enc}} \subseteq X_{\mathsf{Dec}}$. Furthermore, for the addition algorithm $\mathsf{Add}_{\mathsf{GHE}}$ to output valid ciphertexts we require that $X_{\mathsf{Enc}} + X_{\mathsf{Enc}} \subseteq X_{\mathsf{Dec}}$.

Let r_{Enc} be the smallest value such that $X_{\mathsf{Enc}} \subseteq \mathcal{B}(r_{\mathsf{Enc}})$ and let r_{Dec} be the largest value such that $X_{\mathsf{Dec}} \supseteq \mathcal{B}(r_{\mathsf{Dec}})$. By the spherical property of discrete Gaussian distribution (Lemma 1) we know that, for Samp_1 as above, X_{Enc} is located inside the ball $\mathcal{B}(s\sqrt{n})$ with high probability and $r_{\mathsf{Enc}} = s\sqrt{n}$. For a general Samp algorithm, which is located in $\mathcal{B}(l_{\mathsf{Samp}})$, we have that $r_{\mathsf{Enc}} \leq (n + \sqrt{n} l_{\mathsf{Samp}}) \|\mathbf{B}_I\|$ [24]. For r_{Dec} we know that $r_{\mathsf{Dec}} = 1/(2 \cdot \left\|((\mathbf{B}_J^{sk})^{-1})^T\right\|)$ [24].

For GHE, if $\sqrt{2} r_{\mathsf{Enc}} \leq r_{\mathsf{Dec}}$, both ciphertexts and the sum of two ciphertexts decrypt to the correct message except for negligible probability.

2.4 The revised NTRU encryption scheme

The NTRU encryption scheme variant by Stehlé and Steinfeld [41], which relies on the LWE problem, has the similar structure as Gentry's homomorphic encryption scheme. We modify the NTRU scheme to use a discrete Gaussian distribution as the noise distribution instead of an elliptic Gaussian. The parameters of the scheme, given in Figure 4, are as follows:

- $R = \mathbb{Z}[x]/(x^n + 1)$, where $n \geq 8$ is a power of 2,
- q is a prime, $5 \leq q \leq Poly(n)$, $R_q = R/q$,
- $p \in R_q^{\times}$, $I = (p)$, the plaintext space $\mathcal{P} = R/p$,
- set the noise distribution to be $D_{\mathbb{Z}^n, s, \mathbf{0}}$.

Correctness Let $\psi' = f\pi + p(fe_1 + ge_2) \in R_q$ and $\psi'' = f\pi + p(fe_1 + ge_2) \in R$ (not modulo q), if $\|\psi''\|_\infty \leq q/2$ then the decryption algorithm will output π (see [41, Lemma 12]). We will perform a single homomorphic addition and want to find a bound on the sum of two ciphertexts. Discrete Gaussian samples are bounded by $s\sqrt{n}$ with high probability (Lemma 1) and the message space parameter p is a polynomial with small coefficients, where we let p_i denote the largest coefficient of p. We have

$$\begin{aligned}\|f(\psi_1 + \psi_2)\|_\infty &= \left\|f(\pi_1 + \pi_2) + p_i(f(e_1 + e_1') + g(e_2 + e_2'))\right\|_\infty \\ &\leq 2(p_i^2(s\sqrt{n})^2 + p_i^2 s\sqrt{n} + p_i s\sqrt{n} + p_i + (s\sqrt{n})^2) \\ &\leq 8p_i^2 s^2 n.\end{aligned}$$

```
$\mathsf{KG}_{\mathsf{NTRU}}(n, q \in \mathbb{Z}, p \in R_q^\times, s > 0)$ :
    while $(f \mod q) \notin R_q^\times$ do
        $f' \leftarrow D_{\mathbb{Z}^n,s,\mathbf{0}}$
        $f = p \cdot f' + 1$
    while $(g \mod q) \notin R_q^\times$ do
        $g \leftarrow D_{\mathbb{Z}^n,s,\mathbf{0}}$
    $h = pg/f \in R_q$
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow (h, f)$
    return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{Enc}_{\mathsf{NTRU}}(\mathsf{pk} = h, s, \pi \in \mathcal{P})$ :
    $e_1, e_2 \leftarrow D_{\mathbb{Z}^n,s,\mathbf{0}}$
    $\psi \leftarrow \pi + pe_1 + he_2 \in R_q$
    return $\psi$
$\mathsf{Dec}_{\mathsf{NTRU}}(\mathsf{sk} = f, \psi)$ :
    $\psi' = f \cdot \psi \in R_q$
    $\pi \leftarrow \psi' \mod p$
    return $\pi$
$\mathsf{Add}_{\mathsf{NTRU}}(\psi_1, \psi_2)$ :
    $\psi \leftarrow \psi_1 + \psi_2 \in R_q$
    return $\psi$
```

Figure 4: The algorithms of the revised NTRU encryption scheme [41].

Standard deviation $s \geq \eta_\epsilon(\mathbb{Z}^n)$ and has to satisfy $\eta_\epsilon(\mathbb{Z}^n) \leq s$ and $8p_i^2 s^2 n < q/2$ for decryption to be correct with high probability. Then both ciphertexts and the sum of two ciphertexts, in the revised NTRU encryption scheme, decrypt to the correct message except for negligible probability.

2.5 Hard lattice problems

The following lattice problems, that are assumed to be hard, are used in the paper.

Definition 3 (Shortest Vector Problem (SVP)). Given a basis $\mathbf{B}$ for a n-dimensional lattice L, output a nonzero vector $\mathbf{v} \in L$ of length at most $\lambda_1(L)$.

Definition 4 (Ideal Shortest Independent Vector Problem (SIVP)). Fix a polynomial ring R and positive real $\gamma \geq 1$. Let $\mathbf{B}_I$ be a basis for an ideal lattice I of R. Given $\mathbf{B}_I$, and parameters, output a basis $\mathbf{B}'_I$ of I with $||\mathbf{B}'_I|| \leq \gamma \cdot \lambda_n(I)$.

Reduce Hard problems to the semantic security of Gentry's encryption scheme. The following two theorems describe Gentry's reduction from worst-case SIVP (believed to be a hard problem) to the semantic security of the encryption scheme GHE, via the ideal independent vector improvement problem (IVIP).

Theorem 1 (Gentry [24, Corollary 14.7.1], reduce IVIP to semantic security)**.** *Suppose that* $s_{\mathsf{IVIP}} < (\sqrt{2}s\epsilon - 4n^2(\max\{\|\mathbf{B}_I\|\})^2)/(n^4\gamma_\times(R)\,\|f\|\,\max\{\|\mathbf{B}_I\|\})$*, where* s *is the Gaussian deviation parameter in the encryption scheme* GHE*. Also suppose that* $s/2$ *exceeds the smoothing parameter of* I*, that* IdealGen *always outputs an ideal* J *with* $s \cdot \sqrt{n} < \lambda_1(J)$*, and that* $[R : I]$ *is prime. Finally, suppose that there is an algorithm* $\mathcal{A}$ *that breaks the semantic security of* GHE *with advantage* ϵ*. Then there is a quantum algorithm that solves* s_{IVIP}-IVIP *for an* $\epsilon/4$ *(up to negligible factors) weight fraction of bases output by* IdealGen.

Theorem 2 (Gentry [24, Theorem 19.2.3 and Corollary 19.2.5], reduce SIVP to IVIP)**.** *Suppose* $d_{\mathsf{SIVP}} = (3 \cdot e)^{1/n} \cdot d_{\mathsf{IVIP}}$*, where* e *is Euler's constant. Suppose that there is an algorithm* $\mathcal{A}$ *that solves* s_{IVIP}-IVIP *for parameter* $s_{\mathsf{IVIP}} > 16 \cdot \gamma_\times(R)^2 \cdot n^5 \cdot \|f\| \cdot g(n)$ *for some* $g(n)$ *that is* $\omega(\sqrt{\log n})$*, whenever the given ideal has* $\det(J) \in [a, b]$*, where* $[a, b] = [d^n_{\mathsf{IVIP}}, 2 \cdot d^n_{\mathsf{IVIP}}]$*. Assume that invertible prime ideals with norms in* $[a, b]$ *are not negligibly sparse. Then, there is an algorithm* $\mathcal{B}$ *that solves worst-case* d_{SIVP}-SIVP.

In summary we have the following informal result, which we will use to prove that our GHE-BKEM (see Section 5.4) is post-quantum secure.

Theorem 3 (Gentry [24])**.** *If there exists an algorithm that breaks the semantic security of* GHE *with parameters chosen as in Theorem 1 and Theorem 2, then there exists a quantum algorithm that solves worst-case* SIVP.

Reduce Hard problems to the semantic security of the revised NTRU **encryption scheme** We define the ring learning with error problem as follows. For $s \in R_q$ and error distribution D over R_q, let $A_{s,D}$ be a distribution that outputs tuples of the form $(a, as + e)$, where a is sampled uniformly at random from R_q and e is sampled from D. The problem is to distinguish between tuples sampled from $A_{s,D}$ and uniformly random ones.

We now introduce the tools necessary to analyze the NTRU construction.

Definition 5 (Ring-LWE)**.** Let $\mathcal{D}$ be a distribution over a family of distributions, each over R_q. The *Ring Learning With Errors Problem* with parameters q, and $\mathcal{D}$ ($\mathsf{R\text{-}LWE}_{q,\mathcal{D}}$) is as follows. Let D be sampled from $\mathcal{D}$ and s be sampled uniformly at random from R_q. Given access to an oracle $\mathcal{O}$

that produces samples in R_q^2, distinguish whether $\mathcal{O}$ outputs samples from the distribution $A_{s,D}$ or $U(R_q^2)$. The distinguishing advantage should be non-negligible.

Lyubashevsky et al. [36] proposed a reduction from SIVP or SVP (both are thought to be hard problems) to R-LWE.

Theorem 4 (Lyubashevsky et al. [36]). *Let $\alpha < \sqrt{\log n/n}$ and $q = 1 \mod 2n$ be a poly(n)-bounded prime such that $\alpha q \geq \omega(\sqrt{\log n})$. Then there is a polynomial-time quantum reduction from $O(\sqrt{n}/\alpha)$-approximate* SIVP *(or* SVP*) on ideal lattices to* R-LWE$_{q,D_s}$ *given only $l(\geq 1)$ samples, where $s = \alpha \cdot (nl/\log(nl))^{1/4}$.*

We consider a different variant of the R-LWE problem, namely R-LWE$^{\times}_{\mathsf{HNF}}$, which is the same as R-LWE$_{q,\mathcal{D}}$ except for the oracle $\mathcal{O}$ that outputs samples from the distribution $A^{\times}_{s,D}$ or $U(R_q^2)$, where $A^{\times}_{s,D}$ outputs $(a, as + e)$ with $a \in R_q^{\times}, s \in D$. The analysis in the end of Section 2 of Stehlé and Steinfeld [41] shows that when $q = \Omega(n)$, R-LWE$^{\times}_{\mathsf{HNF}}$ remains hard.

The security proof of NTRU encryption scheme is similar to the security proof of Lemma 3.8 of Stehlé and Steinfeld [41]. The proof relies on the uniformity of public key and $p \in R_q^{\times}$. We chose a slightly different error distribution for our construction in Section 5.4, but adaption to our setting is straightforward.

Lemma 3. *Let $n \geq 8$ be a power of 2 such that $\Phi = x^n + 1$ splits into n irreducible factors modulo prime $q \geq 5$. Let $0 < \epsilon < 1/3$, $p \in R_q^{\times}$ and $s \geq 2n\sqrt{\ln(8nq)} \cdot q^{1/2+\epsilon}$. For any* IND-CPA *adversary $\mathcal{A}$ against* NTRU *encryption scheme, there exists an adversary $\mathcal{B}$ solving* R-LWE$^{\times}_{\mathsf{HNF}}$ *such that*

$$\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{NTRU}}(\mathcal{A}) \leq \mathbf{Adv}_{\mathsf{R\text{-}LWE}^{\times}_{\mathsf{HNF}}}(\mathcal{B}) + q^{-\Omega(n)}.$$

3 Blinded KEM

The blinded KEM primitive is the most important building block that BDGJ used to construct their key transport protocol [10] – also required are a signature scheme, a public-key encryption scheme, a hash function and a key derivation function. In this paper we only focus on blinded KEMs.

A *blinded KEM* scheme $\mathsf{BKEM} = (\mathsf{KG}, \mathsf{Encap}, \mathsf{Blind}, \mathsf{Decap}, \mathsf{Unblind})$ is parameterized by a key encapsulation mechanism $\mathsf{KEM} = (\mathsf{KG}, \mathsf{Encap}, \mathsf{Decap})$, a blinding algorithm Blind and an unblinding algorithm $\mathsf{Unblind}$. The *key generation* algorithm KG outputs an encapsulation key $ek \in \mathcal{K}_E$ and a decapsulation key $dk \in \mathcal{K}_D$. The *encapsulation* algorithm Encap takes as input an encapsulation key and outputs a (file encryption) key $k \in \mathcal{K}_F$ together with an encapsulation $C \in \mathcal{C}$ of that key. The *blinding* algorithm takes as input an encapsulation key and an encapsulation and outputs a blinded encapsulation $\tilde{C} \in \mathcal{C}$ and an unblinding key $uk \in \mathcal{K}_U$. The *decapsulation* algorithm Decap takes a decapsulation key and a (blinded) encapsulation as input and outputs a (blinded) key $\tilde{k} \in \mathcal{K}_B$. The *unblinding* algorithm takes as input an unblinding key and a blinded key and outputs a key.

Definition 6 (Correctness of a BKEM)**.** We say that a blinded KEM scheme BKEM has *(1 − ϵ)-correctness* if:

$$\mathbf{Pr}[\mathsf{Unblind}_{uk}(\tilde{k}) = k] \geq 1 - \epsilon,$$

for $(ek, dk) \leftarrow \mathsf{KG}$, $(C, k) \leftarrow \mathsf{Encap}_{ek}$, $(\tilde{C}, uk) \leftarrow \mathsf{Blind}_{ek}(C)$ and $\tilde{k} \leftarrow \mathsf{Decap}_{dk}(\tilde{C})$.

(A KEM scheme KEM has $(1 - \epsilon)$-*correctness* if

$$\mathbf{Pr}[\mathsf{Decap}_{dk}(C) = k] \geq 1 - \epsilon,$$

where $(ek, dk) \leftarrow \mathsf{KG}$ and $(C, k) \leftarrow \mathsf{Encap}_{ek}$.)

We parameterize all BKEM schemes by a public key encryption scheme (PKE), since any PKE scheme can trivially be turned into a KEM. We modify the above definition to be a PKE-based BKEM, where the KEM algorithms are described in Figure 5.

Definition 7 (PKE-based BKEM)**.** We call BKEM a *PKE-based BKEM* if the underlying scheme $\mathsf{KEM} = (\mathsf{KG}, \mathsf{Encap}, \mathsf{Decap})$ is parameterized by a PKE scheme $\mathsf{PKE} = (\mathsf{KG}_{\mathsf{PKE}}, \mathsf{Enc}, \mathsf{Dec})$ as described in Figure 5.

```
$\underline{\mathsf{KG}(\lambda):}$
    $\mathsf{pk}, \mathsf{sk} \leftarrow \mathsf{KG}_{\mathsf{PKE}}(\lambda)$
    $(ek, dk) \leftarrow (\mathsf{pk}, \mathsf{sk})$
    $\textbf{return } ek, dk$

$\underline{\mathsf{Encap}_{ek}:}$
    $k \xleftarrow{\$} \mathcal{M}$
    $C \leftarrow \mathsf{Enc}_{ek}(k)$
    $\textbf{return } C, k$

$\underline{\mathsf{Decap}_{dk}(\tilde{C}):}$
    $\tilde{k} \leftarrow \mathsf{Dec}_{dk}(\tilde{C})$
    $\textbf{return } \tilde{k}$
```

Figure 5: KEM algorithms parameterized by a PKE scheme $\mathsf{PKE} = (\mathsf{KG}_{\mathsf{PKE}}, \mathsf{Enc}, \mathsf{Dec})$.

3.1 Security

We define indistinguishability under chosen-plaintext attack (IND-CPA) for public key encryption and indistinguishability (IND) for blinded KEMs, respectively.

Definition 8. Let $\mathsf{PKE} = (\mathsf{KG}_{\mathsf{PKE}}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme. The IND-CPA advantage of any adversary $\mathcal{A}$ against PKE is

$$\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) = 2\left|\Pr[\mathbf{Exp}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) = 1] - 1/2\right|,$$

where the experiment $\mathbf{Exp}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A})$ is given in Figure 6 (left).

Definition 9. Let $\mathsf{BKEM} = (\mathsf{KG}, \mathsf{Encap}, \mathsf{Blind}, \mathsf{Decap}, \mathsf{Unblind})$ be a blinded KEM. The *distinguishing advantage* of any adversary $\mathcal{A}$ against BKEM getting r blinded encapsulations and their blinded decapsulation tuples is

$$\mathbf{Adv}^{\mathsf{IND}}_{\mathsf{BKEM}}(\mathcal{A}, r) = 2\left|\Pr[\mathbf{Exp}^{\mathsf{IND}}_{\mathsf{BKEM}}(\mathcal{A}, r) = 1] - 1/2\right|,$$

where the experiment $\mathbf{Exp}^{\mathsf{IND}}_{\mathsf{BKEM}}(\mathcal{A}, r)$ is given in Figure 6 (right).

The value r represents the number of recipients in the OAGKE protocol of BDGJ – in practice this will often be fairly small, and certainly bounded by the number of users of the system.

4 Homomorphic-based BKEM

We now show how to turn a homomorphic encryption scheme with certain properties into a BKEM, and analyze the security requirements of such a

```
$\mathbf{Exp}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A})$ :
    $b \xleftarrow{\$} \{0,1\}$
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}_{\mathsf{PKE}}$
    $(\mathsf{m}_0, \mathsf{m}_1, \mathtt{state}) \xleftarrow{\$} \mathcal{A}(\mathsf{pk})$
    $C_b \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mathsf{m}_b)$
    $b' \leftarrow \mathcal{A}(\mathtt{state}, C_b)$
    return $b' \overset{?}{=} b$
```

```
$\mathbf{Exp}^{\mathsf{IND}}_{\mathsf{BKEM}}(\mathcal{A}, r)$ :
    $b \xleftarrow{\$} \{0,1\}$
    $(ek, dk) \leftarrow \mathsf{KG}$
    $(C, k_1) \leftarrow \mathsf{Encap}_{ek}$
    $k_0 \xleftarrow{\$} \mathcal{K}_F$
    for $j \in \{1, \ldots, r\}$ do
        $(\tilde{C}_j, uk_j) \leftarrow \mathsf{Blind}_{ek}(C)$
        $\tilde{k}_j \leftarrow \mathsf{Decap}_{dk}(\tilde{C}_j)$
    $b' \leftarrow \mathcal{A}(ek, C, k_b, \{(\tilde{C}_j, \tilde{k}_j)\}_{1 \le j \le r})$
    return $b' \overset{?}{=} b$
```

Figure 6: IND-CPA experiment $\mathbf{Exp}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A})$ for a PKE scheme PKE (left). Indistinguishability experiment $\mathbf{Exp}^{\mathsf{IND}}_{\mathsf{BKEM}}(\mathcal{A}, r)$ for a BKEM scheme BKEM (right).

BKEM. We eventually prove that the homomorphic-based BKEM is post-quantum secure as long as the underlying homomorphic encryption scheme is post-quantum secure.

4.1 Generic homomorphic-based BKEM

We look for PKE schemes with the following homomorphic property: suppose C and C' are two ciphertexts, then $\mathsf{Dec}_{\mathsf{sk}}(C \oplus_1 C') = \mathsf{Dec}_{\mathsf{sk}}(C) \oplus_2 \mathsf{Dec}_{\mathsf{sk}}(C')$, where $\oplus_1$ and $\oplus_2$ denote two group operations.

We construct blinding and unblinding algorithms using this homomorphic property. Suppose the underlying PKE scheme has $1 - \epsilon$-correctness. To blind an encapsulation C (with corresponding file encryption key k) the Blind algorithm creates a fresh encapsulation C' (with corresponding blinding value k') using the Encap_{ek} algorithm, the blinded encapsulation $\tilde{C}$ is computed as $\tilde{C} \leftarrow C \oplus_1 C'$. The unblinding key uk is the inverse element of k' with respect to $\oplus_2$, that is, $uk \leftarrow k'^{-1}$. The blinding algorithms outputs $\tilde{C}$ and uk. The decapsulation algorithm can evaluate the blinded encapsulation because of the homomorphic property. The blinded key $\tilde{k}$ is the output of this decapsulation algorithm, that is, $\tilde{k} \leftarrow \mathsf{Decap}_{dk}(\tilde{C})$. Hence, $\tilde{k} = k \oplus_2 k'$ with probability $1 - 2\epsilon + \epsilon^2$. To unblind $\tilde{k}$ the unblinding algorithm outputs $\tilde{k} \oplus_2 uk$, which is k except for probability $2\epsilon - \epsilon^2$, and so the BKEM scheme has $(1 - 2\epsilon + \epsilon^2)$-correctness. Formally, we define the BKEM scheme constructed above as follows.

Definition 10 (Homomorphic-based BKEM)**.** Let BKEM be a PKE-based BKEM, as in Definition 7. Suppose the underlying public key encryption scheme is a homomorphic encryption scheme $\mathsf{HE} = (\mathsf{KG_{HE}}, \mathsf{Enc}, \mathsf{Dec})$ such that for any ciphertexts $C, C' \in \mathcal{C}$ and any key pair $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KG_{HE}}$ it holds that

$$\mathsf{Dec_{sk}}(C \oplus_1 C') = \mathsf{Dec_{sk}}(C) \oplus_2 \mathsf{Dec_{sk}}(C'),$$

where $(\mathcal{M}, \oplus_2)$ is the plaintext group and $(\mathcal{C}, \oplus_1)$ is the ciphertext group. Furthermore, let the blinding and unblinding algorithms operate according to Figure 7. We call such a scheme BKEM a *homomorphic-based BKEM*.

```
$\underline{\mathsf{Blind}_{ek}(C):}$
    $(C', k') \leftarrow \mathsf{Encap}_{ek}$
    $\tilde{C} \leftarrow C \oplus_1 C'$
    $uk \leftarrow k'^{-1}$
    **return** $\tilde{C}, uk$

$\underline{\mathsf{Unblind}_{uk}(\tilde{k}):}$
    $k \leftarrow \tilde{k} \oplus_2 uk$
    **return** $k$
```

Figure 7: Blinding and unblinding algorithms of the homomorphic based BKEM.

We stress that all BKEM schemes we consider in the rest of this paper are homomorphic-based BKEMs. The homomorphic encryption scheme HE does not need to be fully homomorphic, since we only need one operation in the blinding algorithm: a somewhat group homomorphic encryption scheme is sufficient.

4.2 Security requirements

In the indistinguishability game IND for BKEMs the adversary $\mathcal{A}$ has r blinded samples. If the decryptions of blinded encapsulations output the correct blinded keys, then these r blinded samples are the following two sets: $\{\tilde{C}_i = C \oplus_1 C'_i\}_{1,\ldots,r}$ and $\{\tilde{k}_i = k \oplus_2 k'_i\}_{i=1\ldots r}$, where the encapsulation is C and the real file encryption key is k. We want the blinded samples and the encapsulation to be random looking such that the combination of all these values does not reveal any information about the underlying file encryption key k that is being transported.

First, we show how to choose the blinding values k'_i to make the blinded keys $\tilde{k}_i$ look random. Then, we show how to make the blinded encapsulations $\tilde{C}_i$ look random, which is achievable when $\tilde{C}_i$ looks like a fresh output of the encapsulation algorithm: this idea is similar to circuit privacy [24]. Finally, we show how an IND-CPA-secure HE scheme ensures that the encapsulation does not reveal any information about the file encryption key. With these steps in place, we provide the main theorem in this paper stating how to achieve an IND secure BKEM scheme. In particular, if the underlying HE scheme is post-quantum IND-CPA secure then the corresponding homomorphic-based BKEM scheme is post-quantum IND secure.

Random-looking blinded keys. We want the blinded key to look like a random element of the space containing blinded keys. In the IND game the adversary is given several blinded keys of the form $\tilde{k} = k \oplus_2 k'$, where k is the file encryption key and k' is a blinding value, and wishes to gain information about k.

Let k be sampled uniformly at random from the file encryption key set, denoted $\mathcal{K}_F$, and let k' be sampled uniformly at random from the blinding value set, denoted $\mathcal{K}_R$. We would like that the size of $\mathcal{K}_F$ is large enough to prevent a brute force attacker from guessing k, say $|\mathcal{K}_F| = 2^\lambda$ for some security parameter λ. If $\mathcal{K}_R$ is a small set then the value of any blinded key $\tilde{k} = k \oplus_2 k'$ will be located within a short distance around k, so the adversary can successfully guess k with high probability. We always assume that $\mathcal{K}_R$ is at least as large as $\mathcal{K}_F$.

If a given blinded key $\tilde{k}$ can be expressed as a result of any file encryption key k and a blinding value k', with respect to an operation, then our goal is to ensure that the adversary cannot get any information of the true file encryption key hidden in $\tilde{k}$: ideally we wish it to be indistinguishable from a random element.

Definition 11 (ϵ-blinded blinded key)**.** Let BKEM be a blinded KEM with blinded key set $\mathcal{K}_B$. Let k be sampled uniformly random from the file encryption key set $\mathcal{K}_F$ and let k' be sampled uniformly random from the blinding value set $\mathcal{K}_R$. We define a *ϵ-blinded blinded key set* $\mathsf{S} := \{\tilde{k} \in$

$\mathcal{K}_B \mid \forall k \in \mathcal{K}_F, \exists 1 k' \in \mathcal{K}_R$ such that $\tilde{k} = k \oplus_2 k'\}$: we say that BKEM has ϵ-blinded blinded keys if

$$\mathbf{Pr}\left[\tilde{k} = k \oplus_2 k' \in \mathsf{S} \mid k \xleftarrow{\$} \mathcal{K}_F, k' \xleftarrow{\$} \mathcal{K}_R\right] = 1 - \epsilon.$$

Suppose the adversary is given any number of ϵ-blinded blinded keys from S with the same underlying file encryption key k. By the definition of the ϵ-blinded blinded set the file encryption key k can be any value in $\mathcal{K}_F$ and all values are equally probable. In other words, guessing k, given ϵ-blinded blinded keys, is the same as guessing a random value from $\mathcal{K}_F$. To prevent giving the adversary a better chance at guessing the key k we wish the blinded keys to be located inside the ϵ-blinded blinded key set S with high probability, which means we want ϵ to be small.

Fresh-looking blinded encapsulations. In the IND game for BKEMs the adversary $\mathcal{A}$ gets r blinded samples and has knowledge of the set $\{\tilde{C}_i = C \oplus_1 C'_i\}_{1,\ldots,r}$, where C is an encapsulation of a file encryption key k and C'_i is an encapsulation of a blinding value. We cannot guarantee that the set of the blinded encapsulations do not reveal any information about the encapsulation C. However, if each of these blinded encapsulations looks like a fresh output of the encapsulation algorithm then they are independent and random-looking compared to the encapsulation C. Therefore we want this set to be indistinguishable from the output set of the encapsulation algorithm.

Definition 12 (ϵ-blinded blinded encapsulation)**.** Let $\mathsf{HE\text{-}BKEM}$ be a homomorphic based BKEM. Let ek be any encapsulation key and C_0 be an encapsulation with the underlying file encryption key k_0. We say that $\mathsf{HE\text{-}BKEM}$ has *ϵ-blinded blinded encapsulation* if the statistical distance between the following distributions is at most ϵ:

$$X = \{C_0 \oplus_1 C' \mid k' \xleftarrow{\$} \mathcal{K}_R, C' \leftarrow \mathsf{Enc}_{ek}(k')\},$$
$$Y = \{C \mid k' \xleftarrow{\$} \mathcal{K}_R, C \leftarrow \mathsf{Enc}_{ek}(k_0 \oplus_2 k')\}.$$

This property ensures that the output of the blinding algorithm looks like a fresh encapsulation except for probability ϵ. Note that the BKEM constructions in [10], DH-BKEM [10, Section 4.1] and RSA-BKEM [10, Section 4.2], both have 0-blinded blinded encapsulation.

In most fully homomorphic encryption schemes the product of two ciphertexts is much larger in size compared to the sum of two ciphertexts, hence, it is easier to achieve ϵ-blinded blinded encapsulation for one addition compared to one multiplication. In our constructions we use addition.

Indistinguishability of BKEMs. Furthermore, if we want to achieve indistinguishability of blinded KEMs. We require the underlying homomorphic encryption scheme have some kind of semantic security to protect the message (the file encryption key) in the ciphertext (the encapsulation).

Theorem 5 (Main Theorem)**.** *For negligible ϵ_3, let* BKEM *be a homomorphic based BKEM designed as in Definition 10 from a $(1-\epsilon_3)$-correct homomorphic encryption scheme* HE. *Let the file encryption key k and the blinding value k' be sampled uniformly random from the large sets $\mathcal{K}_F$ and $\mathcal{K}_R$, respectively. Suppose* BKEM *has ϵ_1-blinded blinded encapsulations and ϵ_2-blinded blinded keys. For any adversary $\mathcal{A}$ against* BKEM *getting r blinded encapsulations and their blinded decapsulation samples, there exists an* IND-CPA *adversary $\mathcal{B}$ against* HE *such that*

$$\mathbf{Adv}^{\mathsf{IND}}_{\mathsf{BKEM}}(\mathcal{A}, r) \leq 2(r+1)(\epsilon_1 + \epsilon_2 + \epsilon_3) + \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{HE}}(\mathcal{B})$$

Proof. The proof of the theorem consists of a sequence of games.

Game 0

The first game is the experiment $\mathbf{Exp}^{\mathsf{IND}}_{\mathsf{BKEM}}(\mathcal{A}, r)$, given in Figure 6 (right). Let E_0 be the event that the adversary's guess b' equals b (and let E_i be the corresponding event for Game i). From Definition 9 we have that

$$\mathbf{Adv}^{\mathsf{IND}}_{\mathsf{BKEM}}(\mathcal{A}, r) = 2\left|\Pr[E_0] - 1/2\right|.$$

Game 1

We consider a modified game which is the same as Game 0 except that blinded key given to the adversary is the sum of the file encryption key and the blinding value instead of the decryption of the blinded encapsulation.

More precisely, suppose C is the encapsulation with corresponding file encryption key k. For $1 \leq j \leq r$, let $C'_j + C$ is the blinded encapsulation where C'_j is a fresh encapsulation with corresponding blinding value k'_j. When $\mathcal{A}$ queries for the blinded key of user j, the game outputs $k \oplus_2 k'_j$.

By the homomorphic property of PKE, if C and $C'_1, \ldots, C'_r$ all decrypt to the correct messages, then the output of blinded keys are the same in both Game 1 and Game 0. Hence the difference between Game 1 and Game 0 is upper bounded by the decryption error of PKE as follows.

$$\left|\Pr[E_1] - \Pr[E_0]\right| \leq 1 - (1-\epsilon_3)^{r+1} \approx (r+1)\epsilon_3.$$

Game 2

We consider a modified game which is the same as Game 1 except that blinded encapsulation and blinded key pairs given to the adversary are now independent and random compared to the file encryption key. More precisely, for $1 \leq j \leq r$:

- When $\mathcal{A}$ queries the blinded encapsulation of user j, the game first chooses a random ϵ-blinded blinded key (Definition 11), $\tilde{k}_j \xleftarrow{\$} \mathsf{S}$, and computes an encapsulation of this random key, $\tilde{C}_j \leftarrow \mathsf{Enc}_{ek}(\tilde{k}_j)$, which is given to $\mathcal{A}$.
- When $\mathcal{A}$ queries for the blinded key of user j, the game outputs $\tilde{k}_j$.

Step 1. We first prove that a real pair of blinded key and blinded encapsulation in Game 1 is $(\epsilon_1 + \epsilon_2)$-statistically close to the modified values in Game 2.

Suppose $k_0 \in \mathcal{K}_F$ is the file encryption key and $C_0 \leftarrow \mathsf{Enc}_{ek}(k_0)$ is the encapsulation with k_0, let $X = \{(k_0 \oplus_2 k', C_0 \oplus_1 C') \mid k' \xleftarrow{\$} \mathcal{K}_R, C' \leftarrow \mathsf{Enc}_{ek}(k')\}$ be the statistical distribution of the real pair of blinded key and blinded encapsulation output in Game 1, and $Y = \{(\tilde{k}, \tilde{C}) \mid \tilde{k} \xleftarrow{\$} \mathsf{S}, \tilde{C} \leftarrow \mathsf{Enc}_{ek}(\tilde{k})\}$ be the statistical distribution of the modified values output in Game 2. We define a middle distribution $Z = \{(k_0 \oplus_2 k', C) \mid k' \xleftarrow{\$} \mathcal{K}_R, C \leftarrow \mathsf{Enc}_{ek}(k_0 \oplus_2 k')\}$. We compute the statistical distance between X and Y as follows.

$$
\begin{aligned}
\Delta(X,Y) &\leq \Delta(X,Z)+\Delta(Z,Y)\\
&= \Delta(X,Z)+\frac{1}{2}\Big(\sum_{\substack{\tilde{k}\in\mathcal{K}_B\\ \tilde{C}\in\mathcal{C}}}\Big|\mathbf{Pr}[Z=(\tilde{k},\tilde{C})]-\mathbf{Pr}[Y=(\tilde{k},\tilde{C})]\Big|\Big)\\
&\leq \epsilon_1+\frac{1}{2}\Big(\sum_{\substack{\tilde{k}\in\mathcal{K}_B\\ \tilde{C}\in\mathcal{C}}}\Big|\mathbf{Pr}[Z=(\tilde{k},\tilde{C}) \mid \tilde{k}\in\mathsf{S}]\cdot\mathbf{Pr}[\tilde{k}\in\mathsf{S}]\\
&\qquad +\mathbf{Pr}[Z=(\tilde{k},\tilde{C}) \mid \tilde{k}\notin\mathsf{S}]\cdot\mathbf{Pr}[\tilde{k}\notin\mathsf{S}]-\mathbf{Pr}[Y=(\tilde{k},\tilde{C})]\Big|\Big)\\
&= \epsilon_1+\frac{1}{2}\Big(\sum_{\substack{\tilde{k}\in\mathsf{S}\\ \tilde{C}\in\mathcal{C}}}\Big|\mathbf{Pr}[Z=(\tilde{k},\tilde{C}) \mid \tilde{k}\in\mathsf{S}]\cdot(1-\epsilon_2)-\mathbf{Pr}[Y=(\tilde{k},\tilde{C})]\Big|\\
&\qquad +\sum_{\substack{\tilde{k}\notin\mathsf{S}\\ \tilde{C}\in\mathcal{C}}}\Big|\mathbf{Pr}[Z=(\tilde{k},\tilde{C}) \mid \tilde{k}\notin\mathsf{S}]\cdot\epsilon_2\Big|\Big) \qquad (1)\\
&\leq \epsilon_1+\frac{1}{2}\Big(\sum_{\substack{\tilde{k}\in\mathsf{S}\\ \tilde{C}\in\mathcal{C}}}\Big|\epsilon_2\cdot\mathbf{Pr}[Y=(\tilde{k},\tilde{C})]\Big|+1\cdot\epsilon_2\Big) \qquad (2)\\
&\leq \epsilon_1+\epsilon_2
\end{aligned}
$$

Note that in (1) we split the summation into two parts, namely $\tilde{k}\in\mathsf{S}$ and $\tilde{k}\notin\mathsf{S}$. For $\tilde{k}\in\mathsf{S}$ we have $\mathbf{Pr}[Z=(\tilde{k},\tilde{C}) \mid \tilde{k}\notin\mathsf{S}]\cdot\mathbf{Pr}[\tilde{k}\notin\mathsf{S}]=0$, and for $\tilde{k}\notin\mathsf{S}$ we have $\mathbf{Pr}[Z=(\tilde{k},\tilde{C}) \mid \tilde{k}\in\mathsf{S}]\cdot\mathbf{Pr}[\tilde{k}\in\mathsf{S}]=0$ and $\mathbf{Pr}[Y=(\tilde{k},\tilde{C})]=0$. Furthermore, (2) holds because distributions Z and Y over set S are equal. For r samples:

$$\Big|\Pr[E_2]-\Pr[E_1]\Big| \leq r(\epsilon_1+\epsilon_2).$$

Step 2. Next, we claim that there exists an adversary $\mathcal{B}$ against $\mathsf{IND\text{-}CPA}$ security of HE such that

$$2\Big|\Pr[E_2]-\frac{1}{2}\Big| = \mathbf{Adv}_{\mathsf{HE}}^{\mathsf{IND\text{-}CPA}}(\mathcal{B}).$$

We construct a reduction $\mathcal{B}$ that plays the $\mathsf{IND\text{-}CPA}$ game by running $\mathcal{A}$, that simulates the responses of Game 2 to $\mathcal{A}$ as follows.

1. $\mathcal{B}$ flips a coin $b \xleftarrow{\$} \{0,1\}$,
2. $\mathcal{B}$ queries its IND-CPA challenger to get the public key of its IND-CPA game, and forwards this public key as the encapsulation key to $\mathcal{A}$,
3. $\mathcal{B}$ simulates the encapsulation by randomly choosing two group key k_0, k_1, sends challenge query with input (k_0, k_1) to its IND-CPA challenger, and forwards the response C to $\mathcal{A}$,
4. $\mathcal{B}$ simulates the output of Blind and Decap by using the Encap algorithm. $\mathcal{B}$ samples $\tilde{k} \xleftarrow{\$} \mathsf{S}$, computes $\tilde{C} \leftarrow \mathsf{Enc}_{ek}(\tilde{k})$, and outputs $\tilde{C}$ as the blinded encapsulation and $\tilde{k}$ as the decapsulation of the blinded encapsulation,
5. When $\mathcal{A}$ asks for a challenge, $\mathcal{B}$ sends k_b to $\mathcal{A}$,
6. After $\mathcal{A}$ returns b', $\mathcal{B}$ sends $1 \oplus b \oplus b'$ to the challenger.

If the challenge ciphertext $\mathcal{B}$ received in $\mathbf{Exp}^{\mathsf{IND\text{-}CPA}}_{\mathsf{HE}}(\mathcal{B})$ is C_b, then $\mathcal{B}$ perfectly simulates the inputs of $\mathcal{A}$ in Game 2 when the output of the key is a real key. Otherwise (the challenge ciphertext $\mathcal{B}$ received in $\mathbf{Exp}^{\mathsf{IND\text{-}CPA}}_{\mathsf{HE}}(\mathcal{B})$ is $C_{1\text{-}b}$), k_b is a random key to $\mathcal{A}$ and $\mathcal{B}$ perfectly simulate the inputs of $\mathcal{A}$ in Game 2 when the output of the key is a random key. $\square$

Remark 1. *As a specific case of Theorem 5, the* DH-BKEM *construction of BDGJ has 0-blinded blinded encapsulations and 0-blinded blinded keys, and the indistinguishability of* DH-BKEM *is upper bounded by* DDH *advantage (defined in the real-or-random sense instead of left-or-right). That is*

$$\mathbf{Adv}^{\mathsf{IND}}_{\mathsf{DH\text{-}BKEM}}(\mathcal{A}, r) \leq \mathbf{Adv}^{\mathsf{DDH}}(\mathcal{B}).$$

This observation matches with the result of Boyd et al. [10, Theorem 1].

5 Instantiating Homomorphic-based BKEMs

We provide two homomorphic-based BKEM constructions, based on Gentry's homomorphic encryption scheme (Section 2.3) and the NTRU variant by Stehlé and Steinfeld (Section 2.4). We show that (for some parameters) our BKEM schemes are post-quantum secure, by Theorem 5, as long as the underlying HE schemes are post-quantum secure [24, 36, 41]. We only require the HE scheme to support one homomorphic operation, and we have chosen addition. Our HE schemes do not need to support bootstrapping or any multiplicative depth.

5.1 Two Homomorphic-based BKEM Schemes

Let $\mathsf{HE} = (\mathsf{KG}_{\mathsf{HE}}, \mathsf{Enc}_{\mathsf{HE}}, \mathsf{Dec}_{\mathsf{HE}})$ be a homomorphic encryption scheme described in Section 2.3 or Section 2.4 with $(1-\epsilon_3)$-correctness for negligible ϵ_3. Let L be any full-rank n-dimensional lattice, for any $\epsilon \in (0,1)$, $s \geq \eta_\epsilon(L)$, and $r \geq 2^{\omega(\log(n))} \cdot s$. The abstract construction of HE-BKEM is in Figure 8.

```
$\underline{\mathsf{KG}(\lambda):}$
    $\mathsf{pk}, \mathsf{sk} \leftarrow \mathsf{KG}_{\mathsf{HE}}(\lambda)$
    $(ek, dk) \leftarrow (\mathsf{pk}, \mathsf{sk})$
    return $ek, dk$

$\underline{\mathsf{Encap}_{ek}:}$
    $k \xleftarrow{\$} \mathcal{K}_F$
    $C \leftarrow \mathsf{Enc}_{\mathsf{HE}}(ek, s, k)$
    return $C, k$

$\underline{\mathsf{Blind}_{ek}(C):}$
    $k' \xleftarrow{\$} \mathcal{K}_R$
    $C' \leftarrow \mathsf{Enc}_{\mathsf{HE}}(ek, r, k')$
    $\tilde{C} \leftarrow \mathsf{Add}_{\mathsf{HE}}(C, C')$
    $uk \leftarrow -k' \mod \mathbf{B}$
    return $\tilde{C}, uk$

$\underline{\mathsf{Decap}_{dk}(\tilde{C}):}$
    $\tilde{k} \leftarrow \mathsf{Dec}_{\mathsf{HE}}(dk, \tilde{C})$
    return $\tilde{k}$

$\underline{\mathsf{Unblind}_{uk}(\tilde{k}):}$
    $k \leftarrow \tilde{k} + uk \mod \mathbf{B}$
    return $k$
```

Figure 8: HE-BKEM, where $\mathbf{B}$ is the basis of the plaintext space $\mathcal{P}$.

5.2 Constructions of random-looking blinded keys

We want the blinded keys to be in the ϵ-blinded blinded key set S with high probability, and we analyze the requirements of the blinding values. We provide two constructions of the ϵ-blinded blinded keys set S as follows.

Construction I. A file encryption key of HE-BKEM is a random element located in a subspace of the underlying HE scheme's message space $\mathcal{M}$. We want to take a small file encryption key k and add a large blinding value k' to produce a slightly larger blinded key $\tilde{k}$, hence, the corresponding key sets should satisfy $\mathcal{K}_F \subseteq \mathcal{K}_R \subseteq \mathcal{K}_B \subseteq \mathcal{M}$ Suppose $\mathcal{M}$ is HE scheme's message space with generators $1, x, \ldots, x^{n-1}$ and order q, i.e. $\mathcal{M} = \{d_0 + d_1 x + \cdots + d_{n-1}x^{n-1} \mid d_i \in \mathbb{F}_q\}$. The addition of two elements in $\mathcal{M}$ is defined as follows

$$\begin{aligned}(a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}) &+ (b_0 + b_1 x + \cdots + b_{n-1}x^{n-1}) \\ &= (a_0 + b_0) + (a_1 + b_1)x + \cdots + (a_{n-1} + b_{n-1})x^{n-1}\end{aligned}$$

Suppose $\mathcal{K}_F = \{d_0 + d_1 x + \cdots + d_{n-1}x^{n-1} \mid d_i \in \mathbb{Z}_{\lfloor\sqrt{q/2}\rfloor}\}$ and $\mathcal{K}_R = \{d_0 + d_1 x + \cdots + d_{n-1}x^{n-1} \mid d_i \in \mathbb{Z}_{\lfloor q/2\rfloor}\}$. For any $c_i \in \{\lfloor\sqrt{q/2}\rfloor, \ldots, \lfloor q/2\rfloor\}$ and any $a_i \in \mathbb{Z}_{\lfloor\sqrt{q/2}\rfloor}$ there exists a unique $b_i = c_i - a_i \in \mathbb{Z}_{\lfloor q/2\rfloor}$. As such, for these restricted $c_0 + c_1 x + \cdots + c_{n-1}x^{n-1}$ and for any $a_0 + a_1 x + \cdots a_{n-1}x^{n-1} \in \mathcal{K}_F$ there exists a unique $b_0 + b_1 x + \cdots b_{n-1}x^{n-1} \in \mathcal{K}_R$ such that $(a_0 + a_1 x + \cdots a_{n-1}x^{n-1}) + (b_0 + b_1 x + \cdots b_{n-1}x^{n-1}) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1}$. Then

$$\mathsf{S} = \{d_0 + d_1 x + \cdots + d_{n-1}x^{n-1} \mid d_i \in \{\lfloor\sqrt{q/2}\rfloor, \ldots, \lfloor q/2\rfloor\}\}$$

Note that for any $i \in \{0, \ldots, n-1\}$,

$$\mathbf{Pr}\left[a_i + b_i \in \{\lfloor\sqrt{\tfrac{q}{2}}\rfloor, \ldots, \lfloor\tfrac{q}{2}\rfloor\} \mid a_i \xleftarrow{\$} \mathbb{Z}_{\lfloor\sqrt{\frac{q}{2}}\rfloor}, b_i \xleftarrow{\$} \mathbb{Z}_{\lfloor\frac{q}{2}\rfloor}\right] = 1 - \frac{\lfloor\sqrt{\frac{q}{2}}\rfloor - 1}{\lfloor\frac{q}{2}\rfloor},$$

so the probability that a blinded key is located in the ϵ-blinded blinded set is

$$\mathbf{Pr}\left[\tilde{k} = k + k' \in \mathsf{S} \mid k \xleftarrow{\$} \mathcal{K}_F, k' \xleftarrow{\$} \mathcal{K}_R\right] = \left(1 - \frac{\lfloor\sqrt{\frac{q}{2}}\rfloor - 1}{\lfloor\frac{q}{2}\rfloor}\right)^n \approx 1 - \frac{n}{\lfloor\sqrt{\frac{q}{2}}\rfloor}.$$

In this construction, HE-BKEM has ϵ-blinded blinded keys with $\epsilon = \frac{n}{\lfloor\sqrt{\frac{q}{2}}\rfloor}$. For suitably large q, the above ϵ can be made negligible.

Construction II. Let the file encryption key k be an element in a subset of $\mathcal{M}$: we want to add a random blinding value k' from the whole message space $\mathcal{M}$ to produce a random-looking blinded key $\tilde{k}$, hence, the corresponding key sets should satisfy $\mathcal{K}_F \subseteq \mathcal{K}_R = \mathcal{K}_B = \mathcal{M}$. For any blinded key $\tilde{k} \in \mathcal{M}$ and any file encryption key $k \in \mathcal{K}_F$ there exists a unique random value $k' = \tilde{k} - k \mod \mathbf{B} \in \mathcal{M}$ such that $\tilde{k} = k + k' \mod \mathbf{B}$, thus the ϵ-blinded blinded set S is $\mathcal{M}$ and thus

$$\mathbf{Pr}\left[\tilde{k} = k + k' \mod \mathbf{B} \in \mathsf{S} \mid k \xleftarrow{\$} \mathcal{K}_F, k' \xleftarrow{\$} \mathcal{M}\right] = 1.$$

In this construction, $\mathsf{HE\text{-}BKEM}$ has ϵ-blinded blinded keys with $\epsilon = 0$.

Remark 2. *Both of these constructions can be applied to our* $\mathsf{HE\text{-}BKEM}$ *schemes.*

5.3 Construction of fresh-looking blinded encapsulations

We claim that $\mathsf{HE\text{-}BKEM}$ in Figure 8 has ϵ-blinded blinded encapsulations with negligible ϵ. The idea is to take the small constant ciphertext and add a ciphertext with large error(s) and the resulting ciphertext should look like a fresh ciphertext with large error(s). The details are given in the following lemma.

Lemma 4. *Let* $\mathsf{HE\text{-}BKEM}$ *be a homomorphic based BKEM with the underlying homomorphic encryption scheme described in Section 2.3 or 2.4 Let ek be any encapsulation key, and recall that the encryption algorithm* $\mathsf{Enc_{HE}}(ek, s, \cdot)$ *uses the discrete Gaussian distribution $D_{L,s,\mathbf{0}}$ as the error distribution. Suppose $C_0 = \mathsf{Enc_{HE}}(ek, s, k_0)$ is an encapsulation of the underlying file encryption key k_0. For any $\epsilon \in (0, 1)$, let $s \geq \eta_\epsilon(L)$ and $r \geq 2^{\omega(\log(n))} \cdot s$, then the statistical distance between the following distributions is negligible*

$$X = \{C_0 \oplus_1 C' \mid k' \xleftarrow{\$} \mathcal{K}_R, C' \leftarrow \mathsf{Enc_{HE}}(ek, r, k')\}$$

$$Y = \{C \mid k' \xleftarrow{\$} \mathcal{K}_R, C \leftarrow \mathsf{Enc_{HE}}(ek, r, k_0 \oplus_2 k')\}.$$

Proof. We prove the result for Gentry's scheme; similar analysis for NTRU follows the same approachch. Suppose $C_0 = k_0 + e_0$, where $e_0 \leftarrow D_{L,s,\mathbf{0}}$. Then

$$C_0 \oplus_1 \mathsf{Enc_{HE}}(ek, r, k') = k_0 + e_0 + k' + D_{L,r,\mathbf{0}} = k_0 + k' + e_0 + D_{L,r,\mathbf{0}}.$$

By Lemma 1, we have $\|e_0\| > s\sqrt{n}$ with negligible probability. For $\|e_0\| \leq s\sqrt{n}$, we have $\frac{\|e_0\|}{r} \leq \frac{\sqrt{n}}{2^{\omega(\log(n))}}$, which is negligible for sufficient large n. By Lemma 2, we have $e_0 + D_{L,r,\mathbf{0}} \overset{s}{\approx} D_{L,r,\mathbf{0}}$. Therefore,

$$C_0 \oplus_1 \mathsf{Enc}_{\mathsf{HE}}(ek, r, k') \overset{s}{\approx} k_0 + k' + D_{L,r,\mathbf{0}} = \mathsf{Enc}_{\mathsf{HE}}(ek, r, k_0 \oplus_2 k').$$

□

5.4 Indistinguishability of our HE-BKEM

The HE-BKEM schemes, defined in Section 5.1, have random-looking blinded keys, which follows from the designs discussed in Section 5.2. Furthermore, these schemes have fresh-looking blinded encapsulations, which follows from from Lemma 4 discussed in Section 5.3. The following corollaries show GHE-BKEM and NTRU-BKEM are IND-secure BKEMs with post-quantum security.

Corollary 1. *Let* GHE-BKEM *be a homomorphic-based BKEM described in Section 5.1. For negligible ϵ, ϵ_2, choose parameters as in Lemma 4, Theorem 1 and Thm. 2. Suppose* GHE-BKEM *has ϵ_2-blinded blinded keys. If there is an algorithm that breaks the indistinguishability of* GHE-BKEM, *i.e. the distinguishing advantage of this algorithm against* GHE-BKEM *getting r blinded encapsulation and their blinded decapsulation tuples is non-negligible, then there exists a quantum algorithm that solves worst-case* SIVP.

Proof. By Lemma 4 there exists a negligible ϵ_1 such that GHE-BKEM has ϵ_1-blinded blinded encapsulations. Then we can apply Theorem 5, which states that if there is an algorithm that breaks the indistinguishability of GHE-BKEM then there exists an algorithm breaks IND-CPA security of GHE, and by Theorem 3 we have a quantum algorithm that solves worst-case SIVP. □

Corollary 2. *Let* NTRU-BKEM *be a homomorphic-based BKEM described in Section 5.1. For negligible ϵ, ϵ_2, choose parameters as in Lemma 4, Lemma 3, and Thm. 4. Suppose* NTRU-BKEM *has ϵ_2-blinded blinded keys. If there is an algorithm that breaks indistinguishability of* NTRU-BKEM *then there exists a quantum algorithm that solves $O(\sqrt{n}/\alpha)$-approximate* SIVP *(or* SVP*) on ideal lattices.*

Proof. Similar to the proof of Corollary 1, from Lemma 4 and Theorem 5 we know that if there is an algorithm that breaks the indistinguishability of NTRU-BKEM then there exists an algorithm that breaks IND-CPA security of NTRU. By Lemma 3 there exists an adversary solving $\mathsf{R\text{-}LWE}^{\times}_{\mathsf{HNF}}$ and by Theorem 4 there exists a quantum algorithm that solves SIVP. □

Parameter settings. For our HE-BKEM schemes, the parameters of the underlying homomorphic encryption schemes are chosen from Gentry [24] or Stehlé and Steinfeld [41], which is required to achieve IND-CPA security. Furthermore, our BKEM schemes require that $r = 2^{\omega(\log(n))} \cdot s$, where s is the standard deviations of a "narrow" Gaussian distributions $D_{L,s,\mathbf{0}}$ and r is the standard deviations of a "wider" Gaussian distributions $D_{L,r,\mathbf{0}}$. We also follows the designs discussed in Section 5.2 to construct random-looking blinded keys. We conclude that for these parameter settings our proposed BKEM schemes are post-quantum secure.

References

[1] Abe, M., Gennaro, R., Kurosawa, K., Shoup, V.: Tag-kem/dem: A new framework for hybrid encryption and A new analysis of kurosawa-desmedt KEM. In: Cramer, R. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 3494, pp. 128–146. Springer (2005). https://doi.org/10.1007/11426639_8

[2] Albrecht, M., Bai, S., Ducas, L.: A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In: Robshaw, M., Katz, J. (eds.) CRYPTO (1). pp. 153–178. Springer-Verlag (2016). https://doi.org/10.1007/978-3-662-53018-4_6

[3] Alkim, E., Bos, J.W., Ducas, L., Easterbrook, K., LaMacchia, B., Longa, P., Mironov, I., Nikolaenko, V., Peikert, C., Raghunathan, A., Stebila, D.: FrodoKEM: Learning With Errors Key Encapsulation. https://frodokem.org/files/FrodoKEM-specification-20190330.pdf, Submission to the NIST Post-Quantum Standardization project, round 2

[4] Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange - A new hope. In: Holz, T., Savage, S. (eds.) USENIX Security Symposium. pp. 327–343. USENIX Association (2016)

[5] Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Kyber (version 2.0). https://pq-crystals.org/kyber/data/kyber-specification-round2.pdf, Submission to the NIST Post-Quantum Standardization project, round 2

[6] Aviram, N., Gellert, K., Jager, T.: Session resumption protocols and efficient forward security for TLS 1.3 0-RTT. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT (2). Lecture Notes in Computer Science, vol. 11477, pp. 117–150. Springer (2019). https://doi.org/10.1007/978-3-030-17656-3_5

[7] Bernstein, D.J., Chuengsatiansup, C., Lange, T., van Vredendaal, C.: NTRU prime: Reducing attack surface at low cost. In: Adams, C., Camenisch, J. (eds.) SAC. Lecture Notes in Computer Science, vol. 10719, pp. 235–260. Springer (2017)

[8] Bos, J.W., Costello, C., Ducas, L., Mironov, I., Naehrig, M., Nikolaenko, V., Raghunathan, A., Stebila, D.: Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM Conference on Computer and Communications Security. pp. 1006–1018. ACM (2016). https://doi.org/10.1145/2976749.2978425

[9] Bos, J.W., Lauter, K.E., Loftus, J., Naehrig, M.: Improved security for a ring-based fully homomorphic encryption scheme. In: Stam, M. (ed.) IMACC. Lecture Notes in Computer Science, vol. 8308, pp. 45–64. Springer (2013). https://doi.org/10.1007/978-3-642-45239-0_4

[10] Boyd, C., Davies, G.T., Gjøsteen, K., Jiang, Y.: Offline assisted group key exchange. In: Chen, L., Manulis, M., Schneider, S. (eds.) ISC. Lecture Notes in Computer Science, vol. 11060, pp. 268–285. Springer (2018). https://doi.org/10.1007/978-3-319-99136-8_15

[11] Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In:

Goldwasser, S. (ed.) ITCS. pp. 309–325. ACM (2012). https://doi.org/10.1145/2090236.2090262

[12] Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-lwe and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 6841, pp. 505–524. Springer (2011). https://doi.org/10.1007/978-3-642-22792-9_29

[13] Chen, C., Danba, O., Hoffstein, J., Hülsing, A., Rijneveld, J., Schanck, J.M., Schwabe, P., Whyte, W., Zhang, Z.: NTRU). https://ntru.org/f/ntru-20190330.pdf, Submission to the NIST Post-Quantum Standardization project, round 2

[14] Cheon, J.H., Han, K., Kim, J., Lee, C., Son, Y.: A practical post-quantum public-key cryptosystem based on spLWE. In: Hong, S., Park, J.H. (eds.) ICISC. Lecture Notes in Computer Science, vol. 10157, pp. 51–74 (2016). https://doi.org/10.1007/978-3-319-53177-9_3

[15] Cheon, J.H., Jeong, J., Lee, C.: An algorithm for NTRU problems and cryptanalysis of the GGH multilinear map without a low-level encoding of zero. LMS Journal of Computation and Mathematics **19**(A), 255–266 (2016). https://doi.org/10.1112/S1461157016000371

[16] Cheon, J.H., Kim, A., Kim, M., Song, Y.S.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT (1). Lecture Notes in Computer Science, vol. 10624, pp. 409–437. Springer (2017). https://doi.org/10.1007/978-3-319-70694-8_15

[17] Cohn-Gordon, K., Cremers, C., Garratt, L., Millican, J., Milner, K.: On ends-to-ends encryption: Asynchronous group messaging with strong security guarantees. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) CCS. pp. 1802–1819. ACM (2018). https://doi.org/10.1007/978-3-030-17656-3_5

[18] Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext at-

tack. Cryptology ePrint Archive, Report 2001/108 (2001), https://eprint.iacr.org/2001/108

[19] Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM J. Comput. **33**(1), 167–226 (2003). https://doi.org/10.1137/S0097539702403773

[20] D'Anvers, J., Karmakar, A., Roy, S.S., Vercauteren, F.: Saber: Module-LWR based key exchange, cpa-secure encryption and cca-secure KEM. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) AFRICACRYPT. Lecture Notes in Computer Science, vol. 10831, pp. 282–305. Springer (2018). https://doi.org/10.1007/978-3-319-89339-6_16

[21] Dent, A.W.: A designer's guide to KEMs. In: Paterson, K.G. (ed.) IMACC. Lecture Notes in Computer Science, vol. 2898, pp. 133–151. Springer (2003). https://doi.org/10.1007/978-3-540-40974-8_12

[22] Derler, D., Jager, T., Slamanig, D., Striecks, C.: Bloom filter encryption and applications to efficient forward-secret 0-rtt key exchange. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT (3). Lecture Notes in Computer Science, vol. 10822, pp. 425–455. Springer (2018). https://doi.org/10.1007/978-3-319-78372-7_14

[23] Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012), https://eprint.iacr.org/2012/144

[24] Gentry, C.: A Fully Homomorphic Encryption Scheme. Ph.D. thesis, Stanford University, Stanford, CA, USA (2009), aAI3382729

[25] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Dwork, C. (ed.) STOC. pp. 197–206. ACM (2008). https://doi.org/10.1145/1374376.1374407

[26] Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO (1).

Lecture Notes in Computer Science, vol. 8042, pp. 75–92. Springer (2013). https://doi.org/10.1007/978-3-642-40041-4_5

[27] Green, M.D., Miers, I.: Forward secure asynchronous messaging from puncturable encryption. In: IEEE Symposium on Security and Privacy. pp. 305–320. IEEE Computer Society (2015). https://doi.org/10.1109/SP.2015.26

[28] Günther, F., Hale, B., Jager, T., Lauer, S.: 0-RTT key exchange with full forward secrecy. In: Coron, J., Nielsen, J.B. (eds.) EUROCRYPT (3). Lecture Notes in Computer Science, vol. 10212, pp. 519–548 (2017). https://doi.org/10.1007/978-3-319-56617-7_18

[29] Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: Buhler, J. (ed.) ANTS. Lecture Notes in Computer Science, vol. 1423, pp. 267–288. Springer (1998). https://doi.org/10.1007/BFb0054868

[30] Hofheinz, D., Kiltz, E.: Secure hybrid encryption from weakened key encapsulation. In: Menezes, A. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 4622, pp. 553–571. Springer (2007). https://doi.org/10.1007/978-3-540-74143-5_31

[31] Hülsing, A., Rijneveld, J., Schanck, J.M., Schwabe, P.: High-speed key encapsulation from NTRU. In: Fischer, W., Homma, N. (eds.) CHES. Lecture Notes in Computer Science, vol. 10529, pp. 232–252. Springer (2017). https://doi.org/10.1007/978-3-319-66787-4_12

[32] Kurosawa, K., Desmedt, Y.: A new paradigm of hybrid encryption scheme. In: Franklin, M.K. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 3152, pp. 426–442. Springer (2004). https://doi.org/10.1007/978-3-540-28628-8_26

[33] López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Karloff, H.J., Pitassi, T. (eds.) STOC. pp. 1219–1234. ACM (2012). https://doi.org/10.1145/2213977.2214086

[34] Lu, X., Liu, Y., Jia, D., Xue, H., He, J., Zhang, Z., Liu, Z., Yang, H., Li, B., Wang, K.: LAC Lattice-based Cryptosystems, Submission to the NIST Post-Quantum Standardization project, round 2

[35] Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 6110, pp. 1–23. Springer (2010). https://doi.org/10.1007/978-3-642-13190-5_1

[36] Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. J. ACM **60**(6), 43:1–43:35 (Nov 2013). https://doi.org/10.1145/2535925

[37] Lyubashevsky, V., Peikert, C., Regev, O.: A toolkit for Ring-LWE cryptography. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT. Lecture Notes in Computer Science, vol. 7881, pp. 35–54. Springer (2013). https://doi.org/10.1007/978-3-642-38348-9_3

[38] Marlinspike, M., Perrin, T.: The X3DH key agreement protocol. https://signal.org/docs/specifications/x3dh/ (November 2016)

[39] Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. SIAM J. Comput. **37**(1), 267–302 (Apr 2007). https://doi.org/10.1137/S0097539705447360

[40] NIST Post-Quantum Cryptography Standardization. https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization, accessed: 2019-11-15

[41] Stehlé, D., Steinfeld, R.: Making NTRU as secure as worst-case problems over ideal lattices. In: Paterson, K.G. (ed.) EUROCRYPT. pp. 27–47. Lecture Notes In Computer Science, Springer-Verlag (2011). https://doi.org/10.1007/978-3-642-20465-4_4

[42] The messaging layer security (MLS) protocol. Internet draft, in progress. https://datatracker.ietf.org/wg/mls/about, accessed: 2019-11-25

Paper III

Fast and Secure Updatable Encryption

Colin Boyd, Gareth T. Davies, Kristian Gjøsteen and Yao Jiang

Published in the 40th Annual International Cryptology Conference, CRYPTO 2020.

Fast and Secure Updatable Encryption[†]

Colin Boyd[1] Gareth T. Davies[2] Kristian Gjøsteen[1] and Yao Jiang[1]

[1]Norwegian University of Science and Technology, NTNU, Norway.
{colin.boyd,kristian.gjosteen,yao.jiang}@ntnu.no
[2]Bergische Universität Wuppertal, Wuppertal, Germany.
davies@uni-wuppertal.de

Abstract

Updatable encryption allows a client to outsource ciphertexts to some untrusted server and periodically rotate the encryption key. The server can update ciphertexts from an old key to a new key with the help of an update token, received from the client, which should not reveal anything about plaintexts to an adversary.

We provide a new and highly efficient suite of updatable encryption schemes that we collectively call SHINE. In the variant designed for short messages, ciphertext generation consists of applying one permutation and one exponentiation (per message block), while updating ciphertexts requires just one exponentiation. Variants for longer messages provide much stronger security guarantees than prior work that has comparable efficiency. We present a new confidentiality notion for updatable encryption schemes that implies prior notions. We prove that SHINE is secure under our new confidentiality definition while also providing ciphertext integrity.

† © IACR 2020: An extended abstract of this paper appears in the proceedings of IACR CRYPTO 2020, with DOI: 10.1007/978-3-030-56784-2_16. This is the full version.

1 Introduction

The past decades have demonstrated clearly that key compromise is a real threat for deployed systems. The standard technique for mitigating key compromise is to regularly *rotate* the encryption keys – generate new ones and switch the ciphertexts to encryption under the new keys. Key rotation is a well-established technique in applications such as payment cards [Cou18] and cloud storage [KRS$^+$03].

For a local drive or server, key rotation is feasible by decrypting and re-encrypting with a new key, since symmetric encryption operations are fast and parallelizable and bandwidth is often plentiful. When ciphertext storage has been outsourced to some (untrusted) cloud storage provider, bandwidth is often considerably more expensive than computation, and even for small volumes of data it may be prohibitively expensive to download, re-encrypt and upload the entire database even once. This means that key rotation by downloading, decrypting, re-encrypting and reuploading is practically infeasible.

An alternative approach to solving this problem is to use *updatable encryption* (UE), first defined by Boneh et al. [BLMR13] (henceforth BLMR). The user computes a *token* and sends it to the storage server. The token allows the server to update the ciphertexts so that they become encryptions under some new key. Although the token clearly depends on both the old and new encryption keys, knowledge of the token alone should not allow the server to obtain either key. In a typical usage of UE, the cloud storage provider will receive a new token on a periodic basis, and the provider then updates every stored ciphertext. The time period for which a given key is valid for is called an *epoch*.

In the past few years there has been considerable interest in extending the understanding of UE. A series of prominent papers [BLMR13, EPRS17a, LT18a, KLR19a] have provided both new (typically stronger) security definitions and concrete or generic constructions to meet their definitions. (We make a detailed comparison of related work in Section 1.1.1 next.) An important distinction between earlier schemes is whether or not the token (and in particular its size) depends on the ciphertexts to be updated (and in particular the number of ciphertexts). Schemes for which a token is assigned to each ciphertext are *ciphertext-dependent* and were studied by Everspaugh et al. [EPRS17a] (henceforth EPRS). If the token is independent of the ciphertexts to be updated, such as in BLMR [BLMR15], we have a *ciphertext-independent*[1] scheme. A clear and important goal is to limit the bandwidth required and so, in general, one should prefer ciphertext-independent schemes. Thus, as with the most recent work [LT18a, KLR19a], we focus on such schemes in this paper. The ciphertext update procedure, performed by the server, may be *deterministic* or *randomized* – note that in the latter case the server is burdened with producing (good) randomness and using it correctly.

Despite the considerable advances of the past few years, there remain some important open questions regarding basic properties of UE. In terms of security, various features have been added to protect against stronger adversaries. Yet it is not obvious what are the realistic and optimal security goals of UE and whether they have been achieved. In terms of efficiency, we only have a few concrete schemes to compare. As may be expected, schemes with stronger security are generally more expensive but it remains unclear whether this cost is necessary. In this paper we make contributions to both of these fundamental questions by defining **new and stronger security properties** and showing that these can be achieved with **more efficient concrete UE schemes**.

Security. The main security properties that one would expect from updatable encryption are by now well studied; however the breadth of information that is possible to protect in this context is more subtle than at first glance. Consider, for example, a journalist who stores a contact list with a cloud storage provider. At some point, the storage is compromised and an adversary recovers the ciphertexts. At this point, it may be important that the cryptography does not reveal which of the contacts are recent, and which are old. That is, it must be hard to decide if some ciphertext was recently created, or if it has been updated from a ciphertext stored in an earlier epoch.

So how do we define realistic adversaries in this environment? A natu-

[1]Note that Boneh et al. [[BLMR15], § Definition 7.6] use ciphertext-independence to mean that the updated ciphertext should have the same distribution as a fresh ciphertext (i.e. independent of the ciphertext in the previous epoch) – we follow the nomenclature of Lehmann and Tackmann [LT18a].

ral first step for security in updatable encryption is *confidentiality* of ciphertexts – given a single ciphertext, the adversarial server should not be able to determine anything about the underlying plaintext. The security model here must take into account that this adversary could be in possession of a number of prior keys or update tokens, and snapshot access to the storage database in different epochs. The next step is to consider *unlinkability* between different epochs arising from the ciphertext update procedure: given a ciphertext for the current epoch, the adversary should not be able to tell which ciphertext (that existed in the previous epoch) a current ciphertext was updated from. Both of these properties can be naturally extended to chosen-ciphertext (CCA) security via provision of a decryption oracle.

These steps have been taken by prior work, but unfortunately even a combination of these properties is not enough to defend against our motivating example. Previous security definitions cannot guarantee that the adversary is unable to distinguish between a ciphertext new in the current epoch and an updated ciphertext from an earlier epoch. We give a single new security property that captures this requirement and implies the notions given in prior work. Therefore we believe that this definition is *the natural confidentiality property* that is required for updatable encryption.

An additional factor to consider is integrity: the user should be confident that their ciphertexts have not been modified by the adversarial server. While prior work has shown how to define and achieve integrity in the context of updatable encryption, a composition result of the style given by Bellare and Namprempre for symmetric encryption [BN08] – the combination of CPA security and integrity of ciphertexts gives CCA security – has been missing. We close this gap.

Efficiency and Functionality. Although UE is by definition a form of symmetric key cryptography, techniques from asymmetric cryptography appear to be needed to achieve the required functionality in a sensible fashion. All of the previous known schemes with security proofs use exponentiation in both the encryption and update functions, even for those with limited security properties. Since a modern database may contain large numbers of files, efficiency is critical both for clients who will have to encrypt plaintexts initially and for servers who will have to update ciphertexts for all of their users.

To bridge the gap between the academic literature and deployments of encrypted outsourced storage, *it is crucial to design fast schemes*. We present three novel UE schemes that not only satisfy our strong security definitions (CCA and ciphertext integrity), but in the vast majority of application scenarios are also at least twice as fast (in terms of computation each message block) as any previous scheme with comparable security level.

The *ciphertext expansion* of a scheme says how much the size of a ciphertext grows compared to the size of the message. For a cloud server that stores vast numbers of files, it is naturally crucial to minimize the ciphertext expansion rate. It is also desirable to construct UE schemes that can encrypt *arbitrarily large files*, since a client might want to upload media files such as images or videos. Prior schemes that have achieved these two properties have only been secure in comparatively weak models. Our construction suitable for long messages – enabling encryption of arbitrarily large files with almost no ciphertext expansion – is secure in our strong sense and is thus the first to bridge this gap.

1.1 Related Work

1.1.1 Security Models for UE.

We regard the sequential, epoch-based corruption model of Lehmann and Tackmann [LT18a] (LT18) as the most suitable execution environment to capture the threats in updatable encryption. In this model, the adversary advances to the next epoch via an oracle query. It can choose to submit its (single) challenge when it pleases, and it can later update the challenge ciphertext to the 'current' epoch. Further, the adversary is allowed to adaptively corrupt epoch (i.e. file encryption) keys and update tokens at any point in the game: only at the end of the adversary's execution does the challenger determine whether a trivial win has been made possible by some combination of the corruption queries and the challenge.

LT18 introduced two notions: IND-ENC asks the adversary to submit two plaintexts and distinguish the resulting ciphertext, while possibly having corrupted tokens (but of course not keys) linking this challenge ciphertext to prior or later epochs. Further, they introduced IND-UPD: the adversary provides two ciphertexts that it received via regular encryption-oracle queries in the previous epoch, and has to work out which one has been updated. They observed[2] that plaintext information can be leaked not only through the encryption procedure, but also via updates. For schemes with deterministic updates, the adversary would trivially win if it could acquire the update token that takes the adversarially-provided ciphertexts into the challenge epoch, hence the definition for this setting, named detIND-UPD, is different from that for the randomized setting, named randIND-UPD.

LT18's IND-UPD definition was not the first approach to formalizing the desirable property of *unlinkability* of ciphertexts, which attempts to specify that given two already-updated ciphertexts, the adversary cannot tell if the plaintext is the same. Indeed EPRS (UP-REENC) and later KLR19 (UP-REENC-CCA) also considered this problem, in the ciphertext-dependent update and CCA-secure setting respectively. KLR19 [[KLR19a], § Appendix A] stated that "an even stronger notion [than IND-UPD] might be desirable: namely that fresh and re-encrypted ciphertexts are indistinguishable... which is not guaranteed by UP-REENC" – we will answer this open question later on in our paper.

In the full version of their work [BLMR15], BLMR introduced a security definition for UE denoted update – an extension of a model of symmetric proxy re-encryption. This non-sequential definition is considerably less adaptive than the later work of LT18, since the adversary's key/token corruption queries and ciphertext update queries are very limited. Further, they only considered schemes with deterministic update algorithms.

EPRS [EPRS17a] provided (non-sequential) definitions for updatable authenticated encryption, in the ciphertext-dependent setting. Their work (inherently) covered CCA security and ciphertext integrity (CTXT). These definitions were ambiguous regarding adaptivity: these issues have since been fixed in the full version [EPRS17b].

KLR19 attempted to provide stronger security guarantees for ciphertext-independent UE than LT18, concentrating on chosen-ciphertext secu-

[2]The proceedings and full versions of LT18 stated that "IND-ENC *security cannot guarantee anything about the security of updates. In fact, a scheme where the update algorithm* UE.Upd *includes all the old ciphertexts* $C_0, \ldots, C_e$ *in the updated ciphertext* C_{e+1} *could be considered* IND-ENC *secure, but clearly lose all security if a single old key gets compromised.*" This line of argument is flawed, and in fact IND-ENC rules out schemes of this form: encryptions were always fresh at some point. This claim was corrected and clarified in a June 2019 presentation by the first author [Leh19], and further elaborated on in an update to the full version in December 2019 [LT18b].

rity (and the weaker replayable CCA) in addition to integrity of plaintexts and ciphertexts. We revisit these definitions later on, and show how a small modification to their INT-CTXT game gives rise to natural composition results.

In practice, LT18's randIND-UPD definition insists that the ciphertext update procedure Upd requires the server to generate randomness for updating each ciphertext. Further, a scheme meeting both IND-ENC and IND-UPD can still leak the epoch in which the file was uploaded (the 'age' of the ciphertext). While it is arguable that metadata is inherent in outsourced storage, the use of updatable encryption is for high-security applications, and it would not be infeasible to design a system that does not reveal meta-data, which is clearly impossible if the underlying cryptosystem reveals the meta-data.

Recent work by Jarecki et al. [JKR19] considers the key wrapping entity as a separate entity from the data owner or the storage server. While this approach seems promising, their security model is considerably weaker than those considered in our work or the papers already mentioned in this section: the adversary must choose whether to corrupt the key management server (and get the epoch key) or the storage server (and get the update token) for each epoch, and thus it cannot dynamically corrupt earlier keys or tokens at a later stage.

1.1.2 Constructions of Ciphertext-Independent UE

The initial description of updatable encryption by Boneh et al. [BLMR13] was motivated by providing a symmetric-key version of proxy re-encryption (see below). BLMR imagined doing this in a symmetric manner, where each epoch is simply one period in which re-encryption (rotation) has occurred. Their resulting scheme, denoted BLMR, deploys a key-homomorphic PRF, yet the nonce attached to a ciphertext ensures that IND-UPD cannot be met (the scheme pre-dates the IND-UPD notion).

The symmetric-Elgamal-based scheme of LT18, named RISE, uses a randomized update algorithm and is proven to meet randIND-UPD and IND-ENC under DDH. These proofs entail a seemingly unavoidable loss – a cubic term in the total number of epochs – our results also have this factor. LT18 also presented an extended version of the scheme by BLMR, denoted BLMR+, where the nonce is encrypted: they showed that this scheme meets a weak version of IND-UPD called weakIND-UPD, in which if the adversary corrupts the token that links the challenge epoch to the epoch immediately after then a trivial win condition is triggered.

The aim of KLR19 was to achieve stronger security than BLMR, EPRS and LT18 in the ciphertext-independent setting: in particular CCA security and integrity protection. They observed that the structure of RISE ensures that ciphertext integrity cannot be achieved: access to just one update token allows the storage provider to construct ciphertexts of messages of its choice. Their generic constructions, based on encrypt-and-MAC and the Naor-Yung paradigm, are strictly less efficient than RISE. We show how to achieve CCA security and integrity protections with novel schemes that are comparably efficient with RISE.

1.1.3 Related Primitives

Proxy re-encryption (PRE) allows a ciphertext that is decryptable by some secret key to be re-encrypted such that it can be decrypted by some other key. Security models for PRE are closer to those for encryption than the strictly sequential outsourced-storage-centric models for UE, and as observed by Lehmann and Tackmann [LT18a] the concepts of allowable corruptions and trivial wins for UE need considerable care when translating to the (more general) PRE setting. Unlinkability is not necessarily desired in PRE – updating the entire ciphertext may not be essential for a PRE scheme to be deemed secure – thus even after conversion to the symmetric setting, prior schemes [AFGH05, CH07] cannot meet the indistinguishability requirements that we ask of UE schemes. Recent works by Lee [Lee17] and Davidson et al. [DDLM19] have highlighted the links between the work of BLMR and EPRS and PRE, and in particular the second work gives a public-key variant of the (sequential) IND-UPD definition of LT18. Myers and Shull [MS18] presented security models for hybrid proxy re-encryption, and gave a single-challenge version of the UP-IND notion of EPRS. While the models are subtly different, the techniques for achieving secure UE and PRE are often similar: in particular rotating keys via exponentiation to some simple function of old and new key (RISE is essentially a combination of Blaze et al.'s symmetric version of ElGamal [BBS98] and ciphertext randomization). Further, the symmetric-key PRE scheme of Sakurai et al. [SNS17] is at a high level similar to SHINE (their all-

or-nothing-transform as an inner layer essentially serves the same purpose as the ideal cipher in SHINE), but in a security model that does not allow dynamic corruptions. Their approach includes – this natural approach is somewhat similar to the schemes that we introduce later in the paper.

Tokenization schemes aim to protect short secrets, such as credit card numbers, using deterministic encryption and deterministic updates: this line of work reflects the PCI DSS standard [Cou18] for the payment card industry. Provable security of such schemes was initially explored by Diaz-Santiago et al. [DRC14] and extended to the updatable setting by Cachin et al. [CCFL17]. While much of the formalism in the model of Cachin et al. has been used in recent works on UE (in particular the epoch-based corruption model), the requirements on ciphertext indistinguishability are stronger in the UE setting, where we expect probabilistic encryption of (potentially large) files.

1.2 Contributions

Our first major contribution is defining the $\mathsf{xxIND\text{-}UE\text{-}atk}$ security notion for updatable encryption schemes, for $(\mathsf{xx}, \mathsf{atk}) \in \{(\mathsf{det}, \mathsf{CPA}), (\mathsf{rand}, \mathsf{CPA}), (\mathsf{det}, \mathsf{CCA})\}$, and comprehensively analyzing its relation to other, existing[3] security notions ($\mathsf{xxIND\text{-}ENC\text{-}atk}$, $\mathsf{xxIND\text{-}UPD\text{-}atk}$). Our single definition requires that ciphertexts output by the encryption algorithm are indistinguishable from ciphertexts output by the update algorithm. We show that our new notion is strictly stronger even than combinations of prior notions, both in the randomized- and deterministic-update settings under chosen-plaintext attack and chosen-ciphertext attack. This not only gives us the unlinkability desired by prior works, but also answers the open question posed by KLR19 mentioned on page . Fig. 18 describes the relationship between our new notion $\mathsf{xxIND\text{-}UE\text{-}atk}$ and prior notions.

After a slight tweak to KLR19's definitions for CTXT and CCA, we show the following generic composition result for ciphertext independent updatable encryption schemes: CPA security plus CTXT security implies CCA security. Combining this result with the relations from $\mathsf{detIND\text{-}UE\text{-}atk}$

[3]The notions $\mathsf{IND\text{-}ENC}$, $\mathsf{randIND\text{-}UPD}$ and $\mathsf{detIND\text{-}UPD}$ (which we denote as $\mathsf{IND\text{-}ENC\text{-}CPA}$, $\mathsf{randIND\text{-}UPD\text{-}CPA}$ and $\mathsf{detIND\text{-}UPD\text{-}CPA}$, resp.) are from LT18. The notions $\mathsf{UP\text{-}IND\text{-}CCA}$ and $\mathsf{UP\text{-}REENC\text{-}CCA}$ ($\mathsf{detIND\text{-}ENC\text{-}CCA}$ and $\mathsf{detIND\text{-}UPD\text{-}CCA}$, resp.) are from KLR19. LT18 and KLR19 both build upon the definitions given by EPRS.

above, we show that the combination of $\mathsf{detIND\text{-}UE\text{-}CPA}$ and $\mathsf{INT\text{-}CTXT}$ yields $\mathsf{detIND\text{-}yy\text{-}CCA}$ *for all* $\mathsf{yy} \in \{\mathsf{UE}, \mathsf{ENC}, \mathsf{UPD}\}$.

Our second major contribution is in designing a new, fast updatable encryption scheme SHINE. Our scheme is based on a random-looking permutation combined with the exponentiation map in a cyclic group, and comes in a number of variants: SHINE0, MirrorSHINE and OCBSHINE, for small messages, medium-sized messages and arbitrarily large messages respectively. In Fig. 1, we provide a comparison of security, ciphertext expansion and efficiency between our new schemes and those from prior literature. We also further the understanding of schemes with deterministic update mechanisms. In particular, we identify the properties that are necessary of such schemes to meet a generalized version of our detIND-UE-atk notion. Another important contribution is that we further improve on the existing epoch insulation techniques that have been used to create proofs of security in the strong corruption environment we pursue. These have been shown to be very useful for studying updatable encryption schemes, and we expect our new techniques to be useful in the future.

1.3 Further Discussion

We have had to make a number of practical design decisions for our new UE scheme SHINE. The main idea is to permute the (combination of nonce and) message and then exponentiate the resulting value, with different mechanisms for enforcing ciphertext integrity depending on the flavor that is being used (which is in turn defined by the desired message length). In this subsection we give some motivation for why we believe that these choices are reasonable.

Deterministic updates. Since we will require indistinguishability of ciphertexts, we know that the UE encryption algorithm should be randomized. The update algorithm may or may not be randomized, however. All known schemes indicate that randomized updates are more expensive than deterministic updates, but there is a small, well-understood security loss in moving to deterministic updates: an adversary with an update token in an appropriate epoch can trivially distinguish between an update of a known ciphertext and other ciphertexts in the next epoch. As a result, in the

	IND	INT	$\lvert M\rvert$	$\lvert C\rvert$	Enc (Upd)
BLMR [BLMR13]	(det, ENC, CPA)	✘	$l\lvert\mathbb{G}\rvert$	$(l+1)\lvert\mathbb{G}\rvert$	$l\mathbf{E}$
BLMR+ [BLMR13, LT18a]	(weak, UE, CPA)	✘	$l\lvert\mathbb{G}\rvert$	$(l+1)\lvert\mathbb{G}\rvert$	$l\mathbf{E}$
RISE [LT18a]	(rand, UE, CPA)	✘	$1\lvert\mathbb{G}\rvert$	$2\lvert\mathbb{G}\rvert$	$2\mathbf{E}$
SHINE0[CPA] § 5.1.1	(det, UE, CPA)	✘	$(1-\gamma)\lvert\mathbb{G}\rvert$	$1\lvert\mathbb{G}\rvert$	$1\mathbf{E}$
NYUE [KLR19a]	(rand, ENC, RCCA) (rand, UPD, RCCA)	✘	$1\lvert\mathbb{G}_1\rvert$	$(34\lvert\mathbb{G}_1\rvert, 34\lvert\mathbb{G}_2\rvert)$	$(60\mathbf{E}, 70\mathbf{E})$
NYUAE [KLR19a]	(rand, ENC, RCCA) (rand, UPD, RCCA)	PTXT	$1\lvert\mathbb{G}_1\rvert$	$(58\lvert\mathbb{G}_1\rvert, 44\lvert\mathbb{G}_2\rvert)$	$(110\mathbf{E}, 90\mathbf{E})$
E&M [KLR19a]	(det, ENC, CCA) (det, UPD, CCA)	CTXT	$1\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$	$3\mathbf{E}$
SHINE0 § 5.1.1	(det, UE, CCA)	CTXT	$(1-2\gamma)\lvert\mathbb{G}\rvert$	$1\lvert\mathbb{G}\rvert$	$1\mathbf{E}$
MirrorSHINE § 5.1.2	(det, UE, CCA)	CTXT	$(1-\gamma)\lvert\mathbb{G}\rvert$	$2\lvert\mathbb{G}\rvert$	$2\mathbf{E}$
OCBSHINE § 5.1.3	(det, UE, CCA)	CTXT	$l\lvert\mathbb{G}\rvert$	$(l+2)\lvert\mathbb{G}\rvert$	$(l+2)\mathbf{E}$

Figure 1: Comparison of security, ciphertext expansion and efficiency for updatable encryption schemes. (xx, yy, atk) represents the best possible xxIND-yy-atk notion that each scheme can achieve. **E** represents the cost of an exponentiation, for encryption Enc and ciphertext update Upd. γ represents the bit-size of the used nonce as a proportion of the group element bit-size. For NYUE and NYUAE, size/cost is in pairing groups $\mathbb{G}_1, \mathbb{G}_2$. SHINE0[CPA] is SHINE0 with a zero-length integrity tag. BLMR, BLMR+ and OCBSHINE support encryption of arbitrary size messages (of l blocks), with $\lvert M\rvert \approx l\lvert\mathbb{G}\rvert$.

detIND-UE-CPA case the adversary is only forbidden from obtaining one token compared to randIND-UE-CPA. Furthermore, UE schemes with randomized updates cannot achieve CTXT and CCA security, which is possible for the deterministic-update setting. We believe that the minor CPA security loss is a small price to pay for stronger security (CTXT and CCA) and efficiency gain, in particular to reduce computations in the UE encryption and update algorithms and also improve ciphertext expansion.

Bi-directional key updates. In principle, the token used to update ciphertexts need not be sufficient to derive the new key from the old key. But for every known practical scheme, this derivation is indeed easy. Moreover, for every known practical scheme including ours, the *old* key can be derived from the *new* key (and token). While *uni-directional* update algorithms are desirable, constructing efficient protocols has so far been elusive: this has technical implications for how security notions are defined.

Limited number of epochs. In many applications that we would like to consider, the user of the storage service will control when updates occur (perhaps when an employee with access to key material leaves the organisation, or if an employee loses a key-holding device): this indicates that the total number of key rotations in the lifetime of a storage system might be numbered in the thousands, and in particular could be considerably smaller than the number of outsourced files.

1.4 Organization

After introducing syntax and preliminaries in Section 2, we detail the necessary formalism for security modeling in updatable encryption in Section 3. In Section 4 we define our new confidentiality property IND-UE and show how it implies prior notions; in Section 5 we give our new scheme, SHINE, including intuition behind its security analysis and implementation options. We show the security properties that can be met, in our new framework, by prior work schemes BLMR and RISE (LT18) in Section B and C respectively.

2 Preliminaries

Pseudocode **return** $b' \overset{?}{=} b$ is used as shorthand for **if** $b' = b$ **then return** 1 // **else return** 0, with an output of 1 indicating adversarial success. We use the concrete security framework, defining adversarial advantage as probability of success in the security game, and avoid statements of security with respect to security notions. In the cases where we wish to indicate that notion A implies notion B (for some fixed primitive), i.e. an adversary's advantage against B carries over to an advantage against A, we show this by bounding these probabilities.

2.1 Hardness Assumptions

For the definition of DDH, CDH and later on, we assume the existence of a group-generation algorithm that is parameterized by λ and outputs a cyclic group $\mathbb{G}$ of order q (where q is of length λ bits) and a generator g. We adapt the definition of pseudorandom functions from Boneh et al. [BLMR13].

Definition 1 (DDH)**.** Fix a cyclic group $\mathbb{G}$ of prime order q with generator g. The advantage of an algorithm $\mathcal{A}$ solving the *Decision Diffie-Hellman (*DDH*)* problem for $\mathbb{G}$ and g is

$$\mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{A}}(\lambda) = \left|\mathbf{Pr}[\mathbf{Exp}^{\mathsf{DDH\text{-}1}}_{\mathbb{G},\,\mathcal{A}}(\lambda) = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{DDH\text{-}0}}_{\mathbb{G},\,\mathcal{A}}(\lambda) = 1]\right|$$

where the experiment $\mathbf{Exp}^{\mathsf{DDH\text{-}b}}_{\mathbb{G},\,\mathcal{A}}$ is given in Fig. 2.

Definition 2 (CDH)**.** Fix a cyclic group $\mathbb{G}$ of prime order q with generator g. The advantage of an algorithm $\mathcal{A}$ solving the *Computational Diffie-Hellman (*CDH*)* problem for $\mathbb{G}$ and g is

$$\mathbf{Adv}^{\mathsf{CDH}}_{\mathbb{G},\mathcal{A}}(\lambda) = \mathbf{Pr}[\mathbf{Exp}^{\mathsf{CDH}}_{\mathbb{G},\,\mathcal{A}}(\lambda) = 1]$$

where the experiment $\mathbf{Exp}^{\mathsf{CDH}}_{\mathbb{G},\,\mathcal{A}}$ is given in Fig. 3.

Definition 3 (PRF)**.** Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be an efficiently computable function, where $\mathcal{K}$ is called the key space, $\mathcal{X}$ is the domain, and $\mathcal{Y}$ is the range. The PRF advantage for $\mathcal{A}$ against F is given by

$$\mathbf{Adv}^{\mathsf{PRF}}_{F,\,\mathcal{A}}(\lambda) = \left|\mathbf{Pr}[\mathbf{Exp}^{\mathsf{PRF\text{-}1}}_{F,\,\mathcal{A}}(\lambda) = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{PRF\text{-}0}}_{F,\,\mathcal{A}}(\lambda) = 1]\right|$$

where the experiment $\mathbf{Exp}^{\mathsf{PRF\text{-}b}}_{F,\,\mathcal{A}}$ is given in Fig. 4.

```
$\mathbf{Exp}^{\mathsf{DDH\text{-}b}}_{\mathbb{G},\mathcal{A}}(\lambda)$
1 : $x, y, r \xleftarrow{\$} \mathbb{Z}_q$
2 : $X \leftarrow g^x; Y \leftarrow g^y$
3 : if $\mathrm{b} = 0$
4 :     $Z \leftarrow g^{xy}$
5 : else
6 :     $Z \leftarrow g^r$
7 : $\mathrm{b}' \leftarrow \mathcal{A}(g, X, Y, Z)$
8 : return $\mathrm{b}'$
```

Figure 2: DDH experiment $\mathbf{Exp}^{\mathsf{DDH\text{-}b}}_{\mathbb{G},\mathcal{A}}$

```
$\mathbf{Exp}^{\mathsf{CDH}}_{\mathbb{G},\mathcal{A}}(\lambda)$
1 : $x, y \xleftarrow{\$} \mathbb{Z}_q$
2 : $X \leftarrow g^x$
3 : $Y \leftarrow g^y$
4 : $Z \leftarrow \mathcal{A}(g, X, Y)$
5 : if $Z = g^{xy}$
6 :     return 1
7 : else
8 :     return 0
```

Figure 3: CDH experiment $\mathbf{Exp}^{\mathsf{CDH}}_{\mathbb{G},\mathcal{A}}$

```
$\mathbf{Exp}^{\mathsf{PRF\text{-}b}}_{F,\mathcal{A}}(\lambda)$
1 : if $\mathrm{b} = 0$
2 :     $\mathrm{k} \xleftarrow{\$} \mathcal{K}$
3 :     $f(\cdot) \leftarrow F(\mathrm{k}, \cdot)$
4 : else
5 :     $f(\cdot) \xleftarrow{\$} \{f : \mathcal{X} \to \mathcal{Y}\}$
6 : $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.f}()$
7 : return $\mathrm{b}'$

$\mathcal{O}.f(x)$
8 : if $x \notin \mathcal{X}$
9 :     return $\perp$
10 : else
11 :     return $f(x)$
```

Figure 4: PRF experiment $\mathbf{Exp}^{\mathsf{PRF\text{-}b}}_{F,\mathcal{A}}$

2.2 Updatable Encryption

We follow the syntax of prior work [KLR19a], defining an Updatable Encryption (UE) scheme as a tuple of algorithms $\{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ that operate in epochs, these algorithms are described in Fig. 5. A scheme is defined over some plaintext space $\mathcal{MS}$, ciphertext space $\mathcal{CS}$, key space $\mathcal{KS}$ and token space $\mathcal{TS}$. We specify integer $n+1$ as the (total) number of epochs over which a UE scheme can operate, though this is only for proof purposes. Correctness [KLR19a] is defined as expected: fresh encryptions and updated ciphertexts should decrypt to the correct mes-

Algorithm	Rand/Det	Input	Output	Syntax
Key Gen: $\mathsf{UE.KG}$	Rand	λ	k_e	$k_e \xleftarrow{\$} \mathsf{UE.KG}(\lambda)$
Token Gen: $\mathsf{UE.TG}$	Det	k_e, k_{e+1}	Δ_{e+1}	$\Delta_{e+1} \leftarrow \mathsf{UE.TG}(k_e, k_{e+1})$
Encryption: $\mathsf{UE.Enc}$	Rand	M, k_e	C_e	$C_e \xleftarrow{\$} \mathsf{UE.Enc}(k_e, M)$
Decryption: $\mathsf{UE.Dec}$	Det	C_e, k_e	M' or $\perp$	$\{M'/\perp\} \leftarrow \mathsf{UE.Dec}(k_e, C_e)$
Update Ctxt: $\mathsf{UE.Upd}$	Rand/det	C_e, Δ_{e+1}	C_{e+1}	$C_{e+1} \xleftarrow{\$} \mathsf{UE.Upd}(\Delta_{e+1}, C_e)$

Figure 5: Syntax of algorithms defining an Updatable Encryption scheme UE.

sage under the appropriate epoch key. In contrast to prior work, we only consider deterministic token generation algorithms – all schemes in prior literature and our schemes allow the token to be produced deterministically from epoch keys alone.

In addition to enabling ciphertext updates, in many schemes the token allows ciphertexts to be 'downgraded': performing some analog of the $\mathsf{UE.Upd}$ operation on a ciphertext C created in (or updated to) epoch e yields a valid ciphertext in epoch e-1. Such a scheme is said to have *bi-directional ciphertext updates*[4]. Furthermore, for many constructions, the token additionally enables key derivation, given one adjacent key. If this can be done in both directions – i.e. knowledge of k_e and Δ_{e+1} allows derivation of k_{e+1} AND knowledge of k_{e+1} and Δ_{e+1} allows derivation of k_e – then such schemes are referred to by LT18 as having *bi-directional key updates*. If such derivation is only possible in one 'direction' then the scheme is said to have *uni-directional key updates*. Much of the prior literature on updatable encryption has distinguished these notions: we stress that all schemes and definitions of security considered in this paper have bi-directional key updates and bi-directional ciphertext updates.

[4]For example if the Upd procedure exponentiates all ciphertext components using the token, as done in SHINE, then Upd itself is sufficient to demonstrate this property.

3 Security Models for Updatable Encryption

We consider a number of indistinguishability-based confidentiality games and integrity games for assessing security of updatable encryption schemes. The environment provided by the challenger attempts to give as much power as possible to adversary $\mathcal{A}$. The adversary may call for a number of oracles, and after $\mathcal{A}$ has finished running the challenger computes whether or not any of the actions enabled a trivial win. The available oracles are described in Fig. 6. An overview of the oracles $\mathcal{A}$ has access to in each security game is provided in Fig. 7.

Confidentiality. A generic representation of all confidentiality games described in this paper is detailed in Fig. 8. The current epoch is advanced by an adversarial call to $\mathcal{O}.\mathsf{Next}$ – simulating $\mathsf{UE.KG}$ and $\mathsf{UE.TG}$ – and keys and tokens (for the current or any prior epoch) can be corrupted via $\mathcal{O}.\mathsf{Corr}$. The adversary can encrypt arbitrary messages via $\mathcal{O}.\mathsf{Enc}$, and update these 'non-challenge' ciphertexts via $\mathcal{O}.\mathsf{Upd}$. In CCA games, the adversary can additionally call decryption oracle $\mathcal{O}.\mathsf{Dec}$ (with some natural restrictions to prevent trivial wins). At some point $\mathcal{A}$ makes its challenge by providing two inputs, and receives the challenge ciphertext – and in later epochs can receive an updated version by calling $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$ (computing this value is actually done by $\mathcal{O}.\mathsf{Next}$, a call to $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$ returns it). $\mathcal{A}$ can then interact with its other oracles again, and eventually outputs its guess bit. The flag phase tracks whether or not $\mathcal{A}$ has made its challenge, and we always give the epoch in which the challenge happens a special identifier $\tilde{\mathsf{e}}$. If $\mathcal{A}$ makes any action that would lead to a trivial win, the flag twf is set as 1 and $\mathcal{A}$'s output is discarded and replaced by a random bit. We follow the bookkeeping techniques of LT18 and KLR19, using the following sets to track ciphertexts and their updates that can be known to the adversary.

- $\mathcal{L}$: List of non-challenge ciphertexts (from $\mathcal{O}.\mathsf{Enc}$ or $\mathcal{O}.\mathsf{Upd}$) with entries of form $(\mathsf{c}, \mathrm{C}, \mathsf{e})$, where query identifier c is a counter incremented with each new $\mathcal{O}.\mathsf{Enc}$ query.
- $\tilde{\mathcal{L}}$: List of updated challenge ciphertexts (created via $\mathcal{O}.\mathsf{Next}$, received by adversary via $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$), with entries of form $(\tilde{\mathrm{C}}, \mathsf{e})$.

Further, we use the following lists that track epochs only.

```
$\mathcal{O}.\mathsf{Enc}(\mathrm{M})$
  $\mathrm{C} \leftarrow \mathsf{UE.Enc}(\mathrm{k_e}, \mathrm{M})$
  $\mathsf{c} \leftarrow \mathsf{c} + 1$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}, \mathsf{e})\}$
  $\mathbf{return}\ \mathrm{C}$

$\mathcal{O}.\mathsf{Dec}(\mathrm{C})$
  $\underline{\mathsf{twf} \leftarrow 1\ \mathbf{if}}$
      $\mathsf{phase} = 1\ \mathbf{and}\ \mathrm{C} \in \tilde{\mathcal{L}}$
  $\mathrm{M}'\ \mathbf{or}\ \perp \leftarrow \mathsf{UE.Dec}(\mathrm{k_e}, \mathrm{C})$
  $\mathbf{return}\ \mathrm{M}'\ \mathbf{or}\ \perp$

$\mathcal{O}.\mathsf{Next}()$
  $\mathsf{e} \leftarrow \mathsf{e} + 1$
  $\mathrm{k_e} \xleftarrow{\$} \mathsf{UE.KG}(\lambda)$
  $\Delta_\mathsf{e} \xleftarrow{\$} \mathsf{UE.TG}(\mathrm{k_{e\text{-}1}}, \mathrm{k_e})$
  $\mathbf{if}\ \mathsf{phase} = 1$
      $\tilde{\mathrm{C}}_\mathsf{e} \leftarrow \mathsf{UE.Upd}(\Delta_\mathsf{e}, \tilde{\mathrm{C}}_{\mathsf{e\text{-}1}})$

$\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}-1})$
  $\mathbf{if}\ (j, \mathrm{C}_{\mathsf{e}-1}, \mathsf{e}-1) \notin \mathcal{L}$
      $\mathbf{return}\ \perp$
  $\mathrm{C}_\mathsf{e} \leftarrow \mathsf{UE.Upd}(\Delta_\mathsf{e}, \mathrm{C}_{\mathsf{e}-1})$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(j, \mathrm{C}_\mathsf{e}, \mathsf{e})\}$
  $\mathbf{return}\ \mathrm{C}_\mathsf{e}$
```

```
$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$
  $\mathbf{if}\ \hat{\mathsf{e}} > \mathsf{e}$
      $\mathbf{return}\ \perp$
  $\mathbf{if}\ \mathsf{inp} = \mathsf{key}$
      $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
      $\mathbf{return}\ \mathrm{k}_{\hat{\mathsf{e}}}$
  $\mathbf{if}\ \mathsf{inp} = \mathsf{token}$
      $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
      $\mathbf{return}\ \Delta_{\hat{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
  $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{\mathrm{C}}_\mathsf{e}, \mathsf{e})\}$
  $\mathbf{return}\ \tilde{\mathrm{C}}_\mathsf{e}$

$\mathcal{O}.\mathsf{Try}(\tilde{\mathrm{C}})$
  $\boxed{\mathbf{if}\ \mathsf{phase} = 1}$
      $\boxed{\mathbf{return}\ \perp}$
  $\boxed{\mathsf{phase} \leftarrow 1}$
  $\underline{\mathsf{twf} \leftarrow 1\ \mathbf{if}}$
      $\mathsf{e} \in \mathcal{K}^*\ \mathbf{or}\ \tilde{\mathrm{C}} \in \tilde{\mathcal{L}}^*$
  $\mathrm{M}'\ \mathbf{or}\ \perp \leftarrow \mathsf{UE.Dec}(\mathrm{k_e}, \tilde{\mathrm{C}})$
  $\mathbf{if}\ \mathrm{M}' \neq \perp$
      $\mathsf{win} \leftarrow 1$
```

Figure 6: Oracles in security games for updatable encryption. The boxed lines in $\mathcal{O}$.Try only apply to INT-CTXT$^\mathsf{s}$: in this game the adversary is allowed to query the $\mathcal{O}$.Try oracle only once. Computing $\tilde{\mathcal{L}}^*$ is discussed in Section 3.2.

Notion	$\mathcal{O}$.Enc	$\mathcal{O}$.Dec	$\mathcal{O}$.Next	$\mathcal{O}$.Upd	$\mathcal{O}$.Corr	$\mathcal{O}$.Upd$\tilde{\mathrm{C}}$	$\mathcal{O}$.Try
detIND-yy-CPA	✓	×	✓	✓	✓	✓	×
randIND-yy-CPA	✓	×	✓	✓	✓	✓	×
detIND-yy-CCA	✓	✓	✓	✓	✓	✓	×
INT-CTXT	✓	×	✓	✓	✓	×	✓

Figure 7: Oracles the adversary is allowed to query in different security games, where yy $\in \{\mathsf{ENC}, \mathsf{UPD}, \mathsf{UE}\}$. $\times$ indicates the adversary does not have access to the corresponding oracle,✓ indicates the adversary has access to the corresponding oracle.

- $\mathcal{C}$: List of epochs in which adversary learned updated version of challenge ciphertext (via CHALL or $\mathcal{O}$.Upd$\tilde{\mathrm{C}}$).
- $\mathcal{K}$: List of epochs in which the adversary corrupted the encryption key.
- $\mathcal{T}$: List of epochs in which the adversary corrupted the update token.

All experiments necessarily maintain some state, but we omit this for readability reasons. The challenger's state is $\mathbf{S} \leftarrow \{\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T}\}$, and the system state in the current epoch is given by st $\leftarrow (\mathrm{k}_{\mathsf{e}}, \Delta_{\mathsf{e}}, \mathbf{S}, \mathsf{e})$.

An at-a-glance overview of CHALL for various security definitions is given in Fig. 9. For security games such as LT18's IND-UPD notion, where the adversary must submit as its challenge two ciphertexts (that it received from $\mathcal{O}$.Enc) and one is updated, the game must also track in which epochs the adversary has updates of these ciphertexts. We will later specify a version of our new xxIND-UE-atk notion that allows the adversary to submit a ciphertext that existed in any epoch prior to the challenge epoch, not just the one immediately before: this introduces some additional bookkeeping (discussed further in Section 3.2).

A note on nomenclature: the adversary can make its challenge query to receive *the challenge ciphertext*, and then acquire *updates of the challenge ciphertext* via calls to $\mathcal{O}$.Upd$\tilde{\mathrm{C}}$, and additionally it can calculate *challenge-equal ciphertexts* via applying tokens it gets via $\mathcal{O}$.Corr queries.

```
$\mathbf{Exp}^{\mathsf{xxIND\text{-}yy\text{-}atk\text{-}b}}_{\mathsf{UE},\ \mathcal{A}}(\lambda)$
1 :  do Setup
2 :  $\mathsf{CHALL} \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},(\mathcal{O}.\mathsf{Dec}),\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
3 :  $\mathsf{phase} \leftarrow 1; \tilde{\mathsf{e}} \leftarrow \mathsf{e}$
4 :  Create $\tilde{\mathrm{C}}$ with $\mathsf{CHALL}$; $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{\mathrm{C}}_{\mathsf{e}}, \mathsf{e})\}$
5 :  $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},(\mathcal{O}.\mathsf{Dec}),\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}})$
6 :  $\underline{\mathsf{twf} \leftarrow 1 \textbf{ if}}$
7 :      $\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$ or
8 :      $\mathsf{xx} = \mathsf{det}$ and $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$
9 :  if $\mathsf{twf} = 1$
10 :     $\mathrm{b}' \xleftarrow{\$} \{0, 1\}$
11 :  return $\mathrm{b}'$
```

```
$\mathbf{Setup}(\lambda)$
12 :  $\mathrm{k}_0 \leftarrow \mathsf{UE.KG}(\lambda)$
13 :  $\Delta_0 \leftarrow \perp$
14 :  $\mathsf{e}, \mathsf{c} \leftarrow 0$
15 :  $\mathsf{phase}, \mathsf{twf} \leftarrow 0$
16 :  $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$
```

Figure 8: Generic description of confidentiality experiment $\mathbf{Exp}^{\mathsf{xxIND\text{-}yy\text{-}atk\text{-}b}}_{\mathsf{UE},\ \mathcal{A}}$ for scheme UE and adversary $\mathcal{A}$, for $\mathsf{xx} \in \{\mathsf{det}, \mathsf{rand}\}$, $\mathsf{yy} \in \{\mathsf{ENC}, \mathsf{UPD}, \mathsf{UE}\}$ and $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA}\}$. We do not consider (and thus do not formally define) randIND-yy-CCA; only in detIND-yy-CCA games does $\mathcal{A}$ have access to $\mathcal{O}$.Dec. CHALL is the challenge input provided by $\mathcal{A}$: how to perform Create $\tilde{\mathrm{C}}$ with CHALL is shown in Fig. 11, Fig. 12 and Fig. 17. Trivial win conditions, i.e. deciding the value of twf and computing $\mathcal{K}^*, \mathcal{C}^*, \mathcal{I}^*$, are discussed in Section 3.2.

When appropriate, we will restrict our experiments to provide definitions of security that are more suitable for assessing schemes with deterministic update mechanisms. For such schemes, access to the update token for the challenge epoch ($\Delta_{\tilde{\mathsf{e}}}$) allows the adversary to trivially win detIND-UPD-atk and detIND-UE-atk for $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA}\}$. Note however that the definitions are not restricted to schemes with deterministic updates: such schemes are simply insecure in terms of randIND-UPD-CPA and randIND-UE-CPA.

Ciphertext Integrity. In ciphertext integrity (CTXT) game, the adversary is allowed to make calls to oracles $\mathcal{O}$.Enc, $\mathcal{O}$.Next, $\mathcal{O}$.Upd and $\mathcal{O}$.Corr.

At some point $\mathcal{A}$ attempts to provide a forgery via $\mathcal{O}.\mathsf{Try}$; as part of this query the challenger will assess if it is valid. We distinguish between the single-$\mathcal{O}.\mathsf{Try}$ case ($\mathsf{INT\text{-}CTXT^s}$) and the multi-$\mathcal{O}.\mathsf{Try}$ case ($\mathsf{INT\text{-}CTXT}$). Here, "valid" means decryption outputs a message (i.e. not $\perp$). In the single-$\mathcal{O}.\mathsf{Try}$ case, $\mathcal{A}$ can continue making oracle queries after its $\mathcal{O}.\mathsf{Try}$ query, however this is of no benefit since it has already won or lost. In the multi-$\mathcal{O}.\mathsf{Try}$ case, $\mathcal{A}$ can make any number of $\mathcal{O}.\mathsf{Try}$ queries: as long as it wins once, it wins the ciphertext integrity game. Formally, the definition of ciphertext integrity is given in Definition 4.

Definition 4. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. Then the $\mathsf{INT\text{-}CTXT}$ advantage of an adversary $\mathcal{A}$ against UE is defined as

$$\mathbf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{UE},\,\mathcal{A}}(\lambda) = \mathbf{Pr}[\mathbf{Exp}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{UE},\,\mathcal{A}} = 1]$$

where the experiment $\mathbf{Exp}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{UE},\,\mathcal{A}}$ is given in Fig. 6 and Fig. 10. Particularly, if $\mathcal{A}$ is allowed to ask only one $\mathcal{O}.\mathsf{Try}$ query, denote such notion as $\mathsf{INT\text{-}CTXT^s}$.

Note that $\mathsf{INT\text{-}CTXT}$ trivially implies $\mathsf{INT\text{-}CTXT^s}$. We can prove that $\mathsf{INT\text{-}CTXT^s}$ implies $\mathsf{INT\text{-}CTXT}$ too, with loss upper-bounded by the number of $\mathcal{O}.\mathsf{Try}$ queries. We prove this result in Lemma 1. KLR19 defined ciphertext integrity with one $\mathcal{O}.\mathsf{Try}$ query plus access to $\mathcal{O}.\mathsf{Dec}$, and the game ends when the $\mathcal{O}.\mathsf{Try}$ query happens. It is hard to prove the generic relation among CPA, CTXT and CCA using this formulation. Notice that decryption oracles give the adversary power to win the CTXT game even it only has one $\mathcal{O}.\mathsf{Try}$ query. The adversary can send its forgery to the decryption oracle to test if it is valid (if $\mathcal{O}.\mathsf{Dec}$ outputs a message and not $\perp$) – thus $\mathcal{A}$ can continue to send forgeries to $\mathcal{O}.\mathsf{Dec}$ until a valid one is found, and then send this as a $\mathcal{O}.\mathsf{Try}$ query (and win the game). So intuitively, a decryption oracle is equivalent to multiple $\mathcal{O}.\mathsf{Try}$ queries. Proving that all these variants of CTXT definitions are equivalent to each other is straightforward, with the loss upper-bounded by the sum of $\mathcal{O}.\mathsf{Try}$ queries and decryption queries.

	CHALL	Output of " Create $\tilde{\mathrm{C}}$ with CHALL" (in $\tilde{\mathsf{e}}$)
xxIND-ENC-atk	$\bar{\mathrm{M}}_0, \bar{\mathrm{M}}_1$	$\mathsf{UE.Enc}(\mathrm{k}_{\tilde{\mathsf{e}}}, \bar{\mathrm{M}}_0)$ **or** $\mathsf{UE.Enc}(\mathrm{k}_{\tilde{\mathsf{e}}}, \bar{\mathrm{M}}_1)$
xxIND-UPD-atk	$\bar{\mathrm{C}}_0, \bar{\mathrm{C}}_1$	$\mathsf{UE.Upd}(\Delta_{\tilde{\mathsf{e}}}, \bar{\mathrm{C}}_0)$ **or** $\mathsf{UE.Upd}(\Delta_{\tilde{\mathsf{e}}}, \bar{\mathrm{C}}_1)$
xxIND-UE-atk	$\bar{\mathrm{M}}, \bar{\mathrm{C}}$	$\mathsf{UE.Enc}(\mathrm{k}_{\tilde{\mathsf{e}}}, \bar{\mathrm{M}})$ **or** $\mathsf{UE.Upd}(\Delta_{\tilde{\mathsf{e}}}, \bar{\mathrm{C}})$

Figure 9: Intuitive description of challenge inputs and outputs in confidentiality games for updatable encryption schemes, for $(\mathsf{xx}, \mathsf{atk}) \in \{(\mathsf{det}, \mathsf{CPA}), (\mathsf{rand}, \mathsf{CPA}), (\mathsf{det}, \mathsf{CCA})\}$. Full definitions are given in Section 3.1 and 4.1.

$\mathbf{Exp}_{\mathsf{UE},\,\mathcal{A}}^{\mathsf{INT\text{-}CTXT}}(\lambda)$

```
1 : do Setup
2 : win ← 0
3 : $\mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Try}}(\lambda)$
4 : if twf = 1
5 :     win ← 0
6 : return win
```

Figure 10: INT-CTXT security notion for updatable encryption scheme UE and adversary $\mathcal{A}$. Deciding twf and computing $\mathcal{L}^*$ are discussed in Section 3.2.

Remark 1. The definition of INT-CTXT is more natural for defining ciphertext integrity, however, it is easier to use $\mathsf{INT\text{-}CTXT}^{\mathsf{s}}$ notion to prove ciphertext integrity for specific UE schemes. As INT-CTXT and $\mathsf{INT\text{-}CTXT}^{\mathsf{s}}$ is equivalent, we use both definitions in this paper.

Lemma 1. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. For any INT-CTXT adversary $\mathcal{A}$ against UE that queries as most Q_T $\mathcal{O}.\mathsf{Try}$ queries, there exists an $\mathsf{INT\text{-}CTXT}^{\mathsf{s}}$ adversary $\mathcal{B}_1$ against UE such that

$$\mathbf{Adv}_{\mathsf{UE},\,\mathcal{A}}^{\mathsf{INT\text{-}CTXT}}(\lambda) \leq Q_T \cdot \mathbf{Adv}_{\mathsf{UE},\,\mathcal{B}_1}^{\mathsf{INT\text{-}CTXT}^{\mathsf{s}}}(\lambda).$$

Proof. As definition 4, we have

$$\mathbf{Adv}_{\mathsf{UE},\,\mathcal{A}}^{\mathsf{INT\text{-}CTXT}}(\lambda) = \mathbf{Pr}[\mathbf{Exp}_{\mathsf{UE},\,\mathcal{A}}^{\mathsf{INT\text{-}CTXT}} = 1].$$

We define Q_T games, for the i-th game $\mathcal{G}_i$, it is identical to $\mathsf{INT\text{-}CTXT}$ game except for the challenger only responses the i-th $\mathcal{O}.\mathsf{Try}$ query and returns $\perp$ to the rest of $\mathcal{O}.\mathsf{Try}$ queries. Then we have

$$\mathbf{Pr}[\mathbf{Exp}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{UE},\,\mathcal{A}} = 1] \leq \sum_{i=1}^{Q_T} \mathbf{Pr}[\mathcal{G}_i = 1].$$

Then we claim that for any $i \in \{1, \ldots, Q_T\}$ there exists an adversary

$$\mathbf{Pr}[\mathcal{G}_i = 1] = \mathbf{Adv}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{UE},\,\mathcal{B}_{1,i}}(\lambda).$$

We can construct the reduction $\mathcal{B}_{1,i}$ playing $\mathsf{INT\text{-}CTXT^s}$ game and simulating the responses of $\mathcal{G}_i$ by submitting the i-th $\mathcal{O}.\mathsf{Try}$ query to its $\mathsf{INT\text{-}CTXT^s}$ challenger and returns $\perp$ for the rest $\mathcal{O}.\mathsf{Try}$ queries. Other queries and the final result can be passed from $\mathsf{INT\text{-}CTXT^s}$ game to $\mathcal{G}_i$. Then we have the desired result. □

3.1 Existing Definitions of Confidentiality

Here we describe existing confidentiality notions given by LT18 and KLR19, including formal definitions for their $\mathsf{IND\text{-}yy\text{-}CPA}$ and $\mathsf{IND\text{-}yy\text{-}CCA}$ notions, respectively. (Note that KLR19 used UP-REENC to refer to the the unlinkability notion that we and LT18 call $\mathsf{IND\text{-}UPD}$). We will define our new security notion in Section 4.1 and compare the relationship between all notions in Section 4.2.

Definition 5. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. Then the $\mathsf{xxIND\text{-}ENC\text{-}atk}$ advantage, for $(\mathsf{xx}, \mathsf{atk}) \in \{(\mathsf{det}, \mathsf{CPA}), (\mathsf{rand}, \mathsf{CPA}), (\mathsf{det}, \mathsf{CCA})\}$, of an adversary $\mathcal{A}$ against UE is defined as

$$\mathbf{Adv}^{\mathsf{xxIND\text{-}ENC\text{-}atk}}_{\mathsf{UE},\,\mathcal{A}}(\lambda) = \left| \mathbf{Pr}[\mathbf{Exp}^{\mathsf{xxIND\text{-}ENC\text{-}atk\text{-}1}}_{\mathsf{UE},\,\mathcal{A}} = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{xxIND\text{-}ENC\text{-}atk\text{-}0}}_{\mathsf{UE},\,\mathcal{A}} = 1] \right|,$$

where the experiment $\mathbf{Exp}^{\mathsf{xxIND\text{-}ENC\text{-}atk\text{-}b}}_{\mathsf{UE},\,\mathcal{A}}$ is given in Fig. 6, Fig. 8 and Fig. 11.

Definition 6. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. Then the $\mathsf{xxIND\text{-}UPD\text{-}atk}$ advantage, for $(\mathsf{xx}, \mathsf{atk}) \in \{(\mathsf{det}, \mathsf{CPA}), (\mathsf{rand}, \mathsf{CPA}), (\mathsf{det}, \mathsf{CCA})\}$, of an adversary $\mathcal{A}$ against UE is defined as

$$\mathbf{Adv}^{\mathsf{xxIND\text{-}UPD\text{-}atk}}_{\mathsf{UE},\,\mathcal{A}}(\lambda) = \left|\mathbf{Pr}[\mathbf{Exp}^{\mathsf{xxIND\text{-}UPD\text{-}atk\text{-}1}}_{\mathsf{UE},\,\mathcal{A}} = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{xxIND\text{-}UPD\text{-}atk\text{-}0}}_{\mathsf{UE},\,\mathcal{A}} = 1]\right|,$$

where the experiments $\mathbf{Exp}^{\mathsf{xxIND\text{-}UPD\text{-}atk\text{-}b}}_{\mathsf{UE},\,\mathcal{A}}$ are given in Fig. 6, Fig. 8 and Fig. 12.

```
$\mathbf{Exp}^{\mathsf{xxIND\text{-}ENC\text{-}atk\text{-}b}}_{\mathsf{UE},\,\mathcal{A}}(\lambda)$
1 : $(\bar{\mathrm{M}}_0, \bar{\mathrm{M}}_1) \leftarrow \mathcal{A}$
2 : Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}_0, \bar{\mathrm{M}}_1)$
3 :     if $|\bar{\mathrm{M}}_0| \neq |\bar{\mathrm{M}}_1|$
4 :         return $\perp$
5 :     $\tilde{\mathrm{C}} \xleftarrow{\$} \mathsf{UE.Enc}(\mathrm{k}_{\tilde{\mathrm{e}}}, \bar{\mathrm{M}}_{\mathrm{b}})$
6 :     return $\tilde{\mathrm{C}}$
```

Figure 11: Challenge call definition for $\mathsf{xxIND\text{-}ENC\text{-}atk}$ security experiment; the full experiment is given in combination with Fig. 6 and Fig. 8.

```
$\mathbf{Exp}^{\mathsf{xxIND\text{-}UPD\text{-}atk\text{-}b}}_{\mathsf{UE},\,\mathcal{A}}(\lambda)$
1 : $(\bar{\mathrm{C}}_0, \bar{\mathrm{C}}_1) \leftarrow \mathcal{A}$
2 : Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{C}}_0, \bar{\mathrm{C}}_1)$
3 :     if $|\bar{\mathrm{C}}_0| \neq |\bar{\mathrm{C}}_1|$
4 :         return $\perp$
5 :     if $(\bar{\mathrm{C}}_0, \tilde{\mathrm{e}}\text{-}1) \notin \mathcal{L}$ or $(\bar{\mathrm{C}}_1, \tilde{\mathrm{e}}\text{-}1) \notin \mathcal{L}$
6 :         return $\perp$
7 :     $\tilde{\mathrm{C}} \xleftarrow{\$} \mathsf{UE.Upd}(\Delta_{\tilde{\mathrm{e}}}, \bar{\mathrm{C}}_{\mathrm{b}})$
8 :     return $\tilde{\mathrm{C}}$
```

Figure 12: Challenge call definition for $\mathsf{xxIND\text{-}UPD\text{-}atk}$ security experiment; the full experiment is given in combination with Fig. 6 and Fig. 8.

We do not define $\mathsf{randIND\text{-}ENC\text{-}CCA}$ or $\mathsf{randIND\text{-}UPD\text{-}CCA}$ – these notions were formalized by KLR19. Note that trivial win via direct update (see Section 3.2) is never triggered in the $\mathsf{detIND\text{-}ENC\text{-}CPA}$ game. Thus, $\mathsf{randIND\text{-}ENC\text{-}CPA}$ is equivalent to $\mathsf{detIND\text{-}ENC\text{-}CPA}$. For simplicity, we will often denote the notion $\mathsf{xxIND\text{-}ENC\text{-}CPA}$ as $\mathsf{IND\text{-}ENC\text{-}CPA}$.

Remark 2. LT18 defined $\mathsf{weakIND\text{-}ENC\text{-}CPA}$ and $\mathsf{weakIND\text{-}UPD\text{-}CPA}$ for analyzing BLMR+, a modification of BLMR's scheme where the nonce is encrypted using symmetric encryption. In this notion, the adversary trivially loses if it obtains an update token linking the challenge epoch to the epoch before or after. We show that $\mathsf{BLMR+}$ is $\mathsf{weakIND\text{-}UE\text{-}CPA}$ secure in Section B.

3.2 Trivial Win Conditions

3.2.1 Trivial Win Conditions in Confidentiality Games

Trivial wins via keys and ciphertexts. The following is for analyzing all confidentiality games. We again follow LT18 in defining the epoch identification sets $\mathcal{C}^*$, $\mathcal{K}^*$ and $\mathcal{T}^*$ as the extended sets of $\mathcal{C}$, $\mathcal{K}$ and $\mathcal{T}$ in which the adversary has learned or inferred information via its acquired tokens. These extended sets are used to exclude cases in which the adversary trivially wins, i.e. if $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$, then there exists an epoch in which the adversary knows the epoch key and a valid update of the challenge ciphertext. Note that the challenger computes these sets once the adversary has finished running. We employ the following algorithms of LT18 (for bi-directional updates):

- $\mathcal{K}^* \leftarrow \{\mathsf{e} \in \{0,...,n\} | \mathsf{CorrK}(\mathsf{e}) = \mathsf{true}\}$, where $\mathsf{CorrK}(\mathsf{e}) = \mathsf{true}$ if and only if $(\mathsf{e} \in \mathcal{K}) \vee (\mathsf{CorrK}(\mathsf{e\text{-}1}) \wedge \mathsf{e} \in \mathcal{T}) \vee (\mathsf{CorrK}(\mathsf{e+1}) \wedge \mathsf{e+1} \in \mathcal{T})$;
- $\mathcal{T}^* \leftarrow \{\mathsf{e} \in \{0,...,n\} | (\mathsf{e} \in \mathcal{T}) \vee (\mathsf{e} \in \mathcal{K}^* \wedge \mathsf{e\text{-}1} \in \mathcal{K}^*)\}$;
- $\mathcal{C}^* \leftarrow \{\mathsf{e} \in \{0,...,n\} | \mathsf{ChallEq}(\mathsf{e}) = \mathsf{true}\}$, where $\mathsf{ChallEq}(\mathsf{e}) = \mathsf{true}$ if and only if $(\mathsf{e} = \tilde{\mathsf{e}}) \vee (\mathsf{e} \in \mathcal{C}) \vee (\mathsf{ChallEq}(\mathsf{e\text{-}1}) \wedge \mathsf{e} \in \mathcal{T}^*) \vee (\mathsf{ChallEq}(\mathsf{e+1}) \wedge \mathsf{e+1} \in \mathcal{T}^*)$.

Trivial wins via direct updates. For $\mathsf{yy} \in \{\mathsf{UE}, \mathsf{UPD}\}$ and $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA}\}$, the following is for analyzing $\mathsf{detIND\text{-}yy\text{-}atk}$ security notions, where the adversary provides as its challenge one or two ciphertexts that it received from $\mathcal{O}.\mathsf{Enc}$. The challenger needs to use $\mathcal{L}$ to track the information the adversary has about these challenge input values.

Define a new list $\mathcal{I}$ as the list of epochs in which the adversary learned an updated version of the ciphertext(s) given as a challenge input. Furthermore, define $\mathcal{I}^*$ to be the extended set in which the adversary has learned or inferred information via token corruption. We will use this set to exclude cases which the adversary trivially wins, i.e. if $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$, then there exists an epoch in which the adversary knows the updated ciphertext of $\bar{\mathrm{C}}$ and a valid challenge-equal ciphertext. For deterministic updates, the adversary can simply compare these ciphertexts to win the game. In particular, if $\bar{\mathrm{C}}$ is restricted to come from $\tilde{\mathsf{e}} - 1$ (recall the challenge epoch is $\tilde{\mathsf{e}}$), then the condition $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$ is equivalent to the win condition that LT18 used for $\mathsf{IND\text{-}UPD}$: $\Delta_{\tilde{\mathsf{e}}} \in \mathcal{T}^*$ or $\mathcal{A}$ did $\mathcal{O}.\mathsf{Upd}(\bar{\mathrm{C}})$ in $\tilde{\mathsf{e}}$. Our generalization is necessary for a variant of $\mathsf{xxIND\text{-}UE\text{-}atk}$ that we define later in which the challenge ciphertext input can come from any prior epoch, and not just the epoch immediately before the one in which the challenge is made.

To compute $\mathcal{I}$, find an entry in $\mathcal{L}$ that contains challenge input $\bar{\mathrm{C}}$. Then for that entry, note the query identifier c, scan $\mathcal{L}$ for other entries with this identifier, and add into list $\mathcal{I}$ all found indices:

$$\mathcal{I} \leftarrow \{\mathsf{e} \in \{0, \ldots, n\} | (\mathsf{c}, \cdot, \mathsf{e}) \in \mathcal{L}\}.$$

Then compute $\mathcal{I}^*$ as follows:

$$\mathcal{I}^* \leftarrow \{\mathsf{e} \in \{0, \ldots, n\} | \mathsf{ChallinputEq}(\mathsf{e}) = \mathsf{true}\},$$

where $\mathsf{ChallinputEq}(\mathsf{e}) = \mathsf{true}$ if and only if $(\mathsf{e} \in \mathcal{I}) \vee (\mathsf{ChallinputEq}(\mathsf{e}\text{-}1) \wedge \mathsf{e} \in \mathcal{T}^*) \vee (\mathsf{ChallinputEq}(\mathsf{e}\text{+}1) \wedge \mathsf{e}\text{+}1 \in \mathcal{T}^*)$.

Additionally, if the adversary submits two ciphertexts $\bar{\mathrm{C}}_0, \bar{\mathrm{C}}_1$ as challenge (as in $\mathsf{xxIND\text{-}UPD\text{-}atk}$), we compute $\mathcal{I}_i, \mathcal{I}_i^*, i \in \{0, 1\}$ first and then use $\mathcal{I}^* = \mathcal{I}_0^* \cup \mathcal{I}_1^*$ to check the trivial win condition. An example of trivial win conditions $\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$ and $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$ is shown in Fig. 16.

We do not consider this trivial win condition for the ENC notion, as there is no ciphertext in the challenge input value, i.e. $\mathcal{I}^* = \emptyset$. Thus, the assessment $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$ in experiment $\mathbf{Exp}_{\mathsf{UE},\,\mathcal{A}}^{\mathsf{detIND\text{-}ENC\text{-}atk\text{-}b}}$ (see Fig. 8) will never be true.

Trivial wins via decryptions. For $\mathsf{yy} \in \{\mathsf{UE}, \mathsf{ENC}, \mathsf{UPD}\}$, the following is for analyzing $\mathsf{detIND\text{-}yy\text{-}CCA}$ where the adversary has access to $\mathcal{O}.\mathsf{Dec}$. We follow the trivial win analysis in KLR19: suppose the adversary knows a challenge ciphertext $(\tilde{\mathrm{C}}, \mathsf{e}_0) \in \tilde{\mathcal{L}}$ and tokens from epoch $\mathsf{e}_0 + 1$ to epoch e, then the adversary can update the challenge ciphertext from epoch e_0 to epoch e. If $\mathcal{A}$ sends the updated ciphertext to $\mathcal{O}.\mathsf{Dec}$ this will reveal the underlying message, and $\mathcal{A}$ trivially wins the game: we shall exclude this type of attack.

Define $\tilde{\mathcal{L}}^*$ to be the extended set of $\tilde{\mathcal{L}}$ in which the adversary has learned or inferred information via token corruption. Whenever $\mathcal{O}.\mathsf{Dec}$ receives a ciphertext located in $\tilde{\mathcal{L}}^*$, the challenger will set the trivial win flag twf to be 1. The list $\tilde{\mathcal{L}}^*$ is updated while the security game is running. After the challenge query happens, the challenger updates $\tilde{\mathcal{L}}^*$ whenever an element is added to list $\tilde{\mathcal{L}}$ or a token is corrupted. In Fig. 13 we show how list $\tilde{\mathcal{L}}^*$ is updated.

3.2.2 Trivial Win Conditions in Ciphertext Integrity Games

We again follow the trivial win analysis in KLR19. In ciphertext integrity games for updatable encryption, we do not consider the randomized update setting as the adversary can update an old ciphertext via a corrupted token to provide any number of new valid forgeries to the Try query to trivially win this game.

Trivial wins via keys. If an epoch key is corrupted, then the adversary can use this key to forge ciphertexts in this epoch. We exclude this trivial win.

Trivial wins via ciphertexts. Suppose the adversary knows a ciphertext $(\mathrm{C}, \mathsf{e}_0) \in \mathcal{L}$ and tokens from epoch $\mathsf{e}_0 + 1$ to epoch e, then the adversary

```
Update $\tilde{\mathcal{L}}^*$
1 : if challenge query or $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$ happens
2 :     $\tilde{\mathcal{L}}^* \leftarrow \tilde{\mathcal{L}}^* \cup \{(\tilde{\mathrm{C}}, \cdot)\}$
3 : if phase = 1 and $\mathcal{O}.\mathsf{Corr}(\mathsf{token}, \cdot)$ happens
4 :     for $i \in \mathcal{T}^*$ and $(\tilde{\mathrm{C}}_{i-1}, i-1) \in \tilde{\mathcal{L}}^*$ do
5 :         $\tilde{\mathcal{L}}^* \leftarrow \tilde{\mathcal{L}}^* \cup \{(\tilde{\mathrm{C}}_i, i)\}$
```

Figure 13: Update procedure for list $\tilde{\mathcal{L}}^*$

```
Update $\mathcal{L}^*$
1 : if $\mathcal{O}.\mathsf{Enc}$ or $\mathcal{O}.\mathsf{Upd}$ happens
2 :     $\mathcal{L}^* \leftarrow \mathcal{L}^* \cup \{(\cdot, \mathrm{C}, \cdot)\}$
3 : if $\mathcal{O}.\mathsf{Corr}(\mathsf{token}, \cdot)$ happens
4 :     for $i \in \mathcal{T}^*$ do
5 :         for $(j, \mathrm{C}_{i-1}, i-1) \in \mathcal{L}^*$ do
6 :             $\mathrm{C}_i \leftarrow \mathsf{UE.Upd}(\Delta_i, \mathrm{C}_{i-1})$
7 :             $\mathcal{L}^* \leftarrow \mathcal{L}^* \cup \{(j, \mathrm{C}_i, i)\}$
```

Figure 14: Update procedure for list $\mathcal{L}^*$.

can provide a forgery by updating C to epoch e. We shall exclude this type of forgeries.

Define $\mathcal{L}^*$ to be the extended set of $\mathcal{L}$ in which the adversary has learned or inferred information via token corruption. If $\mathcal{O}.\mathsf{Try}$ receives a ciphertext located in $\mathcal{L}^*$, the challenger will set twf to 1. The list $\mathcal{L}^*$ is updated while the security game is running. Ciphertexts output by $\mathcal{O}.\mathsf{Enc}$ and $\mathcal{O}.\mathsf{Upd}$ are known to the adversary. Furthermore, whenever a token is corrupted, the challenger may update list $\mathcal{L}^*$ as well. In Fig. 14 we show how list $\mathcal{L}^*$ is updated.

3.3 Firewall Technique

In order to prove security for updatable encryption in the epoch-based model with strong corruption capabilities, cryptographic separation is required between the epochs in which the adversary knows key material, and those in which it knows challenge-equal ciphertexts (acquired/calculated via queries to $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$ and $\mathcal{O}.\mathsf{Corr}(\Delta)$). To ensure this, we follow prior work in explicitly defining the 'safe' or *insulated* regions, as we explain below. These regions insulate epoch keys, tokens and ciphertexts: outside of an insulated region a reduction in a security proof can generate keys and tokens itself, but within these regions it must embed its challenge while still providing the underlying adversary with access to the appropriate oracles. A thorough discussion of how we leverage these insulated regions in proofs is given in Section 5.3.

To understand the idea of firewalls, consider any security game (for bi-directional schemes) in which the trivial win conditions are *not* triggered. If the adversary $\mathcal{A}$ corrupts all tokens then either it never corrupts any keys or it never asks for a challenge ciphertext. Suppose that $\mathcal{A}$ does ask for a challenge ciphertext in epoch $\tilde{\mathsf{e}}$ [5]. Then there exists an (unique) epoch continuum around $\tilde{\mathsf{e}}$ such that no keys in this epoch continuum, and no tokens in the boundaries of this epoch continuum are corrupted. Moreover, we can assume that all tokens within this epoch continuum are corrupted, because once the adversary has finished corrupting keys, it can corrupt any

[5] In the situation that the adversary does not corrupt any keys to the left or the right (or both) of the challenge epoch, the insulated region thus extends to the boundary (or boundaries) of the epoch continuum.

remaining tokens that do not ‘touch’ those corrupted keys. This observation is first used in the $\mathsf{IND\text{-}UPD}$ proof of RISE provided by Lehmann and Tackmann [LT18a], and Klooß et al. [KLR19a] provided an extended description of this ‘key insulation’ technique. We name these epoch ranges *insulated regions* and their boundaries to be *firewalls*.

Definition 7. An *insulated region* with *firewalls* fwl and fwr is a consecutive sequence of epochs $(\mathsf{fwl}, \dots, \mathsf{fwr})$ for which:

- no key in the sequence of epochs $(\mathsf{fwl}, \dots, \mathsf{fwr})$ is corrupted;
- the tokens Δ_{fwl} and $\Delta_{\mathsf{fwr}+1}$ are not corrupted (if they exist);
- all tokens $(\Delta_{\mathsf{fwl}+1}, \dots, \Delta_{\mathsf{fwr}})$ are corrupted (if any exist).

We denote the firewalls bordering the special insulated region that contains $\tilde{\mathsf{e}}$ as $\hat{\mathsf{fwl}}$ and $\hat{\mathsf{fwr}}$ – though note that there could be (many, distinct) insulated regions elsewhere in the epoch continuum. Specifically, when the adversary asks for updated versions of the challenge ciphertext, the epoch in which this query occurs must also fall within (what the challenger later calculates as) an insulated region. In Fig. 15 we give an algorithm $\mathsf{FW\text{-}Find}$ for computing firewall locations. The list $\mathcal{FW}$ tracks, and appends a label to, each insulated region and its firewalls. Observe that if an epoch is a left firewall, then neither the key nor the token for that epoch are corrupted. From the left firewall, since we assume that all tokens are corrupted, track to the right until either a token is not corrupted or a key is.

3.3.1 Example of Epoch Corruption and Trivial Wins

In Fig. 16 we indicate the trivial win conditions and insulated regions for a particular adversarial corruption strategy, in the $\mathsf{detIND\text{-}UE^*\text{-}CPA}$ experiment (this notion chosen here to demonstrate how the challenger populates its lists). Suppose challenge epoch $\tilde{\mathsf{e}} = 8$, and further assume $\mathcal{K}^* = \{1, 6, 9\}$, and $\mathcal{T}^* = \{3, 4, 8\}$, meaning that $\mathcal{C}^* = \{7, 8\}$. Suppose $\bar{\mathrm{C}}$ is in epoch 1 and the adversary has asked $\mathcal{O}.\mathsf{Upd}(\bar{\mathrm{C}})$ in epoch 2, so $\bar{\mathrm{C}}_2, \bar{\mathrm{C}}_3, \bar{\mathrm{C}}_4$ are updated ciphertexts of $\bar{\mathrm{C}}$, therefore $\mathcal{I}^* = \{1, 2, 3, 4\}$. So $\mathcal{C}^* \cap \mathcal{K}^* = \emptyset$ and $\mathcal{I}^* \cap \mathcal{C}^* = \emptyset$, the trivial win conditions have not occurred. Then we see insulated regions: $\{0\}$ is the first insulated region, $\{2, 3, 4\}$ is the second insulated region, etc. We compute $\mathcal{T}^* \cup \mathcal{K}^* = \{1, 3, 4, 6, 8, 9\}$, so

```
FW-Find
1 : $\mathcal{FW} \leftarrow \emptyset$
2 : $j = 0$
3 : for $\mathsf{e} \in \{0, ..., n\}$ do
4 :    if $\mathsf{e} \in \neg(\mathcal{T}^* \cup \mathcal{K}^*)$
5 :       $j \leftarrow j + 1$
6 :       $\mathsf{fwl}_j \leftarrow \mathsf{e}$
7 :    if $(\mathsf{e} + 1 \notin \mathcal{T}^*)$ and $(\mathsf{e} \notin \mathcal{K}^*)$
8 :       $\mathsf{fwr}_j \leftarrow \mathsf{e}$
9 :       $\mathcal{FW} \leftarrow \{(j, \mathsf{fwl}_j, \mathsf{fwr}_j)\}$
```

Figure 15: Algorithm FW-Find for computing all firewalls.

$\neg(\mathcal{T}^* \cup \mathcal{K}^*) = \{0, 2, 5, 7\}$: using FW-Find we know this is the set of left firewalls, and the right firewalls are $\{0, 4, 5, 8\}$.

Epoch	{0}	1	{2	3	4}	{5}	6	{7	ẽ}	9
Key	×	k_1	×	×	×	×	k_6	×	×	k_9
Token		×	×	Δ_3	Δ_4	×	×	×	Δ_8	×
$\tilde{\mathrm{C}}$	×	×	×	×	×	×	×	$\tilde{\mathrm{C}}_7$	$\tilde{\mathrm{C}}_8$	×
$\bar{\mathrm{C}}$	×	$\bar{\mathrm{C}}$	$\bar{\mathrm{C}}_2$	$\bar{\mathrm{C}}_3$	$\bar{\mathrm{C}}_4$	×	×	×	×	×

Figure 16: An example of trivial win conditions and insulated regions incurred by an adversary playing detIND-UE*-CPA, where $\tilde{\mathrm{C}}$ indicates challenge ciphertexts, $\bar{\mathrm{C}}$ indicates challenge input, × indicates the keys/tokens/ciphertexts not revealed to the adversary, and {} indicates insulated regions.

4 On the Security of Updates

In this section we present a new notion of security for updatable encryption schemes, which we denote xxIND-UE-atk. This notion captures both security of fresh encryptions (i.e. implies xxIND-ENC-atk) and unlinkability (i.e. implies xxIND-UPD-atk). We first explain the new notion and then describe its relation to previous notions. Then, we prove a generic relationship among CPA, CTXT and CCA to complete the picture for security notions for UE schemes.

4.1 A New Definition of Confidentiality

In the security game for $\mathsf{xxIND\text{-}UE\text{-}atk}$, the adversary submits one message and a ciphertext from an earlier epoch that the adversary received via a call to $\mathcal{O}.\mathsf{Enc}$. The challenger responds with either an encryption of that message or an update of that earlier ciphertext, in the challenge (current) epoch $\tilde{\mathsf{e}}$.

Definition 8 ($\mathsf{xxIND\text{-}UE\text{-}atk}$)**.** Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. Then the $\mathsf{xxIND\text{-}UE\text{-}atk}$ advantage, for $(\mathsf{xx}, \mathsf{atk}) \in \{(\mathsf{det}, \mathsf{CPA}), (\mathsf{rand}, \mathsf{CPA}), (\mathsf{det}, \mathsf{CCA})\}$, of an adversary $\mathcal{A}$ against UE is defined as

$$\mathbf{Adv}^{\mathsf{xxIND\text{-}UE\text{-}atk}}_{\mathsf{UE},\ \mathcal{A}}(\lambda) = \left|\mathbf{Pr}[\mathbf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}1}}_{\mathsf{UE},\ \mathcal{A}} = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}0}}_{\mathsf{UE},\ \mathcal{A}} = 1]\right|,$$

where the experiment $\mathbf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}b}}_{\mathsf{UE},\ \mathcal{A}}$ is given in Fig. 6, Fig. 8 and Fig. 17.

Note that $\mathsf{randIND\text{-}UE\text{-}CPA}$ is strictly stronger than $\mathsf{detIND\text{-}UE\text{-}CPA}$, since the adversary has strictly more capabilities. A generalized version of $\mathsf{xxIND\text{-}UE\text{-}atk}$, denoted $\mathsf{xxIND\text{-}UE^*\text{-}atk}$, is also given in Fig. 17. In this game the input challenge ciphertext can come from (i.e. be known to $\mathcal{A}$ in) any prior epoch, not just the epoch immediately before $\tilde{\mathsf{e}}$. Note that $\mathsf{xxIND\text{-}UE\text{-}atk}$ is a special case of $\mathsf{xxIND\text{-}UE^*\text{-}atk}$. Under some fairly weak requirements (that all schemes discussed in this paper satisfy) we can prove that $\mathsf{xxIND\text{-}UE\text{-}atk}$ implies $\mathsf{xxIND\text{-}UE^*\text{-}atk}$ – we prove this result in Section. 4.2.1.

Remark 3. The definition of $\mathsf{xxIND\text{-}UE\text{-}atk}$ is more concise and intuitively easier to understand than that of $\mathsf{xxIND\text{-}UE^*\text{-}atk}$, however in Theorem 2.1 and Theorem 2.2 in Section 4.2.1 we show that these two versions of security notions are equivalent. This result and our generic proof techniques mean that all results in this paper that hold for $\mathsf{xxIND\text{-}UE\text{-}atk}$, also hold for $\mathsf{xxIND\text{-}UE^*\text{-}atk}$, and vice versa.

```
$\mathbf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}b}}_{\mathsf{UE},\,\mathcal{A}}(\lambda)$
1 : $(\bar{\mathrm{M}}, \bar{\mathrm{C}}) \leftarrow \mathcal{A}$
2 : $\underline{\text{Create } \tilde{\mathrm{C}} \text{ with } (\bar{\mathrm{M}}, \bar{\mathrm{C}})}$
3 :     if $(\bar{\mathrm{C}}, \tilde{\mathsf{e}} - 1) \notin \mathcal{L}$
4 :         return $\perp$
5 :     if $\mathrm{b} = 0$
6 :         $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}} \leftarrow \mathsf{UE.Enc}(\mathrm{k}_{\tilde{\mathsf{e}}}, \bar{\mathrm{M}})$
7 :     else
8 :         $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}} \leftarrow \mathsf{UE.Upd}(\Delta_{\tilde{\mathsf{e}}}, \bar{\mathrm{C}})$
9 :     return $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
```

```
$\mathbf{Exp}^{\mathsf{xxIND\text{-}UE^*\text{-}atk\text{-}b}}_{\mathsf{UE},\,\mathcal{A}}(\lambda)$
1 : $(\bar{\mathrm{M}}, (\bar{\mathrm{C}}, \mathsf{e}')) \leftarrow \mathcal{A}$
2 : $\underline{\text{Create } \tilde{\mathrm{C}} \text{ with } (\bar{\mathrm{M}}, (\bar{\mathrm{C}}, \mathsf{e}'))}$
3 :     if $(\bar{\mathrm{C}}, \mathsf{e}') \notin \mathcal{L}$
4 :         return $\perp$
5 :     if $\mathrm{b} = 0$
6 :         $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}} \leftarrow \mathsf{UE.Enc}(\mathrm{k}_{\tilde{\mathsf{e}}}, \bar{\mathrm{M}})$
7 :     else
8 :         $\tilde{\mathrm{C}}_{\mathsf{e}'} \leftarrow \bar{\mathrm{C}}$
9 :         for $j \in \{\mathsf{e}'+1, \ldots, \tilde{\mathsf{e}}\}$ do
10 :            $\tilde{\mathrm{C}}_j \leftarrow \mathsf{UE.Upd}(\Delta_j, \tilde{\mathrm{C}}_{j-1})$
11 :     return $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
```

Figure 17: Challenge call definition for xxIND-UE-atk and xxIND-UE*-atk security experiments; the full experiment is defined in Fig. 6 and Fig. 8.

Remark 4. In Section C.1 we show that the RISE scheme presented by LT18 is randIND-UE-CPA secure under DDH. While this result is perhaps unsurprising[6], the proof techniques we use are novel and may be of independent interest. We give an Oracle-DDH-like game that inherits the epoch-based nature of the updatable encryption security model, and then use this as a bridge to prove security.

4.2 Relations among Security Notions

In Fig. 18 we show the relationship between the new and existing UE security notions. Note that our new notion is strictly stronger than the IND-ENC and IND-UPD notions presented in prior work, and is in fact stronger than the combination of the prior notions. Further, we show that the generic relation among CPA, CTXT and CCA, that CPA security coupled with ciphertext integrity implies CCA security, also holds for updatable encryption schemes. The relationships are proven via Theorem 2.2 to 2.8, and Theo-

[6]LT18 had already shown that RISE is IND-ENC-CPA and randIND-UPD-CPA, so our result shows that it is stronger than was previously known.

rem 3, which follow next.

Theorem 2 (Informal Theorem)**.** The relationship among the security notions xxIND-UE-atk, xxIND-ENC-atk and xxIND-UPD-atk are as in Fig. 18. This is proven via Theorem 2.2 to 2.8, and Theorem 3.

$$\begin{tikzcd}
\begin{array}{c}\text{IND-ENC-CPA}\\ \text{+randIND-UPD-CPA}\end{array} \arrow[d, Rightarrow, "/" marking, "2.7"] & \begin{array}{c}\text{IND-ENC-CPA}\\ \text{+detIND-UPD-CPA}\end{array} \arrow[d, Rightarrow, "/" marking, "2.7"] & \begin{array}{c}\text{IND-ENC-CCA}\\ \text{+detIND-UPD-CCA}\end{array} \arrow[d, Rightarrow, "/" marking, "2.7"] \\
\text{randIND-UE-CPA} \arrow[r, Rightarrow, shift left, "\text{Def. } 8"] \arrow[d, Rightarrow, "2.5"] \arrow[dd, Rightarrow, bend right, "2.3"'] & \text{detIND-UE-CPA} \arrow[l, Rightarrow, shift left, "/" marking, "2.8"] \arrow[r, Rightarrow, "+\text{CTXT}", "3"'] \arrow[d, Rightarrow, "2.6"] \arrow[dd, Rightarrow, bend right, "2.4+2.2"] & \text{detIND-UE-CCA} \arrow[d, Rightarrow, "2.6"'] \arrow[dd, Rightarrow, bend left, "2.4+2.2"] \\
\text{randIND-UPD-CPA} & \text{detIND-UPD-CPA} \arrow[r, Rightarrow, "+\text{CTXT}", "3"'] & \text{detIND-UPD-CCA} \\
\text{IND-ENC-CPA} & \text{IND-ENC-CPA} \arrow[r, Rightarrow, "+\text{CTXT}", "3"'] & \text{detIND-ENC-CCA}
\end{tikzcd}$$

Figure 18: Relations among confidentiality notions xxIND-yy-atk for xx $\in$ $\{$det, rand$\}$, yy $\in$ $\{$UE, ENC, UPD$\}$, atk $\in$ $\{$CPA, CCA$\}$, and ciphertext integrity (INT-CTXT). Arrow labelling refers to Theorem numbering except where otherwise specified.

4.2.1 Relations between IND-UE and IND-UE*

Properties of Deterministic Updates Here we will use an an alternative representation of UE.Enc that specifies a deterministic algorithm with randomness as input, i.e. $C_e \leftarrow \mathsf{UE.Enc}(k_e, M; r)$.

One of our main contributions is a scheme with a deterministic update mechanism – we now discuss some of the properties of such schemes. The first two properties, simulatable token generation and randomness-preserving updates, were introduced by Klooß et al. [KLR19a]. Simulatable token generation states that the real token looks like a token generated from a token simulation algorithm, as we consider bi-directional updates we omit the generation of the reverse token. Randomness-preserving states that the update of a ciphertext looks like an encryption of the same message, with the same randomness, under the new key.

Definition 9 (Simulatable token [KLR19a]). Let UE be an updatable encryption scheme. We say that UE has simulatable token generation if it has the following property: There is a PPT algorithm $\mathsf{SimTG}(\lambda)$ which samples a token Δ. Furthermore, for arbitrary (fixed) $\mathrm{k}_{\mathsf{old}} \xleftarrow{\$} \mathsf{UE.KG}(\lambda)$ following distributions of Δ are identical:

- $\{\Delta \mid \Delta \xleftarrow{\$} \mathsf{SimTG}(\lambda)\}$
- $\{\Delta \mid \mathrm{k}_{\mathsf{new}} \xleftarrow{\$} \mathsf{UE.KG}(\lambda), \Delta \leftarrow \mathsf{UE.TG}(\mathrm{k}_{\mathsf{old}}, \mathrm{k}_{\mathsf{new}})\}$

Notice that BLMR, $\mathsf{BLMR+}$, RISE, SHINE all have simulatable token generation. Furthermore, the simulatable token generation algorithms of these UE schemes generates a token by randomly picking a token from the token space, i.e. $\mathsf{SimTG} : \Delta \xleftarrow{\$} \mathcal{TS}$.

Definition 10 (Randomness-preserving [KLR19a]). Let UE be an updatable encryption scheme. We say that $\mathsf{UE.Upd}$ (for UE) is randomness-preserving if the following holds: First, as usually assumed, UE encrypts with uniformly chosen randomness. Second, all keys $(\mathrm{k}^{\mathsf{old}}, \mathrm{k}^{\mathsf{new}}) \xleftarrow{\$} \mathsf{UE.KG}(\lambda)$, tokens $\Delta^{\mathsf{new}} \xleftarrow{\$} \mathsf{UE.TG}(\mathrm{k}^{\mathsf{old}}, \mathrm{k}^{\mathsf{new}})$, plaintext m and randomness r, we have

$$\mathsf{UE.Upd}(\Delta^{\mathsf{new}}, \mathrm{C}^{\mathsf{old}}) = \mathsf{UE.Enc}(\mathrm{k}^{\mathsf{new}}, m; r),$$

where $\mathrm{C}^{\mathsf{old}} = \mathsf{UE.Enc}(\mathrm{k}^{\mathsf{old}}, m; r)$.

Suppose $\mathrm{C}_i = \mathsf{UE.Enc}(\mathrm{k}_i, m; r)$, and C_j is an updated ciphertext of C_i from epoch i to epoch j. Randomness-preserving property makes sure that $\mathrm{C}_j = \mathsf{UE.Enc}(\mathrm{k}_j, m; r)$, which means a (updated) ciphertext under some epoch key is uniquely decided by the message and randomness.

We define an even weaker property which we call update-preserving: any update sequence starting and ending at the same key that starts with the same ciphertext will result in the same ciphertext.

Definition 11 (Update-preserving)**.** Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme with deterministic update algorithm. We say that $\mathsf{UE.Upd}$ (for UE) is update-preserving if the following holds: for any two sequence of key pairs with the same start key and end key $(\mathrm{k}_i, \mathrm{k}_{i+1}, ..., \mathrm{k}_j)$ and $(\mathrm{k}'_i, \mathrm{k}'_{i+1}, ..., \mathrm{k}'_j)$, where $\mathrm{k}_i(= \mathrm{k}'_i), \mathrm{k}_{i+1}, \mathrm{k}'_{i+1}, ..., \mathrm{k}_{j-1}, \mathrm{k}'_{j-1}, \mathrm{k}_j(= \mathrm{k}'_j) \xleftarrow{\$} \mathsf{UE.KG}(\lambda)$, and tokens $\Delta_l \xleftarrow{\$} \mathsf{UE.TG}(\mathrm{k}_{l-1}, \mathrm{k}_l)$, $\Delta'_l \xleftarrow{\$} \mathsf{UE.TG}(\mathrm{k}'_{l-1}, \mathrm{k}'_l)$, f or any ciphertext C_i in epoch i, we have $\mathrm{C}_j = \mathrm{C}'_j$ where $\mathrm{C}'_i = \mathrm{C}_i, \mathrm{C}_l = \mathsf{UE.Upd}(\Delta_l, \mathrm{C}_{l-1})$, $\mathrm{C}'_l = \mathsf{UE.Upd}(\Delta'_l, \mathrm{C}'_{l-1})$ for $l = i+1, ..., j$.

The diagrams in Fig. 19 show how the property works, with the left-hand side indicating keys and tokens and the right-hand side showing ciphertexts.

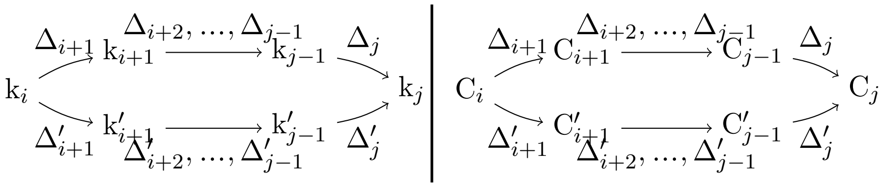
$$\begin{tikzcd}
 & \mathrm{k}_{i+1} \arrow[r, "{\Delta_{i+2}, ..., \Delta_{j-1}}"] & \mathrm{k}_{j-1} \arrow[dr, "\Delta_j"] & \\
\mathrm{k}_i \arrow[ur, "\Delta_{i+1}"] \arrow[dr, "\Delta'_{i+1}"'] & & & \mathrm{k}_j \\
 & \mathrm{k}'_{i+1} \arrow[r, "{\Delta'_{i+2}, ..., \Delta'_{j-1}}"'] & \mathrm{k}'_{j-1} \arrow[ur, "\Delta'_j"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & \mathrm{C}_{i+1} \arrow[r, "{\Delta_{i+2}, ..., \Delta_{j-1}}"] & \mathrm{C}_{j-1} \arrow[dr, "\Delta_j"] & \\
\mathrm{C}_i \arrow[ur, "\Delta_{i+1}"] \arrow[dr, "\Delta'_{i+1}"'] & & & \mathrm{C}_j \\
 & \mathrm{C}'_{i+1} \arrow[r, "{\Delta'_{i+2}, ..., \Delta'_{j-1}}"'] & \mathrm{C}'_{j-1} \arrow[ur, "\Delta'_j"'] &
\end{tikzcd}$$

Figure 19: Keys and updated ciphertexts in Definition 11

We have that update-preserving property implies that the updated ciphertext is uniquely determined by $(\mathrm{C}_i, k_j, j\text{-}i)$, where C_i is the beginning ciphertext for updating, $j\text{-}i$ decides how many updates have occurred, and k_j decides the value of the ending epoch's epoch key.

We now define another property which states that ciphertexts encrypted under one key can be simulated by ciphertexts encrypted under another key. All schemes in this paper meet this property.

Definition 12 (Simulatable Encryption)**.** Let UE be an updatable encryption scheme. For all keys $\mathrm{k}^{\mathsf{old}}, \mathrm{k}^{\mathsf{new}} \xleftarrow{\$} \mathsf{UE.KG}(\lambda)$, tokens $\Delta^{\mathsf{new}} \xleftarrow{\$} \mathsf{UE.TG}(\mathrm{k}^{\mathsf{old}}, \mathrm{k}^{\mathsf{new}})$, plaintext m, define X^{old}, X^{new} to be the statistical distribution of the ciphertexts output by $\mathsf{UE.Enc}(\mathrm{k}^{\mathsf{old}}, m)$, $\mathsf{UE.Enc}(\mathrm{k}^{\mathsf{new}}, m)$, resp.. We say that UE has simulatable encryption if update algorithm keeps the ciphertext distribution, i.e. $\mathsf{UE.Upd}(\Delta^{\mathsf{new}}, X^{\mathsf{old}}) \overset{dist}{=} X^{\mathsf{new}}$.

Note that we do not restrict that the update algorithm is probabilistic.

This means when the update algorithm is deterministic, it will not add randomness to the updated ciphertext, and it maintains the ciphertext distribution. For example, suppose $U(\mathbb{Z})$ is a uniform distribution over $\mathbb{Z}$, and for any integer Δ, let $\mathsf{UE.Upd}(\Delta, x) = x + \Delta$, then $\mathsf{UE.Upd}(\Delta, U(\mathbb{Z})) = U(\mathbb{Z})$. This definition looks similar to the definition of perfect re-encryption provided by Klooß et al. [KLR19a], which mandates that update has the same distribution as decrypt-then-encrypt. Perfect re-encryption requires the update algorithm is probabilistic, which makes it possible for any updated ciphertext looks like a fresh encryption. The simulatable encryption property is to make sure that the update algorithm can keep the distribution of encryption – however it is not necessary to require that the update algorithm is probabilistic.

All schemes discussed in this paper satisfy all of the above properties. Note that randomness-preserving property is strictly stronger than the update-preserving property and simulatable encryption. Obviously, if a scheme is randomness-preserving then it is also update-preserving and has simulatable encryption. However, the update-preserving property does not imply randomness-preserving property, even with simulatable encryption. To see this, construct a deterministic update variant of the RISE scheme (Section C) such that the randomness r updates to $r+2$: this scheme has the update-preserving property and simulatable encryption, but not the randomness-preserving property.

xxIND-UE-atk implies xxIND-UE*-atk. We prove xxIND-UE-atk implies xxIND-UE*-atk in this section, and consequently we have the result that xxIND-UE-atk and xxIND-UE*-atk are equivalent.

randIND-UE-CPA implies randIND-UE*-CPA.

Theorem 2.1. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. For any randIND-UE*-CPA adversary $\mathcal{A}$ against UE, there exists an randIND-UE-CPA adversary $\mathcal{B}_{2.1}$ against UE such that

$$\mathbf{Adv}_{\mathsf{UE},\,\mathcal{A}}^{\mathsf{randIND\text{-}UE^*\text{-}CPA}}(\lambda) \leq \mathbf{Adv}_{\mathsf{UE},\,\mathcal{B}_{2.1}}^{\mathsf{randIND\text{-}UE\text{-}CPA}}(\lambda).$$

Proof. We construct a reduction $\mathcal{B}_{2.1}$: before the epoch counter is incremented, every ciphertext is updated using the available update oracles. This needs to happen when the adversary moves to the next epoch, so that it is always possible to provide a valid challenge input to the reducton's own randIND-UE-CPA challenger and respond with a valid challenge output to the adversary.

More precisely, when the adversary makes the randIND-UE*-CPA challenge query, the reduction make its own randIND-UE-CPA query, submitting the ciphertext provided by the adversary but updated to the epoch one before the challenge epoch that both algorithms are in. This should give the exact same result as updating the older ciphertext. Consequently, and since all other oracle queries can just be forwarded, the reduction perfectly simulates the randIND-UE*-CPA game. We have the required result. □

detIND-UE-atk implies detIND-UE*-atk.

Proof technique of Theorem 2.2. The proof uses the firewall technique, where the reduction will 'pause' its own epoch continuum while responding to the adversary's queries. The main left firewall in the detIND-UE*-atk game is an epoch in which the detIND-UE-atk reduction can possibly ask for a valid challenge query. Before the left firewall, the reduction sends the queries received from the adversary $\mathcal{A}$ to its detIND-UE-atk challenger, and forwards responses to $\mathcal{A}$. Within the firewalls, the reduction stops asking any $\mathcal{O}$.Next queries, and instead simulates the responses of each query to provide answers to $\mathcal{A}$. Because of this action, the detIND-UE-atk challenger will stay in epoch $\hat{\mathsf{fwl}}$. When $\mathcal{A}$ makes the detIND-UE*-atk challenge queries (if the trivial win conditions of the detIND-UE*-atk game are not satisfied then the trivial win conditions of the detIND-UE-atk game will be not satisfied as well), the reduction makes its detIND-UE-atk query using the old ciphertext (in $\hat{\mathsf{fwl}} - 1$) instead. After receiving the response, the reduction updates its challenge ciphertext to the challenge epoch to reply to $\mathcal{A}$. After the right firewall, the query responses are calculated and forwarded, in the same manner as before the left firewall.

Theorem 2.2. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme has simulatable token generation, update-preserving property and simulatable encryption property.

For any detIND-UE*-atk adversary $\mathcal{A}$ against UE, where atk $\in$ {CPA,

$\mathsf{CCA}\}$, there exists an $\mathsf{detIND\text{-}UE\text{-}atk}$ adversary $\mathcal{B}_{2.2}$ against UE such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE^*\text{-}atk}}_{\mathsf{UE},\,\mathcal{A}}(\lambda) \leq (n+1)^2 \cdot \mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}atk}}_{\mathsf{UE},\,\mathcal{B}_{2.2}}(\lambda).$$

Proof. We use three steps to prove this result.

(Step 1.) Consider a modified version of $\mathsf{detIND\text{-}UE^*\text{-}atk}$. For $\mathrm{b} \in \{0,1\}$, define experiments $\mathbf{Exp}^{\mathsf{INT}_1\text{-}\mathrm{b}}$ to be the same as $\mathbf{Exp}^{\mathsf{detIND\text{-}UE^*\text{-}atk\text{-}b}}$ except that the experiments randomly pick $\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}$, and if $\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}$ are not the firewalls around challenge epoch $\tilde{\mathsf{e}}$, then the experiment returns a random bit b'. More formally, the values $\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}$ are the desired firewalls if the challenge is made inside – i.e. $\tilde{\mathsf{e}} \in [\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}]$ – and they actually consitute an insulate region, i.e. $(, \hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}) \in \mathcal{FW}$)).

These firewalls $\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}$ could take any value in $\{0, ..., n\}$, so this loss is upper bounded by $(n+1)^2$. We have

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE^*atk}}_{\mathsf{UE},\mathcal{A}}(\lambda) \leq (n+1)^2 \mathbf{Adv}^{\mathsf{INT}_1}_{\mathsf{UE},\mathcal{A}}.$$

(Step 2.) Then we consider experiments $\mathbf{Exp}^{\mathsf{INT}_2\text{-}\mathrm{b}}$, which is the same as $\mathbf{Exp}^{\mathsf{INT}_1\text{-}\mathrm{b}}$ except for: in the insulated region all encryptions are updated ciphertexts of ciphertexts encrypted in left firewall $\hat{\mathsf{fwl}}$ By this we mean that if the adversary asks for any $\mathcal{O}.\mathsf{Enc}$, $\mathcal{O}.\mathsf{Dec}$ and challenge query, the responses work as follows:

- $\mathcal{O}.\mathsf{Enc}(\mathrm{M})$: if called in an epoch $\hat{\mathsf{fwl}} < \mathsf{e} \leq \hat{\mathsf{fwr}}$, encrypt the message in left firewall $\hat{\mathsf{fwl}}$, then update the ciphertext to epoch e, and return the updated ciphertext.

- $\mathcal{O}.\mathsf{Dec}(\mathrm{C})$: if called in an epoch $\hat{\mathsf{fwl}} < \mathsf{e} \leq \hat{\mathsf{fwr}}$, reverse update the ciphertext from e back to $\hat{\mathsf{fwl}}$, then decrypt the updated ciphertext, and return the decrypted value.

- challenge query, on input $(\bar{\mathrm{M}}, (\bar{\mathrm{C}}, \mathsf{e}'))$: if $\mathrm{b} = 0$, encrypt the message $\bar{\mathrm{M}}$ in left firewall $\hat{\mathsf{fwl}}$, then update the ciphertext to the challenge epoch $\tilde{\mathsf{e}}$; if $\mathrm{b} = 1$, update ciphertext $\bar{\mathrm{C}}$ from epoch e' to epoch $\tilde{\mathsf{e}}$. Return the challenge ciphertext.

Since we assume that UE has the simulatable encryption property, both operations are possible so we have

$$\mathbf{Adv}^{\mathsf{INT}_1}_{\mathsf{UE},\,\mathcal{A}}(\lambda) = \mathbf{Adv}^{\mathsf{INT}_2}_{\mathsf{UE},\,\mathcal{A}}(\lambda).$$

(Step 3.) We construct a reduction $\mathcal{B}_{2.2}$, detailed in Fig. 20 and Fig. 21, that is playing the detIND-UE-atk game and runs $\mathcal{A}$. We claim that

$$\mathbf{Adv}^{\mathsf{INT}_2}_{\mathsf{UE},\,\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}atk}}_{\mathsf{UE},\,\mathcal{B}_{2.2}}(\lambda).$$

If $\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}$ are the desired firewalls, then $[\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}] \subseteq \mathcal{C}^*$. If the trivial win conditions in $\mathbf{Exp}^{\mathsf{INT}_2\text{-b}}$ are not set (the same result as the trivial win conditions in $\mathbf{Exp}^{\mathsf{detIND\text{-}UE^*\text{-}atk\text{-}b}}$), i.e. $\mathcal{I}^* \cap \mathcal{C}^* = \emptyset$, then $\mathcal{I}^* \cap [\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}] = \emptyset$. That means $\mathcal{A}$ never asks $\mathcal{O}.\mathsf{Upd}(\bar{\mathrm{C}})$ and $\mathcal{O}.\mathsf{Corr}(\mathsf{token})$ in $\hat{\mathsf{fwl}}$. So the reduction uses the relevant challenge input to ask a challenge query to its own detIND-UE-atk challenger in epoch $\hat{\mathsf{fwl}}$, and it will not trivially lose.

Before the epoch counter is incremented, every ciphertext is updated using the available update oracles. This needs to happen when the adversary moves to the next epoch, so that it is always possible to provide a valid challenge input to the reducton's own detIND-UE-atk challenger and respond with a valid challenge output to the adversary.

Within the firewalls, the reduction simulates all ciphertexts and uses the list $\mathcal{FL}$ and the list $\tilde{\mathcal{FL}}$ to track non-challenge ciphertexts and challenge-equal ciphertexts, respectively. When the challenge query happens with input $(\bar{\mathrm{M}}, (\bar{\mathrm{C}}, \mathsf{e}'))$, the reduction can find all updated versions of $\bar{\mathrm{C}}$ by checking the first entry of the list $\mathcal{L}$. The reduction uses the ciphertext in epoch $\hat{\mathsf{fwl}}$-1 with the same query identifier c as a challenge input, sending to its own detIND-UE-atk challenger. (Note that $\mathsf{e}' < \hat{\mathsf{fwl}}$, otherwise, $[\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}] \subseteq \mathcal{I}^*$ and the trivial win condition is triggered.) After receiving the response from the detIND-UE-atk challenger, $\mathcal{B}_{2.2}$ updates the received ciphertext to the challenge epoch to reply $\mathcal{A}$.

Eventually $\mathcal{B}_{2.2}$ receives b' from $\mathcal{A}$, and simply outputs b' to its own detIND-UE-atk challenger. When $\mathcal{B}_{2.2}$ interacts with $\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}atk\text{-}b}}_{\mathsf{UE},\,\mathcal{B}_{2.2}}$, $\mathcal{B}_{2.2}$ can perfectly simulate $\mathbf{Exp}^{\mathsf{INT}_2\text{-b}}_{\mathsf{UE},\,\mathcal{A}}$ to $\mathcal{A}$. Then we have the required result. □

4.2.2 Relations among IND-ENC, IND-UPD, and IND-UE

In this section, we analyze the relations among notions with different challenge input. Since similar proof techniques are used in Theorems 2.3, 2.4, 2.5 and 2.6, of these we give full proof details only for Theorem 2.3.

```
Reduction $\mathcal{B}_{2.2}$ playing $\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}atk\text{-}}b}_{\mathsf{UE},\ \mathcal{B}_{2.2}}$
1 :  $\mathbf{do\ Setup}; \mathcal{FL}, \tilde{\mathcal{FL}}, \mathcal{L}, \tilde{\mathcal{L}} \leftarrow \emptyset$
2 :  $\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}} \xleftarrow{\$} \{0, \ldots, n\}$
3 :  $\bar{\mathrm{M}}, (\bar{\mathrm{C}}, \mathsf{e}') \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},(\mathcal{O}.\mathsf{Dec}),\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
4 :  $\mathsf{phase} \leftarrow 1$
5 :  Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}, (\bar{\mathrm{C}}, \mathsf{e}'))$, $\mathbf{get}\ \tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
6 :  $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},(\mathcal{O}.\mathsf{Dec}),\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}})$
7 :  $\underline{\mathsf{twf} \leftarrow 1\ \mathbf{if}}$
8 :      $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ or $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$
9 :  if ABORT occurred or $(\cdot, \hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}) \notin \mathcal{FW}$ or $\mathsf{twf} = 1$
10 :     $\mathrm{b}' \xleftarrow{\$} \{0, 1\}$
11 : return $\mathrm{b}'$
```

Figure 20: Reduction $\mathcal{B}_{2.2}$ for proof of Theorem 2.2. The adversary may call for $\mathcal{O}$.Dec in the CCA game. The oracles in Fig. 21 show how $\mathcal{B}_{2.2}$ responds to $\mathcal{A}$, including calls to oracles in its own detIND-UE-atk game.

Theorem 2.3. Let UE be a UE scheme. For any IND-ENC-CPA adversary $\mathcal{A}$ against UE, there exists an randIND-UE-CPA adversary $\mathcal{B}_{2.3}$ against UE such that

$$\mathbf{Adv}^{\mathsf{IND\text{-}ENC\text{-}CPA}}_{\mathsf{UE},\ \mathcal{A}}(\lambda) \leq 2 \cdot \mathbf{Adv}^{\mathsf{randIND\text{-}UE\text{-}CPA}}_{\mathsf{UE},\ \mathcal{B}_{2.3}}(\lambda).$$

Proof. We construct a reduction $\mathcal{B}_{2.3}$ running the randIND-UE-CPA experiment which will simulate the responses of queries made by the adversary $\mathcal{A}$. To provide a valid non-challenge ciphertext to its own challenger, $\mathcal{B}_{2.3}$ must run $\mathcal{A}$ out of step with its own game, so epoch 0 as far as $\mathcal{A}$ is concerned is actually epoch 1 for $\mathcal{B}_{2.3}$, and so on.

1. $\mathcal{B}_{2.3}$ chooses $\mathrm{b} \xleftarrow{\$} \{0, 1\}$.
2. $\mathcal{B}_{2.3}$ receives the setup parameters from its randIND-UE-CPA challenger, chooses $\mathrm{M} \xleftarrow{\$} \mathcal{MS}$ and calls $\mathcal{O}.\mathsf{Enc}(\mathrm{M})$ which returns some C_0. Then $\mathcal{B}_{2.3}$ calls $\mathcal{O}$.Next once and sends the setup parameters to $\mathcal{A}$.

```
$\mathcal{O}.\mathsf{Enc}(\mathrm{M})$
$\mathsf{c} \leftarrow \mathsf{c} + 1$
if $\mathsf{e} \notin \{\widehat{\mathsf{fwl}}+1, \ldots, \widehat{\mathsf{fwr}}\}$
    call $\mathcal{O}.\mathsf{Enc}(\mathrm{M})$, get $\mathrm{C}_{\mathsf{e}}$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}_{\mathsf{e}}, \mathsf{e})\}$
if $\mathsf{e} \in \{\widehat{\mathsf{fwl}}+1, \ldots, \widehat{\mathsf{fwr}}\}$
    call $\mathcal{O}.\mathsf{Enc}(\mathrm{M})$, get $\mathrm{C}_{\widehat{\mathsf{fwl}}}$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}_{\widehat{\mathsf{fwl}}}, \widehat{\mathsf{fwl}})\}$
    for $j \in \{\widehat{\mathsf{fwl}}+1, \ldots, \mathsf{e}\}$ do
        $\mathrm{C}_j \leftarrow \mathsf{UE.Upd}(\Delta_j, \mathrm{C}_{j-1})$
    $\mathcal{FL} \leftarrow \mathcal{FL} \cup \{(\mathsf{c}, \mathrm{C}_{\mathsf{e}}, \mathsf{e})\}$
return $\mathrm{C}_{\mathsf{e}}$
```

```
$\mathcal{O}.\mathsf{Next}$
if $\mathsf{e} \in \{1, \ldots, \widehat{\mathsf{fwl}}\text{-}1\}$ then
    for $(\mathsf{c}, \mathrm{C}_{\mathsf{e}-1}, \mathsf{e}-1) \in \mathcal{L}$ do
        call $\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}\text{-}1})$, get $\mathrm{C}_{\mathsf{e}}$
        $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}_{\mathsf{e}}, \mathsf{e})\}$
    call $\mathcal{O}.\mathsf{Next}$
if $\mathsf{e} \in \{\widehat{\mathsf{fwr}}+1, \ldots, n\}$ then
    call $\mathcal{O}.\mathsf{Next}$
if $\mathsf{e} \in \{\widehat{\mathsf{fwl}}, \ldots, \widehat{\mathsf{fwr}}\text{-}1\}$ then
    $\mathsf{e} \leftarrow \mathsf{e}{+}1, \Delta_{\mathsf{e}} \xleftarrow{\$} \mathsf{SimTG}(\lambda)$
if $\mathsf{e} = \widehat{\mathsf{fwr}}$ then
    for $j \in \{\widehat{\mathsf{fwl}}+1, \ldots, \widehat{\mathsf{fwr}}+1\}$ do
        call $\mathcal{O}.\mathsf{Next}$
        for $(\mathsf{c}, \mathrm{C}_{j\text{-}1}, j\text{-}1) \in \mathcal{L}$ do
            call $\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{j\text{-}1}$, get $\mathrm{C}_j$
            $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}_j, j)\}$
        for $(\tilde{\mathrm{C}}_{j\text{-}1}, j\text{-}1) \in \tilde{\mathcal{L}}$ do
            call $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}(\tilde{\mathrm{C}}_{j\text{-}1}))$, get $\tilde{\mathrm{C}}_j$
            $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{\mathrm{C}}_j, j)\}$
```

```
$\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}-1})$
if $(\mathsf{c}, \mathrm{C}_{\mathsf{e}-1}, \mathsf{e}-1) \notin \mathcal{L} \cup \mathcal{FL}$
    return $\perp$
if $\mathsf{e} \in \{1, \ldots, \widehat{\mathsf{fwl}}\}$
    call $\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}-1})$, get $\mathrm{C}_{\mathsf{e}}$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}_{\mathsf{e}}, \mathsf{e})\}$
if $\mathsf{e} \in \{\widehat{\mathsf{fwr}}+2, \ldots, n\}$
    call $\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}-1})$, get $\mathrm{C}_{\mathsf{e}}$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}_{\mathsf{e}}, \mathsf{e})\}$
if $\mathsf{e} \in \{\widehat{\mathsf{fwl}}+1, \ldots, \widehat{\mathsf{fwr}}\}$
    $\mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{UE.Upd}(\Delta_{\mathsf{e}}, \mathrm{C}_{\mathsf{e}-1})$
    $\mathcal{FL} \leftarrow \mathcal{FL} \cup \{(\mathsf{c}, \mathrm{C}_{\mathsf{e}}, \mathsf{e})\}$
if $\mathsf{e} = \widehat{\mathsf{fwr}}+1$
    find $(\mathsf{c}, \mathrm{C}_{\mathsf{e}}, \mathsf{e}) \in \mathcal{L}$
return $\mathrm{C}_{\mathsf{e}}$
```

```
$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$
do $\mathsf{Check}(\mathsf{inp}, \hat{\mathsf{e}}; \mathsf{e}; \widehat{\mathsf{fwl}}, \widehat{\mathsf{fwr}})$
if $\mathsf{inp} = \mathsf{key}$
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
    return $\mathrm{k}_{\hat{\mathsf{e}}}$
if $\mathsf{inp} = \mathsf{token}$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
    if $\hat{\mathsf{e}} \in \{1, \ldots, \widehat{\mathsf{fwl}}\text{-}1\}$
        call $\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$
        get $\Delta_{\hat{\mathsf{e}}}$
    if $\{\widehat{\mathsf{fwr}}+2, \ldots, n\}$
        call $\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$
        get $\Delta_{\hat{\mathsf{e}}}$
    if $\hat{\mathsf{e}} \in \{\widehat{\mathsf{fwl}}+1, \ldots, \widehat{\mathsf{fwr}}\}$
        find $\Delta_{\hat{\mathsf{e}}}$
    return $\Delta_{\hat{\mathsf{e}}}$
```

Figure 21: Part 1. Oracles used in the proof of Theorem 2.2.

```
Create $\tilde{C}$ with $(\bar{M}, (\bar{C}, \mathsf{e}'))$
59 :  if $\tilde{\mathsf{e}} \notin \{\widehat{\mathsf{fwl}}, \ldots, \widehat{\mathsf{fwr}}\}$
60 :    ABORT
61 :  if $(\mathsf{c}, \bar{C}, \mathsf{e}') \notin \mathcal{L}$
62 :    ABORT
63 :  find $(\mathsf{c}, C_{\widehat{\mathsf{fwl}}-1}, \widehat{\mathsf{fwl}} - 1) \in \mathcal{L}$
64 :  call cq $(\bar{M}, C_{\widehat{\mathsf{fwl}}-1})$
65 :  get $\tilde{C}_{\widehat{\mathsf{fwl}}}$
66 :  $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{C}_{\widehat{\mathsf{fwl}}}, \widehat{\mathsf{fwl}})\}$
67 :  for $j \in \{\widehat{\mathsf{fwl}}+1, \ldots, \widehat{\mathsf{fwr}}\}$ do
68 :    $\tilde{C}_j \leftarrow \mathsf{UE.Upd}(\Delta_j, \tilde{C}_{j-1})$
69 :    $\tilde{\mathcal{FL}} \leftarrow \tilde{\mathcal{FL}} \cup \{(\tilde{C}_j, j)\}$
70 :  return $\tilde{C}_{\tilde{\mathsf{e}}}$
```

```
$\mathcal{O}.\mathsf{Upd}\tilde{C}$
71 :  if $\mathsf{e} \in \{1, \ldots, \widehat{\mathsf{fwl}}\text{-}1\}$
72 :    return $\perp$
73 :  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
74 :  if $\mathsf{e} \in \{\widehat{\mathsf{fwl}}\}$
75 :    find $(\tilde{C}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{L}}$
76 :  if $\mathsf{e} \in \{\widehat{\mathsf{fwl}}+1, \ldots, \widehat{\mathsf{fwr}}\}$
77 :    find $(\tilde{C}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{FL}}$
78 :  if $\mathsf{e} \in \{\widehat{\mathsf{fwr}}+1, \ldots, n\}$
79 :    call $\mathcal{O}.\mathsf{Upd}\tilde{C}$, get $\tilde{C}_{\mathsf{e}}$
80 :  return $\tilde{C}_{\mathsf{e}}$
```

```
$\mathcal{O}.\mathsf{Dec}(C)$
81 :  if phase $\leftarrow 1$ and $C \in \tilde{\mathcal{L}}^*$
82 :    $\mathsf{twf} \leftarrow 1$
83 :  return $\perp$
84 :  if $\mathsf{e} \notin \{\widehat{\mathsf{fwl}}+1, \ldots, \widehat{\mathsf{fwr}}\}$
85 :    call $\mathcal{O}.\mathsf{Dec}(C)$, get $M'/\perp$
86 :  if $\mathsf{e} \in \{\widehat{\mathsf{fwl}}+1, \ldots, \widehat{\mathsf{fwr}}\}$
87 :    for $j \in \{\mathsf{e}, \ldots, \widehat{\mathsf{fwl}}+1\}$ do
88 :      $C_{j-1} \leftarrow \mathsf{UE.Upd}^{-1}(\Delta_j, C_j)$
89 :    call $\mathcal{O}.\mathsf{Dec}(C_{\widehat{\mathsf{fwl}}})$, get $M'/\perp$
90 :  return $M'$ or $\perp$
```

Figure 21: Part 2. Oracles used in the proof of Theorem 2.2. In line 64 of the Create $\tilde{C}$ description, **call cq** means that the reduction makes its own challenge query with these values.

3. (a) Whenever $\mathcal{B}_{2.3}$ receives the queries $\mathcal{O}.\mathsf{Enc}$, $\mathcal{O}.\mathsf{Upd}$, $\mathcal{O}.\mathsf{Corr}$ from $\mathcal{A}$, $\mathcal{B}_{2.3}$ sends these queries to its randIND-UE-CPA challenger, and forwards the responses to $\mathcal{A}$.

   (b) Whenever $\mathcal{O}.\mathsf{Next}$ is called by $\mathcal{A}$, $\mathcal{B}_{2.3}$ randomly chooses a message $\mathrm{M} \xleftarrow{\$} \mathcal{MS}$ and calls $\mathcal{O}.\mathsf{Enc}(\mathrm{M})$ to receive some C_{e}, and then calls $\mathcal{O}.\mathsf{Next}$.

4. At some point, in epoch $\tilde{\mathsf{e}}$ (for its game), $\mathcal{B}_{2.3}$ receives the challenge query $(\bar{\mathrm{M}}_0, \bar{\mathrm{M}}_1)$ from $\mathcal{A}$. Then $\mathcal{B}_{2.3}$ sends $(\bar{\mathrm{M}}_{\mathrm{b}}, \mathrm{C}_{\tilde{\mathsf{e}}-1})$ as challenge to its own randIND-UE-CPA challenger. After receiving the challenge ciphertext, $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$, from its challenger, $\mathcal{B}_{2.3}$ sends $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$ to $\mathcal{A}$.

5. $\mathcal{B}_{2.3}$ continues to answer $\mathcal{A}$'s queries using its own oracles, now including $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$.

6. Finally $\mathcal{B}_{2.3}$ receives the output bit b' from $\mathcal{A}$. If $\mathrm{b} = \mathrm{b}'$ then $\mathcal{B}_{2.3}$ returns 0. Otherwise $\mathcal{B}_{2.3}$ returns 1.

We now bound the advantage of $\mathcal{B}_{2.3}$. The point is that whenever $\mathcal{B}_{2.3}$ returns a random encryption to $\mathcal{A}$, $\mathcal{B}_{2.3}$'s probability of winning is exactly 1/2 because the bit b' from $\mathcal{A}$ is independent of its choice of b. This happens with probability 1/2. However, when $\mathcal{B}_{2.3}$ returns a "correct" value to $\mathcal{A}$ (an encryption of $\bar{\mathrm{M}}_0$ or $\bar{\mathrm{M}}_1$), then $\mathcal{B}_{2.3}$'s probability of winning is the same as the probability that $\mathcal{A}$ wins.

First note that, as usual,

$$\mathbf{Adv}^{\mathsf{randIND\text{-}UE\text{-}CPA}}_{\mathsf{UE},\mathcal{B}_{2.3}} = \\ \left|\mathbf{Pr}[\mathbf{Exp}^{\mathsf{randIND\text{-}UE\text{-}CPA\text{-}1}}_{\mathsf{UE},\,\mathcal{B}_{2.3}} = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{randIND\text{-}UE\text{-}CPA\text{-}0}}_{\mathsf{UE},\,\mathcal{B}_{2.3}} = 1]\right|.$$

We claim that $\mathbf{Pr}[\mathbf{Exp}^{\mathsf{randIND\text{-}UE\text{-}CPA\text{-}1}}_{\mathsf{UE},\,\mathcal{B}_{2.3}} = 1] = 1/2$ because in this case $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$ is independent of b and so b' must also be independent of b. Then we have:

$$\begin{aligned}
&\mathbf{Adv}^{\mathsf{randIND\text{-}UE\text{-}CPA}}_{\mathsf{UE},\mathcal{B}_{2.3}} \\
&= \left| \frac{1}{2} - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{randIND\text{-}UE\text{-}CPA\text{-}0}}_{\mathsf{UE},\ \mathcal{B}_{2.3}} = 1] \right| \\
&= \left| \frac{1}{2} - \left(\frac{1}{2}\mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}CPA\text{-}0}}_{\mathsf{UE},\ \mathcal{A}} = 1] + \frac{1}{2}\mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}CPA\text{-}1}}_{\mathsf{UE},\ \mathcal{A}} = 0] \right) \right| \\
&= \left| \frac{1}{2} - \frac{1}{2}\mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}CPA\text{-}0}}_{\mathsf{UE},\ \mathcal{A}} = 1] - \frac{1}{2}\left(1 - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}CPA\text{-}1}}_{\mathsf{UE},\ \mathcal{A}} = 1]\right) \right| \\
&= \left| \frac{1}{2} \cdot \left(\mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}CPA\text{-}0}}_{\mathsf{UE},\ \mathcal{A}} = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}CPA\text{-}1}}_{\mathsf{UE},\ \mathcal{A}} = 1] \right) \right| \\
&= \frac{1}{2} \cdot \mathbf{Adv}^{\mathsf{IND\text{-}ENC\text{-}CPA}}_{\mathsf{UE},\ \mathcal{A}}. \qquad \square
\end{aligned}$$

A remark on Theorem 2.4. Directly proving that detIND-UE-atk implies detIND-ENC-atk is very challenging, and in fact the difficulty is the same as proving that detIND-UE-atk implies detIND-UE*-atk. Since we have proved detIND-UE-atk implies detIND-UE*-atk in Theorem 2.2 in Section 4.2.1, we do not repeat the similar proof approach here. We just prove detIND-UE*-atk implies detIND-ENC-atk, which is easy. We follow a very similar approach to the proof of Theorem 2.3. The detIND-UE*-atk (reduction) adversary can ask for tokens almost as freely as the detIND-ENC-atk adversary without incurring the trivial win conditions. But since there should at least one token, in an epoch before (include) challenge epoch $\tilde{\mathsf{e}}$, is unknown to the adversary, we again have to run our reduction out of step with the detIND-ENC-atk adversary (essentially creating an artificial epoch).

Theorem 2.4. Let UE $= \{$UE.KG, UE.TG, UE.Enc, UE.Dec, UE.Upd$\}$ be an updatable encryption scheme. For any detIND-ENC-atk adversary $\mathcal{A}$ against UE, where atk $\in \{$CPA, CCA$\}$, there exists a detIND-UE*-atk adversary $\mathcal{B}_{2.4}$ against UE such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}ENC\text{-}atk}}_{\mathsf{UE},\ \mathcal{A}}(\lambda) \leq 2 \cdot \mathbf{Adv}^{\mathsf{detIND\text{-}UE^*\text{-}atk}}_{\mathsf{UE},\ \mathcal{B}_{2.4}}(\lambda).$$

Proof. Similar to the proof strategy used in Theorem 2.3, we construct a detIND-UE*-atk adversary $\mathcal{B}_{2.4}$ against UE to simulate the responses to queries made by detIND-ENC-atk adversary $\mathcal{A}$. $\mathcal{B}_{2.4}$ chooses $\mathrm{b} \xleftarrow{\$} \{0, 1\}$.

Since $\mathcal{B}_{2.4}$ is allowed to takes its challenge ciphertext from any epoch in its $\mathsf{detIND\text{-}UE^*\text{-}atk}$ experiment, it can in particular uses the random ciphertext created in epoch 0, C_0. Note that, in contrast to Step 3 (b) of the simulation in the proof of Theorem 2.3, there is now no need for $\mathcal{B}_{2.4}$ to generate a random ciphertext for each epoch. Also $\mathcal{B}_{2.4}$ will never ask for $\mathcal{O}.\mathsf{Upd}(\mathrm{C}_0)$ to its own $\mathsf{detIND\text{-}UE^*\text{-}atk}$ challenger.

When receiving the challenge query $(\bar{\mathrm{M}}_0, \bar{\mathrm{M}}_1)$ from $\mathcal{A}$, the reduction $\mathcal{B}_{2.4}$ sends $(\bar{\mathrm{M}}_\mathrm{b}, \mathrm{C}_0)$ as challenge to its own $\mathsf{detIND\text{-}UE^*\text{-}atk}$ challenger. Since $\mathcal{A}$ has no view of epoch 0, $\mathcal{A}$ cannot ask for Δ_1. In addition, $\mathcal{A}$ will never ask for $\mathcal{O}.\mathsf{Upd}(\mathrm{C}_0)$.

Thus the trivial win condition $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$ will not be satisfied in the $\mathsf{detIND\text{-}UE^*\text{-}atk}$ game. The other trivial win(s), i.e. trivial win via keys and ciphertexts (and trivial wins via decryptions), will also pass from $\mathsf{detIND\text{-}ENC\text{-}atk}$ game to $\mathsf{detIND\text{-}UE^*\text{-}atk}$ game. The result follows using the same calculation as in the proof of Theorem 2.3. $\square$

Theorem 2.5. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. For any $\mathsf{randIND\text{-}UPD\text{-}CPA}$ adversary $\mathcal{A}$ against UE, there exists an $\mathsf{randIND\text{-}UE\text{-}CPA}$ adversary $\mathcal{B}_{2.5}$ against UE such that

$$\mathbf{Adv}^{\mathsf{randIND\text{-}UPD\text{-}CPA}}_{\mathsf{UE},\,\mathcal{A}}(\lambda) \leq 2 \cdot \mathbf{Adv}^{\mathsf{randIND\text{-}UE\text{-}CPA}}_{\mathsf{UE},\,\mathcal{B}_{2.5}}(\lambda).$$

Proof. Similarly to the proof method used in Theorem 2.3, we construct a $\mathsf{randIND\text{-}UE\text{-}CPA}$ reduction $\mathcal{B}_{2.5}$ against UE to simulate the responses of queries made by $\mathsf{randIND\text{-}UPD\text{-}CPA}$ adversary $\mathcal{A}$. However in this case it is not necessary for the reduction to be out-of-step with the adversary. $\mathcal{B}_{2.5}$ first chooses $\mathrm{b} \overset{\$}{\leftarrow} \{0,1\}$. $\mathcal{B}_{2.5}$ forwards all queries from $\mathcal{A}$ to its own oracles, and when it receives challenge query $(\bar{\mathrm{C}}_0, \bar{\mathrm{C}}_1)$ from $\mathcal{A}$, $\mathcal{B}_{2.5}$ samples a random message M, and sends $(\mathrm{M}, \bar{\mathrm{C}}_\mathrm{b})$ to the $\mathsf{randIND\text{-}UE\text{-}CPA}$ challenger. The result follows using a similar calculation to that in the proof of Theorem 2.3. $\square$

Theorem 2.6. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme For any $\mathsf{detIND\text{-}UPD\text{-}atk}$ adversary $\mathcal{A}$ against UE, where $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA}\}$, there exists an $\mathsf{detIND\text{-}UE\text{-}atk}$ adversary $\mathcal{B}_{2.6}$ against UE such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UPD\text{-}atk}}_{\mathsf{UE},\,\mathcal{A}}(\lambda) \leq 2 \cdot \mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}atk}}_{\mathsf{UE},\,\mathcal{B}_{2.6}}(\lambda).$$

Proof. The proof follows exactly the same steps as that of Theorem 2.5, in addition to the observation that in the det versions and CCA versions of the games, the trivial win flag twf is either triggered for both adversary and reduction or for neither when $\mathcal{O}.\mathsf{Corr}$ queries are made by the underlying adversary. $\square$

Theorem 2.7. Each of the following hold for $(\mathsf{xx}, \mathsf{atk}) \in \{(\mathsf{det}, \mathsf{CPA}), (\mathsf{rand}, \mathsf{CPA}), (\mathsf{det}, \mathsf{CCA})\}$.
(i). Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme and let α_{ENC} be the xxIND-ENC-atk advantage of an adversary $\mathcal{A}$ against UE. Then there exists a modified scheme UE' such that $\mathcal{A}$'s xxIND-ENC-atk advantage against UE' is (still) α_{ENC}, and there exists an xxIND-UE-atk adversary $\mathcal{B}$ against UE' with advantage 1.
(ii). Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme and let α_{UPD} be the xxIND-UPD-atk advantage of an adversary $\mathcal{A}$ against UE. Then there exists a modified scheme UE' such that $\mathcal{A}$'s xxIND-UPD-atk advantage against UE' is (still) α_{UPD}, and there exists an xxIND-UE-atk adversary $\mathcal{B}$ against UE' with advantage 1.
(iii). Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme and let α_{ENC} be the xxIND-ENC-atk advantage of an adversary $\mathcal{A}_{\mathsf{ENC}}$ against UE and α_{UPD} be the xxIND-UPD-atk advantage of an adversary . Then there exists a modified scheme UE' such that $\mathcal{A}_{\mathsf{ENC}}$'s xxIND-ENC-atk advantage against UE' is (still) α_{ENC}, $\mathcal{A}_{\mathsf{UPD}}$'s xxIND-UPD-atk advantage against UE' is (still) α_{UPD}, and there exists an xxIND-UE-atk adversary $\mathcal{B}$ against UE' with advantage 1.

Proof. All three are demonstrated using the same counterexample. All algorithms for UE' are the same as for UE, except $\mathsf{UE}'.\mathsf{Enc}$ is defined by modifying UE.Enc to append the epoch number in which the ciphertext was initially created. This change does not affect an adversary's ability to win the xxIND-ENC-atk or xxIND-UPD-atk games but trivially breaks xxIND-UE-atk security. $\square$

A remark on Theorem 2.8. We construct an updatable encryption scheme which is detIND-UE-CPA secure but not randIND-UE-CPA secure to prove that detIND-UE-CPA does not imply randIND-UE-CPA. Note that in Section 5 we will prove that SHINE is detIND-UE-CPA secure (yet it is trivially not randIND-UE-CPA secure), which provides an example to support this result. However the proof that SHINE is detIND-UE-CPA in the ideal cipher model (if DDH holds). Here we demonstrate the theorem based on a weaker assumption, namely the existence of pseudorandom functions.

Proof technique of Theorem 2.8. We use a xxIND-UE-CPA secure UE scheme to construct a new UE scheme $\mathsf{UE}^{\mathsf{new}}$, where we use a PRF to make a part of the new update algorithm deterministic. Because of this $\mathsf{UE}^{\mathsf{new}}$ will not be randIND-UE-CPA secure. In order to bound the detIND-UE-CPA security of $\mathsf{UE}^{\mathsf{new}}$, we need to make sure the newly added deterministic part of the updates will not make $\mathsf{Enc}(m)$ and $\mathsf{Upd}(\mathrm{C})$ distinguishable – this is where we need the PRF.

Theorem 2.8. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme, and define a new updatable encryption scheme $\mathsf{UE}^{\mathsf{new}}$ in Fig. 22, built using pseudorandom function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{X}$. Then, for any detIND-UE-CPA adversary $\mathcal{A}$ against $\mathsf{UE}^{\mathsf{new}}$ that asks at most $\mathrm{Q_E}$ queries to $\mathcal{O}.\mathsf{Enc}$ before it makes its challenge, there exists a PRF adversary $\mathcal{B}^{\mathsf{PRF}}$ against F and a detIND-UE-CPA adversary $\mathcal{B}_{2.8}$ against UE such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{UE}^{\mathsf{new}},\,\mathcal{A}}(\lambda) \leq (n+1) \cdot (\mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{UE},\,\mathcal{B}_{2.8}}(\lambda) + 2 \cdot \mathbf{Adv}^{\mathsf{PRF}}_{F,\,\mathcal{B}^{\mathsf{PRF}}} + \frac{2{\mathrm{Q_E}}^2}{|\mathcal{X}|}),$$

and there exists a randIND-UE-CPA adversary $\mathcal{C}$ against $\mathsf{UE}^{\mathsf{new}}$ that wins with probability 1.

Proof. $\underline{\mathsf{UE}^{\mathsf{new}}\text{ is not randIND-UE-CPA secure.}}$ If the token in the challenge epoch is corrupted then the adversary can compare r in the value it receives with the value it provided and therefore trivially win. So $\mathsf{UE}^{\mathsf{new}}$ is not randIND-UE-CPA secure.
$\underline{\mathsf{UE}^{\mathsf{new}}\text{ is detIND-UE-CPA secure.}}$ We proceed in three steps.

(Step 1.) Consider a modified version of detIND-UE-CPA. For $\mathrm{b} \in \{0,1\}$, define experiments $\mathbf{Exp}^{\mathsf{INT}_1\text{-}\mathsf{b}}$ to be the same as $\mathbf{Exp}^{\mathsf{randIND\text{-}UE\text{-}CPA\text{-}b}}$ except that the experiments randomly pick $\mathsf{e}^* \leftarrow \{0, ..., n\}$, and if $\mathsf{e}^* \neq \tilde{\mathsf{e}}$ the experiments return a random bit for b'. The loss is upper bounded by

```
$\mathsf{UE^{new}.KG}(\lambda)$
1 : $\mathrm{k} \xleftarrow{\$} \mathsf{UE.KG}(\lambda)$
2 : $\mathbf{return}\ \mathrm{k}$
$\mathsf{UE^{new}.TG}(\mathrm{k_e}, \mathrm{k_{e+1}})$
3 : $\Delta'_{\mathsf{e}+1} \leftarrow \mathsf{UE.TG}(\mathrm{k_e}, \mathrm{k_{e+1}})$
4 : $\mathrm{fk_{e+1}} \xleftarrow{\$} \mathcal{K}$
5 : $\mathbf{return}\ (\Delta'_{\mathsf{e}+1}, \mathrm{fk_{e+1}})$
$\mathsf{UE^{new}.Enc}(\mathrm{k_e}, \mathrm{M})$
6 : $r_{\mathsf{e}} \xleftarrow{\$} \mathcal{X}$
7 : $\mathrm{C}'_{\mathsf{e}} \xleftarrow{\$} \mathsf{UE.Enc}(\mathrm{k_e}, \mathrm{M})$
8 : $\mathbf{return}\ (r_{\mathsf{e}}, \mathrm{C}'_{\mathsf{e}})$
$\mathsf{UE^{new}.Dec}(\mathrm{k_e}, \mathrm{C_e})$
9 : $\mathbf{parse}\ \mathrm{C_e} = (r_{\mathsf{e}}, \mathrm{C}'_{\mathsf{e}})$
10 : $\mathrm{M}'/\perp \leftarrow \mathsf{UE.Dec}(\mathrm{k_e}, \mathrm{C}'_{\mathsf{e}})$
11 : $\mathbf{return}\ \mathrm{M}'$
$\mathsf{UE^{new}.Upd}(\Delta_{\mathsf{e}+1}, \mathrm{C_e})$
12 : $\mathbf{parse}\ \Delta_{\mathsf{e}+1} = (\Delta'_{\mathsf{e}+1}, \mathrm{fk_{e+1}})$
13 : $\mathbf{parse}\ \mathrm{C_e} = (r_{\mathsf{e}}, \mathrm{C}'_{\mathsf{e}})$
14 : $r_{\mathsf{e}+1} \leftarrow F(\mathrm{fk_{e+1}}, r_{\mathsf{e}})$
15 : $\mathrm{C}'_{\mathsf{e}+1} \leftarrow \mathsf{UE.Upd}(\Delta'_{\mathsf{e}+1}, \mathrm{C}'_{\mathsf{e}})$
16 : $\mathbf{return}\ (r_{\mathsf{e}+1}, \mathrm{C}'_{\mathsf{e}+1})$
```

Figure 22: Updatable encryption scheme $\mathsf{UE^{new}}$ for proof of Theorem 2.8, built from PRF F and updatable encryption scheme UE.

$n+1$. Then:

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{UE^{new}},\mathcal{A}}(\lambda) \leq (n+1) \cdot \mathbf{Adv}^{\mathsf{INT_1}}_{\mathsf{UE^{new}},\mathcal{A}}(\lambda).$$

(Step 2.) Then we consider modified experiments $\mathbf{Exp}^{\mathsf{INT_2\text{-}b}}$, which are the same as $\mathbf{Exp}^{\mathsf{INT_1\text{-}b}}$ except that the first element of ciphertexts in the guessed epoch e^* is a uniformly random element. We show that the ability to notice this change is upper bounded by PRF advantage. More precisely, experiment $\mathbf{Exp}^{\mathsf{INT_2\text{-}b}}$ tracks randomness in the set X (initialized as empty), and when adversary asks an $\mathcal{O}.\mathsf{Upd}$ or challenge query:

- For $\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}^*\text{-}1})$: parse $\Delta_{\mathsf{e}^*} = (\Delta'_{\mathsf{e}^*},)$, parse $\mathrm{C}_{\mathsf{e}^*\text{-}1} = (r_{\mathsf{e}^*\text{-}1}, \mathrm{C}'_{\mathsf{e}^*\text{-}1})$, if $r_{\mathsf{e}^*\text{-}1} \in X$, then the experiment aborts; otherwise, set $X \leftarrow X \cup \{r_{\mathsf{e}^*\text{-}1}\}$, randomly choose $r_{\mathsf{e}^*} \xleftarrow{\$} \mathcal{X}$. Return $(r_{\mathsf{e}^*}, \mathsf{UE.Upd}(\Delta'_{\mathsf{e}^*}, \mathrm{C}'_{\mathsf{e}^*\text{-}1}))$.
- For challenge input $(\bar{\mathrm{M}}, (r, \bar{\mathrm{C}}))$: parse $\Delta_{\mathsf{e}^*} = (\Delta'_{\mathsf{e}^*},)$, if $\mathsf{e}^* \neq \tilde{\mathsf{e}}$ or $r \in X$, then the experiment aborts; otherwise, set $X \leftarrow X \cup \{r\}$. If $\mathrm{b} = 0$, return $\mathsf{UE^{new}.Enc}(\mathrm{k}_{\mathsf{e}^*}, \bar{\mathrm{M}})$. If $\mathrm{b} = 1$, randomly choose $r_{\mathsf{e}^*} \xleftarrow{\$} \mathcal{X}$, return $(r_{\mathsf{e}^*}, \mathsf{UE.Upd}(\Delta'_{\mathsf{e}^*}, \bar{\mathrm{C}}))$.

```
Reduction $\mathcal{B}^{\mathsf{PRF}}$ playing $\mathbf{Exp}_{F,\,\mathcal{B}^{\mathsf{PRF}}}^{\mathsf{PRF\text{-}b}}$
1 :  do Setup
2 :  $\bar{\mathrm{M}}, (r, \bar{\mathrm{C}}) \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
3 :  $\mathsf{phase} \leftarrow 1$
4 :  Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}, (r, \bar{\mathrm{C}}))$, get $\tilde{\mathrm{C}}_{\mathsf{e}^*}$
5 :  $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}}_{\mathsf{e}^*})$
6 :  $\mathsf{twf} \leftarrow 1$ if
7 :      $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ or $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$
8 :  if ABORT occurred or $\mathsf{twf} = 1$
9 :      $\mathrm{b}' \xleftarrow{\$} \{0,1\}$
10 :     return $\mathrm{b}'$
11 : if $\mathrm{b}' = \mathrm{b}$
12 :     return 0
13 : else
14 :     return 1
```

Figure 23: Part 1. Reduction $\mathcal{B}^{\mathsf{PRF}}$ for proof of Theorem 2.8, $\mathcal{B}^{\mathsf{PRF}}$ simulates $\mathcal{O}.\mathsf{Enc}$, $\mathcal{O}.\mathsf{Next}$, $\mathcal{O}.\mathsf{Corr}$ and $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$ queries as described in Fig. 6.

Note that

$$\begin{aligned}\mathbf{Adv}_{\mathsf{UE}^{\mathsf{new}},\,\mathcal{A}}^{\mathsf{INT}_1}(\lambda) &= \left|\mathbf{Pr}[\mathbf{Exp}_{\mathsf{UE}^{\mathsf{new}},\,\mathcal{A}}^{\mathsf{INT}_1\text{-}1} = 1] - \mathbf{Pr}[\mathbf{Exp}_{\mathsf{UE}^{\mathsf{new}},\,\mathcal{A}}^{\mathsf{INT}_1\text{-}0} = 1]\right| \\ &\le \mathbf{Adv}_{\mathsf{UE}^{\mathsf{new}},\,\mathcal{A}}^{\mathsf{INT}_2}(\lambda) \\ &\quad + \left|\mathbf{Pr}[\mathbf{Exp}_{\mathsf{UE}^{\mathsf{new}},\,\mathcal{A}}^{\mathsf{INT}_2\text{-}1} = 1] - \mathbf{Pr}[\mathbf{Exp}_{\mathsf{UE}^{\mathsf{new}},\,\mathcal{A}}^{\mathsf{INT}_1\text{-}1} = 1]\right| \\ &\quad + \left|\mathbf{Pr}[\mathbf{Exp}_{\mathsf{UE}^{\mathsf{new}},\,\mathcal{A}}^{\mathsf{INT}_2\text{-}0} = 1] - \mathbf{Pr}[\mathbf{Exp}_{\mathsf{UE}^{\mathsf{new}},\,\mathcal{A}}^{\mathsf{INT}_1\text{-}0} = 1]\right|\end{aligned}$$

For $\mathrm{b} \in \{0,1\}$, we wish to prove that

$$\left|\mathbf{Pr}[\mathbf{Exp}_{\mathsf{UE}^{\mathsf{new}},\,\mathcal{A}}^{\mathsf{INT}_2\text{-}\mathrm{b}} = 1] - \mathbf{Pr}[\mathbf{Exp}_{\mathsf{UE}^{\mathsf{new}},\,\mathcal{A}}^{\mathsf{INT}_1\text{-}\mathrm{b}} = 1]\right| \le \mathbf{Adv}_F^{\mathsf{PRF}} + \frac{\mathrm{Q_E}^2}{|\mathcal{X}|}.$$

Suppose $\mathcal{A}$ is an adversary who is trying to distinguish $\mathbf{Exp}_{\mathsf{UE}^{\mathsf{new}},\,\mathcal{A}}^{\mathsf{INT}_2\text{-}\mathrm{b}}$ from $\mathbf{Exp}_{\mathsf{UE}^{\mathsf{new}},\,\mathcal{A}}^{\mathsf{INT}_1\text{-}\mathrm{b}}$. We construct a PRF reduction $\mathcal{B}^{\mathsf{PRF}}$, detailed in Fig. 23,

```
Setup(λ)
15 : $\mathrm{k}_0 \leftarrow \mathsf{UE.KG}(\lambda); \Delta_0 \leftarrow \perp;\ \ \mathsf{e} \leftarrow 0;\ \ \mathsf{phase}, \mathsf{twf} \leftarrow 0; \mathsf{e}^* \xleftarrow{\$} \{0, \ldots, n\}$
16 : $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T}, X \leftarrow \emptyset$
$\mathcal{O}.\mathsf{Next}^{\dagger}$
$\mathcal{O}.\mathsf{Upd}(r_{\mathsf{e}-1}, \mathrm{C}'_{\mathsf{e}-1})$
17 : if $(\cdot, (r_{\mathsf{e}-1}, \mathrm{C}'_{\mathsf{e}-1}), \mathsf{e}-1) \notin \mathcal{L}$
18 :     return $\perp$
19 : if $\mathsf{e} \neq \mathsf{e}^*$
20 :     $\mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{UE}^{\mathsf{new}}.\mathsf{Upd}((\Delta'_{\mathsf{e}}, \mathrm{fk}_{\mathsf{e}}), (r_{\mathsf{e}-1}, \mathrm{C}'_{\mathsf{e}-1}))$
21 : if $\mathsf{e} = \mathsf{e}^*$
22 :     if $r_{\mathsf{e}-1} \in X$
23 :         return ABORT
24 :     $r_{\mathsf{e}} \leftarrow \mathcal{O}.f(r_{\mathsf{e}-1})$    // embed
25 :     $X \leftarrow X \cup \{r_{\mathsf{e}-1}\}; \mathrm{C}'_{\mathsf{e}} \leftarrow \mathsf{UE.Upd}(\Delta'_{\mathsf{e}}, \mathrm{C}'_{\mathsf{e}-1}); \mathrm{C}_{\mathsf{e}} \leftarrow (r_{\mathsf{e}}, \mathrm{C}'_{\mathsf{e}})$
26 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_{\mathsf{e}}, \mathsf{e})\}$
27 : return $\mathrm{C}_{\mathsf{e}}$
Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}, (r, \bar{\mathrm{C}}))$
28 : if $(\cdot, (r, \bar{\mathrm{C}}), \tilde{\mathsf{e}}-1) \notin \mathcal{L}$ or $\mathsf{e}^* \neq \tilde{\mathsf{e}}$ or $r \in X$
29 :     return $\perp$
30 : $X \leftarrow X \cup \{r\}$
31 : if $\mathrm{b} = 0$
32 :     $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}} \leftarrow \mathsf{UE}^{\mathsf{new}}.\mathsf{Enc}(\mathrm{k}_{\tilde{\mathsf{e}}}, \bar{\mathrm{M}})$
33 : else
34 :     $r_{\tilde{\mathsf{e}}} \leftarrow \mathcal{O}.f(r)$    // embed
35 :     $\tilde{\mathrm{C}}'_{\tilde{\mathsf{e}}} \leftarrow \mathsf{UE.Upd}(\Delta'_{\tilde{\mathsf{e}}}, \bar{\mathrm{C}}); \tilde{\mathrm{C}}_{\tilde{\mathsf{e}}} \leftarrow (r_{\tilde{\mathsf{e}}}, \tilde{\mathrm{C}}'_{\tilde{\mathsf{e}}})$
36 : return $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
```

Figure 23: Part 2. Reduction $\mathcal{B}^{\mathsf{PRF}}$ for proof of Theorem 2.8, $\mathcal{B}^{\mathsf{PRF}}$ simulates $\mathcal{O}.\mathsf{Enc}$, $\mathcal{O}.\mathsf{Next}$, $\mathcal{O}.\mathsf{Corr}$ and $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$ queries as described in Fig. 6. Recall that the PRF advantage in Definition 3, $\mathcal{O}.f$ replies with $\mathsf{F}(\mathrm{k}, r)$ or a random value. † indicates $\mathrm{fk}_{\mathsf{e}^*}$ are skipped in the generation.

against F to simulate the responses of queries made by $\mathcal{A}$. $\mathcal{B}^{\mathsf{PRF}}$ first guesses when $\mathcal{A}$ is going to ask a challenge query (assume e^*) and in that epoch $\mathcal{B}^{\mathsf{PRF}}$ does bookkeeping for the randomness in X (initialized as empty set). Note that the reduction generates all keys and tokens except for $\mathrm{fk}_{\mathsf{e}^*}$. Update randomness in epoch e^* is simulated by sending the randomness to the PRF challenger and forwarding the response to $\mathcal{A}$.

Eventually $\mathcal{B}^{\mathsf{PRF}}$ receives the guess from $\mathcal{A}$, and outputs 0 if $\mathcal{A}$ guesses that it is in $\mathbf{Exp}^{\mathsf{INT}_1\text{-b}}$, and 1 if $\mathcal{A}$ guesses that it is in $\mathbf{Exp}^{\mathsf{INT}_2\text{-b}}$.

When $\mathcal{B}^{\mathsf{PRF}}$ interacts with $\mathbf{Exp}^{\mathsf{PRF}\text{-}0}$, it can simulate $\mathbf{Exp}^{\mathsf{INT}_1\text{-b}}$ perfectly except with a negligible probability. The negligible term is due to $\mathcal{B}^{\mathsf{PRF}}$ aborts the game. Since the number of existed randomnesses is small compared to the number of possible random elements, the probability that $\mathcal{B}^{\mathsf{PRF}}$ aborts the game is upper bounded by $\frac{\mathrm{Q_E}^2}{|\mathcal{X}|}$. When $\mathcal{B}^{\mathsf{PRF}}$ interacts with $\mathbf{Exp}^{\mathsf{PRF}\text{-}1}$, it can perfectly simulate $\mathbf{Exp}^{\mathsf{INT}_2\text{-b}}$, thus we have the required result.

(Step 3.) Now we conclude that the advantage of winning INT_2 is upper bounded by $\mathsf{detIND\text{-}UE\text{-}CPA}$ advantage (against UE). Suppose an INT_2 adversary $\mathcal{A}$ is trying to attack $\mathsf{UE}^{\mathsf{new}}$. We construct a $\mathsf{detIND\text{-}UE\text{-}CPA}$ reduction $\mathcal{B}_{2.8}$, detailed in Fig. 24, attacking UE and runs $\mathcal{A}$. $\mathcal{B}_{2.8}$ first guesses when $\mathcal{A}$ is going to ask a challenge query (assume e^*) and in that

```
Reduction $\mathcal{B}_{2.8}$ playing $\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{UE},\ \mathcal{B}_{2.8}}$
1 :  do Setup
2 :  $\bar{\mathrm{M}}, (r, \bar{\mathrm{C}}) \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
3 :  $\mathsf{phase} \leftarrow 1$
4 :  Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}, (r, \bar{\mathrm{C}}))$, get $\tilde{\mathrm{C}}_{\mathsf{e}^*}$
5 :  $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}}_{\mathsf{e}^*})$
6 :  $\underline{\mathsf{twf} \leftarrow 1 \textbf{ if}}$
7 :     $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ or $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$
8 :  if ABORT occurred or $\mathsf{twf} = 1$
9 :     $\mathrm{b}' \xleftarrow{\$} \{0, 1\}$
10 : return $\mathrm{b}'$
```

Figure 24: Part 1. Reduction $\mathcal{B}_{2.8}$ for proof of Theorem 2.8.

Setup(λ)

```
11 :  $\mathrm{k}_0 \leftarrow \mathsf{UE.KG}(\lambda), \mathsf{e}^* \xleftarrow{\$} \{0, \ldots, n\}$
12 :  $\Delta_0 \leftarrow \perp;\ \mathsf{e} \leftarrow 0;\ \mathsf{phase}, \mathsf{twf} \leftarrow 0$
13 :  $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T}, X \leftarrow \emptyset$
```

$\mathcal{O}.\mathsf{Enc}(\mathrm{M})$

```
14 :  call $\mathcal{O}.\mathsf{Enc}(\mathrm{M})$, get $\mathrm{C}'$
15 :  $r \xleftarrow{\$} \mathcal{X}$
16 :  return $(r, \mathrm{C}')$
```

$\mathcal{O}.\mathsf{Next}$

```
17 :  call $\mathcal{O}.\mathsf{Next}$
18 :  $\mathrm{fk}_{\mathsf{e}+1} \xleftarrow{\$} \mathcal{K}$
19 :  if $\mathsf{phase} = 1$
20 :     $\tilde{r}_{\mathsf{e}+1} = F(\mathrm{fk}_{\mathsf{e}+1}, \tilde{r}_{\mathsf{e}})$
21 :     $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{((\tilde{r}_{\mathsf{e}+1}, \cdot), \mathsf{e}+1)\}$
```

$\mathcal{O}.\mathsf{Upd}((r_{\mathsf{e}-1}, \mathrm{C}'_{\mathsf{e}-1}))$

```
22 :  if $(\cdot, (r_{\mathsf{e}-1}, \mathrm{C}'_{\mathsf{e}-1}), \mathsf{e}-1) \notin \mathcal{L}$
23 :     return $\perp$
24 :  call $\mathcal{O}.\mathsf{Upd}(\mathrm{C}'_{\mathsf{e}-1})$, get $\mathrm{C}'_{\mathsf{e}}$
25 :  if $\mathsf{e} \neq \mathsf{e}^*$
26 :     $r_{\mathsf{e}} = F(\mathrm{fk}_{\mathsf{e}}, r)$
27 :  if $\mathsf{e} = \mathsf{e}^*$
28 :     if $r_{\mathsf{e}-1} \in X$
29 :        return ABORT
30 :     $X \leftarrow X \cup \{r_{\mathsf{e}-1}\}, r_{\mathsf{e}} \xleftarrow{\$} \mathcal{X}$
31 :  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, (r_{\mathsf{e}}, \mathrm{C}'_{\mathsf{e}}), \mathsf{e})\}$
32 :  return $(r_{\mathsf{e}}, \mathrm{C}'_{\mathsf{e}})$
```

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$

```
33 :  call $\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$
34 :  get $\perp$ or $\mathrm{k}_{\hat{\mathsf{e}}}$ or $\Delta'_{\hat{\mathsf{e}}}$
35 :  return $\perp$ or $\mathrm{k}_{\hat{\mathsf{e}}}$ or $(\Delta'_{\hat{\mathsf{e}}}, \mathrm{fk}_{\hat{\mathsf{e}}})$
```

Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}, (r, \bar{\mathrm{C}}))$

```
36 :  if $(\cdot, (r, \bar{\mathrm{C}}), \tilde{\mathsf{e}}-1) \notin \mathcal{L}$
37 :    or $\mathsf{e}^* \neq \tilde{\mathsf{e}}$ or $r \in X$
38 :      return $\perp$
39 :  $X \leftarrow X \cup \{r\}$
40 :  call cq with $(\bar{\mathrm{M}}, \bar{\mathrm{C}})$
41 :  get $\tilde{\mathrm{C}}'$   // embed
42 :  $\tilde{r}_{\tilde{\mathsf{e}}} \xleftarrow{\$} \mathcal{X}$
43 :  $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{((\tilde{r}_{\tilde{\mathsf{e}}}, \tilde{\mathrm{C}}'), \tilde{\mathsf{e}})\}$
44 :  return $(\tilde{r}_{\tilde{\mathsf{e}}}, \tilde{\mathrm{C}}')$
```

$\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$

```
45 :  call $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$, get $\tilde{\mathrm{C}}'_{\mathsf{e}}$   // embed
46 :  $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{((\tilde{r}_{\mathsf{e}}, \tilde{\mathrm{C}}'_{\mathsf{e}}), \mathsf{e})\}$
47 :  return $(\tilde{r}_{\mathsf{e}}, \tilde{\mathrm{C}}'_{\mathsf{e}})$
```

Figure 24: Part 2. Reduction $\mathcal{B}_{2.8}$ for proof of Theorem 2.8. In line 40 of the Create $\tilde{\mathrm{C}}$ description, **call cq** means that the reduction makes its own challenge query with these values.

epoch $\mathcal{B}_{2.8}$ does bookkeeping for the randomness in X (initialized as empty set).

Eventually $\mathcal{B}_{2.8}$ receives b' from $\mathcal{A}$, and simply outputs b'. Then $\mathcal{B}_{2.8}$ perfectly simulates $\mathbf{Exp}^{\mathsf{INT}_2\text{-b}}$ to $\mathcal{A}$. We have the required result

$$\mathbf{Adv}^{\mathsf{INT}_2}_{\mathsf{UE}^{\mathsf{new}},\, \mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{UE},\, \mathcal{B}_{2.8}}(\lambda). \qquad \square$$

4.2.3 Relation among CPA, CTXT and CCA Security

Theorem 3. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. For any detIND-yy-CCA adversary $\mathcal{A}$ against UE, there exists an INT-CTXT adversary $\mathcal{B}_{3a}$ and an detIND-yy-CPA adversary $\mathcal{B}_{3b}$ against UE such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}yy\text{-}CCA}}_{\mathsf{UE},\, \mathcal{A}}(\lambda) \leq 2\mathbf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{UE},\, \mathcal{B}_{3a}}(\lambda) + \mathbf{Adv}^{\mathsf{detIND\text{-}yy\text{-}CPA}}_{\mathsf{UE},\, \mathcal{B}_{3b}}(\lambda)$$

where $\mathsf{yy} \in \{\mathsf{UE}, \mathsf{ENC}, \mathsf{UPD}\}$.

We now prove Theorem 3, which states that, for $\mathsf{yy} \in \{\mathsf{UE}, \mathsf{ENC}, \mathsf{UPD}\}$, the combination of detIND-yy-CPA security and INT-CTXT security yields detIND-yy-CCA. The proof proceeds via a single game hop.

Proof. **Game 0**

The first game is the experiment $\mathbf{Exp}^{\mathsf{detIND\text{-}yy\text{-}CCA}}_{\mathsf{UE},\, \mathcal{A}}$, given in Fig. 11 (or Fig. 12 or Fig. 17). From Def. 5 (or Def. 6 or Def. 8) we have

$$\mathbf{Adv}^{\mathsf{detIND\text{-}yy\text{-}CCA}}_{\mathsf{UE},\, \mathcal{A}}(\lambda) = 2\Big|\mathbf{Pr}[\mathcal{G}_0 = 1] - 1/2\Big|.$$

Game 1

In this game we introduce an event `bad` that is triggered if the adversary asks its decryption oracle for something that would count as a forgery, and then show that the success probability of a distinguisher between the two games is bounded by INT-CTXT. Then, we bound the success probability in this modified game by an adversary against detIND-yy-CCA. We modify $\mathcal{O}$.Dec, such that the boxed statements only run in Game 1:

```
$\mathcal{O}.\mathsf{Dec}(\mathrm{C})$
1 :  $\mathbf{if}\ \mathsf{phase} \leftarrow 1\ \mathbf{and}\ \mathrm{C} \in \tilde{\mathcal{L}}^*$
2 :    $\mathbf{return}\ \perp$
3 :  $\mathrm{M}/\perp \leftarrow \mathsf{UE.Dec}(\mathrm{k_e}, \mathrm{C})$
4 :  $\boxed{\mathbf{if}\ \mathrm{C} \notin \mathcal{L}^*\ \mathbf{and}\ \mathrm{M}' \neq \perp}$
5 :    $\boxed{\mathsf{bad} \leftarrow \mathsf{true}}$
6 :    $\boxed{\mathbf{return} \perp}$
```

We have

$$\Big|\mathbf{Pr}[\mathcal{G}_0 = 1] - 1/2\Big| \leq \Big|\mathbf{Pr}[\mathcal{G}_0 = 1] - \mathbf{Pr}[\mathcal{G}_1 = 1]\Big| + \Big|\mathbf{Pr}[\mathcal{G}_1 = 1] - 1/2\Big|$$

We claim that there exists an INT-CTXT adversary $\mathcal{B}_{3a}$ against UE such that

$$\Big|\mathbf{Pr}[\mathcal{G}_0 = 1] - \mathbf{Pr}[\mathcal{G}_1 = 1]\Big| \leq \mathbf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{UE},\, \mathcal{B}_{3a}}(\lambda),$$

and there exists an detIND-yy-CPA adversary $\mathcal{B}_{3b}$ such that

$$2\Big|\mathbf{Pr}[\mathcal{G}_1 = 1] - 1/2\Big| \leq \mathbf{Adv}^{\mathsf{detIND\text{-}yy\text{-}CPA}}_{\mathsf{UE},\, \mathcal{B}_{3b}}(\lambda).$$

Claim 1.

$$\Big|\mathbf{Pr}[\mathcal{G}_0 = 1] - \mathbf{Pr}[\mathcal{G}_1 = 1]\Big| = \mathbf{Pr}[\mathsf{bad} = \mathsf{true\ in}\ \mathcal{G}_1] = \mathbf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{UE},\, \mathcal{B}_{3a}}(\lambda).$$

We construct a reduction $\mathcal{B}_{3a}$ playing INT-CTXT that simulates the environment of $\mathcal{G}_1$ to $\mathcal{A}$. $\mathcal{B}_{3a}$ starts by picking a random bit b, then runs $\mathcal{A}$ answering its queries as follows. For a challenge query with input ($\tilde{\mathrm{C}}_0$ or $\bar{\mathrm{M}}_0$, $\tilde{\mathrm{C}}_1$ or $\bar{\mathrm{M}}_1$). If b =0, $\mathcal{B}_{3a}$ sends $\tilde{\mathrm{C}}_0$ (or $\bar{\mathrm{M}}_0$) to its $\mathcal{O}$.Upd (or $\mathcal{O}$.Enc); b =1, $\mathcal{B}_{3a}$ sends $\tilde{\mathrm{C}}_1$ (or $\bar{\mathrm{M}}_1$) to its $\mathcal{O}$.Upd (or $\mathcal{O}$.Enc); eventually, returns the response to $\mathcal{A}$. Furthermore, $\mathcal{B}_{3a}$ simulates $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$ by sending the challenge ciphertext to $\mathcal{O}$.Upd, and forwards the response to $\mathcal{A}$.

For a decryption query $\mathcal{O}$.Dec with input C: If $\mathrm{C} \in \mathcal{L}^*$, $\mathcal{B}_{3a}$ checks the corresponding message in list $\mathcal{L}^*$ (adversary $\mathcal{B}_{3a}$ does bookkeeping for this list and additionally stores message in this list, list is updated by $\mathcal{O}$.Enc, $\mathcal{O}$.Upd, $\mathcal{O}$.Corr. Hence this simulation is feasible by an INT-CTXT adversary.), and returns it to $\mathcal{A}$. If $\mathrm{C} \notin \mathcal{L}^*$, $\mathcal{B}_{3a}$ returns $\perp$ to $\mathcal{A}$, and sends C to its $\mathcal{O}$.Try oracle.

$\mathcal{B}_{3a}$ perfectly simulates $\mathcal{G}_1$. Notice that $\mathcal{G}_0$ and $\mathcal{G}_1$ are identical until $\mathsf{UE.Dec}(\mathrm{k_e}, \mathrm{C}) \neq \perp$ and $\mathrm{C} \notin \mathcal{L}^*$ happens (which causes $\mathsf{bad} = \mathsf{true}$ in $\mathcal{G}_1$): denote this event to be E. Thus, we have $\left|\mathbf{Pr}[\mathcal{G}_0 = 1] - \mathbf{Pr}[\mathcal{G}_1 = 1]\right| = \mathbf{Pr}[\mathsf{bad} = \mathsf{true}\text{ in }\mathcal{G}_1] = \mathbf{Pr}[E]$. If event E happens, which results in $\mathsf{win} = 1$ in the $\mathsf{INT\text{-}CTXT}$ game, that means C is a valid forgery in $\mathsf{INT\text{-}CTXT}$ game. So $\mathbf{Pr}[E] = \mathbf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{UE},\, \mathcal{B}_{3a}}(\lambda)$.
<u>Claim 2.</u>

$$2\left|\mathbf{Pr}[\mathcal{G}_1 = 1] - 1/2\right| \leq \mathbf{Adv}^{\mathsf{detIND\text{-}yy\text{-}CPA}}_{\mathsf{UE},\, \mathcal{B}_{3b}}(\lambda).$$

We only need to consider how the $\mathsf{detIND\text{-}yy\text{-}CPA}$ adversary $\mathcal{B}_{3b}$ simulates the $\mathcal{O}.\mathsf{Dec}$ oracle. Since $\mathcal{B}_{3b}$ knows $\tilde{\mathcal{C}}^*$ and $\mathcal{L}^*$, whenever $\mathcal{A}$ asks $\mathcal{O}.\mathsf{Dec}$ with C, $\mathcal{B}_{3b}$ checks if $\mathrm{C} \in \tilde{\mathcal{C}}^*$ or $\mathrm{C} \notin \mathcal{L}^*$, and if so, responds $\perp$. Otherwise, $\mathcal{B}_{3b}$ checks the corresponding message in list $\mathcal{L}^*$ (adversary $\mathcal{B}_{3b}$ does bookkeeping for this list and additionally stores message in this list as well, list is updated by $\mathcal{O}.\mathsf{Enc}$, $\mathcal{O}.\mathsf{Upd}$, $\mathcal{O}.\mathsf{Corr}$. Hence this simulation is feasible by an $\mathsf{detIND\text{-}yy\text{-}CPA}$ adversary), and returns it to $\mathcal{A}$. □

5 The SHINE Schemes

We now describe our new UE scheme SHINE (<u>S</u>ecure <u>H</u>omomorphic <u>I</u>deal-cipher <u>N</u>once-based <u>E</u>ncryption). The encryption algorithm uses a permutation to obfuscate the input to the exponentiation function. Updating a ciphertext simply requires exponentiation once by the update token, which itself is the quotient of the current epoch key and the previous epoch key. The scheme comes in three flavors: SHINE0 is presented in Fig. 25 and takes in short messages and only uses a single permutation. The second flavor, MirrorSHINE, is provided in Fig. 26 and runs two different permutations with the same input. The third flavor OCBSHINE is given in Fig. 28 and is for applications with arbitrarily long messages, using a family of permutations.

We discuss implementation details of the SHINE schemes in Section 5.6. In particular, for each scheme in the SHINE suite, it is necessary to embed the output of the permutation (a regular block cipher) into an appropriate DDH-hard group.

Our proofs of security, given as Theorem 4 for confidentiality and Theorem 5 for integrity, bound an adversary's $\mathsf{detIND\text{-}UE\text{-}CPA}$ ($\mathsf{INT\text{-}CTXT^s}$)

advantage by DDH (CDH), and are provided in the ideal cipher model. Furthermore, combining the results of Theorem 3, Theorem 4 and Theorem 5, we have that the suite of SHINE schemes (i.e. SHINE0, MirrorSHINE and OCBSHINE) are detIND-UE-CCA secure.

5.1 Construction of SHINE Schemes

5.1.1 SHINE via Zero Block: SHINE0.

Suppose a message space of $\mathcal{MS} = \{0,1\}^{\mathrm{m}}$ and random nonce space $\mathcal{N} = \{0,1\}^{\mathrm{v}}$. The encryption algorithm feeds as input to the permutation a nonce, the message, and a zero string. The decryption algorithm will return $\perp$ if the decrypted value does not end with 0^{t}. The SHINE0 scheme is defined in Fig. 25. If ciphertext integrity is not required (or file/ciphertext integrity is performed in some other manner), then SHINE0 without the zero block results in a scheme (denoted SHINE0[CPA]) that is still detIND-UE-CPA secure.

```
SHINE0.KG(λ)
1 :  $\mathrm{k} \xleftarrow{\$} \mathbb{Z}_q^*$
2 :  return $\mathrm{k}$
SHINE0.TG($\mathrm{k_e}, \mathrm{k_{e+1}}$)
3 :  $\Delta_{\mathrm{e+1}} \leftarrow \frac{\mathrm{k_{e+1}}}{\mathrm{k_e}}$
4 :  return $\Delta_{\mathrm{e+1}}$
SHINE0.Enc($\mathrm{k_e}, \mathrm{M}$)
5 :  $\mathrm{N} \xleftarrow{\$} \mathcal{N}$
6 :  $\mathrm{C_e} \leftarrow (\pi(\mathrm{N}\|\mathrm{M}\|0^{\mathrm{t}}))^{\mathrm{k_e}}$
7 :  return $\mathrm{C_e}$

SHINE0.Dec($\mathrm{k_e}, \mathrm{C_e}$)
8 :  $a \leftarrow \pi^{-1}(\mathrm{C_e}^{1/\mathrm{k_e}})$
9 :  parse† $a$ as $\mathrm{N}'\|\mathrm{M}'\|\mathrm{Z}$
10 : if $\mathrm{Z} = 0^{\mathrm{t}}$
11 :     return $\mathrm{M}'$
12 : else
13 :     return $\perp$
SHINE0.Upd($\Delta_{\mathrm{e+1}}, \mathrm{C_e}$)
14 : $\mathrm{C_{e+1}} \leftarrow (\mathrm{C_e})^{\Delta_{\mathrm{e+1}}}$
15 : return $\mathrm{C_{e+1}}$
```

Figure 25: Updatable encryption scheme SHINE0. Note that there may be an additional embedding step after the permutation π, as discussed in Section 5.6. †: $\|\mathrm{N}'\| = \mathrm{v}$, $\|\mathrm{M}'\| = \mathrm{m}$, $\|\mathrm{Z}\| = \mathrm{t}$.

5.1.2 SHINE via Double Encryption: MirrorSHINE

The construction of MirrorSHINE is similar to SHINE0, except that instead of padding a zero block after the message, the encryption algorithm processes each message and nonce twice using two different random permutations. For authentication, we compare if two ciphertext blocks have the same underlying message and nonce. The idea of the authentication is similar to SHINE0, however here we use the difference of the underlying message and nonce as the "zero block". Compared to SHINE0, MirrorSHINE does one more exponentiation in both encryption and update and requires two ciphertext elements per message, but larger messages are supported (when using the same – for example standardized – group).

```
MirrorSHINE.KG(λ)
1 :  $k \xleftarrow{\$} \mathbb{Z}_q^*$
2 :  return $k$
MirrorSHINE.TG($k_e, k_{e+1}$)
3 :  $\Delta_{e+1} \leftarrow \frac{k_{e+1}}{k_e}$
4 :  return $\Delta_{e+1}$
MirrorSHINE.Enc($k_e$, M)
5 :  $N \xleftarrow{\$} \mathcal{N}$
6 :  $C_e^1 \leftarrow (\pi_1(N||M))^{k_e}$
7 :  $C_e^2 \leftarrow (\pi_2(N||M))^{k_e}$
8 :  $C_e \leftarrow (C_e^1, C_e^2)$
9 :  return $C_e$

MirrorSHINE.Dec($k_e, C_e$)
10 :  parse $C_e = (C_e^1, C_e^2)$
11 :  $a_1 \leftarrow \pi_1^{-1}((C_e^1)^{1/k_e})$
12 :  $a_2 \leftarrow \pi_2^{-1}((C_e^2)^{1/k_e})$
13 :  parse‡ $a_1$ as $N'||M'$
14 :  if $a_1 = a_2$
15 :    return $M'$
16 :  else
17 :    return $\perp$
MirrorSHINE.Upd($\Delta_{e+1}, C_e$)
18 :  parse $C_e = (C_e^1, C_e^2)$
19 :  $C_{e+1}^1 \leftarrow (C_e^1)^{\Delta_{e+1}}$
20 :  $C_{e+1}^2 \leftarrow (C_e^2)^{\Delta_{e+1}}$
21 :  return $C_{e+1}^1, C_{e+1}^2$
```

Figure 26: Updatable encryption scheme MirrorSHINE, where π_1, π_2 are two different random permutations. Note that there may be an additional embedding step after the permutations π_1 and π_2, as discussed in Section 5.6. ‡: $\|N'\| = v$, $\|M'\| = m$.

5.1.3 SHINE for Long Messages via Checksum: OCBSHINE.

The schemes SHINE0 and MirrorSHINE both require that the message space be smaller than the size of an element of the exponentiation group. This ciphertext expansion is undesirable in many practical scenarios, and so we wish to construct a SHINE scheme which gives us (almost) no ciphertext expansion and can be applied to arbitrarily long messages. We build a new SHINE scheme, OCBSHINE, with these properties.

The construction of OCBSHINE is inspired by the authenticated encryption scheme OCB [RBBK01]. Different from OCB mode, the nonce is encrypted inside the ciphertext instead of sending it along with the ciphertext. In order to determine the length of the last message block, the encryption algorithm of OCB mode removes some bits of the last ciphertext block to reveal this information. However in our setting, the output of the permutations are (mapped to) the input of the exponentiation function: thus all bits of permutation outputs must be included. Therefore, OCBSHINE includes the length of the last message block in the first ciphertext component. If ciphertext integrity is not required, then OCBSHINE can be improved by removing the last ciphertext block.

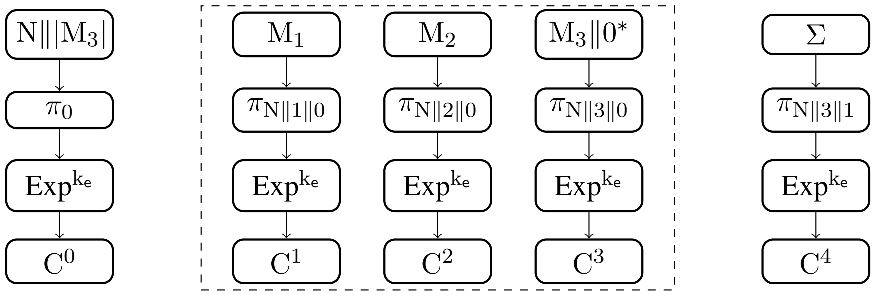

Figure 27: Diagram describing how the OCBSHINE encryption algorithm works on message $\mathrm{M} = (\mathrm{M}_1, \mathrm{M}_2, \mathrm{M}_3)$. $\Sigma = \mathrm{M}_1 \oplus \mathrm{M}_2 \oplus \mathrm{M}_3\|0^*$. There may be an additional embedding step after the permutations, as discussed in Section 5.6.

OCBSHINE is formally defined in Fig. 28 and the encryption process is pictorially represented in Fig. 27; we give an intuitive description here. Suppose the blocksize is m, and the encryption algorithm OCBSHINE.Enc has input message M. By "partition M into $M_1, \ldots, M_l$" we mean setting $l \leftarrow \max\{\lceil |M|/m \rceil, 1\}$ and dividing M into l blocks, i.e. $M_1, \ldots, M_l$, where $|M_1| = \ldots = |M_{l-1}| = m$. The last message block M_l is padded with zeros to make it length m before computing the permutation output and the checksum, i.e $M_l \| 0^*$ with $|M_l \| 0^*| = m$. Let $a = \lceil \log(m) \rceil$, so the length of M_l ($|M_l| \leq m$) can be written as an a-bit representation.

```
OCBSHINE.KG($\lambda$)
1 :  $k \xleftarrow{\$} \mathbb{Z}_q^*$
2 :  return $k$

OCBSHINE.TG($k_e, k_{e+1}$)
3 :  $\Delta_{e+1} \leftarrow \frac{k_{e+1}}{k_e}$
4 :  return $\Delta_{e+1}$

OCBSHINE.Enc($k_e$, M)
5 :  partition M into $M_1, \ldots, M_l$
6 :  $\Sigma \leftarrow \oplus_{i=1}^{l\text{-}1} M_i \oplus M_l \| 0^*$
7 :  $N \xleftarrow{\$} \mathcal{N}$
8 :  $C^0 \leftarrow (\pi_0(N \| |M_l|))^{k_e}$
9 :  $C^{l+1} \leftarrow (\pi_{N\|l\|1}(\Sigma))^{k_e}$
10 : for $i = 1, \ldots, l\text{-}1$ do
11 :     $C^i \leftarrow (\pi_{N\|i\|0}(M_i))^{k_e}$
12 : $C^l \leftarrow (\pi_{N\|l\|0}(M_l \| 0^*))^{k_e}$
13 : $C_e \leftarrow (C^0, \ldots, C^l, C^{l+1})$
14 : return $C_e$

OCBSHINE.Dec($k_e, C_e$)
15 : parse $C_e = (C^0, \ldots, C^l, C^{l+1})$
16 : $N' \| A' \leftarrow \pi_0^{-1}((C^0)^{1/k_e})$
17 : $\Sigma' \leftarrow \pi_{N'\|l\|1}^{-1}((C^{l+1})^{1/k_e})$
18 : for $i = 1, \ldots, l$ do
19 :     $M_i' \leftarrow \pi_{N'\|i\|0}^{-1}((C^i)^{1/k_e})$
20 : if $\Sigma' = \oplus_{i=1}^{l} M_i'$
21 :     $M' \leftarrow (M_1', \ldots, M_l'[x])$
22 :     return $M'$
23 : else
24 :     return $\perp$

OCBSHINE.Upd($\Delta_{e+1}, C_e$)
25 : parse $C_e = (C^0, \ldots, C^l, C^{l+1})$
26 : for $i = 0, \ldots, l + 1$ do
27 :     $C_{e+1}^i \leftarrow (C_e^i)^{\Delta_{e+1}}$
28 : return $C_{e+1}$
```

Figure 28: Updatable encryption scheme OCBSHINE. Note that there may be an additional embedding step after the permutations, as discussed in Section 5.6. In line 21, $M_l[x]$ represents the first A' bits of M_l.

Let $\mathrm{Perm}(m)$ be the set of all permutations on $\{0, 1\}^m$. Randomly choose $\pi_0 \xleftarrow{\$} \mathrm{Perm}(m)$, and use this permutation to randomize the concatenation of the nonce N and an a-bit representation of the last message block length. Then, index the (random) permutations used to encrypt message blocks by the nonce and a counter. Let $\mathrm{Perm}(S, m)$ be the set of all mappings from S to permutations on $\{0,1\}^m$. Suppose the nonce space is $\mathcal{N} = \{0,1\}^{m-a}$, $S = \mathcal{N} \times \mathbb{N}^* \times \{0,1\}$, for each $(\mathrm{N} \in \mathcal{N}, i \in \mathbb{N}^*, b \in \{0,1\})$, set $\pi_{\mathrm{N}\|i\|b} \xleftarrow{\$} \mathrm{Perm}(\mathcal{N} \times \mathbb{N}^* \times \{0,1\}, m)$, which form a random permutation family: we use these permutations to randomize message blocks and the checksum.

5.2 Security of SHINE

All three SHINE schemes, i.e. SHINE0, MirrorSHINE and OCBSHINE, have the same security properties, and the proofs are very similar for each flavor. We refer to SHINE to mean the family containing all these three schemes. In Theorem 4, we show that SHINE is detIND-UE-CPA in the ideal cipher model, if DDH holds. In Theorem 5, we show that SHINE is INT-CTXT$^{\mathsf{s}}$, and therefore INT-CTXT (INT-CTXT and INT-CTXT$^{\mathsf{s}}$ are equivalent, recall Section 3), in the ideal cipher model, if CDH holds. The loss incurred by this proof is the normal $(n+1)^3$ (or $(n+1)^2$ for INT-CTXT) and also the number of encryption queries the adversary makes before it makes its challenge: to avoid the issues described in Section 5.3 we not only need to guess the locations of the challenge firewalls but also the ciphertext that the adversary will submit as its challenge.

The ideal cipher model, a version of which was initially given by Shannon [Sha49] and shown to be equivalent to the random oracle model by Coron et al. [CPS08], gives all parties access to a permutation chosen randomly from all possible key-permutation possibilities of appropriate length. The SHINE schemes exponentiate the output of the permutation by the epoch key to encrypt, so our reduction can ‘program’ the transformation from permutation outputs to group elements.

In the following two Theorems we detail the security properties met by SHINE, i.e. detIND-UE-CPA, INT-CTXT and thus detIND-UE-CCA. Note that this is the strongest known security property for updatable encryption schemes with deterministic updates. In Section 5.3 we discuss the challenges that arise in the proofs of these two theorems, and in Section 5.4 and Section 5.5 we describe the novel techniques and methods used in the proofs.

Theorem 4 (SHINE is detIND-UE-CPA)**.** Let SHINE be the UE scheme described in Fig. 25 (or Fig. 26 or Fig. 28), where SHINE $\in$ {SHINE0, MirrorSHINE, OCBSHINE}. For any ideal cipher model adversary $\mathcal{A}$ (that makes max $\mathrm{Q_E}$ encryption queries before its challenge), there exists an adversary $\mathcal{B}_4$ such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{SHINE},\,\mathcal{A}}(\lambda) \leq O(1)(n+1)^3 \cdot \mathrm{Q_E} \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_4}(\lambda).$$

This is proven via Theorem 4.1, and Theorem 4.3 and Theorem 4.4 in Section 5.4.

Theorem 5 (SHINE is INT-CTXT$^{\mathsf{s}}$)**.** Let SHINE be the UE scheme described in Fig. 25 (or Fig. 26 or Fig. 28), where SHINE $\in$ {SHINE0, MirrorSHINE, OCBSHINE}.For any ideal cipher model adversary $\mathcal{A}$ (that makes max $\mathrm{Q_E}$ encryption queries before calling $\mathcal{O}$.Try), there exists an adversary $\mathcal{B}_5$ such that

$$\mathbf{Adv}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{SHINE},\,\mathcal{A}}(\lambda) \leq O(1)(n+1)^2 \cdot \mathrm{Q_E} \cdot \mathbf{Adv}^{\mathsf{CDH}}_{\mathcal{B}_5} + \text{negligible terms}$$

This is proven via Theorem 5.1, and Theorem 5.3 and Theorem 5.4 in Section 5.5.

Remark 5. Combining the results of Theorem 3, Theorem 4 and Theorem 5, we have that SHINE is detIND-UE-CCA.

Remark 6. Proofs for SHINE0 are extendable to the other two SHINE schemes (MirrorSHINE and OCBSHINE), so we only show full proof details of Theorem 4.1 and Theorem 5.1 in Section 5.4 and Section 5.5, respectively.

5.3 Proof Challenges in Schemes with Deterministic Updates

In each variant of SHINE all ciphertext components are raised to the epoch key, so the update mechanism transforms a ciphertext for epoch e to one for $\mathsf{e}+1$ by raising this value to $\frac{\mathrm{k_{e+1}}}{\mathrm{k_e}}$. We now highlight the difficulties in creating security proofs for such 'single-component' updatable encryption schemes. Randomness is used in creation of the initial ciphertext (via

N) but updates are completely deterministic, and thus in any reduction it is necessary to provide consistent ciphertexts to the adversary (i.e. the N value must be consistent). The (cryptographic) separation gained by using the firewall technique (see Section 3.3 for discussion and definition) assists with producing (updates of) non-challenge ciphertexts, but embedding any challenge value while also providing answers to the $\mathcal{O}.\mathsf{Corr}$ queries of the underlying adversary is very challenging.

The regular key insulation technique as introduced by LT18 – where the reduction constructs one hybrid for each epoch – does not work. Specifically, in any reduction to a DDH-like assumption, it is not possible to provide a challenge ciphertext in a left or right sense (to the left of this challenge ciphertext are of some form, and to the right of this challenge ciphertext are of some other form) if the underlying adversary asks for tokens around the challenge epoch: deterministic updates mean that tokens will make these ciphertexts of the same form and this gap will be easily distinguishable.

We counteract this problem by constructing a *hybrid argument across insulated regions*. This means that in each hybrid, we can embed at one firewall of the insulated region, and simulate all tokens within that insulated region to enable answering queries to both $\mathcal{O}.\mathsf{Upd}$ and $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$. The reduction's distinguishing task is thus ensured to be at the boundaries of the insulated regions, the firewalls, so any (non-trivial) win for the underlying adversary is ensured to carry through directly to the reduction.

5.4 SHINE is detIND-UE-CPA

We now explain how we bound the advantage of any adversary playing the detIND-UE-CPA game for SHINE by the advantage of a reduction playing DDH.

Proof Method for Confidentiality: Constructing a Hybrid Argument across Insulated Regions.

Notice that the non-corrupted key space is the union set of all insulated regions, i.e. $\{0, 1, ..., n\} \setminus \mathcal{K}^* = \cup_{(j,\mathsf{fwl}_j,\mathsf{fwr}_j)\in\mathcal{FW}}\{\mathsf{fwl}_j, ..., \mathsf{fwr}_j\}$. If the trivial win conditions are not triggered and the adversary knows a challenge-equal ciphertext in some epoch within an insulated region, then since the adversary knows all tokens in that insulated region, the adversary will know all challenge-equal ciphertexts in that insulated region. Then we have

$$\mathcal{C}^* = \cup_{(j,\mathsf{fwl}_j,\mathsf{fwr}_j)\in S\subseteq\mathcal{FW}}\{\mathsf{fwl}_j, \ldots, \mathsf{fwr}_j\},$$

where S is a subset of firewall list $\mathcal{FW}$.

We apply the firewall technique to set up hybrid games such that in hybrid i, we embed within the i-th insulated region: this means that to the left of the i-th insulated region the game responds with the $\mathrm{b} = 1$ case of the detIND-UE-CPA experiment, and to the right of the i-th insulated region it gives an encryption of the challenge input message as output, i.e. $\mathrm{b} = 0$. This means we have one hybrid for each insulated region, moving left-to-right across the epoch space.

We construct a reduction $\mathcal{B}$ playing the DDH experiment in hybrid i. Initially, $\mathcal{B}$ guesses the location of the i-th insulated region. If the underlying adversary has performed a corrupt query within this insulated region that would lead to the reduction failing, the reduction aborts the game. We use the algorithm Check described in Fig. 29 to check if this event happens.

```
Check(inp, ê; e; fwl, fwr)
1 :  if $\hat{\mathsf{e}} > \mathsf{e}$
2 :     return $\perp$
3 :  if $\mathsf{inp} = \mathsf{key}$ and $\hat{\mathsf{e}} \in \{\mathsf{fwl}, \ldots, \mathsf{fwr}\}$
4 :     return ABORT
5 :  if $\mathsf{inp} = \mathsf{token}$ and $\hat{\mathsf{e}} \in \{\mathsf{fwl}, \mathsf{fwr}+1\}$
6 :     return ABORT
```

Figure 29: Algorithm Check, used in proofs in this section. In Check, ê is the epoch in the adversary's request, and e is the current epoch.

In particular, within the insulated region, the reduction can simulate challenge ciphertexts and non-challenge ciphertexts using its DDH tuple. Furthermore, ciphertexts can be moved around within the insulated region by tokens.

Remark 7. We note that the problem of challenge insulation in schemes with deterministic updates was also observed independently by Klooß et al. [[KLR19b], § B.2]. Their solution (though in the different context of CCA security of UE with certain properties) is to form a hybrid argument with a hybrid for each epoch, and essentially guess an epoch r which is the first token 'after' the hybrid index that the adversary has not corrupted, and use the inherent 'gap' in the adversary's knowledge continuum to replace challenge updates across this gap with encryptions of just one of the challenge messages. It is not clear if this approach would work for showing detIND-UE-CPA (or IND-ENC-CPA) of SHINE. We conjecture that even if it were possible to construct a reduction in this vein, our approach enables a more direct proof: in particular we do not need to assume specific additional properties of the UE scheme in question for it to work.

SHINE0 is detIND-UE-CPA.

Theorem 4.1 (SHINE0 is detIND-UE-CPA)**.** Let $\mathbb{G}$ be a group of order q (a λ-bit prime) with generator g, and let SHINE0 be the updatable encryption scheme described in Fig. 25. For any detIND-UE-CPA adversary $\mathcal{A}$ against SHINE0 that asks at most $\mathrm{Q_E}$ queries to $\mathcal{O}$.Enc before it makes its challenge, there exists an adversary $\mathcal{B}_{4.1}$ against DDH such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{SHINE0},\ \mathcal{A}}(\lambda) \leq 2(n+1)^3 \cdot \mathrm{Q_E} \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\ \mathcal{B}_{4.1}}(\lambda).$$

Proof. Play hybrid games. We begin by partitioning non-corrupted key space as follows: $\{0,1,\dots,n\} \setminus \mathcal{K}^* = \cup_{(j,\mathsf{fwl}_j,\mathsf{fwr}_j)\in\mathcal{FW}}\{\mathsf{fwl}_j,\dots,\mathsf{fwr}_j\}$, where fwr_i and fwr_i are firewalls of the i-th insulated region. Recall the definition from Section 3.3 and firewall computing algorithm FW-Find in Fig. 15: $\mathsf{fwl}_i, \mathsf{fwr}_i$ are firewalls of the i-th insulated region if $(i, \mathsf{fwl}_i, \mathsf{fwr}_i) \in \mathcal{FW}$.

For $\mathrm{b} \in \{0,1\}$, define game $\mathcal{G}_i^{\mathrm{b}}$ as $\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}b}}_{\mathsf{SHINE0},\ \mathcal{A}}$ except for:

- The game randomly picks an integer h, and if the challenge input $\bar{\mathrm{C}}$ is not an updated ciphertext of the h-th $\mathcal{O}$.Enc query, it (aborts and) returns a random bit for b'. This loss is upper-bounded by $\mathrm{Q_E}$.

- The game randomly picks $\mathsf{fwl}_i, \mathsf{fwr}_i \xleftarrow{\$} \{0,\dots,n\}$ and if $\mathsf{fwl}_i, \mathsf{fwr}_i$ are not the i-th firewalls, returns a random bit for b'. This loss is upper-bounded by $(n+1)^2$.

- For the challenge (made in epoch $\tilde{\mathsf{e}}$, input $(\bar{\mathrm{M}}, \bar{\mathrm{C}})$): If $\tilde{\mathsf{e}} < \mathsf{fwl}_i$ then return a ciphertext with respect to $\bar{\mathrm{C}}$, if $\tilde{\mathsf{e}} > \mathsf{fwr}_i$ return a ciphertext with $\bar{\mathrm{M}}$, and if $\mathsf{fwl}_i \leq \tilde{\mathsf{e}} \leq \mathsf{fwr}_i$ then return a ciphertext with $\bar{\mathrm{M}}$ when $\mathrm{b} = 0$, return a ciphertext with respect to $\bar{\mathrm{C}}$ when $\mathrm{b} = 1$.

- After $\mathcal{A}$ outputs b', returns b' if $\mathsf{twf} \neq 1$ or some additional trivial win condition triggers.

If $\mathrm{h}, \mathsf{fwl}_i, \mathsf{fwr}_i$ are the desired values, then $\mathcal{G}_1^0$ is $\mathbf{Exp}_{\mathsf{SHINE0},\,\mathcal{A}}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}0}}$, i.e. all challenge-equal ciphertexts are encryptions of $\bar{\mathrm{M}}$. And there exists some l (the total number of insulated regions, bounded by $n+1$), game $\mathcal{G}_l^1$ is $\mathbf{Exp}_{\mathsf{SHINE0},\,\mathcal{A}}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}1}}$, i.e. all challenges are updates of $\bar{\mathrm{C}}$. Let E to be the event that $\mathrm{h}, \mathsf{fwl}_i, \mathsf{fwr}_i$ are the desired values, notice that $\mathbf{Pr}[\mathcal{G}_i^{\mathrm{b}} = 1|\neg\mathrm{E}] = \frac{1}{2}$ for any $1 \leq i \leq n+1$ and $\mathrm{b} \in \{0,1\}$. Then

$$\begin{aligned}
\mathbf{Pr}[\mathcal{G}_l^1 = 1] &= \mathbf{Pr}[\mathcal{G}_l^1 = 1|\mathrm{E}] \cdot \mathbf{Pr}[\mathrm{E}] + \mathbf{Pr}[\mathcal{G}_l^1 = 1|\neg\mathrm{E}] \cdot \mathbf{Pr}[\neg\mathrm{E}] \\
&= \mathbf{Pr}[\mathbf{Exp}_{\mathsf{SHINE0},\,\mathcal{A}}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}1}} = 1] \cdot \frac{1}{(n+1)^2\mathrm{Q_E}} \\
&\quad + \frac{1}{2} \cdot (1 - \frac{1}{(n+1)^2\mathrm{Q_E}}), \text{ and} \\
\mathbf{Pr}[\mathcal{G}_1^0 = 1] &= \mathbf{Pr}[\mathbf{Exp}_{\mathsf{SHINE0},\,\mathcal{A}}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}0}} = 1] \cdot \frac{1}{(n+1)^2\mathrm{Q_E}} \\
&\quad + \frac{1}{2} \cdot (1 - \frac{1}{(n+1)^2\mathrm{Q_E}}).
\end{aligned}$$

Thus we have that

$$\begin{aligned}
\left|\mathbf{Pr}[\mathcal{G}_l^1 = 1] - \mathbf{Pr}[\mathcal{G}_1^0 = 1]\right| &= \frac{1}{(n+1)^2\mathrm{Q_E}} \Big|\mathbf{Pr}[\mathbf{Exp}_{\mathsf{SHINE0},\,\mathcal{A}}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}1}} = 1] \\
&\qquad - \mathbf{Pr}[\mathbf{Exp}_{\mathsf{SHINE0},\,\mathcal{A}}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}0}} = 1]\Big| \\
&= \frac{1}{(n+1)^2\mathrm{Q_E}} \cdot \mathbf{Adv}_{\mathsf{SHINE0},\,\mathcal{A}}^{\mathsf{detIND\text{-}UE\text{-}CPA}}(\lambda).
\end{aligned}$$

Notice that all queries in $\mathcal{G}_{i-1}^1$ and $\mathcal{G}_i^0$ have the equal responses: for the challenge query and $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$, if called in epoch in first $i-1$ insulated regions, the reduction returns a ciphertext with respect to $\bar{\mathrm{C}}$, otherwise returns an encryption of $\bar{\mathrm{M}}$. Therefore, for any $l(\leq n+1)$, $|\mathbf{Pr}[\mathcal{G}_l^1 = 1] - \mathbf{Pr}[\mathcal{G}_1^0 = 1]| \leq \sum_{i=1}^{l} |\mathbf{Pr}[\mathcal{G}_i^1 = 1] - \mathbf{Pr}[\mathcal{G}_i^0 = 1]|$. We prove that for any $1 \leq i \leq l$, $|\mathbf{Pr}[\mathcal{G}_i^1 = 1] - \mathbf{Pr}[\mathcal{G}_i^0 = 1]| \leq 2\mathbf{Adv}_{\mathbb{G},}^{\mathsf{DDH}}(\lambda)$. We only prove one of these l hybrids, the rest can be similarly proven.

In hybrid i. Suppose that $\mathcal{A}_i$ is an adversary attempting to distinguish $\mathcal{G}_i^0$ from $\mathcal{G}_i^1$. For all queries concerning epochs outside of the i-th insulated region the responses will be equal in either game, so we assume that $\mathcal{A}_i$ asks for at least one challenge ciphertext in an epoch within the i-th insulated region (then $[\mathsf{fwl}_i, \mathsf{fwr}_i] \subseteq \mathcal{C}^*$) and this is where we will embed DDH tuples in our reduction.

We construct a reduction $\mathcal{B}_{4.1}$, detailed in Fig. 30, that is playing the standard DDH game and will simulate the responses of queries made by adversary $\mathcal{A}_i$.

The reduction $\mathcal{B}_{4.1}$ receives DDH tuples (X, Y, Z), flips a coin b and simulates game $\mathcal{G}_i^{\mathrm{b}}$. Whenever the reduction needs to provide an output of $\pi(\cdot)$ to $\mathcal{A}$, it chooses ('programs') some random value r such that $\pi(\cdot) = g^r$. Then, we use the fact that $(g^r)^{\mathrm{k_e}} = (g^{\mathrm{k_e}})^r$ and use the $g^{\mathrm{k_e}}$ values as 'public keys' to allow simulation. In this setting, decryption (i.e. π^{-1}) is simply a lookup to this mapping of the ideal cipher π. A summary of the technical simulations follows:

Initially,

1. $\mathcal{B}_{4.1}$ guesses the values of $\mathrm{h}, \mathsf{fwl}_i, \mathsf{fwr}_i$.
2. $\mathcal{B}_{4.1}$ generates all keys and tokens except for $\mathrm{k}_{\mathsf{fwl}_i}, \ldots, \mathrm{k}_{\mathsf{fwr}_i}$, Δ_{fwl_i}, $\Delta_{\mathsf{fwr}_i+1}$. If $\mathcal{A}_i$ ever corrupts these keys and tokens – which indicates the firewall guess is wrong – the reduction aborts the game.
3. $\mathcal{B}_{4.1}$ computes the public values of keys in an epoch:
   - $\mathsf{e} \notin \{\mathsf{fwl}_i, \ldots, \mathsf{fwr}_i\}$: $\mathcal{B}_{4.1}$ computes $\mathsf{PK_e} = g^{\mathrm{k_e}}$;
   - $\mathsf{e} \in \{\mathsf{fwl}_i, \ldots, \mathsf{fwr}_i\}$: $\mathcal{B}_{4.1}$ embeds DDH value Y as $\mathsf{PK}_{\mathsf{fwl}_i}$. More precisely, if $\mathrm{b} = 0, \mathsf{PK}_{\mathsf{fwl}_i} = Y$, otherwise $\mathsf{PK}_{\mathsf{fwl}_i} = Y^{\mathrm{k}_{\mathsf{fwl}_i - 1}}$ (since $g^{\Delta_{\mathsf{fwl}_i}} = Y$). Then $\mathcal{B}_{4.1}$ uses tokens $\Delta_{\mathsf{fwl}_i+1}, \ldots,$ Δ_{fwr_i} to compute the remaining public key values $\mathsf{PK_e}$ in the insulated region.

To simulate a non-challenge ciphertext that is:

- not the h-th query to $\mathcal{O}.\mathsf{Enc}$: $\mathcal{B}_{4.1}$ generates a random value r for each encryption (so that the randomness will be consistent for calls that $\mathcal{A}_i$

```
Reduction $\mathcal{B}_{4.1}$ playing $\mathbf{Exp}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_{4.1}}$
1 :  receive $(g, X, Y, Z)$
2 :  do Setup
3 :  $\bar{\mathrm{M}}, \bar{\mathrm{C}} \leftarrow \mathcal{A}^{\mathtt{ors}}(\lambda)$
4 :  $\mathsf{phase} \leftarrow 1$
5 :  Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}, \bar{\mathrm{C}})$, get $\tilde{\mathrm{C}}_{\tilde{\mathrm{e}}}$
6 :  $\mathrm{b}' \leftarrow \mathcal{A}^{\mathtt{ors}, \mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}}_{\tilde{\mathrm{e}}})$
7 :  $\underline{\mathsf{twf} \leftarrow 1 \textbf{ if}}$
8 :      $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ or
9 :      $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$
10 : if ABORT occurred or $\mathsf{twf} = 1$
11 :     $\mathrm{b}' \xleftarrow{\$} \{0, 1\}$
12 :     return $\mathrm{b}'$
13 : if $(i, \mathsf{fwl}_i, \mathsf{fwr}_i) \notin \mathcal{FW}$
14 :     $\mathrm{b}' \xleftarrow{\$} \{0, 1\}$
15 :     return $\mathrm{b}'$
16 : if $\mathrm{b}' = \mathrm{b}$
17 :     return 0
18 : else
19 :     return 1
```

```
Setup($\lambda$)
20 : $\mathrm{b} \xleftarrow{\$} \{0, 1\}$
21 : $\mathrm{k}_0 \leftarrow \mathsf{SHINE0.KG}(\lambda)$
22 : $\Delta_0 \leftarrow \perp$
23 : $\mathsf{e}, \mathsf{c}, \mathsf{phase}, \mathsf{twf} \leftarrow 0$
24 : $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$
25 : $\mathsf{fwl}_i, \mathsf{fwr}_i \xleftarrow{\$} \{0, \ldots, n\}$
26 : $\mathrm{h} \xleftarrow{\$} \{1, \ldots, \mathrm{Q_E}\}$
27 : for $j \in \{0, \ldots, \mathsf{fwl}_i\text{-}1\}$ do
28 :     $\mathrm{k}_j \xleftarrow{\$} \mathbb{Z}_q^*; \Delta_j \leftarrow \dfrac{\mathrm{k}_j}{\mathrm{k}_{j-1}}$ $\bowtie$
29 :     $\mathsf{PK}_j \leftarrow g^{\mathrm{k}_j}$
30 : for $j \in \{\mathsf{fwr}_i\text{+}1, \ldots, n\}$ do
31 :     $\mathrm{k}_j \xleftarrow{\$} \mathbb{Z}_q^*; \Delta_j \leftarrow \dfrac{\mathrm{k}_j}{\mathrm{k}_{j-1}}$ $\bowtie$
32 :     $\mathsf{PK}_j \leftarrow g^{\mathrm{k}_j}$
33 : if $\mathrm{b} = 0$
34 :     $\mathsf{PK}_{\mathsf{fwl}_i} \leftarrow Y; \mathrm{C} \xleftarrow{\$} \mathbb{G}$
35 : else
36 :     $\mathsf{PK}_{\mathsf{fwl}_i} \leftarrow Y^{\mathrm{k}_{\mathsf{fwl}_i - 1}}; \mathrm{C} \leftarrow X$
37 : for $j \in \{\mathsf{fwl}_i\text{+}1, \ldots, \mathsf{fwr}_i\}$ do
38 :     $\Delta_j \xleftarrow{\$} \mathbb{Z}_q^*; \mathsf{PK}_j \leftarrow \mathsf{PK}_{j-1}^{\Delta_j}$
```

Figure 30: Part 1. Reduction $\mathcal{B}_{4.1}$ for proof of Theorem 4.1, in hybrid i. Moving left-to-right through embedding DDH tuples in the i-th insulated region: when $\mathrm{b} = 1$, embedding DDH tuples to token values to move left to random; when $\mathrm{b} = 0$, embedding DDH tuples to key values to move right to random. inf encodes fixed programming information: it marks an epoch if $\mathsf{c} = \mathrm{h}$, otherwise it is the random encryption exponent. On lines 3 and 6, $\mathtt{ors}$ refers to the set $\{\mathcal{O}.\mathsf{Enc}, \mathcal{O}.\mathsf{Next}, \mathcal{O}.\mathsf{Upd}, \mathcal{O}.\mathsf{Corr}\}$. On lines 28 and 31, $\bowtie$ indicates Δ_0 and $\Delta_{\mathsf{fwr}_i+1}$ are skipped in the computation.

```
$\mathcal{O}.\mathsf{Enc}(\mathrm{M})$
  $\mathsf{c} \leftarrow \mathsf{c} + 1$
  **if** $\mathsf{c} = \mathrm{h}$
    $\mathrm{C}_{\mathsf{e}} \leftarrow \mathrm{C}; \mathsf{inf} \leftarrow \mathsf{e}$
  **else**
    $\mathsf{inf} \xleftarrow{\$} \mathbb{Z}_q^*$
    $\pi(\mathrm{N}||\mathrm{M}) \leftarrow g^{\mathsf{inf}}$
    $\mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{PK}_{\mathsf{e}}^{\mathsf{inf}}$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}_{\mathsf{e}}, \mathsf{e}; \mathsf{inf})\}$
  **return** $\mathrm{C}_{\mathsf{e}}$
```

```
$\mathcal{O}.\mathsf{Next}$
  $\mathsf{e} \leftarrow \mathsf{e} + 1$
```

```
$\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}-1})$
  **if** $(\mathsf{c}, \mathrm{C}_{\mathsf{e}-1}, \mathsf{e}-1; \mathsf{inf}) \notin \mathcal{L}$
    **return** $\perp$
  **if** $\mathsf{c} = \mathrm{h}$
    $\mathrm{C}_{\mathsf{e}} \leftarrow \mathrm{C}_{\mathsf{e}-1}^{\Delta_{\mathsf{e}}}$
  **else**
    $\mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{PK}_{\mathsf{e}}^{\mathsf{inf}}$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}_{\mathsf{e}}, \mathsf{e}; \mathsf{inf})\}$
  **return** $\mathrm{C}_{\mathsf{e}}$
```

```
$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$
  **do** $\mathsf{Check}(\mathsf{inp}, \hat{\mathsf{e}}; \mathsf{e}; \mathsf{fwl}_i, \mathsf{fwr}_i)$
  **if** $\mathsf{inp} = \mathsf{key}$
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
    **return** $\mathrm{k}_{\hat{\mathsf{e}}}$
  **if** $\mathsf{inp} = \mathsf{token}$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
    **return** $\Delta_{\hat{\mathsf{e}}}$
```

```
Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}, \bar{\mathrm{C}})$
  **if** $(\mathrm{h}, \bar{\mathrm{C}}, \tilde{\mathsf{e}}-1; \mathsf{inf}) \notin \mathcal{L}$
    **return** ABORT
  **if** $\mathrm{b} = 0$
    $\pi(\mathrm{N}||\bar{\mathrm{M}}) \leftarrow X; \tilde{\mathrm{C}}_{\mathsf{fwl}_i} \leftarrow Z$
  **else**
    $\pi(\mathrm{N}||\bar{\mathrm{M}}) \xleftarrow{\$} \mathbb{G}$
    $\tilde{\mathrm{C}}_{\mathsf{fwl}_i} \leftarrow Z^{\prod_{j=\mathsf{inf}+1}^{\mathsf{fwl}_i-1} \Delta_j}$
  **for** $j \in \{0, \ldots, \mathsf{fwl}_i - 1\}$ **do**
    $\tilde{\mathrm{C}}_j \leftarrow \bar{\mathrm{C}}^{(\prod_{k=0}^{j} \Delta_k)/(\prod_{k=0}^{\tilde{\mathsf{e}}-1} \Delta_k)}$    // left
  **for** $j \in \{\mathsf{fwl}_i + 1, \ldots, \mathsf{fwr}_i\}$ **do**
    $\tilde{\mathrm{C}}_j \leftarrow \tilde{\mathrm{C}}_{j-1}^{\Delta_j}$    // embed
  **for** $j \in \{\mathsf{fwr}_i + 1, \ldots, n\}$ **do**
    $\tilde{\mathrm{C}}_j \leftarrow (\pi(\mathrm{N}||\bar{\mathrm{M}}))^{k_j}$    // right
  $\tilde{\mathcal{L}} \leftarrow \cup_{j=0}^{n} \{(\tilde{\mathrm{C}}_j, j)\}$
  **return** $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
```

```
$\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
  **find**$(\tilde{\mathrm{C}}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{L}}$
  **return** $\tilde{\mathrm{C}}_{\mathsf{e}}$
```

Figure 30: Part 2. Reduction $\mathcal{B}_{4.1}$ for proof of Theorem 4.1, in hybrid i.

makes to $\mathcal{O}.\mathsf{Upd}$) and programs the ideal cipher with $\mathrm{C}_{\mathsf{e}} = \mathsf{PK}_{\mathsf{e}}^{r}$. To respond to $\mathcal{O}.\mathsf{Upd}$ queries, the reduction computes $\mathrm{C}_{\mathsf{e}'} = \mathsf{PK}_{\mathsf{e}'}^{r}$ to update a non-challenge ciphertext to epoch e'.

- the h-th query to $\mathcal{O}.\mathsf{Enc}$: $\mathcal{B}_{4.1}$ embeds either a random ciphertext ($\mathrm{b} = 0$) or a DDH value ($\mathrm{b} = 1$) to the encryption ($\mathrm{C}_{\mathsf{e}_\mathrm{h}}$). Furthermore, the reduction uses tokens $\Delta_0, \ldots, \Delta_{\mathsf{fwl}_i - 1}$ to update the h-th encryption. Note that $\bar{\mathrm{C}}$ is an update of the h-th encryption. The adversary can not ask for update of the h-th encryption in an epoch $\mathsf{e} \geq \mathsf{fwl}_i$, as this would trigger the trivial win condition $[\mathsf{fwl}_i, \mathsf{fwr}_i] \subseteq \mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$.

To simulate challenge-equal ciphertext in an epoch that is:

- to the left of the i-th insulated region: $\mathcal{B}_{4.1}$ simulates $\mathsf{SHINE0.Upd}(\bar{\mathrm{C}})$ using the tokens that it created itself.

- within the i-th insulated region: $\mathcal{B}_{4.1}$ simulates $\mathsf{SHINE0.Upd}(\bar{\mathrm{C}})$ if $\mathrm{b} = 1$, and simulates $\mathsf{SHINE0.Enc}(\bar{\mathrm{M}})$ if $\mathrm{b} = 0$. More precisely, $\mathcal{B}_{4.1}$ embeds DDH value X to ciphertext information of challenge input, embeds DDH value Z to the challenge ciphertext. Which means if $\mathrm{b} = 1$, the reduction will give the value X to $\mathrm{C}_{\mathsf{e}_\mathrm{h}}$, $Z^{\prod_{j=\mathsf{e}_\mathrm{h}+1}^{\mathsf{fwl}_i - 1} \Delta_j}$ to $\tilde{\mathrm{C}}_{\mathsf{fwl}_i}$ (recall $g^{\Delta_{\mathsf{fwl}_i}} = Y$) since

$$\tilde{\mathrm{C}}_{\mathsf{fwl}_i} = \mathsf{SHINE0.Upd}(\bar{\mathrm{C}}) = \bar{\mathrm{C}}_{\mathsf{fwl}_i - 1}^{\Delta_{\mathsf{fwl}_i}} = (\mathrm{C}_{\mathsf{e}_\mathrm{h}}^{\Delta_{\mathsf{fwl}_i}})^{\prod_{j=\mathsf{e}_\mathrm{h}+1}^{\mathsf{fwl}_i - 1} \Delta_j}.$$

  If $\mathrm{b} = 0$, the reduction will give X to $\pi(\mathrm{N}||\bar{\mathrm{M}})$, and Z to $\tilde{\mathrm{C}}_{\mathsf{fwl}_i}$ (recall $\mathsf{PK}_{\mathsf{fwl}_i} = Y$) since $\tilde{\mathrm{C}}_{\mathsf{fwl}_i} = \mathsf{SHINE0.Enc}(\bar{\mathrm{M}}) = \pi(\mathrm{N}||\bar{\mathrm{M}})^{\mathrm{k}_{\mathsf{fwl}_i}}$. Furthermore, the reduction uses tokens $\Delta_{\mathsf{fwl}_i + 1}, \ldots, \Delta_{\mathsf{fwr}_i}$ to update $\tilde{\mathrm{C}}_{\mathsf{fwl}_i}$ to simulate all challenge ciphertext in epochs within the insulated region.

- to the right of the i-th insulated region: the reduction $\mathcal{B}_{4.1}$ simulates $\mathsf{SHINE0.Enc}(\bar{\mathrm{M}})$ using the keys that it created itself.

Eventually, $\mathcal{B}_{4.1}$ receives the output bit b' from $\mathcal{A}_i$. If $\mathrm{b}' = \mathrm{b}$, then $\mathcal{B}_{4.1}$ guesses that it has seen real DDH tuples (returns 0 to its DDH challenger), otherwise, $\mathcal{B}_{4.1}$ guesses that it has seen random DDH tuples (returns 1).

If $\mathcal{B}_{4.1}$ receives a real DDH tuple, then $\mathcal{B}_{4.1}$ perfectly simulates the environment of $\mathcal{A}_i$ in $\mathcal{G}_i^{\mathrm{b}}$. If $\mathcal{B}_{4.1}$ receives a random DDH tuple, then $\mathcal{B}_{4.1}$ wins with probability 1/2. After some computation similar to that in the proof of Theorem 2.3 we have $\mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_{4.1}}(\lambda) = \frac{1}{2}|\mathbf{Pr}[\mathcal{G}_i^1 = 1] - \mathbf{Pr}[\mathcal{G}_i^0 = 1]|$. $\square$

SHINE0 is IND-ENC-CPA Secure. As a corollary of Theorem 2.4 and Theorem 4.1, SHINE0 is IND-ENC-CPA – however in Appendix A we give a tighter proof – eliminating the $\mathrm{Q_E}$ term – by directly proving the IND-ENC-CPA security of SHINE0. The proof follows a similar strategy to that of Theorem 4.1, with one hybrid for each insulated region.

MirrorSHINE is detIND-UE-CPA.

Theorem 4.3 (MirrorSHINE is detIND-UE-CPA)**.** Let $\mathbb{G}$ be a group of order q (a λ-bit prime) with generator g, and let MirrorSHINE be the updatable encryption scheme described in Fig. 26. For any detIND-UE-CPA adversary $\mathcal{A}$ against MirrorSHINE that asks at most $\mathrm{Q_E}$ queries to $\mathcal{O}$.Enc before it makes its challenge, there exists an adversary $\mathcal{B}_{4.3}$ against DDH such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{MirrorSHINE},\,\mathcal{A}}(\lambda) \le 2(n+1)^3 \cdot \mathrm{Q_E} \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_{4.3}}(\lambda).$$

Proof. The proof is similar to the security proof of Theorem 4.1, except that the reduction will embed two random group elements to the image of π_1, π_2 while it simulates the output of the $\mathcal{O}$.Enc oracle, instead of embedding one random group element to the image of π. We will not give a detailed construction of the reduction in this proof, since we will give a more general construction in the proof of Theorem 4.4. $\square$

OCBSHINE is detIND-UE-CPA. For convenience, we denote a vector with $l+2$ elements to be $\vec{C} = (C^0, \ldots, C^{l+1})$. We also use the shorthand $\vec{A} \leftarrow \vec{B}^c$ (component-wise exponentiation) and $\vec{A} \leftarrow B^{\vec{c}}$ (common-base exponentiation) to mean the following:

```
$\vec{A} \leftarrow \vec{B}^c$
1 : for $j = 0, \ldots, l+1$ do
2 :     $A^j \leftarrow (B^j)^c$
```

```
$\vec{A} \leftarrow B^{\vec{c}}$
1 : for $j = 0, \ldots, l+1$ do
2 :     $A^j \leftarrow B^{c_j}$
```

Theorem 4.4 (OCBSHINE is detIND-UE-CPA)**.** Let $\mathbb{G}$ be a group of order q (a λ-bit prime, $\lambda = m$) with generator g, and let OCBSHINE be the updatable encryption scheme described in Fig. 28. For any detIND-UE-CPA adversary $\mathcal{A}$ against OCBSHINE that asks at most $\mathrm{Q_E}$ queries to $\mathcal{O}$.Enc before it makes its challenge, there exists an adversary $\mathcal{B}_{4.4}$ against DDH such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{OCBSHINE},\,\mathcal{A}}(\lambda) \leq 2(n+1)^3 \cdot \mathrm{Q_E} \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_{4.4}}(\lambda).$$

Proof. The proof is similar to the security proof of Theorem 4.1, except that here we are dealing with vector of ciphertexts. To simulate the output of $\mathcal{O}$.Enc, the reduction will embed a vector of random group elements to the image of the random permutation family. The reduction is detailed in Fig. 31. □

5.5 SHINE is INT-CTXT$^{\mathsf{s}}$

We then explain how we bound the advantage of any adversary playing the INT-CTXT$^{\mathsf{s}}$ game for SHINE by the advantage of a reduction playing CDH.

Proof Method for ciphertext integrity. In the INT-CTXT$^{\mathsf{s}}$ game, the challenger will keep a list of consistent values for ciphertexts (i.e. the underlying permutation output). Suppose $\tilde{\mathrm{C}}$ is a forgery attempt sent to the $\mathcal{O}$.Try query in epoch $\tilde{\mathrm{e}}$. Let $\tilde{c} = (\tilde{\mathrm{C}})^{1/\mathrm{k}_{\tilde{\mathrm{e}}}}$ be the underlying permutation output.

If $\tilde{c}$ is a new value, since we have that π (or (π_1, π_2) or $(\pi_0, \{\pi_{\mathrm{N}\|i\|b}\})$) is a random permutation then the INT-CTXT$^{\mathsf{s}}$ challenger simulates the preimage of $\tilde{c}$ under the corresponding random permutation(s) to be a random string(s). So the probability that this (these) random string(s) matches the integrity tag is negligible, and this carries over to the probability that the adversary wins the INT-CTXT$^{\mathsf{s}}$ game.

If $\tilde{c}$ is an already-existing value, and suppose this event happens with probability p. We construct a reduction playing the CDH game such that it wins CDH game with probability $p \cdot \frac{1}{\mathrm{Q_E}(n+1)^2}$. Similar to the proof method of Theorem 4.1, we construct a reduction playing the CDH experiment by guessing the location of the firewalls around the challenge epoch. The reduction embeds the CDH value and simulates the INT-CTXT$^{\mathsf{s}}$ game, using

Reduction $\mathcal{B}_{4.4}$ playing $\mathbf{Exp}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_{4.4}}$

```
1 :  $\textbf{receive } (g, X, Y, Z)$
2 :  $\textbf{do Setup}$
3 :  $\vec{\mathrm{M}}, \vec{\mathrm{C}} \leftarrow \mathcal{A}^{\texttt{ors}}(\lambda)$
4 :  $\mathsf{phase} \leftarrow 1$
5 :  Create $\tilde{\mathrm{C}}$ with $(\vec{\mathrm{M}}, \vec{\mathrm{C}})$, $\textbf{get } \tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
6 :  $\mathrm{b}' \leftarrow \mathcal{A}^{\texttt{ors}, \mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}})$
7 :  $\underline{\mathsf{twf} \leftarrow 1 \textbf{ if }}$
8 :    $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ $\textbf{or}$
9 :    $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$
10 : $\textbf{if}$ ABORT occurred $\textbf{or}$ $\mathsf{twf} = 1$
11 :   $\mathrm{b}' \xleftarrow{\$} \{0,1\}$
12 :   $\textbf{return } \mathrm{b}'$
13 : $\textbf{if } (i, \mathsf{fwl}_i, \mathsf{fwr}_i) \notin \mathcal{FW}$
14 :   $\mathrm{b}' \xleftarrow{\$} \{0,1\}$
15 :   $\textbf{return } \mathrm{b}'$
16 : $\textbf{if } \mathrm{b}' = \mathrm{b}$
17 :   $\textbf{return } 0$
18 : $\textbf{else}$
19 :   $\textbf{return } 1$
```

$\mathbf{Setup}(\lambda)$

```
20 : $\mathrm{b} \xleftarrow{\$} \{0,1\}$
21 : $\mathrm{k}_0 \leftarrow \mathsf{SHINE0.KG}(\lambda)$
22 : $\Delta_0 \leftarrow \perp$
23 : $\mathsf{e}, \mathsf{c}, \mathsf{phase}, \mathsf{twf} \leftarrow 0$
24 : $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$
25 : $\mathsf{fwl}_i, \mathsf{fwr}_i \xleftarrow{\$} \{0, \ldots, n\}$
26 : $\mathrm{h} \xleftarrow{\$} \{1, \ldots, \mathrm{Q_E}\}$
27 : $\textbf{for } j \in \{0, \ldots, \mathsf{fwl}_i\text{-}1\} \textbf{ do}$
28 :   $\mathrm{k}_j \xleftarrow{\$} \mathbb{Z}_q^*; \Delta_j \leftarrow \frac{\mathrm{k}_j}{\mathrm{k}_{j-1}}^{\bowtie}$
29 :   $\mathsf{PK}_j \leftarrow g^{\mathrm{k}_j}$
30 : $\textbf{for } j \in \{\mathsf{fwr}_i\text{+}1, \ldots, n\} \textbf{ do}$
31 :   $\mathrm{k}_j \xleftarrow{\$} \mathbb{Z}_q^*; \Delta_j \leftarrow \frac{\mathrm{k}_j}{\mathrm{k}_{j-1}}^{\bowtie}$
32 :   $\mathsf{PK}_j \leftarrow g^{\mathrm{k}_j}$
33 : $\textbf{if } \mathrm{b} = 0$
34 :   $\mathsf{PK}_{\mathsf{fwl}_i} \leftarrow Y$
35 :   $\vec{\mathrm{C}} \xleftarrow{\$} \mathbb{G}^{l+2}$
36 : $\textbf{else}$
37 :   $\mathsf{PK}_{\mathsf{fwl}_i} \leftarrow Y^{\mathrm{k}_{\mathsf{fwl}_i - 1}}$
38 :   $\vec{r} \xleftarrow{\$} (\mathbb{Z}_q^*)^{l+2}$
39 :   $\vec{\mathrm{C}} \xleftarrow{\$} X^{\vec{r}}$
40 : $\textbf{for } j \in \{\mathsf{fwl}_i\text{+}1, \ldots, \mathsf{fwr}_i\} \textbf{ do}$
41 :   $\Delta_j \xleftarrow{\$} \mathbb{Z}_q^*$
42 :   $\mathsf{PK}_j \leftarrow \mathsf{PK}_{j-1}^{\Delta_j}$
```

Figure 31: Part 1. Reduction $\mathcal{B}_{4.4}$ for proof of Theorem 4.4, in hybrid i. On lines 3 and 6, $\texttt{ors}$ refers to the set $\{\mathcal{O}.\mathsf{Enc}, \mathcal{O}.\mathsf{Next}, \mathcal{O}.\mathsf{Upd}, \mathcal{O}.\mathsf{Corr}\}$. On lines 28 and 31, $\bowtie$ indicates Δ_0 and $\Delta_{\mathsf{fwr}_i+1}$ are skipped in the computation.

```
$\mathcal{O}.\mathsf{Enc}(\vec{\mathrm{M}})$
43 :  $\mathsf{c} \leftarrow \mathsf{c} + 1$
44 :  if $\mathsf{c} = \mathrm{h}$
45 :      $\vec{\mathrm{C}}_{\mathsf{e}} \leftarrow \vec{\mathrm{C}}; \mathsf{inf} \leftarrow \mathsf{e}$
46 :  else
47 :      $\vec{\mathsf{inf}} \xleftarrow{\$} (\mathbb{Z}_q^*)^{l+2}$
48 :      $\Pi(\vec{\mathrm{M}}) \xleftarrow{\$} g^{\vec{\mathsf{inf}}}$   / see caption
49 :      $\vec{\mathrm{C}}_{\mathsf{e}} \leftarrow \mathsf{PK}_{\mathsf{e}}^{\vec{\mathsf{inf}}}$
50 :  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \vec{\mathrm{C}}_{\mathsf{e}}, \mathsf{e}; \mathsf{inf} \textbf{ or } \vec{\mathsf{inf}})\}$
51 :  return $\vec{\mathrm{C}}_{\mathsf{e}}$

$\mathcal{O}.\mathsf{Next}$
52 :  $\mathsf{e} \leftarrow \mathsf{e} + 1$

$\mathcal{O}.\mathsf{Upd}(\vec{\mathrm{C}}_{\mathsf{e}-1})$
53 :  if $(j, \vec{\mathrm{C}}_{\mathsf{e}-1}, \mathsf{e} - 1; \mathsf{inf} \textbf{ or } \vec{\mathsf{inf}}) \notin \mathcal{L}$
54 :      return $\perp$
55 :  if $j = \mathrm{h}$
56 :      $\vec{\mathrm{C}}_{\mathsf{e}} \leftarrow \vec{\mathrm{C}}_{\mathsf{e}-1}^{\Delta_{\mathsf{e}}}$
57 :  else
58 :      $\vec{\mathrm{C}}_{\mathsf{e}} \leftarrow \mathsf{PK}_{\mathsf{e}}^{\vec{\mathsf{inf}}}$
59 :  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(j, \vec{\mathrm{C}}_{\mathsf{e}}, \mathsf{e}; \mathsf{inf} \textbf{ or } \vec{\mathsf{inf}})\}$
60 :  return $\vec{\mathrm{C}}_{\mathsf{e}}$

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$
61 :  do $\mathsf{Check}(\mathsf{inp}, \hat{\mathsf{e}}; \mathsf{e}; \mathsf{fwl}_i, \mathsf{fwr}_i)$
62 :  if $\mathsf{inp} = \mathsf{key}$
63 :      $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$, return $\mathrm{k}_{\hat{\mathsf{e}}}$
64 :  if $\mathsf{inp} = \mathsf{token}$
65 :      $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$, return $\Delta_{\hat{\mathsf{e}}}$

Create $\tilde{\mathrm{C}}$ with $(\vec{\tilde{\mathrm{M}}}, \vec{\tilde{\mathrm{C}}})$
66 :  if $(\mathrm{h}, \vec{\tilde{\mathrm{C}}}, \tilde{\mathsf{e}} - 1; \mathsf{inf}) \notin \mathcal{L}$
67 :      return ABORT
68 :  if $\mathrm{b} = 0$
69 :      $\vec{s} \xleftarrow{\$} (\mathbb{Z}_q^*)^{l+2}$
70 :      $\Pi(\vec{\tilde{\mathrm{M}}}) \xleftarrow{\$} X^{\vec{s}}$
71 :      $\vec{\tilde{\mathrm{C}}}_{\mathsf{fwl}_i} \leftarrow Z^{\vec{s}}$
72 :  else
73 :      $\Pi(\vec{\tilde{\mathrm{M}}}) \xleftarrow{\$} \mathbb{G}^{l+2}$
74 :      $\vec{\tilde{\mathrm{C}}}_{\mathsf{fwl}_i} \leftarrow \left(Z^{\prod_{j=\mathsf{inf}+1}^{\mathsf{fwl}_i - 1} \Delta_j}\right)^{\vec{r}}$
75 :  for $j \in \{0, \ldots, \mathsf{fwl}_i - 1\}$ do
76 :      $\vec{\tilde{\mathrm{C}}}_j \leftarrow \vec{\tilde{\mathrm{C}}}^{(\prod_{k=0}^{j} \Delta_k)/(\prod_{k=0}^{\tilde{\mathsf{e}}-1} \Delta_k)}$   / left
77 :  for $j \in \{\mathsf{fwl}_i + 1, \ldots, \mathsf{fwr}_i\}$ do
78 :      $\vec{\tilde{\mathrm{C}}}_j \leftarrow \vec{\tilde{\mathrm{C}}}_{j-1}^{\Delta_j}$   / embed
79 :  for $j \in \{\mathsf{fwr}_i + 1, \ldots, n\}$ do
80 :      $\vec{\tilde{\mathrm{C}}}_j \leftarrow \{\Pi(\vec{\tilde{\mathrm{M}}})\}^{k_j}$   / right
81 :  $\tilde{\mathcal{L}} \leftarrow \cup_{j=0}^{n} \{(\vec{\tilde{\mathrm{C}}}_j, j)\}$
82 :  return $\vec{\tilde{\mathrm{C}}}_{\tilde{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$
83 :  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
84 :  find$(\vec{\tilde{\mathrm{C}}}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{L}}$
85 :  return $\vec{\tilde{\mathrm{C}}}_{\mathsf{e}}$
```

Figure 31: Part 2. Reduction $\mathcal{B}_{4.4}$ for proof of Theorem 4.4, in hybrid i. On line 48 of $\mathcal{O}.\mathsf{Enc}$, vector $\Pi(\vec{\mathrm{M}}) = (\pi_0(\mathrm{N}), \pi_{\mathrm{N}\|1\|0}(\mathrm{M}_1), \ldots, \pi_{\mathrm{N}\|l\|0}(\mathrm{M}_l), \pi_{\mathrm{N}\|l\|1}(\Sigma))$.

any successfully-forged ciphertext to compute the CDH output to its CDH challenger.

$\mathsf{SHINE0}$ is $\mathsf{INT\text{-}CTXT^s}$.

Theorem 5.1 ($\mathsf{SHINE0}$ is $\mathsf{INT\text{-}CTXT^s}$)**.** Let $\mathbb{G}$ be a group of order q (a λ-bit prime, where $\lambda = v + m + l$) with generator g, and let $\mathsf{SHINE0}$ be the updatable encryption scheme described in Fig. 25. For any $\mathsf{INT\text{-}CTXT^s}$ adversary $\mathcal{A}$ against $\mathsf{SHINE0}$ that asks at most $\mathrm{Q_E}$ queries to $\mathcal{O}.\mathsf{Enc}$ before it asks its $\mathcal{O}.\mathsf{Try}$ query, there exists an adversary $\mathcal{B}_{5.1}$ against CDH such that

$$\mathbf{Adv}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{SHINE0},\,\mathcal{A}}(\lambda) \leq 1/2^l + \mathrm{Q_E}(n+1)^2\mathbf{Adv}^{\mathsf{CDH}}_{\mathcal{B}_{5.1}}.$$

Proof. Note that the probability of a random string ends by 0^l is $1/2^l$.

In the $\mathsf{INT\text{-}CTXT^s}$ game, whenever, the adversary sends a forgery, suppose C^*, to the $\mathcal{O}.\mathsf{Try}$ oracle. If the trivial win conditions does not trigger, then C^* is a new ciphertext to the challenger and there exists an insulated region around the challenge epoch. We split the proof into two parts based on if ${\mathrm{C}^*}^{1/\mathrm{k_e}}$ is a new value to the challenger:

1. If ${\mathrm{C}^*}^{1/\mathrm{k_e}}$ is a new value, the random permutation π^{-1} will pick a random string a as the output of $\pi^{-1}({\mathrm{C}^*}^{1/\mathrm{k_e}})$. So the probability of a is valid (a $(v+m+l)$-bit string with a l-bit zero string in the end) is upper bounded by $1/2^l$.

2. If ${\mathrm{C}^*}^{1/\mathrm{k_e}}$ is an existed value, denote this event as E_3, we claim that the probability of E_3 happens is very low. Which means it is hard to provide a valid forgery with a known permutation value. In other words, without the knowledge of the encryption key, it is difficult to provide a correct exponentiation. Formally, we prove the following inequality in Lemma 5.2.

   $$\mathbf{Pr}[E_3] = \mathbf{Pr}[{\mathrm{C}^*}^{1/\mathrm{k_e}} \text{ exists}, \mathrm{C}^* \text{ is new}] \leq \mathrm{Q_E}(n+1)^2\mathbf{Adv}^{\mathsf{CDH}}.$$

In order to analyze the security, we define some events:

- $E_1 = \{\mathrm{C}^* \text{ is new}\}$,
- $E_2 = \{{\mathrm{C}^*}^{1/\mathrm{k_e}} \text{ is new}, \mathrm{C}^* \text{ is new}\}$,

- Recall $E_3 = \{\mathrm{C}^{*1/\mathrm{k_e}}$ exists, C^* is new$\}$.

Denote the experiment $\mathbf{Exp}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{SHINE0},\,\mathcal{A}}$ to be $\mathbf{Exp}$. Then we have the following results:

- $\mathbf{Pr}[\mathbf{Exp} = 1 \mid \neg E_1] = 0$.
- We have proved $\mathbf{Pr}[\mathbf{Exp} = 1 \mid E_2] \leq 1/2^l$ in part 1.
- Events $\neg E_1, E_2, E_3$ are disjoint from each other, so $\mathbf{Pr}[\neg E_1]+\mathbf{Pr}[E_2]$ $+\mathbf{Pr}[E_3] = 1$.
- We have proved $\mathbf{Pr}[E_3] \leq \mathrm{Q_E}(n+1)^2\mathbf{Adv}^{\mathsf{CDH}}$ in Lemma 5.2.

Applying the above properties, we can compute the INT-CTXT$^{\mathsf{s}}$ advantage

$$\begin{aligned}
&\mathbf{Adv}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{SHINE0},\,\mathcal{A}}(\lambda)\\
&\quad= \mathbf{Pr}[\mathbf{Exp} = 1]\\
&\quad= \mathbf{Pr}[\mathbf{Exp} = 1 \mid \neg E_1] \cdot \mathbf{Pr}[\neg E_1] + \mathbf{Pr}[\mathbf{Exp} = 1 \mid E_2] \cdot \mathbf{Pr}[E_2]\\
&\qquad + \mathbf{Pr}[\mathbf{Exp} = 1 \mid E_3] \cdot \mathbf{Pr}[E_3]\\
&\quad= \mathbf{Pr}[\mathbf{Exp} = 1 \mid E_2] \cdot \mathbf{Pr}[E_2] + \mathbf{Pr}[\mathbf{Exp} = 1 \mid E_3] \cdot \mathbf{Pr}[E_3]\\
&\quad\leq \mathbf{Pr}[\mathbf{Exp} = 1 \mid E_2] + \mathbf{Pr}[E_3]\\
&\quad\leq 1/2^l + \mathrm{Q_E}(n+1)^2\mathbf{Adv}^{\mathsf{CDH}}.
\end{aligned}$$

Lemma 5.2. Let $\mathcal{A}$ be an INT-CTXT$^{\mathsf{s}}$ adversary against SHINE0 that asks at most $\mathrm{Q_E}$ queries to $\mathcal{O}$.Enc before it asks its $\mathcal{O}$.Try query. Suppose $\tilde{\mathrm{C}}$ is a forgery attempt provided by $\mathcal{A}$ and the corresponding permutation value is $\tilde{c}$. Define E to be the event that $\tilde{c}$ is an existed value but $\tilde{\mathrm{C}}$ is a new value. Then there exists an adversary $\mathcal{B}_{5.2}$ against CDH such that

$$\mathbf{Pr}[E] \leq \mathrm{Q_E}(n+1)^2\mathbf{Adv}^{\mathsf{CDH}}_{\mathcal{B}_{5.2}}$$

Proof. Suppose $\mathcal{A}$ is an adversary against INT-CTXT$^{\mathsf{s}}$ game, and it can provide a forgery such that $\tilde{\mathrm{C}}$ is a new ciphertext but the underlying permutation value is an existed one with probability $\mathbf{Pr}[E]$. We claim that there exists a reduction $\mathcal{B}_{5.2}$, detailed in Figure 32, such that it wins CDH game with probability $\mathbf{Pr}[E] \cdot \frac{1}{\mathrm{Q_E}(n+1)^2}$.

```
Reduction $\mathcal{B}_{5.2}$ playing $\mathbf{Exp}^{\mathsf{CDH}}_{\mathbb{G},\,\mathcal{B}_{5.2}}$
1 :  receive $(g, X, Y)$
2 :  do Setup
3 :  $\mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Try}}(\lambda)$
4 :  if $\mathsf{twf} = 1$ or ABORT occurred
5 :      $\mathsf{win} \leftarrow 0$
6 :  else
7 :      return $\mathsf{win}$

Setup($\lambda$)
8 :  $\mathrm{k}_0 \leftarrow \mathsf{SHINE0.KG}(\lambda)$
9 :  $\Delta_0 \leftarrow \perp$;  $\mathsf{e}, \mathsf{c} \leftarrow 0$;  $\mathsf{win} \leftarrow 0$
10 : $\mathcal{L}^*, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$
11 : $\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}} \xleftarrow{\$} \{0, \ldots, n\}$
12 : $\mathrm{h} \xleftarrow{\$} \{1, \ldots, \mathrm{Q_E}\}$
13 : for $j \in \{0, \ldots, \hat{\mathsf{fwl}}\text{-}1\}$ do
14 :     $\mathrm{k}_j \xleftarrow{\$} \mathbb{Z}_q^*$
15 :     $\Delta_j \leftarrow \frac{\mathrm{k}_j}{\mathrm{k}_{j-1}}$        // except $\Delta_0$
16 :     $\mathsf{PK}_j \leftarrow g^{\mathrm{k}_j}$
17 : for $\{\hat{\mathsf{fwr}}\text{+}1, \ldots, n\}$ do
18 :     $\mathrm{k}_j \xleftarrow{\$} \mathbb{Z}_q^*$
19 :     $\Delta_j \leftarrow \frac{\mathrm{k}_j}{\mathrm{k}_{j-1}}$        // except $\Delta_{\hat{\mathsf{fwr}}+1}$
20 :     $\mathsf{PK}_j \leftarrow g^{\mathrm{k}_j}$
21 : $\mathsf{PK}_{\hat{\mathsf{fwl}}} \leftarrow Y$
22 : for $j \in \{\hat{\mathsf{fwl}}\text{+}1, \ldots, \hat{\mathsf{fwr}}\}$ do
23 :     $\Delta_j \xleftarrow{\$} \mathbb{Z}_q^*$; $\mathsf{PK}_j \leftarrow \mathsf{PK}_{j-1}^{\Delta_j}$

$\mathcal{O}.\mathsf{Enc}(\mathrm{M})$
24 : $\mathsf{c} \leftarrow \mathsf{c} + 1$
25 : if $\mathsf{c} = \mathrm{h}$
26 :     if $\mathsf{e} < \hat{\mathsf{fwl}}$
27 :         $\pi(\mathrm{N}\|\mathrm{M}\|0^l) \leftarrow X$
28 :         $\mathrm{C}_\mathsf{e} \leftarrow X^{\mathrm{k}_\mathsf{e}}$
29 :     else
30 :         return ABORT
31 : else
32 :     $r \xleftarrow{\$} \mathbb{Z}_q^*$;
33 :     $\pi(\mathrm{N}\|\mathrm{M}\|0^l) \leftarrow g^r$
34 :     $\mathrm{C}_\mathsf{e} \leftarrow \mathsf{PK}_\mathsf{e}^r$
35 : $\mathcal{L}^* \leftarrow \mathcal{L}^* \cup \{(\mathsf{c}, \mathrm{C}_\mathsf{e}, \mathsf{e}; r)\}^{\gtrdot}$
36 : return $\mathrm{C}_\mathsf{e}$

$\mathcal{O}.\mathsf{Next}()$
37 : $\mathsf{e} \leftarrow \mathsf{e} + 1$

$\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}-1})$
38 : if $(j, \mathrm{C}_{\mathsf{e}-1}, \mathsf{e} - 1; r) \notin \mathcal{L}^*$
39 :     return $\perp$
40 : if $j = \mathrm{h}$
41 :     if $\mathsf{e} < \hat{\mathsf{fwl}}$
42 :         $\mathrm{C}_\mathsf{e} \leftarrow \mathrm{C}_{\mathsf{e}-1}^{\Delta_\mathsf{e}}$
43 :     else
44 :         return ABORT
45 : else
46 :     $\mathrm{C}_\mathsf{e} \leftarrow \mathsf{PK}_\mathsf{e}^r$
47 : $\mathcal{L}^* \leftarrow \mathcal{L}^* \cup \{(j, \mathrm{C}_\mathsf{e}, \mathsf{e}; r)\}$
48 : return $\mathrm{C}_\mathsf{e}$
```

Figure 32: Part 1. Reduction $\mathcal{B}_{5.2}$ for proof of Lemma 5.2. On line 35, $\gtrdot$ indicates r is empty when $\mathsf{c} = \mathrm{h}$.

```
$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$
    do $\mathsf{Check}(\mathsf{inp}, \hat{\mathsf{e}}; \mathsf{e}; \hat{\mathsf{fwl}}, \hat{\mathsf{fwr}})$
    if $\mathsf{inp} = \mathsf{key}$
        $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
        return $\mathrm{k}_{\hat{\mathsf{e}}}$
    if $\mathsf{inp} = \mathsf{token}$
        $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
        for $i \in \mathcal{T}^*$ do
            for $(j, \mathrm{C}_{i-1}, i-1; r) \in \mathcal{L}^*$ do
                $\mathrm{C}_i \leftarrow \mathcal{O}.\mathsf{Upd}(\mathrm{C}_{i-1})$
                $\mathcal{L}^* \leftarrow \mathcal{L}^* \cup \{(j, \mathrm{C}_i, i; r)\}$
        return $\Delta_{\hat{\mathsf{e}}}$
```

```
$\mathcal{O}.\mathsf{Try}(\tilde{\mathrm{C}})$
    if $\mathsf{phase} = 1$
        return $\perp$
    $\mathsf{phase} \leftarrow 1$
    if $\tilde{\mathsf{e}} \in \mathcal{K}^*$ or $\tilde{\mathrm{C}} \in \mathcal{L}^*$
        $\mathsf{twf} \leftarrow 1$
    if $\tilde{\mathsf{e}} \notin \{\hat{\mathsf{fwl}}, \ldots, \hat{\mathsf{fwr}}\}$
        $\mathsf{twf} \leftarrow 1$
    $y \leftarrow \tilde{\mathrm{C}}^{1/\prod_{\mathsf{e}=\hat{\mathsf{fwl}}+1}^{\mathsf{e}=\tilde{\mathsf{e}}} \Delta_{\mathsf{e}}}$
    call Chall with $y$; get $\mathrm{b}$
    $\mathsf{win} \leftarrow \mathrm{b}$
```

Figure 32: Part 2. Reduction $\mathcal{B}_{5.2}$ for proof of Lemma 5.2.

The reduction will guess the location of firewalls around the epoch when $\mathcal{O}.\mathsf{Try}$ query happens, furthermore, it guess which message (suppose the h-th encryption) might be the underlying message of the forgery. After it receives the CDH values g^a, g^b, it embeds g^a to the h-th encryption as $\pi(\mathrm{N}\|\mathrm{M}\|0^l) \leftarrow g^a$, embeds g^b to the public key value on the left firewall as $\mathsf{PK}_{\hat{\mathsf{fwl}}} = g^{\mathrm{k}_{\hat{\mathsf{fwl}}}} \leftarrow g^b$. Then $\tilde{\mathrm{C}} = g^{ab\prod_{\mathsf{e}=\hat{\mathsf{fwl}}+1}^{\mathsf{e}=\tilde{\mathsf{e}}} \Delta_{\mathsf{e}}}$ with probability $\mathbf{Pr}[E] \cdot \frac{1}{\mathrm{Q_E}(n+1)^2}$, which is the advantage of winning the CDH game. $\square$

$\mathsf{MirrorSHINE}$ **is** $\mathsf{INT\text{-}CTXT^s}$**.**

Theorem 5.3 ($\mathsf{MirrorSHINE}$ is $\mathsf{INT\text{-}CTXT^s}$)**.** Let $\mathbb{G}$ be a group of order q (a λ-bit prime) with generator g, and let $\mathsf{MirrorSHINE}$ be the updatable encryption scheme described in Fig. 26. For any $\mathsf{INT\text{-}CTXT^s}$ adversary $\mathcal{A}$ against $\mathsf{MirrorSHINE}$ that asks at most queries at most $\mathrm{Q_E}$ queries to $\mathcal{O}.\mathsf{Enc}$ before it asks its $\mathcal{O}.\mathsf{Try}$ query, there exists an adversary $\mathcal{B}_{5.3}$ against CDH such that

$$\mathbf{Adv}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{MirrorSHINE},\ \mathcal{A}}(\lambda) \leq 1/2^{|\mathcal{N}|} + 2\mathrm{Q_E}(n+1)^2\mathbf{Adv}^{\mathsf{CDH}}_{\mathcal{B}_{5.3}}$$

Proof. Similar to the proof in Theorem 5.1. Suppose $\tilde{\mathrm{C}} = (\tilde{\mathrm{C}}^1, \tilde{\mathrm{C}}^2)$ is a new ciphertext sent as the $\mathcal{O}.\mathsf{Try}$ query. Then at least one block of the ciphertext, suppose $\tilde{\mathrm{C}}^i$, is new. We split the proof in two parts:

- If $(\tilde{\mathrm{C}}^i)^{1/\mathsf{k_e}}$ is new. The random permutation π_i^{-1} will pick a random string a_i as the output of $\pi_i^{-1}((\tilde{\mathrm{C}}^i)^{1/\mathsf{k_e}})$. So the probability of a_i is valid (satisfy $a_1 = a_2$) is upper bounded by $1/2^{|\mathcal{N}|}$.
- If $(\tilde{\mathrm{C}}^i)^{1/\mathsf{k_e}}$ is an existed value outputted by permutation π_i. Similar to the proof in Theorem 5.1, we can prove that

$$\mathbf{Pr}[(\tilde{\mathrm{C}}^i)^{1/\mathsf{k_e}} \text{ exists}, \tilde{\mathrm{C}}^i \text{ is new }] \leq 2\mathrm{Q_E}(n+1)^2\mathbf{Adv}^{\mathsf{CDH}}_{\mathcal{B}_{5.3}}. \qquad \square$$

$\mathsf{OCBSHINE}$ is $\mathsf{INT\text{-}CTXT^s}$.

Theorem 5.4 ($\mathsf{OCBSHINE}$ is $\mathsf{INT\text{-}CTXT^s}$). Let $\mathbb{G}$ be a group of order q (a λ-bit prime, $\lambda = m$) with generator g, and let $\mathsf{OCBSHINE}$ be the updatable encryption scheme described in Fig. 28. For any $\mathsf{INT\text{-}CTXT^s}$ adversary $\mathcal{A}$ against $\mathsf{OCBSHINE}$ that asks at most $\mathrm{Q_E}$ queries, on messages with at most L message blocks, to $\mathcal{O}.\mathsf{Enc}$ before it asks its $\mathcal{O}.\mathsf{Try}$ query, there exists an adversary $\mathcal{B}_{5.4}$ against CDH such that

$$\mathbf{Adv}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{OCBSHINE},\, \mathcal{A}}(\lambda) \leq \frac{\mathrm{Q_E}^2 + \mathrm{Q_E}}{2^{m-a}} + (\mathrm{L}+1)\mathrm{Q_E}(n+1)^2\mathbf{Adv}^{\mathsf{CDH}}_{\mathcal{B}_{5.4}}.$$

Proof. **Game 0**

The first game is the experiment $\mathbf{Exp}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{OCBSHINE},\, \mathcal{A}}$, given in Fig. 6 and Fig. 10. As definition 4 we have

$$\mathbf{Adv}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{OCBSHINE},\, \mathcal{A}}(\lambda) = \mathbf{Pr}[\mathcal{G}_0 = 1]$$

Game 1

Modify the response of the encryption oracle such that the game randomly picks a nonce and if the nonce repeats, it aborts the game and returns $\mathsf{win} = 0$. This loss is upper-bounded by $\frac{\mathrm{Q_E}^2}{2^{m-a}}$, then we have

$$|\mathbf{Pr}[\mathcal{G}_0 = 1] - \mathbf{Pr}[\mathcal{G}_1 = 1]| \leq \frac{\mathrm{Q_E}^2}{2^{m-a}}$$

Finally, we claim $\mathbf{Pr}[\mathcal{G}_1 = 1] \leq \frac{\mathrm{Q_E}}{2^{m-a}} + (l+1)\mathrm{Q_E}(n+1)^2\mathbf{Adv}^{\mathsf{CDH}}$. Similar to the proof in Theorem 5.1. Suppose $\tilde{\mathrm{C}} = (\tilde{\mathrm{C}}^0, \ldots, \tilde{\mathrm{C}}^{\tilde{l}+1})$ is a forgery attempt sent as the $\mathcal{O}.\mathsf{Try}$ query in epoch $\tilde{\mathsf{e}}$. Suppose $\tilde{c}_i = (\tilde{\mathrm{C}}^i)^{1/\mathrm{k}_{\tilde{\mathsf{e}}}}$, the corresponding message is $\tilde{\mathrm{M}}_i$. Let $\tilde{\mathrm{N}}$ be the underlying nonce and $\tilde{\Sigma}$ be the underlying checksum. We consider the following two situations.

1. If $(\tilde{c}_0, \ldots, \tilde{c}_{\tilde{l}+1})$ is new, we claim that the probability of the adversary correctly guessing $\tilde{\Sigma}$ is upper-bounded by $\frac{\mathrm{Q_E}}{2^{m-a}}$.
   (a) If $\tilde{c}_0$ is new, then either $\tilde{\mathrm{N}}$ is new or $|\tilde{\mathrm{M}}_{\tilde{l}}|$ is new.
      i. If $\tilde{\mathrm{N}}$ is equal to some already existing nonce N, we claim that the probability that this event happens is very low. Since π_0 is a random permutation, the first $(m\text{-}a)$ bits of the preimage of $\tilde{c}_0$ under π_0, which is $\tilde{\mathrm{N}}$, is as likely as any $(m\text{-}a)$-bit string. So the probability that $\tilde{\mathrm{N}}$ collides with one of the existed nonces is upper bounded by $\mathrm{Q_E}/2^{m-a}$.
      ii. If $\tilde{\mathrm{N}}$ is new, therefore $\pi_{\tilde{\mathrm{N}}\|\tilde{l}\|1}$ is a new random permutation. The adversary sees no image and preimage of $\pi_{\tilde{\mathrm{N}}\|\tilde{l}\|1}$. The preimage of $\tilde{c}_{\tilde{l}+1}$ under $\pi_{\tilde{\mathrm{N}}\|\tilde{l}\|1}$, which is $\tilde{\Sigma}$, is as likely as any other m-bit string. So the probability that the adversary correctly guesses $\tilde{\Sigma}$ (which means $\tilde{\Sigma} = \oplus_{i=1}^{l}\tilde{\mathrm{M}}_i$) is 2^{-m}.
   (b) If $\tilde{c}_0 = c_0$ for some already existing $(c_0, \ldots, c_{l+1})$ but $\tilde{l} \neq l$, similar to the above situation $\pi_{\tilde{\mathrm{N}}\|\tilde{l}\|1}$ is a new random permutation. So the probability that the adversary correctly guesses $\tilde{\Sigma}$ is 2^{-m}.
   (c) If $\tilde{c}_0 = c_0$ for some already existing $(c_0, \ldots, c_{l+1})$ and $\tilde{l} = l$ but $\tilde{c}_j \neq c_j$ for some $j \in \{1, \ldots, \tilde{l}\}$. The adversary sees one image of $\pi_{\tilde{\mathrm{N}}\|j\|0}$, that is, c_j. The preimage of $\tilde{c}_j$ under $\pi_{\tilde{\mathrm{N}}\|j\|0}$, which is $\tilde{\mathrm{M}}_j$, is as likely as any m-bit string except for M_j (or $\mathrm{M}_l\|0^*$ when $j = l$). So the probability that the adversary correctly guesses $\tilde{\Sigma}$ is $\frac{1}{2^m-1}$.
   (d) If $\tilde{c}_0 = c_0$ for some already existing $(c_0, \ldots, c_{l+1})$ and $\tilde{l} = l$ and $\tilde{c}_j = c_j$ for all $j \in \{1, \ldots, \tilde{l}\}$ but $\tilde{c}_{\tilde{l}+1} \neq c_{l+1}$. This event is impossible to happen, as $\tilde{\mathrm{N}} = \mathrm{N}, |\tilde{\mathrm{M}}_{\tilde{l}}| = |\mathrm{M}_l|$ and $\tilde{l} = l$ so

$\pi_{\tilde{\mathrm{N}}\|\tilde{l}\|1} = \pi_{\mathrm{N}\|l\|1}$, furthermore $\tilde{\mathrm{M}}_j = \mathrm{M}_j$ for all $j \in \{1, \ldots, \tilde{l}\}$, so $\tilde{\Sigma} = \Sigma$ and then $\tilde{c}_{\tilde{l}+1} = c_{l+1}$.

2. If $(\tilde{c}_0, \ldots, \tilde{c}_{\tilde{l}+1})$ is an already existing value output by the permutations but $\tilde{\mathrm{C}} = (\tilde{\mathrm{C}}^0, \ldots, \tilde{\mathrm{C}}^{\tilde{l}+1})$ is a new ciphertext, then this is equivalent to $(\tilde{c}_0, \ldots, \tilde{c}_{\tilde{l}})$ exists but $(\tilde{\mathrm{C}}^0, \ldots, \tilde{\mathrm{C}}^{\tilde{l}})$ is new, as in the analysis of 1.(d). Similar to the proof in Theorem 5.1, we can prove that the probability of this event happening is upper-bounded by CDH advantage:

$$\begin{aligned}&\mathbf{Pr}[\ (\tilde{c}_0, \ldots, \tilde{c}_{\tilde{l}}) \text{ exists but } (\tilde{\mathrm{C}}^0, \ldots, \tilde{\mathrm{C}}^{\tilde{l}}) \text{ is new }]\\ &\quad \leq (\mathrm{L}+1)\mathrm{Q}_{\mathrm{E}}(n+1)^2\mathbf{Adv}^{\mathsf{CDH}}.\end{aligned}$$

□

5.6 Implementing the SHINE Schemes

In the proofs of Theorem 4 and Theorem 5, we require that π is a random (unkeyed) permutation which must be followed by a mapping to an appropriate group for exponentiation by the epoch key. For the permutation we do not need any specific and strong properties that are provided by modern constructions of block ciphers and sponges. As far as the proof goes, and in practice, the property that we want from this permutation is that given a ciphertext and the inverse of the epoch key k_{e}, the only way to extract useful information about the message is to apply the inverse permutation π^{-1}. The random permutation model (or ideal cipher model) is thus the tool we need here to create a simple interface for this aspect of our proof.

The different members of the SHINE family are suited to different application scenarios. The variants SHINE0 and MirrorSHINE are best suited to cases where messages are of small, fixed size, such as customer credentials (or phone contact details, to return to the motivating example in the Introduction). For applications with longer messages (i.e. larger than the size of the exponentiation group), OCBSHINE is considerably faster and we will assume that these choices are made in our implementation suggestions. This removes any need for larger groups in order to encrypt longer messages. Using larger groups would not only carry a significant performance penalty, but also force us to construct custom large blocklength block ciphers. Although this can be done (and has been for RSA groups [GOR18], where our approach would not work), the analysis is tricky.

Instantiating the ideal permutation. The message block in SHINE0, MirrorSHINE and the final message block in OCBSHINE must be appropriately padded to allow application of the permutation. The permutation could be deployed using a variable-output-length sponge construction, a block cipher or an authenticated encryption scheme with a fixed key and suitably large nonce space. In practice, we suggest to instantiate the random permutation with a block cipher of a suitable block length. AES has only 128-bit blocks which does not match the minimum required size of the group, so we instead suggest block ciphers such as Threefish, or original Rijndael, allowing block lengths of 256 or 512 bits.

Mapping to elliptic curve group. We would like to instantiate our groups using elliptic curves. Using modern techniques it is always possible to find a suitable curve over a field with a size matching the block length of the ideal permutation, but using standard curves like NIST P-256 or P-521 seems desirable. A standard approach [Kob87] is to embed bit strings in the X-coordinate of a point as follows. Note that close to half the field elements are X-coordinates of points. Given a field of size q, we consider a t-bit block as an integer x_0 and find a small integer u such that $u2^t + x_0$ is the X-coordinate of a curve point. If $\log q - t$ is between 8 and 9, this will fail to terminate with probability around 2^{-256} under reasonable assumptions.

With this approach we could use Threefish with 512-bit blocks together with NIST P-521 curve. If we want to use 256-bit blocks from Threefish, or original Rijndael, together with NIST P-256 curve, we can use a standard block cipher iteration trick [RSA78] to reduce the block length from 256 bits, so that embedding in the X-coordinate still works, as follows. With block length $t + \tau$, concatenate a t-bit block with τ leading zeros and apply the block cipher until the τ leading bits of the result are all zeros. Discard these zeros to get a t-bit block. This is fairly cheap as for our purposes 8 or 9 bits will do.

Note that we have constructed an injective embedding of a block into an elliptic curve, not a bijection as assumed in our proofs. When we sample group elements in our proof, we must take care to sample points in the image of our embedding, but this can be done cheaply.

6 Conclusions

In this work we provided a suite of new updatable encryption schemes, collectively called SHINE, and a new definition of security xxIND-UE-atk (that implies prior notions) in which we prove our schemes secure. In the process, we provided a greater understanding of the proof techniques that are inherent in the strong corruption model that is desirable for updatable encryption – in particular in the context of deterministic updates that is desirable in practice.

Acknowledgements. This research was funded by the Research Council of Norway under Project No. 248166. Gareth T. Davies is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreement 802823. We would like to thank Frederik Armknecht, Håvard Raddum and Mohsen Toorani for fruitful discussions in the initial stages of this project, and anonymous reviewers for a number of useful suggestions for improvement. We would also like to thank Adam O'Neill for a number of useful suggestions for improvement. Parts of this work were conducted while Gareth T. Davies was employed by NTNU Trondheim and the University of Paderborn.

References

[AFGH05] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2005, San Diego, California, USA*. The Internet Society, 2005.

[BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In Kaisa Nyberg, editor, *Proceedings of EUROCRYPT 1998*, volume 1403 of *Lecture Notes in Computer Science*, pages 127–144. Springer, 1998.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 410–428. Springer, 2013.

[BLMR15] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. *IACR Cryptology ePrint Archive, Report 2015/220*, 2015.

[BN08] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *J. Cryptology*, 21(4):469–491, 2008.

[CCFL17] Christian Cachin, Jan Camenisch, Eduarda Freire-Stögbuchner, and Anja Lehmann. Updatable tokenization: Formal definitions and provably secure constructions. In Aggelos Kiayias, editor, *Financial Cryptography and Data Security - 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers*, volume 10322 of *Lecture Notes in Computer Science*, pages 59–75. Springer, 2017.

[CH07] Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 185–194. ACM, 2007.

[Cou18] PCI Security Standards Council. Data security standard (PCI DSS v3.2.1), 2018. https://www.pcisecuritystandards.org/.

[CPS08] Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. The random oracle model and the ideal cipher model are equivalent. In David A. Wagner, editor, *Advances in Cryptology*

- CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings, volume 5157 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2008.

[DDLM19] Alex Davidson, Amit Deo, Ela Lee, and Keith Martin. Strong post-compromise secure proxy re-encryption. In Julian Jang-Jaccard and Fuchun Guo, editors, *Information Security and Privacy - 24th Australasian Conference, ACISP 2019, Christchurch, New Zealand, July 3-5, 2019, Proceedings*, volume 11547 of *Lecture Notes in Computer Science*, pages 58–77. Springer, 2019.

[DRC14] Sandra Diaz-Santiago, Lil María Rodríguez-Henríquez, and Debrup Chakraborty. A cryptographic study of tokenization systems. In Mohammad S. Obaidat, Andreas Holzinger, and Pierangela Samarati, editors, *SECRYPT 2014 - Proceedings of the 11th International Conference on Security and Cryptography, Vienna, Austria, 28-30 August, 2014*, pages 393–398. SciTePress, 2014.

[EPRS17a] Adam Everspaugh, Kenneth G. Paterson, Thomas Ristenpart, and Samuel Scott. Key rotation for authenticated encryption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 98–129. Springer, 2017.

[EPRS17b] Adam Everspaugh, Kenneth G. Paterson, Thomas Ristenpart, and Samuel Scott. Key rotation for authenticated encryption. *IACR Cryptology ePrint Archive, Report 2017/527*, 2017.

[GOR18] Craig Gentry, Adam O'Neill, and Leonid Reyzin. A unified framework for trapdoor-permutation-based sequential aggregate signatures. In Michel Abdalla and Ricardo Dahab, editors, *Public-Key Cryptography – PKC 2018*, pages 34–57, Cham, 2018. Springer International Publishing.

[JKR19] Stanislaw Jarecki, Hugo Krawczyk, and Jason K. Resch. Updatable oblivious key management for storage systems. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 379–393. ACM, 2019.

[KLR19a] Michael Klooß, Anja Lehmann, and Andy Rupp. (R)CCA secure updatable encryption with integrity protection. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 68–99. Springer, 2019.

[KLR19b] Michael Klooß, Anja Lehmann, and Andy Rupp. (R)CCA secure updatable encryption with integrity protection. *IACR Cryptology ePrint Archive, Report 2019/222*, 2019.

[Kob87] Neil Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[KRS$^+$03] Mahesh Kallahalla, Erik Riedel, Ram Swaminathan, Qian Wang, and Kevin Fu. Plutus: Scalable secure file sharing on untrusted storage. In Jeff Chase, editor, *Proceedings of the FAST '03 Conference on File and Storage Technologies, March 31 - April 2, 2003, Cathedral Hill Hotel, San Francisco, California, USA*. USENIX, 2003.

[Lee17] Ela Lee. Improved security notions for proxy re-encryption to enforce access control. In Tanja Lange and Orr Dunkelman, editors, *Progress in Cryptology - LATINCRYPT 2017 - 5th International Conference on Cryptology and Information Security in Latin America, Havana, Cuba, September 20-22, 2017, Revised Selected Papers*, volume 11368 of *Lecture Notes in Computer Science*, pages 66–85. Springer, 2017.

[Leh19] Anja Lehmann. Updatable encryption & key rotation (presentation slides), 2019. https://summerschool-croatia.cs.ru.nl/2019/slides/lehmann1.pdf.

[LT18a] Anja Lehmann and Björn Tackmann. Updatable encryption with post-compromise security. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 685–716. Springer, 2018.

[LT18b] Anja Lehmann and Björn Tackmann. Updatable encryption with post-compromise security. *IACR Cryptology ePrint Archive, Report 2018/118*, 2018.

[MS18] Steven Myers and Adam Shull. Practical revocation and key rotation. In Nigel P. Smart, editor, *Topics in Cryptology - CT-RSA 2018 - The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings*, volume 10808 of *Lecture Notes in Computer Science*, pages 157–178. Springer, 2018.

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and kdcs. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 327–346. Springer, 1999.

[RBBK01] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: a block-cipher mode of operation for efficient authenticated encryption. In Michael K. Reiter and Pierangela Samarati, editors, *CCS 2001, Proceedings of the 8th ACM Conference on Computer and Communications Security, Philadelphia, Pennsylvania, USA, November 6-8, 2001*, pages 196–205. ACM, 2001.

[RSA78] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[Sha49] Claude E Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4):656–715, 1949.

[SNS17] Kouichi Sakurai, Takashi Nishide, and Amril Syalim. Improved proxy re-encryption scheme for symmetric key cryptography. In *International Workshop on Big Data and Information Security, IWBIS 2017, Jakarta, Indonesia, September 23-24, 2017*, pages 105–111. IEEE, 2017.

A SHINE0 is IND-ENC -CPA Secure

Theorem 4.2. Let $\mathbb{G}$ be a group of order q (a λ-bit prime) with generator g, and let SHINE0 be the updatable encryption scheme described in Fig. 25. For any IND-ENC -CPA adversary $\mathcal{A}$ against SHINE0, there exists an adversary $\mathcal{B}_{4.2}$ against DDH such that

$$\mathbf{Adv}^{\mathsf{IND\text{-}ENC\text{-}CPA}}_{\mathsf{SHINE0},\,\mathcal{A}}(\lambda) \leq 2(n+1)^3 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_{4.2}}(\lambda).$$

Proof. Similarly to the proof of Theorem 4.1, we use the firewall technique and construct hybrid games. For $\mathrm{b} \in \{0,1\}$, define game $\mathcal{G}_i^{\mathrm{b}}$ as $\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}CPA\text{-}b}}_{\mathsf{SHINE0},\,\mathcal{A}}$ except for:

- The game randomly picks two numbers $\mathsf{fwl}_i, \mathsf{fwr}_i$ and if $\mathsf{fwl}_i, \mathsf{fwr}_i$ are not the i-th firewalls, a random bit is returned for b'. This loss is upper bounded by $(n+1)^2$;
- For challenge made in epoch $\tilde{\mathsf{e}}$ with input $(\bar{\mathrm{M}}_0, \bar{\mathrm{M}}_1)$: If $\tilde{\mathsf{e}} < \mathsf{fwl}_i$ then return a ciphertext of $\bar{\mathrm{M}}_1$, if $\tilde{\mathsf{e}} > \mathsf{fwr}_i$ return a ciphertext of $\bar{\mathrm{M}}_0$, and if $\mathsf{fwl}_i \leq \tilde{\mathsf{e}} \leq \mathsf{fwr}_i$ return a ciphertext of $\bar{\mathrm{M}}_{\mathrm{b}}$;
- After $\mathcal{A}$ outputs b': returns b' if $\mathsf{twf} \neq 1$ or some additional trivial win condition is triggered.

```
Reduction $\mathcal{B}_{4.2}$ playing $\mathbf{Exp}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_{4.2}}$ in hybrid $i$
1 :  receive $(g, X, Y, Z)$
2 :  do Setup
3 :  $\bar{\mathrm{M}}_0, \bar{\mathrm{M}}_1 \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
4 :  $\mathsf{phase} \leftarrow 1$, Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}_0, \bar{\mathrm{M}}_1)$, get $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
5 :  $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}})$
6 :  $\underline{\mathsf{twf} \leftarrow 1}$ if $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$
7 :  if ABORT occurred or $\mathsf{twf} = 1$
8 :      $\mathrm{b}' \xleftarrow{\$} \{0,1\}$, return $\mathrm{b}'$
9 :  if $(i, \mathsf{fwl}_i, \mathsf{fwr}_i) \notin \mathcal{FW}$
10 :     $\mathrm{b}' \xleftarrow{\$} \{0,1\}$, return $\mathrm{b}'$
11 : if $\mathrm{b}' = \mathrm{b}$
12 :     return 0
13 : else
14 :     return 1

Setup$(\lambda)$
19 : $\mathrm{b} \xleftarrow{\$} \{0,1\}; \mathrm{k}_0 \leftarrow \mathsf{SHINE0.KG}(\lambda)$
20 : $\Delta_0 \leftarrow \perp$;  $\mathsf{e} \leftarrow 0$;  $\mathsf{phase}, \mathsf{twf} \leftarrow 0$
21 : $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$
22 : $\mathsf{fwl}_i, \mathsf{fwr}_i \xleftarrow{\$} \{0, \ldots, n\}$
23 : $\mathsf{PK}_{\mathsf{fwl}_i} \leftarrow Y$
24 : for $j \in \{\mathsf{fwl}_i{+}1, \ldots, \mathsf{fwr}_i\}$ do
25 :     $\Delta_j \xleftarrow{\$} \mathbb{Z}_q^*; \mathsf{PK}_j \leftarrow \mathsf{PK}_{j-1}^{\Delta_j}$
26 : for $j \in \{0, \ldots, \mathsf{fwl}_i\text{-}1\} \cup \{\mathsf{fwr}_i{+}1, \ldots, n\}$ do
27 :     $\mathrm{k}_j \xleftarrow{\$} \mathbb{Z}_q^*; \Delta_j \leftarrow \frac{\mathrm{k}_j}{\mathrm{k}_{j-1}}^{\bowtie}; \mathsf{PK}_j \leftarrow g^{\mathrm{k}_j}$
```

Figure 33: Part 1. Reduction $\mathcal{B}_{4.2}$ for proof of Theorem 4.2. Embedding DDH tuples to challenge ciphertexts requires faithful responses to queries within the i-th insulated region. On line 27, $\bowtie$ indicates Δ_0 and $\Delta_{\mathsf{fwr}_i+1}$ are skipped in the computation.

```
O.Enc(M)
28 :  $r \xleftarrow{\$} \mathbb{Z}_q^*$
29 :  $\pi(\mathrm{N}||\mathrm{M}) \leftarrow g^r; \mathrm{C_e} \leftarrow \mathsf{PK}_\mathsf{e}^r$
30 :  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C_e}, \mathsf{e}; r)\}$
31 :  return $\mathrm{C_e}$
O.Next
31 :  $\mathsf{e} \leftarrow \mathsf{e} + 1$
O.Upd(C_{e-1})
32 :  if $(\cdot, \mathrm{C_{e-1}}, \mathsf{e}-1; r) \notin \mathcal{L}$
33 :      return $\perp$
34 :  $\mathrm{C_e} \leftarrow \mathsf{PK}_\mathsf{e}^r$
35 :  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C_e}, \mathsf{e}; r)\}$
36 :  return $\mathrm{C_e}$
O.Corr(inp, ê)
37 :  do Check$(\mathsf{inp}, \hat{\mathsf{e}}; \mathsf{e}; \mathsf{fwl}_i, \mathsf{fwr}_i)$
38 :  if $\mathsf{inp} = \mathsf{key}$
39 :      $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
40 :      return $\mathrm{k}_{\hat{\mathsf{e}}}$
41 :  if $\mathsf{inp} = \mathsf{token}$
42 :      $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
43 :      return $\Delta_{\hat{\mathsf{e}}}$

Create C̃ with (M̄_0, M̄_1)
44 :  $\pi(\mathrm{N}||\bar{\mathrm{M}}_\mathrm{b}) \leftarrow X$
45 :  $\pi(\mathrm{N}||\bar{\mathrm{M}}_{\mathrm{b}\oplus 1}) \xleftarrow{\$} \mathbb{G}$
46 :  $\tilde{\mathrm{C}}_{\mathsf{fwl}_i} \leftarrow Z$
47 :  for $j \in \{0, \ldots, \mathsf{fwl}_i - 1\}$ do
48 :      $\tilde{\mathrm{C}}_j \leftarrow (\pi(\mathrm{N}||\bar{\mathrm{M}}_1))^{k_j}$    // left
49 :  for $j \in \{\mathsf{fwl}_i + 1, \ldots, \mathsf{fwr}_i\}$ do
50 :      $\tilde{\mathrm{C}}_j \leftarrow \tilde{\mathrm{C}}_{j-1}^{\Delta_j}$    // embed
51 :  for $j \in \{\mathsf{fwr}_i + 1, \ldots, n\}$ do
52 :      $\tilde{\mathrm{C}}_j \leftarrow (\pi(\mathrm{N}||\bar{\mathrm{M}}_0))^{k_j}$    // right
53 :  $\tilde{\mathcal{L}} \leftarrow \cup_{j=0}^{n}\{(\tilde{\mathrm{C}}_j, j)\}$
54 :  return $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
O.UpdC̃
55 :  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
56 :  find$(\tilde{\mathrm{C}}_\mathsf{e}, \mathsf{e}) \in \tilde{\mathcal{L}}$
57 :  return $\tilde{\mathrm{C}}_\mathsf{e}$
```

Figure 33: Part 2. Reduction $\mathcal{B}_{4.2}$ for proof of Theorem 4.2.

Similarly to the computation in Theorem 4.1, we have

$$\mathbf{Adv}_{\mathsf{SHINE0},\,\mathcal{A}}^{\mathsf{IND\text{-}ENC\text{-}CPA}}(\lambda) = (n+1)^2 \cdot \left(\sum_{i=1}^{l} |\mathbf{Pr}[\mathcal{G}_i^1 = 1] - \mathbf{Pr}[\mathcal{G}_i^0 = 1]| \right),$$

for some l. Again we need to prove that $|\mathbf{Pr}[\mathcal{G}_i^1 = 1] - \mathbf{Pr}[\mathcal{G}_i^0 = 1]| \leq 2\mathbf{Adv}_{\mathbb{G},}^{\mathsf{DDH}}(\lambda)$.

We construct a reduction $\mathcal{B}_{4.2}$, detailed in Fig. 33, that is playing the standard DDH game and runs $\mathcal{A}_i$. The reduction $\mathcal{B}_{4.2}$ flips a coin b, and simulates $\mathcal{G}_i^{\mathrm{b}}$ by using DDH tuples (X, Y, Z) to output $\mathsf{SHINE0.Enc}(\bar{\mathrm{M}}_{\mathrm{b}})$ in the i-th insulated region. If $\mathcal{A}_i$ guess b correctly, then $\mathcal{B}_{4.2}$ guesses its real DDH tuples, otherwise, $\mathcal{B}_{4.2}$ guess its random DDH tuples. If $\mathcal{B}_{4.2}$ receives a real DDH tuple, then $\mathcal{B}_{4.2}$ perfectly simulates the input of $\mathcal{A}_i$ in $\mathcal{G}_i^{\mathrm{b}}$. If $\mathcal{B}_{4.2}$ receives a random DDH tuple, then $\mathcal{B}_{4.2}$ wins with probability 1/2. After some computation similar to that in the proof of Theorem 2.3 we have that $\mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_{4.2}}(\lambda) = \frac{1}{2}|\mathbf{Pr}[\mathcal{G}_i^1 = 1] - \mathbf{Pr}[\mathcal{G}_i^0 = 1]|$. □

B The BLMR Scheme of Boneh, Lewi, Montgomery, and Raghunathan

We present the original scheme given by Boneh et al. [BLMR13], which we denote by BLMR. The scheme is a direct application of the key-homomorphic PRFs defined in the same paper: the authors observed that the Naor-Reingold-Pinkas PRF [NPR99] is key homomorphic, and presented a number of other constructions based on DLIN and LWE. The updatable encryption scheme BLMR [BLMR13], which is ciphertext-independent and defined in Fig. 34, represented the first UE construction.

To present the schemes and the results in this section, we first need to introduce a definition of a key-homomorphic PRF, and also the regular (left-or-right) IND-CPA security definition for symmetric encryption.

Definition 13 (Key-homomorphic PRF [BLMR13])**.** Let $F : \mathcal{KS} \times \mathcal{X} \to \mathcal{Y}$ be some efficiently-computable function, where $(\mathcal{KS}, \oplus)$ and $(\mathcal{Y}, \otimes)$ are groups. Then, $(F, \oplus, \otimes)$ is a key-homomorphic PRF if F is a PRF, and for every $\mathrm{k}_1, \mathrm{k}_2 \in \mathcal{KS}$ and every $x \in \mathcal{X}$, $F(\mathrm{k}_1, x) \otimes F(\mathrm{k}_2, x) = F(\mathrm{k}_1 \oplus \mathrm{k}_2)$.

Definition 14. Let $\mathsf{SKE} = \{\mathsf{KG}, \mathsf{E}, \mathsf{D}\}$ be an symmetric encryption scheme. Then the IND-CPA advantage of an adversary $\mathcal{A}$ against SKE is defined as

$$\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SKE},\,\mathcal{A}}(\lambda) = \left|\mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\text{-}CPA\text{-}1}}_{\mathsf{SKE},\,\mathcal{A}} = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\text{-}CPA\text{-}0}}_{\mathsf{SKE},\,\mathcal{A}} = 1]\right|,$$

where the experiment $\mathbf{Exp}^{\mathsf{IND\text{-}CPA\text{-}b}}_{\mathsf{SKE},\,\mathcal{A}}$ is given in Fig. 35.

Note that BLMR is trivially insecure in terms of xxIND-UPD-atk (in any of its three flavors) since the adversary can gain the epoch key for the epoch

```
$\mathsf{BLMR.KG}(\lambda)$
1 :  $k_e \xleftarrow{\$} \mathsf{F.KG}(\lambda)$
2 :  return $k_e$

$\mathsf{BLMR.TG}(k_e, k_{e+1})$
3 :  $\Delta_{e+1} \leftarrow k_e \oplus k_{e+1}$
4 :  return $\Delta_{e+1}$

$\mathsf{BLMR.Enc}(k_e, M)$
5 :  $N \xleftarrow{\$} \chi$
6 :  $C_e^1 \leftarrow \mathsf{F}(k_e, N) \otimes M$
7 :  $C_e \leftarrow (C_e^1, N)$
8 :  return $C_e$

$\mathsf{BLMR.Dec}(k_e, C_e)$
9 :  parse $C_e = (C_e^1, N)$
10 : $M' \leftarrow C_e^1 \otimes \mathsf{F}(k_e, N)$
11 : return $M'$

$\mathsf{BLMR.Upd}(\Delta_{e+1}, C_e)$
12 : parse $C_e = (C_e^1, N)$
13 : $C_{e+1} \leftarrow (C_e^1 \otimes \mathsf{F}(\Delta_{e+1}, N), N)$
14 : return $C_{e+1}$
```

Figure 34: Updatable encryption scheme BLMR [BLMR13] for key-homomorphic PRF F.

```
$\mathbf{Exp}_{\mathsf{SKE},\mathcal{A}}^{\mathsf{IND\text{-}CPA\text{-}b}}(\lambda)$
1 :  $k \xleftarrow{\$} \mathsf{KG}$
2 :  $(M_0, M_1, st) \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{E}}(\lambda)$
3 :  if $|M_0| \neq |M_1|$
4 :      return $\perp$
5 :  $\tilde{C} \xleftarrow{\$} \mathsf{SKE.Enc}(k, M_b)$
6 :  $b' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{E}}(\tilde{C})$
7 :  return $b'$

$\mathcal{O}.\mathsf{E}(M)$
8 :  $C \leftarrow \mathsf{SKE.Enc}(k, M)$
9 :  return $C$
```

Figure 35: The experiments defining IND-CPA security for symmetric encryption schemes.

preceding the challenge epoch, allowing decryption of the challenge input ciphertexts and consequently a direct comparison of nonce values between these input ciphertexts and the challenge ciphertext.

LT18 detailed an extension of BLMR, denoted BLMR+, where the nonce is encrypted: this scheme is described in Fig. 36. LT18 showed that BLMR+ is IND-ENC-CPA and weakIND-UPD-CPA secure, however it is not able to achieve detIND-UE-CPA security, since the token contains the encryption key for the nonce value. More precisely, the adversary runs as follows:

- Choose some M_0, call $\mathcal{O}.\mathsf{Enc}(\mathrm{M}_0)$ and receive some C.
- Call $\mathcal{O}.\mathsf{Next}$, choose M_1 (that is distinct from M_0), do $\mathcal{O}.\mathsf{Chall}(\mathrm{C}, \mathrm{M}_1)$ and receive $\tilde{\mathrm{C}}$.
- Call $\mathcal{O}.\mathsf{Next}$, call $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$, call $\mathcal{O}.\mathsf{Corr}(\mathsf{token}, 2)$ and $\mathcal{O}.\mathsf{Corr}(\mathsf{key}, 0)$.
- Do $\mathsf{BLMR}+.\mathsf{Dec}_{\mathrm{k}_0}(\mathrm{C})$ to see its nonce, do $\mathsf{D}_{\mathrm{k}_2^2}(\tilde{\mathrm{C}})$ to see nonce of challenge ciphertext and compare.

This is a very similar attack to the one LT18 used to demonstrate that BLMR+ is not detIND-UPD-CPA secure.

Although BLMR+ is not detIND-UE-CPA secure, we can prove that it is weakIND-UE-CPA secure.

B.1 BLMR+ is weakIND-UE-CPA Secure.

Trivial wins for a weak model. An additional notion weakIND-UPD-CPA was used by LT18 for proving their BLMR+ scheme secure: if the adversary has access to any token or key which could leak the nonce of a challenge input ciphertext, the trivial win flag is triggered if the adversary gains access to any token which could reveal the nonce of a known (version of the) challenge ciphertext (i.e. if $\mathcal{I}^* \cap (\mathcal{K}^* \cup \mathcal{T}^*) \neq \emptyset$, then $\mathsf{twf} \leftarrow 1$ if $\exists \mathsf{e} \in \mathcal{C}^*$ such that e **or** $\mathsf{e}+1 \in \mathcal{T}^*$). This is necessary because the token in BLMR+ contains the symmetric keys that enable decryption and re-encryption of the nonce.

Proof technique of Proposition 6. The proof technique is very similar to the proof of weakIND-UPD-CPA security of BLMR+ in LT18 [LT18a]. We consider two situations of the additional requirements of weakIND-UE-CPA security and provide two proofs based on these situations. We only describe the first proof technique as both proofs use the same strategy. We construct hybrid games across each epoch, such that distinguishing the endpoints represents success in the weakIND-UE-CPA game. Suppose $\mathcal{A}_i$ is an adversary

```
BLMR+.KG(λ)
  $k^1 \xleftarrow{\$} \mathsf{F.KG}(\lambda)$
  $k^2 \xleftarrow{\$} \mathsf{SKE.KG}(\lambda)$
  $k_e \leftarrow (k^1, k^2)$
  $\mathbf{return}\ k_e$

BLMR+.TG(k_e, k_{e+1})
  $\mathbf{parse}\ k_e = (k_e^1, k_e^2)$
  $\mathbf{parse}\ k_{e+1} = (k_{e+1}^1, k_{e+1}^2)$
  $\Delta_{e+1} \leftarrow (k_e^1 \oplus k_{e+1}^1, (k_e^2, k_{e+1}^2))$
  $\mathbf{return}\ \Delta_{e+1}$

BLMR+.Enc(k_e, M)
  $\mathbf{parse}\ k_e = (k_e^1, k_e^2)$
  $N \xleftarrow{\$} \chi$
  $C_e^1 \leftarrow \mathsf{F}(k_e^1, N) \otimes M$
  $C_e^2 \leftarrow \mathsf{SKE.E}(k_e^2, N)$
  $C_e \leftarrow (C_e^1, C_e^2)$
  $\mathbf{return}\ C_e$

BLMR+.Dec(k_e, C_e)
  $\mathbf{parse}\ k_e = (k_e^1, k_e^2)$
  $\mathbf{parse}\ C_e = (C_e^1, C_e^2)$
  $N \leftarrow \mathsf{SKE.D}(k_e^2, C_e^2)$
  $M' \leftarrow C_e^1 \otimes \mathsf{F}(k_e^1, N)$
  $\mathbf{return}\ M'$

BLMR+.Upd(Δ_{e+1}, C_e)
  $\mathbf{parse}\ \Delta_{e+1} = (\Delta'_{e+1}, (k_e^2, k_{e+1}^2))$
  $\mathbf{parse}\ C_e = (C_e^1, C_e^2)$
  $N \leftarrow \mathsf{SKE.D}(k_e^2, C_e^2)$
  $C_{e+1}^1 \leftarrow C_e^1 \otimes \mathsf{F}(\Delta'_{e+1}, N)$
  $C_{e+1}^2 \leftarrow \mathsf{SKE.E}(k_{e+1}^2, N)$
  $C_{e+1} \leftarrow (C_{e+1}^1, C_{e+1}^2)$
  $\mathbf{return}\ C_{e+1}$
```

Figure 36: Updatable encryption scheme BLMR+ [BLMR13, LT18a] for key-homomorphic PRF F and symmetric key encryption scheme SKE.

trying to distinguish games in hybrid i. We consider a modified hybrid game in which the first element of ciphertexts is a uniformly random element. We can prove that the ability to notice this change is upper bounded by PRF advantage. Then, we conclude the proof by switching out the nonce inside the encryption in the second component: noticing this change is upper bounded by IND-CPA advantage of an adversary against SKE.

Proposition 6. Let BLMR+ be the updatable encryption scheme described in Fig. 36. For any weakIND-UE-CPA adversary $\mathcal{A}$ against UE that asks at most Q_E queries to $\mathcal{O}$.Enc before it makes its challenge, there exists an IND-CPA adversary $\mathcal{B}_{6b}^{\mathsf{IND\text{-}CPA}}$ against SKE and an PRF adversary $\mathcal{B}_{6a}^{\mathsf{PRF}}$

against F such that

$$\mathbf{Adv}^{\mathsf{weakIND\text{-}UE\text{-}CPA}}_{\mathsf{BLMR+},\,\mathcal{A}}(\lambda) \leq \\ (n+1)^3 \cdot \left(\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SKE},\,\mathcal{B}^{\mathsf{IND\text{-}CPA}}_{6b}}(\lambda) + 2\mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{F},\,\mathcal{B}^{\mathsf{PRF}}_{6a}}(\lambda) + \frac{2{\mathrm{Q_E}}^2}{|\mathcal{X}|} \right).$$

Proof. The additional requirement of weakIND-UE-CPA security states: If the adversary knows a secret key or a token in epoch $\mathsf{e}^* \in \mathcal{I}^*$, then for any $\mathsf{e} \in \mathcal{C}^*$, if the adversary corrupts Δ_{e} or $\Delta_{\mathsf{e}+1}$ then the adversary trivially loses, i.e. $\mathsf{twf} \leftarrow 1$. We consider two situations (whether or not $\mathcal{I}^* \cap (\mathcal{K}^* \cup \mathcal{T}^*) = \emptyset$) that might happen. The reduction can flip a coin at the beginning of the simulation to guess which situation the adversary will produce and set up the simulation appropriately.

Situation 1. Suppose the adversary knows a secret key or a token in epoch $\mathsf{e}^* \in \mathcal{I}^*$.

(Step 1.) We construct a sequence of hybrid games. Define game $\mathcal{G}_i$ as $\mathbf{Exp}^{\mathsf{weakIND\text{-}UE\text{-}CPA\text{-}b}}_{\mathsf{BLMR+},\,\mathcal{A}}$ except for:

- The challenge input $(\bar{\mathrm{M}}, \bar{\mathrm{C}})$, called in epoch j. If $j \leq i$ then return a ciphertext that is an update of $\bar{\mathrm{C}}$, if $j > i$ then return a ciphertext that is an encryption of $\bar{\mathrm{M}}$.
- After $\mathcal{A}$ outputs b', returns b' if $\mathsf{twf} \neq 1$.

Similarly the advantage $\mathbf{Adv}^{\mathsf{weakIND\text{-}UE\text{-}CPA}}_{\mathsf{BLMR+},\,\mathcal{A}}(\lambda)$ is upper bounded by $|\mathbf{Pr}[\mathcal{G}_{-1} = 1] - \mathbf{Pr}[\mathcal{G}_n = 1]|$. For any i, we prove that

$$|\mathbf{Pr}[\mathcal{G}_i = 1] - \mathbf{Pr}[\mathcal{G}_{i-1} = 1]| \leq \\ \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SKE},\,\mathcal{B}^{\mathsf{IND\text{-}CPA}}_{6b}}(\lambda) + 2\mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{F},\,\mathcal{B}^{\mathsf{PRF}}_{6a}}(\lambda) + \frac{2{\mathrm{Q_E}}^2}{|\mathcal{X}|}.$$

Suppose $\mathcal{A}_i$ is an adversary trying to distinguish $\mathcal{G}_i$ from $\mathcal{G}_{i-1}$. For all queries concerning epochs other than i the responses will be equal in either game, so we assume $\mathcal{A}_i$ asks for a challenge ciphertext in epoch i. That means if the adversary corrupts tokens in epoch i or epoch $i+1$, the trivial win condition is met and the adversary loses.

(Step 2.) We consider a modified game $\mathcal{G}_{\mathsf{PRF}}$ which is the same as $\mathcal{G}_i$ except for: the first element of ciphertexts given to the adversary in epoch i is a uniformly random element in $\mathcal{Y}$. More precisely, in epoch i, when $\mathcal{A}_i$ asks for $\mathcal{O}.\mathsf{Enc}$, $\mathcal{O}.\mathsf{Upd}$ or a challenge-equal ciphertext to game $\mathcal{G}^{\mathrm{b}}_{\mathsf{PRF}}$:

- An $\mathcal{O}.\mathsf{Enc}(\mathrm{M})$ query: randomly choose a nonce $\mathrm{N} \xleftarrow{\$} \mathcal{X} \backslash X$, set $X \leftarrow X \cup \{\mathrm{N}\}$, randomly choose $\mathrm{C}_i^1 \xleftarrow{\$} \mathcal{Y}$, compute $\mathrm{C}_i^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_i^2, \mathrm{N})$, set $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_i, i; \mathrm{N}, \mathrm{M})\}$. Output C_i.
- An $\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{i-1})$ query: proceed if $(\cdot, \mathrm{C}_{i-1}, i-1; \mathrm{N}, \mathrm{M}) \in \mathcal{L}$. If $\mathrm{N} \in X$, then abort the game; otherwise, set $X \leftarrow X \cup \{\mathrm{N}\}$, randomly choose $\mathrm{C}_i^1 \xleftarrow{\$} \mathcal{Y}$, compute $\mathrm{C}_i^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_i^2, \mathrm{N})$, set $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_i, i; \mathrm{N}, \mathrm{M})\}$. Output C_i.
- A challenge-equal ciphertext (with the underlying challenge input $(\bar{\mathrm{M}}_0, \bar{\mathrm{C}})$): proceed if $(\cdot, \bar{\mathrm{C}}, \tilde{\mathsf{e}} - 1; \mathrm{N}_1, \bar{\mathrm{M}}_1) \in \mathcal{L}$. If $\mathrm{N}_1 \in X$, then abort the game; otherwise, set $X \leftarrow X \cup \{\mathrm{N}_1\}$. Randomly choose a nonce $\mathrm{N}_0 \xleftarrow{\$} \mathcal{X} \setminus X$, set $X \leftarrow X \cup \{\mathrm{N}_0\}$, randomly choose $\mathrm{C}_i^1 \xleftarrow{\$} \mathcal{Y}$, compute $\tilde{\mathrm{C}}_i^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_i^2, \mathrm{N}_\mathrm{b})$, $(\tilde{\mathrm{C}}_i, i; \mathrm{N}_\mathrm{b}, \bar{\mathrm{M}}_\mathrm{b}) \in \tilde{\mathcal{L}}$. Output $\tilde{\mathrm{C}}_i$.

We wish to prove that

$$|\mathbf{Pr}[\mathcal{G}_i = 1] - \mathbf{Pr}[\mathcal{G}_{i-1} = 1]| \leq \mathbf{Adv}^{\mathcal{G}_{\mathsf{PRF}}}(\lambda) + 2\mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{F},\, \mathcal{B}^{\mathsf{PRF}}_{6a}}(\lambda) + \frac{2{\mathrm{Q}_\mathrm{E}}^2}{|\mathcal{X}|}.$$

If the following results are true, then we have the above result.

$$|\mathbf{Pr}[\mathcal{G}_i = 1] - \mathbf{Pr}[\mathcal{G}^1_{\mathsf{PRF}} = 1]| \leq \mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{F},\, \mathcal{B}^{\mathsf{PRF}}_{6a}}(\lambda) + \frac{{\mathrm{Q}_\mathrm{E}}^2}{|\mathcal{X}|}$$

and

$$|\mathbf{Pr}[\mathcal{G}_{i-1} = 1] - \mathbf{Pr}[\mathcal{G}^0_{\mathsf{PRF}} = 1]| \leq \mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{F},\, \mathcal{B}^{\mathsf{PRF}}_{6a}}(\lambda) + \frac{{\mathrm{Q}_\mathrm{E}}^2}{|\mathcal{X}|}.$$

We construct an PRF adversary $\mathcal{B}^{\mathsf{PRF}}_{6a}$, detailed in Fig. 37, against F to simulate the responses of queries made by $\mathcal{A}_i$.

The reduction does appropriate bookkeeping for the nonce, message, and ciphertexts in list $\mathcal{L}$. Specifically, in epoch i, $\mathcal{B}^{\mathsf{PRF}}_{6a}$ collects used nonces

Reduction $\mathcal{B}_{6a}^{\mathsf{PRF}}$ playing $\mathbf{Exp}_{\mathsf{F},\,\mathcal{B}_{6a}}^{\mathsf{PRF\text{-}b}}$ in hybrid i

```
1 :  do Setup
2 :  $\bar{\mathrm{M}}_0, \bar{\mathrm{C}} \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
3 :  $\mathsf{phase} \leftarrow 1$
4 :  Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}_0, \bar{\mathrm{C}})$, get $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
5 :  $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}})$
6 :  if $\mathcal{I}^* \cap (\mathcal{K}^* \cup \mathcal{T}^*) = \emptyset$
7 :      return ABORT
8 :  $\underline{\mathsf{twf} \leftarrow 1 \textbf{ if }}$
9 :      $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ or $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$ or
10 :     $(\exists \mathsf{e} \in \mathcal{C}^* : \mathsf{e} \in \mathcal{T}^*$ or $\mathsf{e} + 1 \in \mathcal{T}^*)$
11 : if ABORT occurred or $\mathsf{twf} = 1$
12 :     $\mathrm{b}' \xleftarrow{\$} \{0, 1\}$
13 :     return $\mathrm{b}'$
14 : if $\mathrm{b}' = \mathrm{b}$
15 :     return 0
16 : else
17 :     return 1
```

$\mathbf{Setup}(\lambda)$

```
18 : $\mathrm{b} \xleftarrow{\$} \{0, 1\}$
19 : $\Delta_0 \leftarrow \perp$;  $\mathsf{e} \leftarrow 0$;  $\mathsf{phase}, \mathsf{twf} \leftarrow 0$
20 : $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T}, X \leftarrow \emptyset$
21 : for $j \in \{0, ..., n\}$ do
22 :     $\mathrm{k}_j^{¶} \xleftarrow{\$} \mathsf{BLMR{+}.KG}(\lambda)$
23 :     $\Delta_{j+1}^{\diamond} \xleftarrow{\$} \mathsf{BLMR{+}.TG}(\mathrm{k}_j, \mathrm{k}_{j+1})$
```

Figure 37: Part 1. Reduction $\mathcal{B}_{6a}^{\mathsf{PRF}}$ for proof of Proposition 6. On line 22, ¶ indicates k_i^1 are skipped in the generation; on line 23, $\diamond$ indicates Δ_i and Δ_{i+1} are skipped in the generation.

$\mathcal{O}.\mathsf{Enc}(\mathrm{M})$

```
if $\mathsf{e} \neq i$
    $\mathrm{C}_\mathsf{e} \leftarrow \mathsf{BLMR}+.\mathsf{Enc}(\mathrm{k}_\mathsf{e}, \mathrm{M})$
if $\mathsf{e} = i$
    $\mathrm{N} \xleftarrow{\$} \mathcal{X} \setminus X; X \leftarrow X \cup \{\mathrm{N}\}$
    call $y \leftarrow \mathcal{O}.f(\mathrm{N})$
    $\mathrm{C}_i^1 \leftarrow y \otimes \mathrm{M}$   // embed
    $\mathrm{C}_i^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_i^2, \mathrm{N})$
    $\mathrm{C}_i \leftarrow (\mathrm{C}_i^1, \mathrm{C}_i^2)$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_\mathsf{e}, \mathsf{e}; \mathrm{N}, \mathrm{M})\}$
return $\mathrm{C}_\mathsf{e}$
```

$\mathcal{O}.\mathsf{Next}$

```
$\mathsf{e} \leftarrow \mathsf{e} + 1$
```

$\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}\text{-}1})$

```
if $(\cdot, \mathrm{C}_{\mathsf{e}\text{-}1}, \mathsf{e}\text{-}1; \mathrm{N}, \mathrm{M}) \notin \mathcal{L}$
  or $(\mathsf{e} = i$ and $\mathrm{N} \in X)$
    return $\perp$
if $\mathsf{e} \neq i, i+1$
    $\mathrm{C}_\mathsf{e} \leftarrow \mathsf{BLMR}+.\mathsf{Upd}(\Delta_\mathsf{e}, \mathrm{C}_{\mathsf{e}\text{-}1})$
if $\mathsf{e} = i+1$
    $\mathrm{C}_\mathsf{e} \leftarrow (\mathsf{F}(\mathrm{k}_\mathsf{e}^1, \mathrm{N}) \otimes \mathrm{M}, \mathsf{SKE.E}(\mathrm{k}_\mathsf{e}^2, \mathrm{N}))$
if $\mathsf{e} = i$
    $X \leftarrow X \cup \{\mathrm{N}\}$
    call $y \leftarrow \mathcal{O}.f(\mathrm{N})$
    $\mathrm{C}_i^1 \leftarrow y \otimes \mathrm{M}$   // embed
    $\mathrm{C}_i^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_i^2, \mathrm{N})$
    $\mathrm{C}_i \leftarrow (\mathrm{C}_i^1, \mathrm{C}_i^2)$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_\mathsf{e}, \mathsf{e}; \mathrm{N}, \mathrm{M})\}$
return $\mathrm{C}_\mathsf{e}$
```

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$

```
if $\hat{\mathsf{e}} > \mathsf{e}$ or $\mathsf{e} = i$
  or $(\mathsf{e} = i{+}1$ and $\mathsf{inp} = \mathsf{token})$
    return $\perp$
if $\mathsf{inp} = \mathsf{key}$
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
    return $\mathrm{k}_{\hat{\mathsf{e}}}$
if $\mathsf{inp} = \mathsf{token}$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
    return $\Delta_{\hat{\mathsf{e}}}$
```

Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}_0, \bar{\mathrm{C}})$

```
if $(\cdot, \bar{\mathrm{C}}, \tilde{\mathsf{e}} - 1; \mathrm{N}_1, \bar{\mathrm{M}}_1) \notin \mathcal{L}$ or $\mathrm{N}_1 \in X$
    return $\perp$
$\mathrm{N}_0 \xleftarrow{\$} \mathcal{X} \setminus (X \cup \{\mathrm{N}_1\})$
$X \leftarrow X \cup \{\mathrm{N}_0, \mathrm{N}_1\}$
call $y_\mathrm{b} \leftarrow \mathcal{O}.f(\mathrm{N}_\mathrm{b})$
$\tilde{\mathrm{C}}_i^1 \leftarrow y_\mathrm{b} \otimes \bar{\mathrm{M}}_\mathrm{b}$   // embed
$\tilde{\mathrm{C}}_i^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_i^2, \mathrm{N}_\mathrm{b})$
$\tilde{\mathrm{C}}_i \leftarrow (\tilde{\mathrm{C}}_i^1, \tilde{\mathrm{C}}_i^2)$
for $j \in \{0, \ldots, i-1\}$ do
    $\tilde{\mathrm{C}}_j^1 \leftarrow \mathsf{F}(\mathrm{k}_j^1, \mathrm{N}_1) \otimes \bar{\mathrm{M}}_1$   // left
    $\tilde{\mathrm{C}}_j^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_j^2, \mathrm{N}_1)$   // left
for $j \in \{i+1, \ldots, n\}$ do
    $\tilde{\mathrm{C}}_j^1 \leftarrow \mathsf{F}(\mathrm{k}_j^1, \mathrm{N}_0) \otimes \bar{\mathrm{M}}_0$   // right
    $\tilde{\mathrm{C}}_j^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_j^2, \mathrm{N}_0)$   // right
$\tilde{\mathcal{L}} \leftarrow \cup_{j=0}^{n} \{(\tilde{\mathrm{C}}_j, j)\}$
return $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
```

$\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$

```
$\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
find $(\tilde{\mathrm{C}}_\mathsf{e}, \mathsf{e}) \in \tilde{\mathcal{L}}$
return $\tilde{\mathrm{C}}_\mathsf{e}$
```

Figure 37: Part 2. Reduction $\mathcal{B}_{6a}^{\mathsf{PRF}}$ for proof of Proposition 6. Recall that in the PRF game in Definition 3, the oracle $\mathcal{O}.f$ responds to query input N with either $\mathsf{F}(\mathrm{k}, \mathrm{N})$ or a random value.

in list X (initiated as empty set). Initially, the reduction flips a coin $\mathrm{b} \xleftarrow{\$} \{0,1\}$, simulates the challenge response with $\bar{\mathrm{M}}_0$ if $\mathrm{b} = 0$; otherwise, simulates the challenge response with $\bar{\mathrm{C}}$. The reduction $\mathcal{B}_{6a}^{\mathsf{PRF}}$ generates all keys and tokens except for k_i^1. In epoch i, $\mathcal{B}_{6a}^{\mathsf{PRF}}$ calls its PRF challenger for help computing $\mathsf{F}(\mathrm{k}_i^1, \mathrm{N})$.

Eventually $\mathcal{B}_{6a}^{\mathsf{PRF}}$ receives b' from $\mathcal{A}_i$, and if $\mathrm{b}' = \mathrm{b}$, then $\mathcal{B}_{6a}^{\mathsf{PRF}}$ guesses that it is interacting with the 'real' PRF, i.e. outputs 0 to the PRF challenger, otherwise $\mathcal{B}_{6a}^{\mathsf{PRF}}$ outputs 1.

When $\mathcal{B}_{6a}^{\mathsf{PRF}}$ interacts with $\mathbf{Exp}_{F,\, \mathcal{B}_{6a}^{\mathsf{PRF}}}^{\mathsf{PRF}\text{-}0}$, the simulation of $\mathcal{G}_{i-1}$ (if $\mathrm{b} = 0$) or $\mathcal{G}_i$ (if $\mathrm{b} = 1$) is perfect except if a nonce collision during the game has caused an abort, this term is bounded by $\frac{\mathrm{Q_E}^2}{|\mathcal{X}|}$. When $\mathcal{B}_{6a}^{\mathsf{PRF}}$ interacts with $\mathbf{Exp}_{F,\, \mathcal{B}_{6a}^{\mathsf{PRF}}}^{\mathsf{PRF}\text{-}1}$, the simulation of $\mathcal{G}_{\mathsf{PRF}}^{\mathrm{b}}$ to $\mathcal{A}_i$ is perfect. We have the desired result.

(Step 3.) Suppose $\mathcal{A}_i$ is an adversary trying to distinguish game $\mathcal{G}_{\mathsf{PRF}}^0$ from game $\mathcal{G}_{\mathsf{PRF}}^1$. Then we construct a reduction $\mathcal{B}_{6b}^{\mathsf{IND\text{-}CPA}}$, detailed in Fig. 38, playing the $\mathsf{IND\text{-}CPA}$ game that runs $\mathcal{A}_i$. We claim that

$$\mathbf{Adv}_{\mathcal{A}_i}^{\mathcal{G}_{\mathsf{PRF}}}(\lambda) \leq \mathbf{Adv}_{\mathcal{B}_{6b}^{\mathsf{IND\text{-}CPA}}}^{\mathsf{IND\text{-}CPA}}(\lambda).$$

Reduction $\mathcal{B}_{6b}^{\mathsf{IND\text{-}CPA}}$ generates all keys and tokens except for k_i. In epoch i, $\mathcal{B}_{6b}^{\mathsf{IND\text{-}CPA}}$ uses the $\mathsf{IND\text{-}CPA}$ challenger for assistance in computing $\mathsf{SKE.E}(\mathrm{k}_i^2, \mathrm{N})$. In epoch i, the reduction forwards all nonces of $\mathcal{O}.\mathsf{Enc}$ and $\mathcal{O}.\mathsf{Upd}$ to the $\mathsf{IND\text{-}CPA}$ challenger, and sets the reply in the second part of ciphertext, i.e. C_i^2. For challenge input $(\bar{\mathrm{M}}, \bar{\mathrm{C}})$, suppose $\bar{\mathrm{C}}$ has the underlying nonce N_1, $\mathcal{B}_{6b}^{\mathsf{IND\text{-}CPA}}$ chooses nonce N_0 while encrypting M, sends $(\mathrm{N}_0, \mathrm{N}_1)$ to the $\mathsf{IND\text{-}CPA}$ challenger as challenge input, and sets the reply in the second part of the challenge ciphertext. The following shows how $\mathcal{B}_{6b}^{\mathsf{IND\text{-}CPA}}$ simulates the responses of queries made by $\mathcal{A}_i$:

Eventually, $\mathcal{B}_{6b}^{\mathsf{IND\text{-}CPA}}$ sends the guess bit of $\mathcal{A}$ to the $\mathsf{IND\text{-}CPA}$ challenger. We have the required result.

Situation 2. Suppose the adversary knows none of the secret keys and tokens in epoch $\mathsf{e}^* \in \mathcal{I}^*$.

Since the adversary never knows the nonce in the challenge $\bar{\mathrm{C}}$, we do not need to worry if the adversary knows a token in the challenge epoch or the next epoch will make the adversary trivially win the game.

```
Reduction $\mathcal{B}_{6b}^{\mathsf{IND\text{-}CPA}}$ playing $\mathbf{Exp}_{\mathsf{SKE},\,\mathcal{B}_{6b}}^{\mathsf{IND\text{-}CPA}}$ in hybrid i
1 :  do Setup
2 :  $\bar{\mathrm{M}}_0, \bar{\mathrm{C}} \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
3 :  $\mathsf{phase} \leftarrow 1$
4 :  Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}_0, \bar{\mathrm{C}})$, get $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
5 :  $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}})$
6 :  if if $\mathcal{I}^* \cap (\mathcal{K}^* \cup \mathcal{T}^*) \neq \emptyset$
7 :      return ABORT
8 :  $\underline{\mathsf{twf} \leftarrow 1\textbf{if}}$
9 :      $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ or $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$ or
10 :     $(\exists \mathsf{e} \in \mathcal{C}^* : \mathsf{e} \in \mathcal{T}^*$ or $\mathsf{e} + 1 \in \mathcal{T}^*)$
11 : if ABORT occurred or $\mathsf{twf} = 1$
12 :     $\mathrm{b}' \xleftarrow{\$} \{0, 1\}$
13 :     return $\mathrm{b}'$
14 : if $\mathrm{b}' = \mathrm{b}$
15 :     return 0
16 : else
17 :     return 1

Setup($\lambda$)
18 : $\mathrm{b} \xleftarrow{\$} \{0, 1\}$
19 : $\Delta_0 \leftarrow \perp$; $\mathsf{e} \leftarrow 0$; $\mathsf{phase}, \mathsf{twf} \leftarrow 0$
20 : $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T}, X \leftarrow \emptyset$
21 : for $j \in \{0, ..., n\}$ do
22 :     $\underline{\mathrm{k}_j^{\circledast} \xleftarrow{\$} \mathsf{BLMR{+}.KG}(\lambda)}$
23 :     $\Delta_{j+1}^{\diamond} \xleftarrow{\$} \mathsf{BLMR{+}.TG}(\mathrm{k}_j, \mathrm{k}_{j+1})$
```

Figure 38: Part 1. Reduction $\mathcal{B}_{6b}^{\mathsf{IND\text{-}CPA}}$ for proof of Proposition 6; the simulation is almost the same as $\mathcal{B}_{6a}^{\mathsf{PRF}}$ except for the underlined simulations. On line 22, $\circledast$ indicates k_i is skipped in the generation; on line 23, $\diamond$ indicates Δ_i and Δ_{i+1} are skipped in the generation.

```
$\mathcal{O}.\mathsf{Enc}(\mathrm{M})$
  if $\mathsf{e} \neq i$
    $\mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{BLMR+.Enc}(\mathrm{k}_{\mathsf{e}}, \mathrm{M})$
  if $\mathsf{e} = i$
    $\mathrm{N} \xleftarrow{\$} \mathcal{X} \setminus X; X \leftarrow X \cup \{\mathrm{N}\}$
    $\underline{\mathrm{C}_i^1 \xleftarrow{\$} \mathcal{Y}}$
    $\underline{\textbf{call } y \leftarrow \mathcal{O}.\mathsf{E}(\mathrm{N})}$
    $\underline{\mathrm{C}_i^2 \leftarrow y}$   / embed
    $\mathrm{C}_i \leftarrow (\mathrm{C}_i^1, \mathrm{C}_i^2)$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_{\mathsf{e}}, \mathsf{e}; \mathrm{N}, \mathrm{M})\}$
  return $\mathrm{C}_{\mathsf{e}}$

$\mathcal{O}.\mathsf{Next}$
  $\mathsf{e} \leftarrow \mathsf{e} + 1$

$\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}\text{-}1})$
  if $(\cdot, \mathrm{C}_{\mathsf{e}\text{-}1}, \mathsf{e}\text{-}1; \mathrm{N}, \mathrm{M}) \notin \mathcal{L}$
   or $(\mathsf{e} = i$ and $\mathrm{N} \in X)$
    return $\perp$
  if $\mathsf{e} \neq i, i+1$
    $\mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{BLMR+.Upd}(\Delta_{\mathsf{e}}, \mathrm{C}_{\mathsf{e}\text{-}1})$
  if $\mathsf{e} = i+1$
    $\mathrm{C}_{\mathsf{e}}^1 \leftarrow \mathsf{F}(\mathrm{k}_{\mathsf{e}}^1, \mathrm{N}) \otimes \mathrm{M}$
    $\mathrm{C}_{\mathsf{e}}^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_{\mathsf{e}}^2, \mathrm{N})$
  if $\mathsf{e} = i$
    $X \leftarrow X \cup \{\mathrm{N}\}$; $\underline{\mathrm{C}_i^1 \xleftarrow{\$} \mathcal{Y}}$
    $\underline{\textbf{call } y \leftarrow \mathcal{O}.\mathsf{E}(\mathrm{N})}$
    $\underline{\mathrm{C}_i^2 \leftarrow y}$   / embed
    $\mathrm{C}_i \leftarrow (\mathrm{C}_i^1, \mathrm{C}_i^2)$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_{\mathsf{e}}, \mathsf{e}; \mathrm{N}, \mathrm{M})\}$
  return $\mathrm{C}_{\mathsf{e}}$
```

```
$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$
  if $\hat{\mathsf{e}} > \mathsf{e}$ or $\mathsf{e} = i$
   or $(\mathsf{e} = i{+}1$ and $\mathsf{inp} = \mathsf{token})$
    return $\perp$
  if $\mathsf{inp} = \mathsf{key}$
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
    return $\mathrm{k}_{\hat{\mathsf{e}}}$
  if $\mathsf{inp} = \mathsf{token}$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
    return $\Delta_{\hat{\mathsf{e}}}$

Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}_0, \bar{\mathrm{C}})$
  if $(\cdot, \bar{\mathrm{C}}, \tilde{\mathsf{e}} - 1; \mathrm{N}_1, \bar{\mathrm{M}}_1) \notin \mathcal{L}$ or $\mathrm{N}_1 \in X$
    return $\perp$
  $\mathrm{N}_0 \xleftarrow{\$} \mathcal{X} \setminus (X \cup \{\mathrm{N}_1\})$; $X \leftarrow X \cup \{\mathrm{N}_0, \mathrm{N}_1\}$
  $\underline{\tilde{\mathrm{C}}_i^1 \xleftarrow{\$} \mathcal{Y}}$
  $\underline{\textbf{call } \mathsf{CHALL} \textbf{ with } (\mathrm{N}_0, \mathrm{N}_1), \textbf{get } \tilde{\mathrm{C}}_i^2}$ / embed
  $\tilde{\mathrm{C}}_i \leftarrow (\tilde{\mathrm{C}}_i^1, \tilde{\mathrm{C}}_i^2)$
  for $j \in \{0, \ldots, i-1\}$ do
    $\tilde{\mathrm{C}}_j^1 \leftarrow \mathsf{F}(\mathrm{k}_j^1, \mathrm{N}_1) \otimes \bar{\mathrm{M}}_1$   / left
    $\tilde{\mathrm{C}}_j^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_j^2, \mathrm{N}_1)$   / left
  for $j \in \{i+1, \ldots, n\}$ do
    $\tilde{\mathrm{C}}_j^1 \leftarrow \mathsf{F}(\mathrm{k}_j^1, \mathrm{N}_0) \otimes \bar{\mathrm{M}}_0$   / right
    $\tilde{\mathrm{C}}_j^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_j^2, \mathrm{N}_0)$   / right
  $\tilde{\mathcal{L}} \leftarrow \cup_{j=0}^{n} \{(\tilde{\mathrm{C}}_j, j)\}$
  return $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
  find$(\tilde{\mathrm{C}}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{L}}$
  return $\tilde{\mathrm{C}}_{\mathsf{e}}$
```

Figure 38: Part 2. Reduction $\mathcal{B}_{6b}^{\mathsf{IND\text{-}CPA}}$ for proof of Proposition 6; the simulation is almost the same as $\mathcal{B}_{6a}^{\mathsf{PRF}}$ except for the underlined simulations. Recall that in the IND-CPA game in Definition 14, the encryption oracle $\mathcal{O}.\mathsf{E}$ replies to input N with $\mathsf{SKE.Enc}(\mathrm{k}, \mathrm{N})$.

We use the firewall technique to construct hybrid games: in hybrid i, we embed within the i-th insulated region. This means that to the left of the i-th insulated region the game responds with an update of the challenge input ciphertext and to the right of the i-th insulated region it gives an encryption of the challenge input message. Similarly the advantage $\mathbf{Adv}^{\mathsf{weakIND\text{-}UE\text{-}CPA}}_{\mathsf{BLMR+},\,\mathcal{A}}(\lambda)$ is upper bounded by $(n+1)^2 \cdot |\mathbf{Pr}[\mathcal{G}^1_l = 1] - \mathbf{Pr}[\mathcal{G}^0_1 = 1]|$. For any $1 \leq i \leq l$, we prove that

$$|\mathbf{Pr}[\mathcal{G}^1_i = 1] - \mathbf{Pr}[\mathcal{G}^0_i = 1]| \leq \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SKE},\,\mathcal{B}^{\mathsf{IND\text{-}CPA}}_{6b}}(\lambda) + 2\mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{F},\,\mathcal{B}^{\mathsf{PRF}}_{6a}}(\lambda) + \frac{2\mathrm{Q_E}^2}{|\mathcal{X}|}.$$

Suppose $\mathcal{A}_i$ is an adversary trying to distinguish game $\mathcal{G}^0_i$ from game $\mathcal{G}^1_i$ in hybrid i.

As the above step 2 proof, we consider a modified hybrid game in which the first element of ciphertexts in epoch fwl_i is a uniformly random element in $\mathcal{Y}$. The difference here is that if $\mathcal{A}_i$ asks for encryption queries or challenge queries in an epoch within the i-th insulated region, the reduction will simulate these queries in epoch fwl_i and then output the updated version (updated from epoch fwl_i to the queried epoch). Since the update algorithm of BLMR+ is deterministic, this simulation is valid.

Similarly we can prove the modified hybrid game is indistinguishable from the original hybrid game, and that the distinguishing advantage is upper bounded by the PRF advantage. Finally, similarly to the above step 3 proof (the difference is the same as the difference mentioned in the former paragraph), the advantage is upper bounded by IND-CPA advantage of SKE. We have the following result:

$$\mathbf{Adv}^{\mathsf{weakIND\text{-}UE\text{-}CPA}}_{\mathsf{BLMR+},\,\mathcal{A}}(\lambda) \leq (n+1)^3\left(\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SKE},\,\mathcal{B}^{\mathsf{IND\text{-}CPA}}_{6b}}(\lambda) + 2\mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{F},\,\mathcal{B}^{\mathsf{IND\text{-}CPA}}_{6b}}(\lambda) + \frac{2\mathrm{Q_E}^2}{|\mathcal{X}|}\right). \qquad \square$$

C The RISE Scheme of Lehmann and Tackmann

In this section we discuss the Elgamal-based updatable encryption scheme RISE, developed by Lehmann and Tackmann [LT18a] and given in Fig. 39.

LT18 showed[7] that RISE is randIND-ENC and randIND-UPD-CPA, under DDH. KLR19 observed that for RISE, knowledge of an update token allows the storage host to create arbitrary ciphertexts for messages of its choice: this is a very undesirable feature for an UE scheme, and is not possible for SHINE. This emphasizes the importance of ciphertext integrity for UE schemes.

```
RISE.KG(λ)
1 :  $x \xleftarrow{\$} \mathbb{Z}_q^*$
2 :  $\mathrm{k_e} \leftarrow (x, g^x)$
3 :  return $\mathrm{k_e}$

RISE.TG(k_e, k_e+1)
4 :  parse $\mathrm{k_e} = (x, y)$
5 :  parse $\mathrm{k_{e+1}} = (x', y')$
6 :  $\Delta_{\mathrm{e+1}} \leftarrow (\frac{x'}{x}, y')$
7 :  return $\Delta_{\mathrm{e+1}}$

RISE.Enc(k_e, M)
8 :  parse $\mathrm{k_e} = (x, y)$
9 :  $r \xleftarrow{\$} \mathbb{Z}_q$
10 : $\mathrm{C_e} \leftarrow (y^r, g^r \cdot \mathrm{M})$
11 : return $\mathrm{C_e}$

RISE.Dec(k_e, C_e)
12 : parse $\mathrm{k_e} = (x, y)$
13 : parse $\mathrm{C_e} = (\mathrm{C_1}, \mathrm{C_2})$
14 : $\mathrm{M}' \leftarrow \mathrm{C_2} \cdot \mathrm{C_1}^{-1/x}$
15 : return $\mathrm{M}'$

RISE.Upd(Δ_e+1, C_e)
16 : parse $\Delta_{\mathrm{e+1}} = (\Delta, y')$
17 : parse $\mathrm{C_e} = (\mathrm{C_1}, \mathrm{C_2})$
18 : $r' \xleftarrow{\$} \mathbb{Z}_q$
19 : $\mathrm{C}_1' \leftarrow \mathrm{C}_1^{\Delta} \cdot y'^{r'}$
20 : $\mathrm{C}_2' \leftarrow \mathrm{C_2} \cdot g^{r'}$
21 : $\mathrm{C_{e+1}} \leftarrow (\mathrm{C}_1', \mathrm{C}_2')$
22 : return $\mathrm{C_{e+1}}$
```

Figure 39: Updatable encryption scheme RISE [LT18a] for λ-bit prime q.

C.1 RISE is randIND-UE-CPA Secure

We show that RISE is randIND-UE-CPA under DDH. First, we adapt (an extended version of) the Oracle-DDH experiment to the epoch-based corruption model found in updatable encryption, in a way that ensures that it still reduces to DDH. In this way, we lift a lot of the bookkeeping and com-

[7]On 19th December 2019, the authors updated the full version of their paper [LT18b] to include a new proof of randIND-ENC security, fixing a flaw in the prior proof methodology.

plexity. Then, the reduction from $\mathsf{randIND\text{-}UE\text{-}CPA}$ to this oracle-based game is straightforward. We believe that this two-step proof strategy may be useful for proving security of other UE schemes, under any of the security notions discussed so far.

Oracle-Decision-Diffie Hellman for RISE. We now give an experiment $\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}$, where each exponent represents an epoch key, and corruption of keys and tokens is represented: the game allows the adversary to acquire the difference of two exponents in the form $\mathsf{t}_i = \frac{\mathsf{e}_i}{\mathsf{e}_{i-1}}$. In each 'epoch' the adversary can possibly ask for a challenge via $\mathcal{O}.\mathsf{Chall}$, which returns either a 'real' DDH tuple, with the epoch key defined as one of the exponents, or a random tuple. The game is given in Fig. 40. Just as in the games for UE, the challenger keeps track of the epochs in which the adversary has access to 'updates' of the 'challenge' (via CL^*), and access to the keys (exponents, via EL^*). If these overlap then the adversary can trivially extract from the challenge value whether it is 'real' or 'random' and win, so this is of course ruled out. The syntax also follows the UE games in the sense that once the adversary asks for a challenge, it can only ask for 'later' challenges (i.e. with a higher index) from this oracle – it can of course move this challenge 'backwards' into earlier epochs/indices by applying 'token' t.

Definition 15. Fix a cyclic group $\mathbb{G}$ of prime order q with generator g. The advantage of an algorithm $\mathcal{A}$ solving the *Oracle-Decision Diffie-Hellman for* RISE ($\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}$) problem for $\mathbb{G}$ and g is

$$\mathbf{Adv}_{\mathbb{G},\,\mathcal{A}}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}} = \left|\mathbf{Pr}[\mathbf{Exp}_{\mathbb{G},\,\mathcal{A}}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-}1}(\lambda) = 1] - \mathbf{Pr}[\mathbf{Exp}_{\mathbb{G},\,\mathcal{A}}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-}0}(\lambda) = 1]\right|,$$

where the experiment $\mathbf{Exp}_{\mathbb{G},\,\mathcal{A}}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-}\mathrm{b}}$ is given in Fig. 40.

Proposition 7. Let $\mathbb{G}$ be a group of order q (a λ-bit prime) with generator g, and let RISE be the updatable encryption scheme described in Fig. 39. For any $\mathsf{randIND\text{-}UE\text{-}CPA}$ adversary $\mathcal{A}$ against RISE, there exists an adversary $\mathcal{B}_7$ against DDH such that

$$\mathbf{Adv}_{\mathsf{RISE},\,\mathcal{A}}^{\mathsf{randIND\text{-}UE\text{-}CPA}}(\lambda) = 2(n+1)^3 \cdot \mathbf{Adv}_{\mathbb{G},\,\mathcal{B}_7}^{\mathsf{DDH}}(\lambda).$$

```
$\mathbf{Exp}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-b}}_{\mathbb{G},\,\mathcal{A}}(\lambda)$
1 :  $\mathsf{phase}, i^* \leftarrow 0$
2 :  $\mathsf{EL}^*, \mathsf{CL}^* \leftarrow \emptyset$
3 :  $\mathbf{if}\ \mathrm{b} = 1$
4 :      $w_1, \ldots, w_n \xleftarrow{\$} \mathbb{Z}_q^*$
5 :  $\mathbf{else}$
6 :      $w_1, \ldots, w_n \leftarrow 0$
7 :  $\mathsf{e}_1, \ldots, \mathsf{e}_n \xleftarrow{\$} \mathbb{Z}_q^*$
8 :  $x_1, \ldots, x_n \xleftarrow{\$} \mathbb{Z}_q^*$
9 :  $\mathbf{for}\ i \in \{0, \ldots n\}\ \mathbf{do}$
10 :     $\mathsf{s}_i \leftarrow g^{\mathsf{e}_i}$
11 :     $X_i \leftarrow g^{x_i}$
12 : $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Open}, \mathcal{O}.\mathsf{Difr}, \mathcal{O}.\mathsf{Chall}}(g, \{\mathsf{s}_1, \ldots, \mathsf{s}_n\})$
13 : $\mathbf{if}\ \mathsf{EL}^* \cap \mathsf{CL}^* \neq \emptyset$
14 :     $\mathrm{b}' \xleftarrow{\$} \{0, 1\}$
15 : $\mathbf{return}\ \mathrm{b}'$

$\mathcal{O}.\mathsf{Open}(i)$
16 : $\mathbf{update}\ \mathsf{EL}^*$
17 : $\mathbf{return}\ \mathsf{e}_i$

$\mathcal{O}.\mathsf{Difr}(i)$
18 : $\mathsf{t}_i \leftarrow \frac{\mathsf{e}_i}{\mathsf{e}_{i-1}}$
19 : $\mathbf{update}\ \mathsf{EL}^*, \mathsf{CL}^*$

$\mathcal{O}.\mathsf{Chall}(i)$
20 : $\mathbf{if}\ \mathsf{phase} = 1\ \mathbf{and}\ i < i^*$
21 :     $\mathbf{return}\ \perp$
22 : $\mathsf{CL}^* \leftarrow \mathsf{CL}^* \cup \{i\}$
23 : $Z_i \leftarrow \mathsf{s}_i^{x_i} \cdot g^{w_i}$
24 : $\mathbf{if}\ \mathsf{phase} = 0$
25 :     $i^* \leftarrow i$
26 :     $\mathsf{phase} \leftarrow 1$
27 : $\mathbf{return}\ (X_i, Z_i)$
```

Figure 40: $\mathcal{O}$-$\mathsf{DDH}^{\mathsf{RISE}}$ experiment for $\mathbb{G}$ of order q (λ-bit prime) and generator g.

This theorem is proven by Lemmas 7.1 and 7.2.

Lemma 7.1. Let $\mathbb{G}$ be a group of order q (a λ-bit prime) with generator g. For any adversary $\mathcal{A}$ against $\mathcal{O}$-$\mathsf{DDH}^{\mathsf{RISE}}$, there exists an adversary $\mathcal{B}_{7.1}$ against DDH such that

$$\mathbf{Adv}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}}_{\mathbb{G},\,\mathcal{A}}(\lambda) = (n+1)^3 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_{7.1}}(\lambda),$$

where $n+1$ is the number of exponents in the $\mathcal{O}$-$\mathsf{DDH}^{\mathsf{RISE}}$ game.

Proof. We use a sequence of game hops and a hybrid argument. Define game $\mathcal{G}_i^{\mathrm{b}}$ as $\mathbf{Exp}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-b}}_{\mathbb{G},\,\mathcal{A}}$ except for $\mathcal{O}.\mathsf{Chall}$: if called in index j, if

$j < i$ then return a 'real' sample (with $w_j = 0$), and if $j > i$ return a 'random sample' ($w_j \xleftarrow{\$} \mathbb{Z}_q^*$). Thus $\mathcal{G}_0^1$ is $\mathbf{Exp}_{\mathbb{G}}^{\mathcal{O}\text{-DDH}^{\mathsf{RISE}}\text{-}1}$, i.e. all challenges result in 'random' DDH tuples, and $\mathcal{G}_n^0$ is $\mathbf{Exp}_{\mathbb{G}}^{\mathcal{O}\text{-DDH}^{\mathsf{RISE}}\text{-}0}$, i.e. all challenges result in 'real' DDH tuples. Thus distinguishing $\mathcal{G}_0^1$ from $\mathcal{G}_n^0$ is the task of distinguishing $\mathbf{Exp}_{\mathbb{G},\,\mathcal{A}}^{\mathcal{O}\text{-DDH}^{\mathsf{RISE}}\text{-}1}$ from $\mathbf{Exp}_{\mathbb{G},\,\mathcal{A}}^{\mathcal{O}\text{-DDH}^{\mathsf{RISE}}\text{-}0}$ for adversary $\mathcal{A}$.

Notice that all queries in $\mathcal{G}_{i-1}^0$ and $\mathcal{G}_i^1$ have the equal responses (For $j \leq i-1$, returns a real sample. For $j > i-1$, returns a random sample). We have $\mathbf{Adv}_{\mathbb{G},\,\mathcal{A}}^{\mathcal{O}\text{-DDH}^{\mathsf{RISE}}}(\lambda) = \sum_{i=0}^{n} |\mathbf{Pr}[\mathcal{G}_i^1 = 1] - \mathbf{Pr}[\mathcal{G}_i^0 = 1]|$. Then we prove that $|\mathbf{Pr}[\mathcal{G}_i^1 = 1] - \mathbf{Pr}[\mathcal{G}_i^0 = 1]| \leq (n+1)^2 \cdot \mathbf{Adv}_{\mathbb{G},}^{\mathsf{DDH}}(\lambda)$ for any i.

Let $\mathcal{A}_i$ be an adversary trying to distinguish $\mathcal{G}_i^1$ from $\mathcal{G}_i^0$. For all queries concerning epochs other than i the responses will be equal in either game, so we assume that $\mathcal{A}_i$ asks for a challenge ciphertext in epoch i and this is where we will embed in our reduction. We construct a reduction $\mathcal{B}_{7.1}$, detailed in Fig. 41, that is playing the standard DDH game (Fig. 2) and runs $\mathcal{A}_i$. This reduction guesses the locations of the firewalls around the challenge query: if $\mathcal{A}_i$ adds any of the epochs within this insulated region to its EL^* list then the reduction fails. fwl and fwr could take any value in $\{0, \ldots, n\}$, so this loss is upper bounded by $(n+1)^2$.

For all challenge queries smaller than i the reduction needs to faithfully respond with a 'real' tuple, that is the exponent of Z_j is the product of the exponent used in s_j and the exponent in X_j. These queries must still be consistent with each other, which is why even though the reduction is free to choose x_j it must compute the correct value of s_j. For challenge queries larger than i the reduction produces a random value.

Note that $\mathcal{A}_i$ will have its own CL^* and EL^* lists and $\mathcal{B}_{7.1}$ will simulate these, however we omit this calculation for readability.

If $\mathcal{B}_{7.1}$ is playing $\mathbf{Exp}_{\mathbb{G},\,\mathcal{B}_{7.1}}^{\mathsf{DDH}\text{-}1}$ then it receives a random tuple from its challenger and thus provides a random response to $\mathcal{O}.\mathsf{Chall}(i)$, creating a perfect simulation of $\mathcal{G}_i^1$ to $\mathcal{A}_i$. If $\mathcal{B}_{7.1}$ is playing $\mathbf{Exp}_{\mathbb{G},\,\mathcal{B}_{7.1}}^{\mathsf{DDH}\text{-}0}$ then its tuple is real, providing a perfect simulation of $\mathcal{G}_i^0$. We have the required result. $\square$

Lemma 7.2. Let $\mathbb{G}$ be a group of order q (a λ-bit prime) with generator g, and let RISE be the updatable encryption scheme described in Fig. 39. For any $\mathsf{randIND\text{-}UE\text{-}CPA}$ adversary $\mathcal{A}$ against RISE, there exists an adversary

$\mathcal{B}_{7.2}$ against $\mathcal{O}\text{-DDH}^{\mathsf{RISE}}$ such that

$$\mathbf{Adv}^{\mathsf{randIND\text{-}UE\text{-}CPA}}_{\mathsf{RISE},\ \mathcal{A}}(\lambda) = 2 \cdot \mathbf{Adv}^{\mathcal{O}\text{-DDH}^{\mathsf{RISE}}}_{\mathbb{G},\ \mathcal{B}_{7.2}}(\lambda).$$

Proof. The reduction $\mathcal{B}_{7.2}$ is given in Fig. 42. The reduction $\mathcal{B}_{7.2}$ is playing $\mathcal{O}\text{-DDH}^{\mathsf{RISE}}$ game and runs $\mathcal{A}$. $\mathcal{B}_{7.2}$ flips a coin b, and simulates the experiment $\mathbf{Exp}^{\mathsf{randIND\text{-}UE\text{-}CPA\text{-}b}}_{\mathsf{RISE},\ \mathcal{A}}$ by interacting with its own $\mathcal{O}\text{-DDH}^{\mathsf{RISE}}$ challenger.

To simulate updated non-challenge ciphertexts (i.e. respond to $\mathcal{O}.\mathsf{Upd}$ queries), $\mathcal{B}_{7.2}$ must track the underlying messages for these encryptions so that it can generate valid 'fresh' encryptions using the 'public key' values $\{\mathsf{s}_1, \ldots, \mathsf{s}_n\}$ received from its $\mathcal{O}\text{-DDH}^{\mathsf{RISE}}$ challenger. Since updated non-challenge ciphertexts are of the form $\mathrm{C}_{\mathsf{e}} = (\mathsf{s}_{\mathsf{e}}^r, g^r \cdot \mathrm{M})$, where r is a fresh random value, the s_i values allow $\mathcal{B}_{7.2}$ to successfully simulate updated non-challenge ciphertexts.

To simulate challenge ciphertext (i.e. respond to challenge query or $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$), $\mathcal{B}_{7.2}$ must embed using its own challenge. Recall that in the $\mathcal{O}\text{-DDH}^{\mathsf{RISE}}$ experiment in Fig. 40, a call to $\mathcal{O}.\mathsf{Chall}(i)$ will result in a response $Z_i = g^{\mathrm{k}_{\tilde{\mathsf{e}}} x_i}$ or $g^{\mathrm{k}_{\tilde{\mathsf{e}}} x_i + w_i}$, and $X_i = g^{x_i}$. When $\mathcal{B}_{7.2}$ receives a challenge query $(\bar{\mathrm{M}}_0, \mathrm{C})$ (where $\mathrm{C} = (g^{r_2 \mathrm{k}_{\tilde{\mathsf{e}}-1}}, g^{r_2}\bar{\mathrm{M}}_1)$ for some $\bar{\mathrm{M}}_1$) from $\mathcal{A}$, $\mathcal{B}_{7.2}$ tries to simulate encryption $\mathsf{RISE.Enc}(\mathrm{k}_{\tilde{\mathsf{e}}}, \bar{\mathrm{M}}_0) = (g^{r_1 \mathrm{k}_{\tilde{\mathsf{e}}}}, g^{r_1}\bar{\mathrm{M}}_0)$ or updating $\mathsf{RISE.Upd}(\Delta_{\tilde{\mathsf{e}}-1}, \mathrm{C}) = (\mathrm{C}_1^{\Delta_{\tilde{\mathsf{e}}-1}} g^{\mathrm{k}_{\tilde{\mathsf{e}}} r_3}, \mathrm{C}_2 g^{r_3}) = (g^{\mathrm{k}_{\tilde{\mathsf{e}}}(r_2+r_3)}, g^{r_2+r_3}\bar{\mathrm{M}}_1)$, where r_1, r_3 are fresh random values. $\mathcal{B}_{7.2}$ will embed $Z_{\tilde{\mathsf{e}}}$ to the first part of the challenge ciphertext and embed $X_{\tilde{\mathsf{e}}}$ to the second part of the challenge ciphertext, i.e. $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}} = (Z_{\tilde{\mathsf{e}}}, X_{\tilde{\mathsf{e}}} \cdot \bar{\mathrm{M}}_{\mathrm{b}})$. Similarly, $\mathcal{B}_{7.2}$ can simulate the response of $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$ using the same approach.

Eventually, $\mathcal{B}_{7.2}$ receives the output bit b' from $\mathcal{A}_i$. If $\mathrm{b}' = \mathrm{b}$, then $\mathcal{B}_{7.2}$ returns 0 to its $\mathcal{O}\text{-DDH}^{\mathsf{RISE}}$ challenger, otherwise, $\mathcal{B}_{7.2}$ returns 1.

If $\mathcal{B}_{7.2}$ interacting with $\mathbf{Exp}^{\mathcal{O}\text{-DDH}^{\mathsf{RISE}}\text{-}0}_{\mathbb{G},\ \mathcal{B}_{7.2}}$, then it perfectly simulates the experiment $\mathbf{Exp}^{\mathsf{randIND\text{-}UE\text{-}CPA\text{-}b}}_{\mathsf{RISE},\ \mathcal{A}}$ to $\mathcal{A}$. If $\mathcal{B}_{7.2}$ interacting with the experiment $\mathbf{Exp}^{\mathcal{O}\text{-DDH}^{\mathsf{RISE}}\text{-}1}_{\mathbb{G},\ \mathcal{B}_{7.2}}$, then it wins with probability 1/2. After some computation similar to that in the proof of Theorem 2.3 we have the desired result. □

```
$\mathcal{B}_{7.1}$ playing $\mathbf{Exp}^{\mathsf{DDH\text{-}b}}_{\mathbb{G},\,\mathcal{B}_{7.1}}(\lambda)$ in hybrid $i$
1 :  receive $(g, X, Y, Z)$
2 :  $\mathsf{fwl}, \mathsf{fwr} \xleftarrow{\$} \{0, \ldots, n\}$
3 :  $w_{i+1}, \ldots, w_n \xleftarrow{\$} \mathbb{Z}_q^*$
4 :  $w_0, \ldots, w_{i-1} \leftarrow 0$
5 :  $\mathsf{s}_i \leftarrow Y$
6 :  for $j \in \{0, \ldots, i\text{-}1\} \cup \{i\text{+}1, \ldots, n\}$ do
7 :      $x_j \xleftarrow{\$} \mathbb{Z}_q^*; X_j \leftarrow g^{x_j}$
8 :  for $j \in \{0, \ldots, \mathsf{fwl}\text{-}1\} \cup \{\mathsf{fwr}\text{+}1, \ldots, n\}$ do
9 :      $\mathsf{e}_j \xleftarrow{\$} \mathbb{Z}_q^*; \mathsf{t}_j \leftarrow \dfrac{\mathsf{e}_j}{\mathsf{e}_{j\text{-}1}}^{\dagger}; \mathsf{s}_j \leftarrow g^{\mathsf{e}_j}$
10 : for $j \in \{\mathsf{fwl}\text{+}1, \ldots, \mathsf{fwr}\}$ do
11 :     $\mathsf{t}_j \xleftarrow{\$} \mathbb{Z}_q^*$
12 : for $j \in \{\mathsf{fwl}, \ldots, i\text{-}1\}$ do
13 :     $\mathsf{s}_j \leftarrow Y^{-\prod_{k=j+1}^{i} \mathsf{t}_k}$
14 : for $j \in \{i\text{+}1, \ldots, \mathsf{fwr}\}$ do
15 :     $\mathsf{s}_j \leftarrow Y^{\prod_{k=i+1}^{j} \mathsf{t}_k}$
16 : $\mathrm{b}' \leftarrow \mathcal{A}_i^{\mathcal{O}.\mathsf{Open}, \mathcal{O}.\mathsf{Difr}, \mathcal{O}.\mathsf{Chall}}(g, \{\mathsf{s}_1, \ldots, \mathsf{s}_n\})$
17 : if ABORT occurred then
18 :     return $\mathrm{b}' \xleftarrow{\$} \{0, 1\}$
19 : return $\mathrm{b}'$
```

```
$\mathcal{O}.\mathsf{Open}(j)$
20 : if $j \in \{\mathsf{fwl}, \ldots, \mathsf{fwr}\}$
21 :     return ABORT
22 : return $\mathsf{e}_i$
$\mathcal{O}.\mathsf{Difr}(j)$
23 : if $j \in \{\mathsf{fwl}, \mathsf{fwr}\text{+}1\}$
24 :     return ABORT
25 : return $\mathsf{t}_j$
$\mathcal{O}.\mathsf{Chall}(j)$
26 : if $j = i$
27 :     $X_j \leftarrow X$
28 :     $Z_j \leftarrow Z$
29 : else
30 :     $Z_j \leftarrow \mathsf{s}_j^{\,x_j} g^{w_j}$
31 : return $(X_j, Z_j)$
```

Figure 41: Reduction $\mathcal{B}_{7.1}$ for proof of Lemma 7.1. On line 9, † indicates t_0 and $\mathsf{t}_{\mathsf{fwr}+1}$ are skipped in the computation.

```
$\mathcal{B}_{7.2}$ playing $\mathbf{Exp}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}}_{\mathbb{G},\,\mathcal{B}_{7.2}}$
1 :  receive $g, \{\mathsf{s}_1, \ldots, \mathsf{s}_n\}$
2 :  $\mathsf{e} \leftarrow 0; \mathsf{phase} \leftarrow 0; \mathcal{L} \leftarrow \emptyset$
3 :  $\bar{\mathrm{M}}_0, \mathrm{C} \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
4 :  $\mathsf{phase} \leftarrow 1$
5 :  if $(\cdot, \mathrm{C} = (\mathrm{C}_1, \mathrm{C}_2), \tilde{\mathsf{e}} - 1; \bar{\mathrm{M}}_1) \notin \mathcal{L}$
6 :    return $\perp$
7 :  call $(X_{\tilde{\mathsf{e}}}, Z_{\tilde{\mathsf{e}}}) \leftarrow \mathcal{O}.\mathsf{Chall}(\tilde{\mathsf{e}})$
8 :  $\mathrm{b} \xleftarrow{\$} \{0, 1\}$
9 :  $\tilde{\mathrm{C}} \leftarrow (Z_{\tilde{\mathsf{e}}}, X_{\tilde{\mathsf{e}}} \cdot \bar{\mathrm{M}}_{\mathrm{b}})$     // embed
10 : $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}})$
11 : if $\mathrm{b}' = \mathrm{b}$
12 :   return $0$
13 : else
14 :   return $1$

$\mathcal{O}.\mathsf{Enc}(\mathrm{M})$
15 : $r \xleftarrow{\$} \mathbb{Z}_q^*$
16 : $\mathrm{C}_{\mathsf{e}} \leftarrow (\mathsf{s}_{\mathsf{e}}^r, g^r \cdot \mathrm{M})$
17 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_{\mathsf{e}}, \cdot; \mathrm{M})\}$
18 : return $\mathrm{C}_{\mathsf{e}}$

$\mathcal{O}.\mathsf{Next}$
19 : $\mathsf{e} \leftarrow \mathsf{e} + 1$
```

```
$\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}-1})$
20 : if $(\cdot, \mathrm{C}_{\mathsf{e}-1}, \cdot; \mathrm{M}) \notin \mathcal{L}$
21 :   return $\perp$
22 : $r \xleftarrow{\$} \mathbb{Z}_q^*$
23 : $\mathrm{C}_{\mathsf{e}} \leftarrow (\mathsf{s}_{\mathsf{e}}^r, g^r \cdot \mathrm{M})$
24 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_{\mathsf{e}}, \cdot; \mathrm{M})\}$
25 : return $\mathrm{C}_{\mathsf{e}}$

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$
26 : if $\hat{\mathsf{e}} > \mathsf{e}$
27 :   return $\perp$
28 : if $\mathsf{inp} = \mathsf{key}$
29 :   call $\mathsf{e}_{\hat{\mathsf{e}}} \leftarrow \mathcal{O}.\mathsf{Open}(\hat{\mathsf{e}})$
30 :   $\mathrm{k}_{\hat{\mathsf{e}}} \leftarrow (\mathsf{e}_{\hat{\mathsf{e}}}, g^{\mathsf{e}_{\hat{\mathsf{e}}}})$
31 :   return $\mathrm{k}_{\hat{\mathsf{e}}}$
32 : if $\mathsf{inp} = \mathsf{token}$
33 :   call $\mathsf{t}_{\hat{\mathsf{e}}} \leftarrow \mathcal{O}.\mathsf{Difr}(\hat{\mathsf{e}})$
34 :   $\Delta_{\hat{\mathsf{e}}} \leftarrow (\mathsf{t}_{\hat{\mathsf{e}}}, \mathsf{s}_{\hat{\mathsf{e}}})$
35 :   return $\Delta_{\hat{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$
36 : call $(X_{\mathsf{e}}, Z_{\mathsf{e}}) \leftarrow \mathcal{O}.\mathsf{Chall}(\mathsf{e})$
37 : $\tilde{\mathrm{C}} \leftarrow (Z_{\mathsf{e}}, X_{\mathsf{e}} \cdot \bar{\mathrm{M}}_{\mathrm{b}})$
38 : return $\tilde{\mathrm{C}}_{\mathsf{e}}$
```

Figure 42: Reduction $\mathcal{B}_{7.2}$ for proof of Lemma 7.2.

Paper IV

The Direction of Updatable Encryption does not Matter Much

Yao Jiang

Published in the 26th Annual International Conference on the Theory and Application of Cryptology and Information Security, Asiacrypt 2020.

The Direction of Updatable Encryption does not Matter Much

Yao Jiang

Norwegian University of Science and Technology, NTNU, Norway.
yao.jiang@ntnu.no

Abstract

Updatable encryption schemes allow for key rotation on ciphertexts. A client outsourcing storage of encrypted data to a cloud server can change its encryption key. The cloud server can update the stored ciphertexts to the new key using only a token provided by the client.

This paper solves two open problems in updatable encryption, that of uni-directional vs. bi-directional updates, and post-quantum security.

The main result in this paper is to analyze the security notions based on uni- and bi-directional updates. Surprisingly, we prove that uni- and bi-directional variants of each security notion are equivalent.

The second result in this paper is to provide a new and efficient updatable encryption scheme based on the Decisional Learning with Error assumption. This gives us post-quantum security. Our scheme is bi-directional, but because of our main result, this is sufficient.

† © IACR 2020: An extended abstract of this paper appears in the proceedings of Asiacrypt 2020, with DOI: 10.1007/978-3-030-64840-4_18. This is the full version.

1 Introduction

Consider the following scenario: a client wishes to outsource data to a cloud storage provider with a cryptoperiod (client key lifetime). The cryptoperiod is decided by the client or the cloud storage provider or both. If the key lifetime is expired, the old key is no longer available for either encryption or decryption, a new key must be used in the new cryptoperiod. However, the client might still want to keep the data in the cloud storage in the new cryptoperiod and needs to update the data. The above requirement implies a need to update ciphertext from the old key to the new key. During this process, it is also reasonable to expect that no information of plaintexts are leaked while updating. Another benefit to consider in such a scenario is that it can be used to protect the data and reduce the risk of key compromise over time.

Key rotation is the process of generating a new key and altering ciphertexts from the old key to the new key without changing the underlying massage.

Key rotation can be done by downloading the old ciphertext, decrypting with the old key, re-encrypting with a new key and reuploading the new ciphertext. However, this is expensive. *Updatable encryption* (UE) [BLMR13, EPRS17, LT18, KLR19, BDGJ20, BEKS20] provides a better solution for key rotation. A client generates an *update token* and send it to the cloud server, the cloud server can use this update token to update the ciphertexts from the old key to the new key. In recent years there has been considerable interest in understanding UE, including defining the security notions for UE and constructing UE schemes (we make a detailed comparison of related work in Section 1.1).

Consider the following two variants of updatable encryption schemes: *ciphertext-dependent* schemes and *ciphertext-independent* schemes. If the generation of update token depends on the ciphertext to be updated then the UE scheme is ciphertext-dependent. In ciphertext-dependent schemes, the updating process of a ciphertext requires a specific token which forces the client to download the old ciphertext before this token can be generated. Therefore, ciphertext-dependent schemes are less practical. If the token is independent of the old ciphertext then the UE scheme is ciphertext-independent. Hence, a single token can be used to update all ciphertexts a client owns. As ciphertext-independent updatable encryption schemes are considerably more efficient than ciphertext-dependent schemes, in terms of bandwidth, most recent works [BLMR15, LT18, KLR19, BDGJ20] focus on ciphertext-independent schemes. In this paper, we will focus on such schemes.

Consider the following four variants of update setting for ciphertext-independent UE schemes: *uni-directional ciphertext* updates, *bi-directional ciphertext* updates, *uni-directional key* updates and *bi-directional key* updates. If the update token can only move ciphertexts from the old key to the new key then ciphertext updates in such UE schemes are uni-directional. If the update token can additionally downgrade ciphertexts from the new key to the old key then ciphertext updates in such UE schemes are bi-directional. On the other hand, the update token can potentially be used to derive keys from other keys. In the uni-directional key update setting, the update token can only infer the new key from the old key. While in the bi-directional key update setting, the update token can both upgrade and downgrade keys. Prior works [BLMR15, LT18, KLR19, BDGJ20] focus on UE schemes with bi-directional updates, and no security notion was introduced in uni-directional update setting. We close this gap. Intuitively, UE schemes with uni-directional updates are desirable, such schemes leak less ciphertext/key information to an adversary compared to schemes with bi-directional updates. In this paper, we analyze the relationship between security notions with uni- and bi-directional updates. We show that the (confidentiality and integrity) security of UE schemes are not influenced by uni- or bi-directional updates.

No-directional key updates is another key update setting to consider, where the update token cannot be used to derive keys. A UE scheme with optimal leakage, discussed in [LT18], is a scheme where no token inference (no token can be inferred via keys), keys cannot be updated via a token, and ciphertext updates are only uni-directional. We do not consider no token inference, instead in this work an update token can be computed via two consecutive epoch keys. We show that the no-directional key update variant of a security notion is strictly stronger than the uni- and bi-directional update variant of the same security notion.

While the study of security notions for UE schemes appears promising, existing ciphertext-independent UE schemes are either vulnerable to quantum computers or only achieve weak security. The schemes of Lehmann and Tackmann [LT18], Klooß et al. [KLR19] and Boyd et al. [BDGJ20] base their security on the DDH problem, and thus are only secure in the classical setting. Boneh et al. [BLMR13] constructed key homomorphic PRFs, based on the learning with errors (LWE) problem, and it can be used to construct UE schemes. However, all of these schemes of Boneh et al. [BLMR13] cannot achieve IND-UPD security (introduced in [LT18]).

In this work, we construct a post-quantum secure UE scheme and the security of our construction is based on hard lattice problems. In particular, our scheme provides the randIND-UE-CPA security (introduced in [BDGJ20], stronger than IND-UPD and IND-ENC security).

Efficiency. All of the previous known ciphertext-independent UE schemes with security proofs (RISE, E&M, NYUE (NYUAE), SHINE) have computation cost that are comparable to PKE schemes that rely on the DDH problem, while our scheme has a computation cost that is comparable to PKE schemes that rely on lattice problems.

1.1 Related Work

Security Notions. Boneh et al. [BLMR13] introduced a security definition for UE, however, this notion is less adaptive than the later works [LT18, KLR19, BDGJ20] which allows the adversary to adaptively corrupt epoch keys and update tokens at any point in the game.

In the ciphertext-dependent setting, Everspaugh et al. [EPRS17] provided two security notions, a weak form of ciphertext integrity and re-encryption indistinguishability, that strengthen the security notion in the work of Boneh et al. [BLMR13]. Recently, Boneh et al. [BEKS20] introduced new definitions for updatable encryption in the ciphertext-dependent setting to further strengthen the confidentiality property and the integrity definition in [EPRS17]. Boneh et al. [BEKS20] stated that for authenticated updatable encryption schemes it is necessary to expect that ciphertexts will not reveal how many times they have been updated, which was a desired property independently presented in [BDGJ20].

Lehmann and Tackmann [LT18] introduced two notions to achieve CPA security for ciphertext-independent UE schemes. Their IND-ENC notion requires that ciphertexts output by the encryption algorithm are indistinguishable from each other. Their IND-UPD notion ensures ciphertexts output by the update algorithm are indistinguishable from each other.

Klooß et al. [KLR19] attempted to provide stronger security notions for ciphertext-independent UE than LT18, specifically, CCA security and integrity protection.

Boyd et al. [BDGJ20] provided a new notion IND-UE which states that a ciphertext output by the encryption algorithm is indistinguishable from a ciphertext output by the update algorithm. They showed that the new notion is strictly stronger than any combinations of prior notions, both under CPA and CCA. They also tweaked the CTXT and CCA notions in [KLR19] and showed the following generic composition result: CPA + CTXT $\implies$ CCA.

Constructing Ciphertext-Independent Updatable Encryption Schemes. The UE scheme BLMR in [BLMR13] is an application of key homomorphic PRFs, however, the encrypted nonce in the ciphertext can be decrypted by an update token which makes it impossible for BLMR to achieve IND-UPD security.

In the classical setting, RISE in [LT18] is built from (public-key) ElGamal encryption, which only uses the public key in the update token. The security of RISE is based on the DDH assumption. Klooß et al. [KLR19] provided two generic constructions, based on encrypt-and-MAC (E&M) and the Naor-Yung paradigm (NYUE and NYUAE). The security of E&M is based on the DDH assumption, and the security of NYUE and NYUAE are based on the SXDH assumption. Boyd et al. [BDGJ20] constructed three permutation-based UE schemes, SHINE, which achieves strong security notions based on DDH.

Post-Quantum Secure Schemes. In the past decade, much work has been done on constructing lattice-based post-quantum secure PKE schemes, specifically the NIST Post-Quantum Standardization Project, round 2, submissions: CRYSTALS-KYBER [ABD+b], FrodoKEM [ABD+a], LAC [LLJ+], Round5 [OZS+], SABER [DKRV18], NewHope [ADPS16], NTRU [CDH+, BCLvV17] and Three Bears [Ham]. A natural question is if we can turn a PKE scheme into a UE scheme, where the security of the UE follows from the PKE. We provide a specific UE scheme that is built form an LWE-based

PKE scheme, and prove the security. The LWE-based scheme we use is in some sense very similar to RISE (which is based on ElGamal), however, as with most lattice-based constructions, there are significant technical problems in turning it into a UE scheme (see Section 5.2). Our LWE-based UE construction suggests that there is a limit to how generic any efficient construction can be, a generic construction that abstracts both our construction and RISE remains to be done.

1.2 Our Contributions

Our first contribution is defining six variants of security notions (a combination of three versions of key updates and two versions of ciphertext updates) for updatable encryption and analyzing the relations among these six variants of the same notion.

Our main result is that we demonstrate that our security notions with uni- and bi-directional updates are equivalent. When we analyze the security, we can treat UE schemes with uni-directional updates as with bi-directional updates, the security will not be influenced by the update direction. This means that UE schemes with uni-directional updates will not provide more security than UE schemes with bi-directional updates. This is a surprising result.[1] This result implies that the search for uni-directional updatable encryption scheme seems less important.

Furthermore, we show that security notions with no-directional key updates are strictly stronger than uni- and bi- directional update variants of the corresponding notions. Finding UE schemes with no-directional key updates would be good, but it is much more challenge than finding UE schemes with uni-directional key updates (which is already believed to be

[1]It is possible to construct a scenario where this result will not be true. Let's assume there exists a UE scheme with a leakage function that helps the adversary win the security game. This leakage function could, for example, give the adversary information about plaintexts when it knows enough keys. In this scenario, a UE scheme with uni-directional updates has better security than a UE scheme with bi-directional updates. Because the scheme with uni-directional updates has less key leakage and the leakage function provides less data to the adversary. However, this and similar constructions cannot capture the security we wish to have for UE schemes. In terms of the security expectation of key rotation, the keys used in the past should not reveal any data.

For constructions that do follow the security model and update mechanism for UE schemes, we have this surprising result.

difficult). We leave this as an open problem.

Our second major contribution is constructing an efficient post-quantum secure UE scheme. We analyze how to construct LWE-based updatable encryption schemes and provide one construction. Our construction follows the re-randomization idea of RISE, using public key in the update token to update ciphertexts. We build a suitable post-quantum secure PKE scheme to construct our UE scheme so that the encryption and update algorithms can use a public key as input instead of the secret key. We also show the difficulties of turning a PKE scheme into a UE scheme.

We show that our LWE-based updatable encryption scheme achieves randIND-UE-CPA secure under the DLWE assumption. In the randomized update setting, we show the difference between previous work (RISE, NYUE, NYUAE) and our scheme, and state that the method used in proving the security of LWE-based updatable encryption scheme is different from the previous approach.

1.3 Open Problems

Ideally we want UE schemes with no-directional key updates, no such UE schemes have been constructed so far. Whether such UE schemes exist and how to construct such UE schemes are still open problems.

Furthermore, not that many efficient UE schemes with strong security exist so far. It remains an open challenge to construct UE schemes with chosen ciphertext[2] post-quantum security.

1.4 Organization

We provide the background of updatable encryption in Section 2. In Section 3 we define the six variants of security notions for UE schemes and prove the relations among the six variants of security notions.

In Section 4 we construct an LWE-based PKE scheme LWEPKE and prove that it is secure, this PKE scheme will be used to construct a UE scheme. We then construct an LWE-based UE scheme LWEUE in Section 5,

[2]It is ideal to achieve detIND-UE-CCA security for UE schemes with deterministic updates and to achieve INT-PTXT and randIND-UE-CCA security for UE schemes with randomized updates.

and include the restrictions we encountered when constructing a secure UE scheme from a PKE scheme.

2 Preliminaries

In this section we describe the notation used in this paper and present the necessary background material of updatable encryption. In Appendix A, we provide the background of hard lattice problems.

2.1 Notations

Let λ be the security parameter throughout the paper. Let negl denote as a negligible function. The notation $X \overset{s}{\approx} Y$ (X$\overset{c}{\approx} Y$, resp.) means X is statistically indistinguishable (computationally indistinguishable, resp.) from Y. Let $\mathcal{U}(S)$ denote the uniform distribution over set S.

2.2 Security Notions for Encryption Schemes

We describe the real or random variant of indistinguishability under chosen-plaintext attack (IND\$-CPA) for public key encryption (PKE) and symmetric key encryption (SKE). Note that IND\$-CPA implies IND-CPA.

Definition 1. [The IND\$-CPA notion for PKE] Let $\mathsf{PKE} = (\mathsf{PKE.KG}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a public key encryption scheme. The IND\$-CPA advantage of any adversary $\mathcal{A}$ against PKE is

$$\mathbf{Adv}_{\mathsf{PKE},\ \mathcal{A}}^{\mathsf{IND\$\text{-}CPA}}(1^{\lambda}) = \left|\Pr[\mathbf{Exp}_{\mathsf{PKE},\ \mathcal{A}}^{\mathsf{IND\$\text{-}CPA\text{-}1}} = 1] - \Pr[\mathbf{Exp}_{\mathsf{PKE},\ \mathcal{A}}^{\mathsf{IND\$\text{-}CPA\text{-}0}} = 1]\right|,$$

where the experiment $\mathbf{Exp}_{\mathsf{PKE},\ \mathcal{A}}^{\mathsf{IND\$\text{-}CPA\text{-}b}}$ is given in Figure 1.

Definition 2. [The IND\$-CPA notion for SKE] Let $\mathsf{SKE} = (\mathsf{SKE.KG}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ be a symmetric key encryption scheme. The IND\$-CPA advantage of any adversary $\mathcal{A}$ against SKE is

$$\mathbf{Adv}_{\mathsf{SKE},\ \mathcal{A}}^{\mathsf{IND\$\text{-}CPA}}(1^{\lambda}) = \left|\Pr[\mathbf{Exp}_{\mathsf{SKE},\ \mathcal{A}}^{\mathsf{IND\$\text{-}CPA\text{-}1}} = 1] - \Pr[\mathbf{Exp}_{\mathsf{SKE},\ \mathcal{A}}^{\mathsf{IND\$\text{-}CPA\text{-}0}} = 1]\right|,$$

where the experiment $\mathbf{Exp}_{\mathsf{SKE},\ \mathcal{A}}^{\mathsf{IND\$\text{-}CPA\text{-}b}}$ is given in Figure 2.

```
$\mathbf{Exp}^{\mathsf{IND\$\text{-}CPA\text{-}b}}_{\mathsf{PKE},\,\mathcal{A}}$:
    $(\mathbf{s}, \mathbf{p}) \leftarrow \mathsf{PKE.KG}$
    $(\mathbf{m}, \mathrm{state}) \leftarrow \mathcal{A}(\mathbf{p})$
    if $\mathrm{b} = 0$ then
        $\mathbf{c} \leftarrow \mathsf{PKE.Enc}(\mathbf{p}, \mathbf{m})$
    else
        $\mathbf{c} \xleftarrow{\$} \mathcal{CS}$
    $\mathrm{b}' \leftarrow \mathcal{A}(\mathrm{state}, \mathbf{c})$
    return $\mathrm{b}'$
```

Figure 1: The experiment $\mathbf{Exp}^{\mathsf{IND\$\text{-}CPA\text{-}b}}_{\mathsf{PKE},\,\mathcal{A}}$ for a PKE scheme PKE.

```
$\mathbf{Exp}^{\mathsf{IND\$\text{-}CPA\text{-}b}}_{\mathsf{SKE},\,\mathcal{A}}$:
    $\mathbf{s} \leftarrow \mathsf{SKE.KG}$
    $(\mathbf{m}, \mathrm{state}) \leftarrow \mathcal{A}(1^\lambda)$
    if $\mathrm{b} = 0$ then
        $\mathbf{c} \leftarrow \mathsf{SKE.Enc}(\mathbf{s}, \mathbf{m})$
    else
        $\mathbf{c} \xleftarrow{\$} \mathcal{CS}$
    $\mathrm{b}' \leftarrow \mathcal{A}(\mathrm{state}, \mathbf{c})$
    return $\mathrm{b}'$
```

Figure 2: The experiment $\mathbf{Exp}^{\mathsf{IND\$\text{-}CPA\text{-}b}}_{\mathsf{SKE},\,\mathcal{A}}$ for a SKE scheme SKE.

Definition 3. [Correctness of a PKE] Let PKE = (PKE.KG, PKE.Enc, PKE.Dec) be a public key encryption scheme. We say PKE has *$(1-\epsilon)$-correctness* if: for any message $\mathbf{m}$, any key $(\mathbf{s}, \mathbf{p}) \leftarrow \mathsf{PKE.KG}$

$$\mathbf{Pr}[\mathsf{PKE.Dec}(\mathbf{s}, \mathsf{PKE.Enc}(\mathbf{p}, \mathbf{m})) = \mathbf{m}] \geq 1 - \epsilon.$$

2.3 Updatable Encryption

Updatable encryption (UE) scheme is parameterized by a tuple of algorithms $\{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ that operate in epochs, the epoch starts at 0. The key generation algorithm UE.KG outputs an epoch key $\mathbf{k}_\mathsf{e}$. The token generation algorithm UE.TG takes as input two epoch keys $\mathbf{k}_\mathsf{e}$ and $\mathbf{k}_{\mathsf{e}+1}$ and outputs an update token $\Delta_{\mathsf{e}+1}$, the update token can be used to move ciphertexts from epoch e to $\mathsf{e}+1$. The encryption algorithm UE.Enc takes as input an epoch key $\mathbf{k}_\mathsf{e}$ and a message $\mathbf{m}$ and outputs a ciphertext $\mathbf{c}_\mathsf{e}$. The decryption algorithm UE.Dec takes as input an epoch key $\mathbf{k}_\mathsf{e}$ and a ciphertext $\mathbf{c}_\mathsf{e}$ and outputs a message $\mathbf{m}'$. The update algorithm UE.Upd takes as input an update token $\Delta_{\mathsf{e}+1}$ and a ciphertext $\mathbf{c}_\mathsf{e}$ from epoch e and outputs an updated ciphertext $\mathbf{c}_{\mathsf{e}+1}$. We stress that an update token can be computed via two consecutive epoch keys by token generation algorithm in this paper.

In the updatable encryption setting, the total number of epoch will be a comparatively small integer in practice, we consider the total number of epoch to be bounded in this paper. In particular, we denote l as an upper bound on the last epoch.

Definition 4 (Correctness of a UE)**.** Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. We say UE has *$(1-\epsilon)$-correctness* if: for any message $\mathbf{m}$ and any epochs $\mathsf{e}_1 \leq \mathsf{e}_2 \leq l$, we have

$$\mathbf{Pr}[\mathsf{UE.Dec}(\mathbf{k}_{\mathsf{e}_2}, \mathbf{c}_{\mathsf{e}_2}) = \mathbf{m}] \geq 1 - \epsilon,$$

where $\mathbf{k}_{\mathsf{e}_1}, \ldots, \mathbf{k}_{\mathsf{e}_2} \overset{\$}{\leftarrow} \mathsf{UE.KG}(1^\lambda)$, $\mathbf{c}_{\mathsf{e}_1} \overset{\$}{\leftarrow} \mathsf{UE.Enc}(\mathbf{k}_{\mathsf{e}_1}, \mathbf{m})$, and

$\Delta_j \leftarrow \mathsf{UE.TG}(\mathbf{k}_{j-1}, \mathbf{k}_j), \mathbf{c}_j \leftarrow \mathsf{UE.Upd}(\Delta_j, \mathbf{c}_{j-1}),$ for $j \in \{\mathsf{e}_1{+}1, \ldots, \mathsf{e}_2\}$.

2.4 Existing Security Notions for Updatable Encryption

Klooß et al. [KLR19] and Boyd et al. [BDGJ20] defined the confidentiality and the integrity notions for updatable encryption schemes using experiments that are running between an adversary and a challenger. In each experiment, the adversary may send a number of oracle queries. The main differences between an experiment running the confidentiality game and one running the integrity game are the challenge and win condition. In the confidentiality game, the adversary tries to distinguish a fresh encryption from an updated ciphertext. In the integrity game, the adversary attempts to provide a valid forgery. At the end of an experiment the challenger evaluates whether or not the adversary wins, if a trivial win condition was triggered the adversary will always lose.

We follow the notation of security notions from Boyd et al. [BDGJ20]. An overview of the oracles the adversary has access to in each security game is given in Fig. 3. A generic description of all confidentiality experiments and integrity experiments described in this paper is detailed in Fig. 4 and Fig. 5, resp.. Our oracle algorithms, see Fig. 6, are stated differently than in [BDGJ20] and [KLR19], however, conceptually they are the same. The oracles we use in our security games are as follows, encrypt $\mathcal{O}.\mathsf{Enc}$, decrypt $\mathcal{O}.\mathsf{Dec}$, move to the next epoch $\mathcal{O}.\mathsf{Next}$, update ciphertext $\mathcal{O}.\mathsf{Upd}$, corrupt key or token $\mathcal{O}.\mathsf{Corr}$, ask for the challenge ciphertext $\mathcal{O}.\mathsf{Chall}$, get an updated version of the challenge ciphertext $\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$, or test if a ciphertext is a valid forgery $\mathcal{O}.\mathsf{Try}$. The detailed discussion of trivial win conditions are discussed in Section 2.7.

Notions	$\mathcal{O}$.Enc	$\mathcal{O}$.Dec	$\mathcal{O}$.Next	$\mathcal{O}$.Upd	$\mathcal{O}$.Corr	$\mathcal{O}$.Chall	$\mathcal{O}$.Upd$\tilde{\text{C}}$	$\mathcal{O}$.Try
detIND-UE-CPA	✓	×	✓	✓	✓	✓	✓	×
randIND-UE-CPA	✓	×	✓	✓	✓	✓	✓	×
detIND-UE-CCA	✓	✓	✓	✓	✓	✓	✓	×
randIND-UE-CCA	✓	✓	✓	✓	✓	✓	✓	×
INT-CTXT	✓	×	✓	✓	✓	×	×	✓
INT-PTXT	✓	×	✓	✓	✓	×	×	✓

Figure 3: Oracles given to the adversary in different security games for updatable encryption schemes. × indicates the adversary does not have access to the corresponding oracle, ✓ indicates the adversary has access to the corresponding oracle.

```
$\mathbf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}b}}_{\mathsf{UE},\,\mathcal{A}}$ :
  do Setup; $\mathsf{phase} \leftarrow 0$
  $\mathrm{b}' \leftarrow \mathcal{A}^{oracles}(1^\lambda)$
  if $\Big((\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset)$ or $(\mathsf{xx} = \mathsf{det}$ and
      $(\tilde{\mathsf{e}} \in \mathcal{T}^*$ or $\mathcal{O}.\mathsf{Upd}(\bar{\mathbf{c}})$ is queried$))\Big)$ then
    $\mathsf{twf} \leftarrow 1$
  if $\mathsf{twf} = 1$ then
    $\mathrm{b}' \xleftarrow{\$} \{0,1\}$
  return $\mathrm{b}'$
```

Figure 4: Generic description of the confidentiality experiment $\mathbf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}b}}_{\mathsf{UE},\,\mathcal{A}}$ for updatable encryption scheme UE and adversary $\mathcal{A}$, for $\mathsf{xx} \in \{\mathsf{det}, \mathsf{rand}\}$ and $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA}\}$. The flag phase tracks whether or not $\mathcal{A}$ has queried the $\mathcal{O}$.Chall oracle, $\tilde{\mathsf{e}}$ denotes the epoch in which the $\mathcal{O}$.Chall oracle happens, and twf tracks if the trivial win conditions are triggered. Fig. 3 shows the oracles the adversary have access to in a specific security game. How to compute the leakage sets $\mathcal{K}^*, \mathcal{T}^*, \mathcal{C}^*$ are discussed in Section 2.6.

```
$\mathbf{Exp}^{\mathsf{INT\text{-}atk}}_{\mathsf{UE},\,\mathcal{A}}$
  do Setup; $\mathsf{win} \leftarrow 0$
  $\mathcal{A}^{oracles}(1^\lambda)$
  if $\mathsf{twf} = 1$ then
    $\mathsf{win} \leftarrow 0$
  return $\mathsf{win}$
```

Figure 5: Generic description of the integrity experiment $\mathbf{Exp}^{\mathsf{INT\text{-}atk}}_{\mathsf{UE},\,\mathcal{A}}$ for updatable encryption scheme UE and adversary $\mathcal{A}$, for $\mathsf{atk} \in \{\mathsf{CTXT}, \mathsf{PTXT}\}$. The flag win tracks whether or not the adversary provided a valid forgery and twf tracks if the trivial win conditions are triggered. Fig. 3 shows the oracles the adversary have access to in a specific security game.

For the confidentiality game we have the following additional definitions that we will frequently use. While the security game is running, the adversary may query $\mathcal{O}.\mathsf{Enc}$ or $\mathcal{O}.\mathsf{Upd}$ oracles or corrupt tokens to know some (updated) versions of ciphertexts, we call them *non-challenge ciphertexts*. In addition, the adversary may query $\mathcal{O}.\mathsf{Chall}$ or $\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$ oracles or corrupt tokens to infer some (updated) versions of the challenge ciphertext, we call them *challenge-equal ciphertexts*.

Definition 5. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be a UE scheme. Then the notion advantage, for notion $\in \{\mathsf{detIND\text{-}UE\text{-}CPA}, \mathsf{randIND\text{-}UE\text{-}CPA}, \mathsf{detIND\text{-}UE\text{-}CCA}, \mathsf{randIND\text{-}UE\text{-}CCA}\}$, of an adversary $\mathcal{A}$ against UE is defined as

$$\mathbf{Adv}^{\mathsf{notion}}_{\mathsf{UE},\,\mathcal{A}}(1^\lambda) = \left|\mathbf{Pr}[\mathbf{Exp}^{\mathsf{notion\text{-}1}}_{\mathsf{UE},\,\mathcal{A}} = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{notion\text{-}0}}_{\mathsf{UE},\,\mathcal{A}} = 1]\right|,$$

where the experiment $\mathbf{Exp}^{\mathsf{notion\text{-}b}}_{\mathsf{UE},\,\mathcal{A}}$ is given in Fig. 4 and Fig. 6.

Definition 6. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. Then the notion advantage, for notion $\in \{\mathsf{INT\text{-}CTXT}, \mathsf{INT\text{-}PTXT}\}$, of an adversary $\mathcal{A}$ against UE is defined as

$$\mathbf{Adv}^{\mathsf{notion}}_{\mathsf{UE},\,\mathcal{A}}(1^\lambda) = \mathbf{Pr}[\mathbf{Exp}^{\mathsf{notion}}_{\mathsf{UE},\,\mathcal{A}} = 1],$$

where the experiment $\mathbf{Exp}^{\mathsf{notion}}_{\mathsf{UE},\,\mathcal{A}}$ is given in Fig. 5 and Fig. 6.

```
$\mathbf{Setup}(1^\lambda)$
    $\mathbf{k}_0 \xleftarrow{\$} \mathsf{UE.KG}(1^\lambda)$
    $\Delta_0 \leftarrow \perp; \mathsf{e}, \mathrm{c}, \mathsf{twf} \leftarrow 0$
    $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$
$\mathcal{O}.\mathsf{Enc}(\mathbf{m})$ :
    $\mathrm{c} \leftarrow \mathrm{c} + 1$
    $\mathbf{c} \xleftarrow{\$} \mathsf{UE.Enc}(\mathbf{k}_\mathsf{e}, \mathbf{m})$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathrm{c}, \mathbf{c}, \mathsf{e}; \mathbf{m})\}$
    return $\mathbf{c}$
$\mathcal{O}.\mathsf{Dec}(\mathbf{c})$ :
    $\mathbf{m}'$ or $\perp \leftarrow \mathsf{UE.Dec}(\mathbf{k}_\mathsf{e}, \mathbf{c})$
    if $\Big((\mathsf{xx} = \mathsf{det}$ and $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*)$ or
        $(\mathsf{xx} = \mathsf{rand}$ and $(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*)\Big)$ then
        $\mathsf{twf} \leftarrow 1$    return $\mathbf{m}'$ or $\perp$
$\mathcal{O}.\mathsf{Next}()$ :
    $\mathsf{e} \leftarrow \mathsf{e} + 1$
    $\mathbf{k}_\mathsf{e} \xleftarrow{\$} \mathsf{UE.KG}(1^n)$
    $\Delta_\mathsf{e} \leftarrow \mathsf{UE.TG}(\mathbf{k}_{\mathsf{e}\text{-}1}, \mathbf{k}_\mathsf{e})$
    if $\mathsf{phase} = 1$ then
        $\tilde{\mathbf{c}}_\mathsf{e} \leftarrow \mathsf{UE.Upd}(\Delta_\mathsf{e}, \tilde{\mathbf{c}}_{\mathsf{e}\text{-}1})$
$\mathcal{O}.\mathsf{Upd}(\mathbf{c}_{\mathsf{e}-1})$ :
    if $(j, \mathbf{c}_{\mathsf{e}-1}, \mathsf{e} - 1; \mathbf{m}) \notin \mathcal{L}$ then
        return $\perp$
    $\mathbf{c}_\mathsf{e} \leftarrow \mathsf{UE.Upd}(\Delta_\mathsf{e}, \mathbf{c}_{\mathsf{e}-1})$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(j, \mathbf{c}_\mathsf{e}, \mathsf{e}; \mathbf{m})\}$
    return $\mathbf{c}_\mathsf{e}$
$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$ :
    if $\hat{\mathsf{e}} > \mathsf{e}$ then
        return $\perp$
    if $\mathsf{inp} = \mathsf{key}$ then
        $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
        return $\mathbf{k}_{\hat{\mathsf{e}}}$
$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$ continue :
    if $\mathsf{inp} = \mathsf{token}$ then
        $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
        return $\Delta_{\hat{\mathsf{e}}}$
$\mathcal{O}.\mathsf{Chall}(\bar{\mathbf{m}}, \bar{\mathbf{c}})$ :
    if $\mathsf{phase} = 1$ then
        return $\perp$
    $\mathsf{phase} \leftarrow 1; \tilde{\mathsf{e}} \leftarrow \mathsf{e}$
    if $(\cdot, \bar{\mathbf{c}}, \tilde{\mathsf{e}} - 1; \bar{\mathbf{m}}_1) \notin \mathcal{L}$ then
        return $\perp$
    if $\mathrm{b} = 0$ then
        $\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}} \leftarrow \mathsf{UE.Enc}(\mathbf{k}_{\tilde{\mathsf{e}}}, \bar{\mathbf{m}})$
    else
        $\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}} \leftarrow \mathsf{UE.Upd}(\Delta_{\tilde{\mathsf{e}}}, \bar{\mathbf{c}})$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{\tilde{\mathsf{e}}\}$
    $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}}, \tilde{\mathsf{e}})\}$
    return $\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}}$
$\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$ :
    if $\mathsf{phase} \neq 1$ then
        return $\perp$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
    $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{\mathbf{c}}_\mathsf{e}, \mathsf{e})\}$
    return $\tilde{\mathbf{c}}_\mathsf{e}$
$\mathcal{O}.\mathsf{Try}(\tilde{\mathbf{c}})$ :
    $\mathbf{m}'$ or $\perp \leftarrow \mathsf{UE.Dec}(\mathbf{k}_\mathsf{e}, \tilde{\mathbf{c}})$
    if $\Big(\mathsf{e} \in \mathcal{K}^*$ or $(\mathsf{atk} = \mathsf{CTXT}$ and
        $(\tilde{\mathbf{c}}, \mathsf{e}) \in \mathcal{L}^*)$ or $(\mathsf{atk} = \mathsf{PTXT}$
         and $(\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*)\Big)$ then
        $\mathsf{twf} \leftarrow 1$    if $\mathbf{m}' \neq \perp$ then
        $\mathsf{win} \leftarrow 1$
```

Figure 6: Oracles in security games for updatable encryption. How to compute the leakage sets $\mathcal{K}^*, \mathcal{T}^*, \mathcal{C}^*, \tilde{\mathcal{L}}^*, \tilde{\mathcal{Q}}^*, \mathcal{L}^*, \mathcal{Q}^*$ are discussed in Section 2.6 and Section 2.7.

2.5 Notations of the Leakage Sets

In this section, we describe the definition of leakage sets given by [LT18] and [KLR19], these sets will later be used to check whether the leaked information will allow the adversary trivially win the security game. We analyze some properties of leakage sets and trivial win conditions in Section 3.1.

Epoch Leakage Sets. We use the following sets that track epochs in which the adversary corrupted a key or a token, or learned a version of challenge-ciphertext.

- $\mathcal{K}$: Set of epochs in which the adversary corrupted the epoch key (from $\mathcal{O}.\mathsf{Corr}$).
- $\mathcal{T}$: Set of epochs in which the adversary corrupted the update token (from $\mathcal{O}.\mathsf{Corr}$).
- $\mathcal{C}$: Set of epochs in which the adversary learned a challenge-equal ciphertext (from $\mathcal{O}.\mathsf{Chall}$ or $\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$).

We use $\mathcal{K}^*, \mathcal{T}^*$ and $\mathcal{C}^*$ as the extended sets of $\mathcal{K}$, $\mathcal{T}$ and $\mathcal{C}$ in which the adversary has learned or inferred information via its known tokens. We show how to compute $\mathcal{K}^*, \mathcal{T}^*$ and $\mathcal{C}^*$ in Section 2.6.

Information Leakage Sets. We use the following sets to track ciphertexts and their updates that can be known to the adversary.

- $\mathcal{L}$: Set of non-challenge ciphertexts $(\mathrm{c}, \mathbf{c}, \mathsf{e}; \mathbf{m})$, where query identifier c is a counter incremented with each new $\mathcal{O}.\mathsf{Enc}$ query. The adversary learned these ciphertexts from $\mathcal{O}.\mathsf{Enc}$ or $\mathcal{O}.\mathsf{Upd}$.
- $\tilde{\mathcal{L}}$: Set of challenge-equal ciphertexts $(\tilde{\mathbf{c}}_{\mathsf{e}}, \mathsf{e})$. The adversary learned these ciphertexts from $\mathcal{O}.\mathsf{Chall}$ or $\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$.

In the deterministic update setting, we use $\mathcal{L}^*$ and $\tilde{\mathcal{L}}^*$ as the extended (ciphertext) sets of $\mathcal{L}$ and $\tilde{\mathcal{L}}$ in which the adversary has learned or inferred ciphertexts via its known tokens. In particular, we only use partial information of $\mathcal{L}^*$: the ciphertext and the epoch. Hence, we only track the set $\mathcal{L}^* = \{(\mathbf{c}, \mathsf{e})\}$.

In the randomized update setting, we use $\mathcal{Q}^*$ and $\tilde{\mathcal{Q}}^*$ as the extended (plaintext) sets of $\mathcal{L}$ and $\tilde{\mathcal{L}}$, that contain messages that the adversary can provide a ciphertext of - i.e. a forgery. Similarly, only partial information is needed: the plaintext and the epoch. Hence, we track sets $\mathcal{Q}^*$ and $\tilde{\mathcal{Q}}^*$ as follows.

- $\mathcal{Q}^*$: Set of plaintexts $(\mathbf{m}, \mathsf{e})$. The adversary learned or was able to create a ciphertext in epoch e with the underlying message $\mathbf{m}$.
- $\tilde{\mathcal{Q}}^*$: Set of challenge plaintexts $\{(\bar{\mathbf{m}}, \mathsf{e}), (\bar{\mathbf{m}}_1, \mathsf{e})\}$, where $(\bar{\mathbf{m}}, \bar{\mathbf{c}})$ is the input of challenge query $\mathcal{O}.\mathsf{Chall}$ and $\bar{\mathbf{m}}_1$ is the underlying message of $\bar{\mathbf{c}}$. The adversary learned or was able to create a challenge-equal ciphertext in epoch e with the underlying message $\bar{\mathbf{m}}$ or $\bar{\mathbf{m}}_1$.

Remark 2.1. Based on the definition of these sets, we observe that

a. $(\tilde{\mathbf{c}}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{L}} \iff \mathsf{e} \in \mathcal{C}$,

b. $(\tilde{\mathbf{c}}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{L}}^* \iff \mathsf{e} \in \mathcal{C}^* \iff (\bar{\mathbf{m}}, \mathsf{e}), (\bar{\mathbf{m}}_1, \mathsf{e}) \in \tilde{\mathcal{Q}}^*$.

We will use this remark to discuss how to compute $\mathcal{L}^*$, $\tilde{\mathcal{L}}^*$, $\mathcal{Q}^*$ and $\tilde{\mathcal{Q}}^*$ in Section 2.7.

2.6 Epoch Leakage Sets of Keys, Tokens and Ciphertexts

We follow the bookkeeping techniques and base our notations of the work of Lehmann and Tackmann [LT18], where we further analyze the epoch leakage sets. Specifically, we add a no-directional key update setting. Suppose a security game ends at epoch l, then, for any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \ldots, l\}$, the following algorithms show how to compute the extended sets $\mathcal{K}^*, \mathcal{T}^*$ and $\mathcal{C}^*$ in different update settings.

Key Leakage. The adversary learned all keys in epochs in $\mathcal{K}$. In the no-directional key update setting, the adversary does not have more information about keys except for this set. In the uni-directional key update setting, if the adversary knows a key $\mathbf{k}_{\mathsf{e}}$ and an update token $\Delta_{\mathsf{e}+1}$ then it can infer the next key $\mathbf{k}_{\mathsf{e}+1}$. In the bi-directional key update setting, the adversary can additionally downgrade a key by a known token. In the kk-directional key update setting, for $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$, denote the set $\mathcal{K}^*_{\mathsf{kk}}$ as the extended set of corrupted key epochs. We compute these sets as follows.

No-directional key updates: $\mathcal{K}^*_{\mathsf{no}} = \mathcal{K}$.

Uni-directional key updates:

$$\begin{aligned} &\mathcal{K}^*_{\mathsf{uni}} \leftarrow \{\mathsf{e} \in \{0, \dots, l\} | \mathsf{CorrK}(\mathsf{e}) = \mathsf{true}\} \\ &\quad \mathsf{true} \leftarrow \mathsf{CorrK}(\mathsf{e}) \iff (\mathsf{e} \in \mathcal{K}) \vee (\mathsf{CorrK}(\mathsf{e}\text{-}1) \wedge \mathsf{e} \in \mathcal{T}). \end{aligned} \tag{1}$$

Bi-directional key updates:

$$\begin{aligned} &\mathcal{K}^*_{\mathsf{bi}} \leftarrow \{\mathsf{e} \in \{0, \dots, l\} | \mathsf{CorrK}(\mathsf{e}) = \mathsf{true}\} \\ &\quad \mathsf{true} \leftarrow \mathsf{CorrK}(\mathsf{e}) \iff \\ &\qquad (\mathsf{e} \in \mathcal{K}) \vee (\mathsf{CorrK}(\mathsf{e}\text{-}1) \wedge \mathsf{e} \in \mathcal{T}) \vee (\mathsf{CorrK}(\mathsf{e}\text{+}1) \wedge \mathsf{e}\text{+}1 \in \mathcal{T}). \end{aligned} \tag{2}$$

Token Leakage. A token is known to the adversary is either a corrupted token or a token inferred from two consecutive epoch keys, so the extended set of corrupted token epochs is computed by information in set $\mathcal{T}$ and set $\mathcal{K}^*_{\mathsf{kk}}$. The set $\mathcal{K}^*_{\mathsf{kk}}$ is computed as above depending on the key updates is no- or uni- or bi-directional. Hence, we denote $\mathcal{T}^*_{\mathsf{kk}}$ as the extended set of corrupted token epochs.

$$\mathcal{T}^*_{\mathsf{kk}} \leftarrow \{\mathsf{e} \in \{0, \dots, l\} | (\mathsf{e} \in \mathcal{T}) \vee (\mathsf{e} \in \mathcal{K}^*_{\mathsf{kk}} \wedge \mathsf{e}\text{-}1 \in \mathcal{K}^*_{\mathsf{kk}})\}. \tag{3}$$

Challenge-Equal Ciphertext Leakage. The adversary directly learned all challenge-equal ciphertexts in epochs in $\mathcal{C}$. Additionally, the adversary can infer challenge-equal ciphertexts via tokens. In the uni-directional ciphertext update setting, the adversary can upgrade ciphertexts. In the bi-directional ciphertext update setting, the adversary can additionally downgrade ciphertexts.

We compute the extended set of challenge-equal epochs using the information contained in $\mathcal{C}$ and $\mathcal{T}^*_{\mathsf{kk}}$. The set $\mathcal{T}^*_{\mathsf{kk}}$ is computed as above depending on the key updates is no- or uni- or bi-directional. In the cc-directional ciphertext update setting, for $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$, denote the set $\mathcal{C}^*_{\mathsf{kk},\mathsf{cc}}$ as the extended set of challenge-equal epochs. We compute these sets as follows.

Uni-directional ciphertext updates:

$$\begin{aligned} &\mathcal{C}^*_{\mathsf{kk},\mathsf{uni}} \leftarrow \{\mathsf{e} \in \{0, \dots, l\} | \mathsf{ChallEq}(\mathsf{e}) = \mathsf{true}\} \\ &\quad \mathsf{true} \leftarrow \mathsf{ChallEq}(\mathsf{e}) \iff (\mathsf{e} \in \mathcal{C}) \vee (\mathsf{ChallEq}(\mathsf{e}\text{-}1) \wedge \mathsf{e} \in \mathcal{T}^*_{\mathsf{kk}}). \end{aligned} \tag{4}$$

Bi-directional ciphertext updates:

$$\begin{aligned}
&\mathcal{C}^*_{\mathsf{kk,bi}} \leftarrow \{\mathsf{e} \in \{0,\ldots,l\} | \mathsf{ChallEq}(\mathsf{e}) = \mathsf{true}\} \\
&\quad \mathsf{true} \leftarrow \mathsf{ChallEq}(\mathsf{e}) \iff \\
&\qquad (\mathsf{e} \in \mathcal{C}) \vee (\mathsf{ChallEq}(\mathsf{e\text{-}1}) \wedge \mathsf{e} \in \mathcal{T}^*_{\mathsf{kk}}) \vee (\mathsf{ChallEq}(\mathsf{e{+}1}) \wedge \mathsf{e{+}1} \in \mathcal{T}^*_{\mathsf{kk}}). \quad (5)
\end{aligned}$$

2.7 Trivial Win Conditions

The main benefit of using ciphertext-independent UE scheme is that it offers an efficient way for key rotation, where a single token can be used to update all ciphertexts. However, this property provides the adversary more power, the tokens can be used to gain more information, and gives the adversary more chances to win the security games. We again follow the trivial win analysis in [LT18, KLR19, BDGJ20] and exclude these trivial win conditions in the security games for UE. An overview of the trivial win conditions the challenger will check in each security game is given in Fig. 7 and Fig. 8.

Notions	"$\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$"	"$\tilde{\mathsf{e}} \in \mathcal{T}^*$ **or** $\mathcal{O}.\mathsf{Upd}(\bar{\mathbf{c}})$ is queried"	"$(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*$"	"$(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*$"
detIND-UE-CPA	✓	✓	×	×
randIND-UE-CPA	✓	×	×	×
detIND-UE-CCA	✓	✓	✓	×
randIND-UE-CCA	✓	×	×	✓

Figure 7: Trivial win conditions considered in different confidentiality games for updatable encryption schemes. × indicates the confidentiality notion does not consider the corresponding trivial win condition, ✓ indicates the confidentiality notion considers the corresponding trivial win condition.

Notions	"$\mathsf{e} \in \mathcal{K}^*$"	"$(\tilde{\mathbf{c}}, \mathsf{e}) \in \mathcal{L}^*$"	"$(\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*$"
INT-CTXT	✓	✓	×
INT-PTXT	✓	×	✓

Figure 8: Trivial win conditions considered in different integrity games for updatable encryption schemes. × indicates the integrity notion does not consider the corresponding trivial win condition, ✓ indicates the integrity notion considers the corresponding trivial win condition.

2.7.1 Checking Trivial Win Conditions at the End of a Game

Trivial Wins via Keys and Ciphertexts. The following is used for analyzing all confidentiality games. If there exists an epoch $\mathsf{e} \in \mathcal{K}^* \cap \mathcal{C}^*$ in which the adversary knows the epoch key $\mathbf{k}_{\mathsf{e}}$ and a valid update of the challenge ciphertext $\tilde{\mathbf{c}}_{\mathsf{e}}$, then the adversary can use this epoch key to decrypt the challenge-equal ciphertext and know the underlying challenge plaintext to win the confidentiality game. The trivial win condition "$\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$" is checked in the end of a confidentiality game.

Trivial Wins via Direct Updates. The following is used for analyzing all confidentiality games with deterministic updates. If the adversary knows the update token $\Delta_{\tilde{\mathsf{e}}}$ in the challenge epoch $\tilde{\mathsf{e}}$ or the adversary queried an update oracle on the challenge input ciphertext $\mathcal{O}.\mathsf{Upd}(\bar{\mathbf{c}})$ in epoch $\tilde{\mathsf{e}}$, then it knows the updated ciphertext of $\bar{\mathbf{c}}$ in epoch $\tilde{\mathsf{e}}$ and it can compare the updated ciphertext with the challenge ciphertext to win the confidentiality game. The trivial win condition "$\tilde{\mathsf{e}} \in \mathcal{T}^*$ or $\mathcal{O}.\mathsf{Upd}(\bar{\mathbf{c}})$ is queried" is checked in the end of a confidentiality game.

2.7.2 Checking Trivial Win Conditions while Running a Game

The following overview of trivial win conditions are checked by an oracle. The sets $\tilde{\mathcal{L}}^*, \tilde{\mathcal{Q}}^*, \mathcal{K}^*, \mathcal{L}^*$ and $\mathcal{Q}^*$ are defined in Section 2.5.

- "$(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*$" are checked by $\mathcal{O}.\mathsf{Dec}$ oracles in the detIND-UE-CCA game,
- "$(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*$" are checked by $\mathcal{O}.\mathsf{Dec}$ oracles in the randIND-UE-CCA game,

- "$\mathsf{e} \in \mathcal{K}^*$" are checked by $\mathcal{O}.\mathsf{Try}$ oracles in the $\mathsf{INT\text{-}CTXT}$ game or the $\mathsf{INT\text{-}PTXT}$ game,
- "$(\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*$" are checked by $\mathcal{O}.\mathsf{Try}$ oracles in the $\mathsf{INT\text{-}CTXT}$ game
- "$(\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*$" are checked by $\mathcal{O}.\mathsf{Try}$ oracles in the $\mathsf{INT\text{-}PTXT}$ game.

General Idea. At the moment when the adversary queries a decryption query $\mathcal{O}.\mathsf{Dec}$ or a try query $\mathcal{O}.\mathsf{Try}$, the challenger computes the knowledge the adversary <u>currently</u> has, which is used to check if the adversary can trivially win a security game. More precisely, the challenger uses information in the sets $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T}$ to compute the leakage sets $\tilde{\mathcal{L}}^*$, $\tilde{\mathcal{Q}}^*$, $\mathcal{K}^*$, $\mathcal{L}^*$ and $\mathcal{Q}^*$. Note that the sets $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T}$ contains information the adversary learns at such a moment.

Trivial Wins via Decryptions in the Deterministic Update Setting. The following is used for analyzing the $\mathsf{detIND\text{-}UE\text{-}CCA}$ security notion. In the deterministic update setting, if the adversary knows a challenge-equal ciphertext $(\tilde{\mathbf{c}}_{\mathsf{e}_0}, \mathsf{e}_0) \in \tilde{\mathcal{L}}$ and tokens from epoch $\mathsf{e}_0 + 1$ to epoch e, then the adversary can compute the updated challenge-equal ciphertext $\tilde{\mathbf{c}}_{\mathsf{e}}$ and send it to the decryption oracle to get the underlying message. Eventually, the adversary compares the received message with the challenge plaintexts to trivially win the security game.

We use the set $\tilde{\mathcal{L}}^*$ to check this trivial win condition, recall that $\tilde{\mathcal{L}}^*$ includes all challenge-equal ciphertexts the adversary has learned or inferred. Suppose the adversary queries a decryption oracle $\mathcal{O}.\mathsf{Dec}(\mathbf{c})$ in epoch e, if $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*$ then the response of the decryption oracle leads to a trivial win to the adversary, hence, the challenger will set the trivial win flag to be 1.

By Remark 2.1, we have $(\tilde{\mathbf{c}}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{L}}^* \iff \mathsf{e} \in \mathcal{C}^*$, using this method we can easily compute the set $\tilde{\mathcal{L}}^*$. In Fig. 9 we show how the set $\tilde{\mathcal{L}}^*$ is computed, where the set $\mathcal{C}^*$ is computed by the algorithms discussed in Section 2.6.

Trivial Wins via Decryptions in the Randomized Update Setting. The following is used for analyzing the $\mathsf{randIND\text{-}UE\text{-}CCA}$ security notion. In the randomized update setting, if the adversary knows a challenge-equal ciphertext $(\tilde{\mathbf{c}}_{\mathsf{e}_0}, \mathsf{e}_0) \in \tilde{\mathcal{L}}$ and tokens from epoch $\mathsf{e}_0 + 1$ to epoch e, then the adversary can create arbitrary number of ciphertexts by updating $\tilde{\mathbf{c}}_{\mathsf{e}_0}$ from epoch e_0 to epoch e. Let $\mathbf{c}_{\mathsf{e}}$ denote a ciphertext generated in such a way. Notice that the ciphertext $\mathbf{c}_{\mathsf{e}}$ has the same underlying message as the challenge-equal ciphertext $\tilde{\mathbf{c}}_{\mathsf{e}_0}$. The adversary can send the computed ciphertext $\mathbf{c}_{\mathsf{e}}$ to the decryption oracle to get the underlying message and trivially win the security game.

We use the set $\tilde{\mathcal{Q}}^*$ to check this trivial win condition, recall that $\tilde{\mathcal{Q}}^*$ includes information about challenge plaintexts that the adversary has learned or can create challenge-equal ciphertexts of. Suppose the adversary queries a decryption oracle $\mathcal{O}.\mathsf{Dec}(\mathbf{c})$ in epoch e, if $\mathsf{UE.Dec}(\mathbf{k}_{\mathsf{e}}, \mathbf{c}) = \mathbf{m}'$ and $(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*$ then the response of the decryption oracle leads to a trivial win to the adversary, hence, the challenger will set the trivial win flag to be 1.

By Remark 2.1, we have $(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^* \iff \mathsf{e} \in \mathcal{C}^*$, using this method we can easily compute the set $\tilde{\mathcal{Q}}^*$. Suppose the challenge input is $(\bar{\mathbf{m}}, \bar{\mathbf{c}})$ and the underlying message of $\bar{\mathbf{c}}$ is $\bar{\mathbf{m}}_1$. In Fig. 10 we show how the set $\tilde{\mathcal{Q}}^*$ is computed.

Remark 2.2. Our definition of this trivial win restriction is more generous than that of [KLR19], they disallow the decryption of any ciphertext that decrypts to either of the two challenge plaintexts. We allow the decryption of a ciphertext that decrypts to a challenge plaintext as long as the adversary cannot learn (from $\mathcal{O}.\mathsf{Chall}$ or $\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$) or infer (from tokens) a valid ciphertext of challenge plaintext in that epoch.

```
for $i \in \{0, ..., \mathsf{e}\}$ do
  if $i \in \mathcal{C}^*_{\mathsf{kk,cc}}$ then
    $\tilde{\mathcal{L}}^*_{\mathsf{kk,cc}} \leftarrow \tilde{\mathcal{L}}^*_{\mathsf{kk,cc}} \cup \{(\tilde{\mathbf{c}}_i, i)\}$
```

Figure 9: Algorithm for computing the set $\tilde{\mathcal{L}}^*_{\mathsf{kk,cc}}$, where $\mathsf{kk} \in \{\mathsf{no, uni, bi}\}$ and $\mathsf{cc} \in \{\mathsf{uni, bi}\}$.

```
for $i \in \{0, ..., \mathsf{e}\}$ do
  if $i \in \mathcal{C}^*_{\mathsf{kk,cc}}$ then
    $\tilde{\mathcal{Q}}^*_{\mathsf{kk,cc}} \leftarrow \tilde{\mathcal{Q}}^*_{\mathsf{kk,cc}} \cup \{(\bar{\mathbf{m}}, i)\} \cup \{(\bar{\mathbf{m}}_1, i)\}$
```

Figure 10: Algorithm for computing the set $\tilde{\mathcal{Q}}^*_{\mathsf{kk,cc}}$, where $\mathsf{kk} \in \{\mathsf{no, uni, bi}\}$ and $\mathsf{cc} \in \{\mathsf{uni, bi}\}$.

Trivial Forgeries by Keys. The following is used for analyzing all integrity games. If the adversary knows an epoch key $\mathbf{k}_{\mathsf{e}}$, then the adversary can create arbitrary number of valid forgeries of arbitrary messages under this epoch key $\mathbf{k}_{\mathsf{e}}$.

We use the set $\mathcal{K}^*$ to check this trivial win condition, recall that $\mathcal{K}^*$ includes all epochs the adversary learned or inferred an epoch key. Suppose the adversary queries a try oracle $\mathcal{O}.\mathsf{Try}(\mathbf{c})$ in epoch e, if $\mathsf{e} \in \mathcal{K}^*$ then the challenger will set the trivial win flag to be 1. We use algorithms discussed in Section 2.6 to compute the set $\mathcal{K}^*$.

Trivial Ciphertext Forgeries by Tokens. The following is used for analyzing the INT-CTXT security notion. From [KLR19] we know that only UE schemes with deterministic updates can possibly achieve INT-CTXT security. In the deterministic update setting, if the adversary knows a ciphertext $(\mathrm{c}, \mathbf{c}, \mathsf{e}_0; \mathbf{m}) \in \mathcal{L}$ and tokens from epoch $\mathsf{e}_0 + 1$ to epoch e, then the adversary can create a valid updated ciphertext by updating $\mathbf{c}$ from epoch e_0 to epoch e.

We use the set $\mathcal{L}^*$ to check this trivial win condition, recall that $\mathcal{L}^*$ includes all ciphertexts that can be known or inferred to the adversary. Suppose the adversary queries a try oracle $\mathcal{O}.\mathsf{Try}(\mathbf{c})$ in epoch e, if $(\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*$ then the challenger will set the trivial win flag to be 1. In Fig. 11 we show how the set $\mathcal{L}^*$ is computed.

```
**for** $i \in \{0, \ldots, \mathsf{e}\}$ **do**
  **for** $(\cdot, \mathbf{c}, i; \cdot) \in \mathcal{L}$ **do**
    $\mathcal{L}^*_{\mathsf{kk,cc}} \leftarrow \mathcal{L}^*_{\mathsf{kk,cc}} \cup \{(\mathbf{c}, i)\}$
  **if** $i \in \mathcal{T}^*_{\mathsf{kk}}$ **then**
    **for** $(\mathbf{c}_{i-1}, i-1) \in \mathcal{L}^*_{\mathsf{kk,cc}}$ **do**
      $\mathbf{c}_i \leftarrow \mathsf{UE.Upd}(\Delta_i, \mathbf{c}_{i-1})$
      $\mathcal{L}^*_{\mathsf{kk,cc}} \leftarrow \mathcal{L}^*_{\mathsf{kk,cc}} \cup \{(\mathbf{c}_i, i)\}$
    **if** $\mathsf{cc} = \mathsf{bi}$ **then**
      **for** $(\mathbf{c}_i, i) \in \mathcal{L}^*_{\mathsf{kk,cc}}$ **do**
        $\mathbf{c}_{i-1} \leftarrow \mathsf{UE.Upd}^{-1}(\Delta_i, \mathbf{c}_i)$
        $\mathcal{L}^*_{\mathsf{kk,cc}} \leftarrow \mathcal{L}^*_{\mathsf{kk,cc}} \cup \{(\mathbf{c}_{i-1}, i-1)\}$
```

Figure 11: Algorithm for computing the set $\mathcal{L}^*_{\mathsf{kk,cc}}$, where $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$ and $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$.

```
**for** $i \in \{0, \ldots, \mathsf{e}\}$ **do**
  **for** $(\cdot, \cdot, i; \mathbf{m}) \in \mathcal{L}$ **do**
    $\mathcal{Q}^*_{\mathsf{kk,cc}} \leftarrow \mathcal{Q}^*_{\mathsf{kk,cc}} \cup \{(\mathbf{m}, i)\}$
  **if** $i \in \mathcal{T}^*_{\mathsf{kk}}$ **then**
    **for** $(\mathbf{m}, i-1) \in \mathcal{Q}^*_{\mathsf{kk,cc}}$ **do**
      $\mathcal{Q}^*_{\mathsf{kk,cc}} \leftarrow \mathcal{Q}^*_{\mathsf{kk,cc}} \cup \{(\mathbf{m}, i)\}$
    **if** $\mathsf{cc} = \mathsf{bi}$ **then**
      **for** $(\mathbf{m}, i) \in \mathcal{Q}^*_{\mathsf{kk,cc}}$ **do**
        $\mathcal{Q}^*_{\mathsf{kk,cc}} \leftarrow \mathcal{Q}^*_{\mathsf{kk,cc}} \cup \{(\mathbf{m}, i-1)\}$
```

Figure 12: Algorithm for computing the set $\mathcal{Q}^*_{\mathsf{kk,cc}}$, where $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$ and $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$.

Trivial Plaintext Forgeries by Tokens. The following is used for analyzing the $\mathsf{INT\text{-}PTXT}$ security notion. In the randomized update setting, if the adversary knows a ciphertext $(\mathrm{c}, \mathbf{c}, \mathsf{e}_0; \mathbf{m}) \in \mathcal{L}$ and tokens from epoch $\mathsf{e}_0 + 1$ to epoch e, then the adversary can create arbitrary number of valid forgeries of message $\mathbf{m}$ by updating $\mathbf{c}$ from epoch e_0 to epoch e.

We use the set $\mathcal{Q}^*$ to check this trivial win condition, recall that $\mathcal{Q}^*$ includes information about plaintexts that the adversary has learned or can create ciphertexts of. Suppose the adversary queries a try oracle $\mathcal{O}.\mathsf{Try}(\mathbf{c})$ in epoch e, if $\mathsf{UE.Dec}(\mathbf{k}_\mathsf{e}, \mathbf{c}) = \mathbf{m}'$ and $(\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*$ then the challenger will set the trivial win flag to be 1. In Fig. 12 we show how the set $\mathcal{Q}^*$ is computed.

3 Six Variants of Security Notions

In this section we first define six variants of security notions for updatable encryption schemes. In the end of this section, we compare the relationship among all these variants of each security notion.

For $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$ and $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$, we define $(\mathsf{kk}, \mathsf{cc})$- variants of security notions, where kk refers to UE schemes with kk-directional key updates and cc to cc-directional ciphertext updates.

Definition 7 (The $(\mathsf{kk}, \mathsf{cc})$- variant of confidentiality notions)**.** Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. Then the $(\mathsf{kk}, \mathsf{cc})$-notion advantage, for notion $\in \{\mathsf{detIND\text{-}UE\text{-}CPA}, \mathsf{randIND\text{-}UE\text{-}CPA}, \mathsf{detIND\text{-}UE\text{-}CCA}, \mathsf{randIND\text{-}UE\text{-}CCA}\}$, $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$ and $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$, of an adversary $\mathcal{A}$ against UE is defined as

$$\mathbf{Adv}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}}_{\mathsf{UE},\,\mathcal{A}}(1^\lambda) = \left|\mathbf{Pr}[\mathbf{Exp}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}\text{-}1}_{\mathsf{UE},\,\mathcal{A}} = 1] - \mathbf{Pr}[\mathbf{Exp}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}\text{-}0}_{\mathsf{UE},\,\mathcal{A}} = 1]\right|,$$

where the experiment $\mathbf{Exp}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}\text{-}\mathsf{b}}_{\mathsf{UE},\,\mathcal{A}}$ is the same as the experiment $\mathbf{Exp}^{\mathsf{notion}\text{-}\mathsf{b}}_{\mathsf{UE},\,\mathcal{A}}$ (see Fig. 4 and Fig. 6) except for all leakage sets are both in the kk-directional key update setting and cc-directional ciphertext update setting.

Remark 3.1. Recall that we compute all leakage sets with kk-directional key updates and cc-directional ciphertext updates in Section 2.6 and Section 2.7.

Remark 3.2. Note that the security notion RCCA, which we denote as randIND-UE-CCA, is from [KLR19]. In our definition of this notion is stronger - the adversary has fewer trivial win restrictions - we discuss this difference in Remark 2.2.

Definition 8 (The $(\mathsf{kk}, \mathsf{cc})$- variant of integrity notions)**.** Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. Then the $(\mathsf{kk}, \mathsf{cc})$-notion advantage, for $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$, $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$ and notion $\in \{\mathsf{INT\text{-}CTXT}, \mathsf{INT\text{-}PTXT}\}$, of an adversary $\mathcal{A}$ against UE is defined as

$$\mathbf{Adv}_{\mathsf{UE},\ \mathcal{A}}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}}(1^\lambda) = \mathbf{Pr}[\mathbf{Exp}_{\mathsf{UE},\ \mathcal{A}}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}} = 1],$$

where the experiment $\mathbf{Exp}_{\mathsf{UE},\ \mathcal{A}}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}}$ is the same as $\mathbf{Exp}_{\mathsf{UE},\ \mathcal{A}}^{\mathsf{notion}}$ (see Fig. 5 and Fig. 6) except for all leakage sets are both in the kk-directional key update setting and cc-directional ciphertext update setting.

3.1 Properties of Leakage Sets and Trivial Win Conditions

In this section, we prove some essential properties of key leakage, which will be used to analyze the trivial win conditions. We will use these trivial win properties to prove the relations among six variants of the same security notion in Section 3.2.

3.1.1 Properties of Key Updates

In this section, we look at some properties of sets $\mathcal{K}, \mathcal{T}, \mathcal{K}^*$ and $\mathcal{T}^*$ in terms of uni- and bi-directional key updates.

Firewall and Insulated Region. We first describe the definition of firewall and insulated region, which will be widely used in this paper. Firewall technique (see [LT18, KLR19, BDGJ20]) is used for doing cryptographic seperation. We follow the firewall definition in [BDGJ20] and use firewall set $\mathcal{FW}$ (defined in [BDGJ20]) to track each insulated region and its firewalls.

Definition 9. An *insulated region* with *firewalls* fwl and fwr is a consecutive sequence of epochs $(\mathsf{fwl}, \ldots, \mathsf{fwr})$ for which:

- no key in the sequence of epochs $(\mathsf{fwl}, \ldots, \mathsf{fwr})$ is corrupted, i.e. $\{\mathsf{fwl}, \ldots, \mathsf{fwr}\} \cap \mathcal{K} = \emptyset$;
- the tokens Δ_{fwl} and $\Delta_{\mathsf{fwr}+1}$ are not corrupted (if they exist), i.e. fwl, $\mathsf{fwr} + 1 \notin \mathcal{T}$;
- all tokens $(\Delta_{\mathsf{fwl}+1}, \ldots, \Delta_{\mathsf{fwr}})$ are corrupted, i.e.$\{\mathsf{fwl}+1, \ldots, \mathsf{fwr}\} \subseteq \mathcal{T}$.

Remark 3.3. Based on Definition 9, we notice that all firewalls or all insulated regions (in other words, set $\mathcal{FW}$) are uniquely determined by $\mathcal{K}$ and $\mathcal{T}$. In particular, we denote the union of all insulated regions as set $\mathcal{IR}$, i.e. $\mathcal{IR} = \cup_{(\mathsf{fwl},\mathsf{fwr}) \in \mathcal{FW}} \{\mathsf{fwl}, \ldots, \mathsf{fwr}\}$.

Then we look at the structure of the set $\mathcal{IR}$. Lemma 3.1 states that $\mathcal{IR}$ is the complementary set of $\mathcal{K}^*_{\mathsf{bi}}$. Furthermore, Lemma 3.3 shows that the complementary set of $\mathcal{IR}$ is the union of two types of epoch sets (see Definition 10 and Definition 11).

Lemma 3.1. For any sets $\mathcal{K}, \mathcal{T} \subseteq \{0, \ldots, l\}$, we have $\mathcal{K}^*_{\mathsf{bi}} = \{0, \ldots, l\} \backslash \mathcal{IR}$.

Proof. Note that Δ_0 and Δ_{l+1} do not exist, however, 0 and l can possibly be firewalls. For convenience, we just assume Δ_0 and Δ_{l+1} exist and the adversary is not allowed to corrupt these two tokens. Thus the set of epochs in which the adversary never corrupted the update token is: $\{0, \ldots, l+1\} \setminus \mathcal{T} = \{\bar{\mathsf{e}}_0 := 0, \bar{\mathsf{e}}_1, \ldots, \bar{\mathsf{e}}_t, \bar{\mathsf{e}}_{t+1} := l+1\}$, where $t \geq 0$.

In the bi-directional key update setting, if the adversary has corrupted a key in an epoch e, where $\mathsf{e} \in \{\bar{\mathsf{e}}_{i-1}, \ldots, \bar{\mathsf{e}}_i - 1\}$, then the adversary can infer all keys from epoch $\bar{\mathsf{e}}_{i-1}$ to epoch $\bar{\mathsf{e}}_i - 1$, that is $\{\bar{\mathsf{e}}_{i-1}, \ldots, \bar{\mathsf{e}}_i - 1\} \subseteq \mathcal{K}^*_{\mathsf{bi}}$, because all tokens from epoch $\bar{\mathsf{e}}_{i-1} + 1$ to epoch $\bar{\mathsf{e}}_i - 1$ are corrupted. Otherwise, when no key in the sequence of epochs $\{\bar{\mathsf{e}}_{i-1}, \ldots, \bar{\mathsf{e}}_i - 1\}$ is corrupted, then $\{\bar{\mathsf{e}}_{i-1}, \ldots, \bar{\mathsf{e}}_i - 1\}$ is an insulated region . Therefore, for any i, $\{\bar{\mathsf{e}}_{i-1}, \ldots, \bar{\mathsf{e}}_i - 1\}$ is either an insulated region or a subset of $\mathcal{K}^*_{\mathsf{bi}}$. $\square$

Epoch		{0}		1		2		3		{4		5}		6		7		8	
$\mathcal{K}$		×		×		✓		×		×		×		✓		✓		×	
$\mathcal{T}$	×		×		✓		✓		×		✓		×		×		✓		×
$\mathcal{K}^*_{\mathsf{bi}}$		×		✓		✓		✓		×		×		✓		✓		✓	
$\mathcal{T}^*_{\mathsf{bi}}$	×		×		✓		✓		×		✓		×		✓		✓		×

Figure 13: An example of Lemma 3.1, where $\mathcal{K} = \{2, 6, 7\}$ and $\mathcal{T} = \{2, 3, 5, 8\}$. So $\{0, ..., l + 1\} \setminus \mathcal{T} = \{0, 1, 4, 6, 7, 9\}$, insulated regions are $\{0\}$ and $\{4, 5\}$. $\mathcal{IR} = \{0, 4, 5\}$ and $\mathcal{K}^*_{\mathsf{bi}} = \{1, 2, 3, 6, 7, 8\}$. $\times$ indicates the keys/tokens are not revealed to the adversary, ✓ indicates the keys/tokens are revealed to the adversary.

We define two types of epoch sets in Definition 10 and Definition 11, which will later be used to analyze the structure of $\mathcal{IR}$. An overview of the corruption model of these two epoch sets are shown in Fig. 14.

Definition 10. A set of *type1* epochs is a consecutive sequence of epochs $(\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}})$ for which:

- no key in the sequence of epochs $\{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}} - 1\}$ is corrupted, i.e.$\{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}} - 1\} \cap \mathcal{K} = \emptyset$;
- the key in epoch $\mathsf{e}_{\mathsf{end}}$ is corrupted, i.e.$\mathsf{e}_{\mathsf{end}} \in \mathcal{K}$;
- all tokens $\{\Delta_{\mathsf{e}_{\mathsf{start}}+1}, \ldots, \Delta_{\mathsf{e}_{\mathsf{end}}}\}$ are corrupted, i.e. $\{\mathsf{e}_{\mathsf{start}} + 1, \ldots, \mathsf{e}_{\mathsf{end}}\} \subseteq \mathcal{T}$.

Definition 11. A set of *type2* epochs is a consecutive sequence of epochs $(\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}})$ for which:

- $\{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}}\} \subseteq \mathcal{K}^*_{\mathsf{uni}}$;
- $\{\mathsf{e}_{\mathsf{start}} + 1, \ldots, \mathsf{e}_{\mathsf{end}}\} \subseteq \mathcal{T}^*_{\mathsf{uni}}$.

The following lemma demonstrates that if a key is revealed in the bi-directional key update setting but not in the uni-directional key update setting then the revealed key epoch can stretch to a type 1 epoch set. We use this property to prove Lemma 3.3.

Epoch	$\mathsf{e}_{\mathsf{start}}$		$\mathsf{e}_{\mathsf{start}}+1$		$\ldots$		$\mathsf{e}_{\mathsf{end}-1}$		$\mathsf{e}_{\mathsf{end}}$
$\mathcal{K}$	$\times$		$\times$		$\ldots$		$\times$		$\checkmark$
$\mathcal{T}$		$\checkmark$		$\checkmark$	$\ldots$	$\checkmark$		$\checkmark$	

(a) Type 1 set of epochs

Epoch	$\mathsf{e}_{\mathsf{start}}$		$\mathsf{e}_{\mathsf{start}}+1$		$\ldots$		$\mathsf{e}_{\mathsf{end}}$
$\mathcal{K}^*_{\mathsf{uni}}$	$\checkmark$		$\checkmark$		$\ldots$		$\checkmark$
$\mathcal{T}^*_{\mathsf{uni}}$		$\checkmark$		$\checkmark$	$\ldots$	$\checkmark$	

(b) Type 2 set of epochs

Figure 14: $\times$ indicates the keys/tokens are not revealed to the adversary, $\checkmark$ indicates the keys/tokens are revealed to the adversary.

Lemma 3.2. If $\mathsf{e} \in \mathcal{K}^*_{\mathsf{bi}} \setminus \mathcal{K}^*_{\mathsf{uni}}$, then there exists an epoch (say e_u) after e such that $\mathsf{e}_u \in \mathcal{K}$, $\{\mathsf{e}, \ldots, \mathsf{e}_u - 1\} \cap \mathcal{K} = \emptyset$ and $\{\mathsf{e}+1, \ldots, \mathsf{e}_u\} \subseteq \mathcal{T}$.

Proof. As the assumption and Equations (1, 2), we have $\mathsf{e} \in \mathcal{K}^*_{\mathsf{bi}}$ is inferred from the next epoch key $\mathbf{k}_{\mathsf{e}+1}$ via token $\Delta_{\mathsf{e}+1}$. That is $\mathsf{e}+1 \in \mathcal{K}^*_{\mathsf{bi}}$ and $\mathsf{e}+1 \in \mathcal{T}$. If $\mathsf{e}+1 \notin \mathcal{K}^*_{\mathsf{uni}}$, then $\mathsf{e}+2 \in \mathcal{K}^*_{\mathsf{bi}}$ and $\mathsf{e}+2 \in \mathcal{T}$. Iteratively, we know that there exists an epoch after e, say e_u, such that $\{\mathsf{e}, \ldots, \mathsf{e}_u - 1\} \cap \mathcal{K}^*_{\mathsf{uni}} = \emptyset$, $\mathsf{e}_u \in \mathcal{K}^*_{\mathsf{uni}}$ and $\mathsf{e}+1, \ldots, \mathsf{e}_u \in \mathcal{T}$. Hence, $\{\mathsf{e}, \ldots, \mathsf{e}_u - 1\} \cap \mathcal{K} \subseteq \{\mathsf{e}, \ldots, \mathsf{e}_u - 1\} \cap \mathcal{K}^*_{\mathsf{uni}} = \emptyset$. In particular, we know that $\mathsf{e}_u \in \mathcal{K}$ since $\mathsf{e}_u - 1 \notin \mathcal{K}^*_{\mathsf{uni}}$. $\square$

Lemma 3.3. For any sets $\mathcal{K}, \mathcal{T} \subseteq \{0, \ldots, l\}$, we have $\{0, \ldots, l\} \setminus \mathcal{IR} = (\cup_{\text{type } 1}\{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}}\}) \cup (\cup_{\text{type } 2}\{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}}\})$, where the two types of epoch sets are defined in Definition 10 and Definition 11.

Proof. Suppose $\mathsf{e} \in \{0, \ldots, l\} \setminus \mathcal{IR}$, by Lemma 3.1, we have $\mathsf{e} \in \mathcal{K}^*_{\mathsf{bi}}$. If $\mathsf{e} \notin \mathcal{K}^*_{\mathsf{uni}}$, we can apply Lemma 3.2 and have a set of type 1 epochs, assume $\{\mathsf{e}, \ldots, \mathsf{e}_u\}$. For all $\mathsf{e} \in \mathcal{K}^*_{\mathsf{bi}} \setminus \mathcal{K}^*_{\mathsf{uni}}$, we can find a set of type 1 epochs. Hence, the rest epochs are in the type 2 epoch sets. $\square$

Remark 3.4. As a conclusion of Lemma 3.1 and Lemma 3.3, we have the sequence of all epochs are a union of three types of epoch sets, that are insulated regions, type 1 epochs and type 2 epochs. $\{0, \ldots, l\} = (\cup_{(\mathsf{fwl}, \mathsf{fwr}) \in \mathcal{FW}} \{\mathsf{fwl}, \ldots, \mathsf{fwr}\}) \cup (\cup_{\text{type } 1}\{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}}\}) \cup (\cup_{\text{type } 2}\{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}}\})$.

3.1.2 Trivial Win Equivalences in the Uni- and Bi-Directional Update Setting

In this section we prove seven equivalences of the trivial win conditions. As a result, we have that in any security game if the trivial win conditions in the bi-directional update setting are triggered then the same trivial win conditions in the uni-directional update setting would be triggered. We will use these trivial win equivalences to prove the relation between uni- and bi-directional variants of security notions in Theorem 3.1.

The following two lemmas show that UE schemes with uni-directional updates has less leakage than UE schemes with bi-directional updates.

Lemma 3.4. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C}$ and any $\mathsf{kk} \in \{\mathsf{uni}, \mathsf{bi}\}$, we have $\mathcal{C}^*_{\mathsf{kk},\mathsf{uni}} \subseteq \mathcal{C}^*_{\mathsf{kk},\mathsf{bi}}$, $\tilde{\mathcal{L}}^*_{\mathsf{kk},\mathsf{uni}} \subseteq \tilde{\mathcal{L}}^*_{\mathsf{kk},\mathsf{bi}}$, $\tilde{\mathcal{Q}}^*_{\mathsf{kk},\mathsf{uni}} \subseteq \tilde{\mathcal{Q}}^*_{\mathsf{kk},\mathsf{bi}}$, $\mathcal{L}^*_{\mathsf{kk},\mathsf{uni}} \subseteq \mathcal{L}^*_{\mathsf{kk},\mathsf{bi}}$, and $\mathcal{Q}^*_{\mathsf{kk},\mathsf{uni}} \subseteq \mathcal{Q}^*_{\mathsf{kk},\mathsf{bi}}$.

Proof. For any fixed kk-directional key updates, uni-directional ciphertext updates has less leakage than bi-directional ciphertext updates. More precisely, for any $\mathcal{K}, \mathcal{T}, \mathcal{C}$ and a fixed kk, we compute $\mathcal{K}^*_{\mathsf{kk}}$, $\mathcal{T}^*_{\mathsf{kk}}$, $\mathcal{C}^*_{\mathsf{kk},\mathsf{uni}}$ and $\mathcal{C}^*_{\mathsf{kk},\mathsf{bi}}$ using Equations (1, 2, 3, 4, 5). Then we have $\mathcal{C}^*_{\mathsf{kk},\mathsf{uni}} \subseteq \mathcal{C}^*_{\mathsf{kk},\mathsf{bi}}$. Furthermore, we use algorithms discussed in Section 2.7.2 to compute ciphertext/message leakage sets $\tilde{\mathcal{L}}^*$, $\tilde{\mathcal{Q}}^*$, $\mathcal{L}^*$, $\mathcal{Q}^*$. Similarly we get $\tilde{\mathcal{L}}^*_{\mathsf{kk},\mathsf{uni}} \subseteq \tilde{\mathcal{L}}^*_{\mathsf{kk},\mathsf{bi}}$, $\tilde{\mathcal{Q}}^*_{\mathsf{kk},\mathsf{uni}} \subseteq \tilde{\mathcal{Q}}^*_{\mathsf{kk},\mathsf{bi}}$, $\mathcal{L}^*_{\mathsf{kk},\mathsf{uni}} \subseteq \mathcal{L}^*_{\mathsf{kk},\mathsf{bi}}$, and $\mathcal{Q}^*_{\mathsf{kk},\mathsf{uni}} \subseteq \mathcal{Q}^*_{\mathsf{kk},\mathsf{bi}}$. □

Lemma 3.5. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C}$ and any $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$, we have $\mathcal{K}^*_{\mathsf{uni}} \subseteq \mathcal{K}^*_{\mathsf{bi}}$, $\mathcal{T}^*_{\mathsf{uni}} \subseteq \mathcal{T}^*_{\mathsf{bi}}$, $\mathcal{C}^*_{\mathsf{uni},\mathsf{cc}} \subseteq \mathcal{C}^*_{\mathsf{bi},\mathsf{cc}}$, $\tilde{\mathcal{L}}^*_{\mathsf{uni},\mathsf{cc}} \subseteq \tilde{\mathcal{L}}^*_{\mathsf{bi},\mathsf{cc}}$, $\tilde{\mathcal{Q}}^*_{\mathsf{uni},\mathsf{cc}} \subseteq \tilde{\mathcal{Q}}^*_{\mathsf{bi},\mathsf{cc}}$, $\mathcal{L}^*_{\mathsf{uni},\mathsf{cc}} \subseteq \mathcal{L}^*_{\mathsf{bi},\mathsf{cc}}$ and $\mathcal{Q}^*_{\mathsf{uni},\mathsf{cc}} \subseteq \mathcal{Q}^*_{\mathsf{bi},\mathsf{cc}}$.

Proof. The proof is similar to the proof of Lemma 3.4. For any fixed cc-directional ciphertext updates, uni-directional key updates has less leakage than bi-directional key updates. More precisely, for any $\mathcal{K}, \mathcal{T}, \mathcal{C}$ and a fixed cc, we compute $\mathcal{K}^*_{\mathsf{uni}}$, $\mathcal{K}^*_{\mathsf{bi}}$, $\mathcal{T}^*_{\mathsf{uni}}$, $\mathcal{T}^*_{\mathsf{bi}}$, $\mathcal{C}^*_{\mathsf{uni},\mathsf{cc}}$ and $\mathcal{C}^*_{\mathsf{bi},\mathsf{cc}}$ using Equations (1, 2, 3, 4, 5). Then we have $\mathcal{K}^*_{\mathsf{uni}} \subseteq \mathcal{K}^*_{\mathsf{bi}}$, $\mathcal{T}^*_{\mathsf{uni}} \subseteq \mathcal{T}^*_{\mathsf{bi}}$, and therefore $\mathcal{C}^*_{\mathsf{uni},\mathsf{cc}} \subseteq \mathcal{C}^*_{\mathsf{bi},\mathsf{cc}}$. Furthermore, we use algorithms discussed in Section 2.7.2 to compute ciphertext/message leakage sets $\tilde{\mathcal{L}}^*$, $\tilde{\mathcal{Q}}^*$, $\mathcal{L}^*$, $\mathcal{Q}^*$. Similarly we get $\tilde{\mathcal{L}}^*_{\mathsf{uni},\mathsf{cc}} \subseteq \tilde{\mathcal{L}}^*_{\mathsf{bi},\mathsf{cc}}$, $\tilde{\mathcal{Q}}^*_{\mathsf{uni},\mathsf{cc}} \subseteq \tilde{\mathcal{Q}}^*_{\mathsf{bi},\mathsf{cc}}$, $\mathcal{L}^*_{\mathsf{uni},\mathsf{cc}} \subseteq \mathcal{L}^*_{\mathsf{bi},\mathsf{cc}}$ and $\mathcal{Q}^*_{\mathsf{uni},\mathsf{cc}} \subseteq \mathcal{Q}^*_{\mathsf{bi},\mathsf{cc}}$. □

Equivalence for Trivial Win Condition "$\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$".

Lemma 3.6. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \ldots, l\}$, we have $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,uni}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} \neq \emptyset$.

Proof. For any $\mathcal{K}, \mathcal{T}, \mathcal{C}$, we compute $\mathcal{K}^*_{\mathsf{uni}}, \mathcal{C}^*_{\mathsf{uni,uni}}, \mathcal{K}^*_{\mathsf{bi}}$ and $\mathcal{C}^*_{\mathsf{bi,bi}}$ using Equations (1, 2, 4, 5).

Note that $\mathcal{K}^*_{\mathsf{uni}} \subseteq \mathcal{K}^*_{\mathsf{bi}}$ and $\mathcal{C}^*_{\mathsf{uni,uni}} \subseteq \mathcal{C}^*_{\mathsf{bi,bi}}$, so $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,uni}} \subseteq \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}}$. It suffices to prove

$$\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} \neq \emptyset \Rightarrow \mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,uni}} \neq \emptyset.$$

Suppose $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} \neq \emptyset$. We know that firewalls provide cryptographic separation, which make sure insulated regions are isolated from other insulated regions and the complementary set of all insulated regions. If the adversary never asks for any challenge-equal ciphertext in an epoch in the set $\{0, \ldots, l\} \setminus \mathcal{IR}$, then the adversary cannot infer any challenge-equal ciphertext in this set even in the bi-directional update setting. That is, $\mathcal{C}^*_{\mathsf{bi,bi}} \cap (\{0, \ldots, l\} \setminus \mathcal{IR}) = \emptyset$. However, $\{0, \ldots, l\} \setminus \mathcal{IR} \overset{\text{Lemma 3.1}}{=} \mathcal{K}^*_{\mathsf{bi}}$, then $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \emptyset$, which contradicts with the assumption. Therefore, there exists an epoch $\mathsf{e}' \in \{0, \ldots, l\} \setminus \mathcal{IR}$ such that the adversary has asked for a challenge-equal ciphertext in this epoch, that is $\mathsf{e}' \in \mathcal{C}$.

By Lemma 3.3, we know that e' is located in an epoch set which is either type 1 or type 2. Suppose $\mathsf{e}' \in \{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}}\}$, we know that the epoch key $\mathbf{k}_{\mathsf{e}_{\mathsf{end}}}$ is known to the adversary even in the uni-directional key update setting, i.e. $\mathsf{e}_{\mathsf{end}} \in \mathcal{K}^*_{\mathsf{uni}}$. Furthermore, all tokens $\Delta_{\mathsf{e}'+1}, \ldots, \Delta_{\mathsf{e}_{\mathsf{end}}}$ are known to the adversary even in the uni-directional key update setting. Hence, the adversary can update the challenge-equal ciphertext $\tilde{\mathbf{c}}_{\mathsf{e}'}$ from epoch e' to epoch $\mathsf{e}_{\mathsf{end}}$ to know $\tilde{\mathbf{c}}_{\mathsf{e}_{\mathsf{end}}}$. Which means $\mathsf{e}_{\mathsf{end}} \in \mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,uni}}$, we have $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,uni}} \neq \emptyset$. $\square$

As a corollary of Lemma 3.4 to 3.6, we have the following equivalence. We only provide Corollary 3.1 with a fully detailed proof, since we will use similar proof techniques for Corollary 3.2 to 3.5.

Corollary 3.1. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \ldots, l\}$, we have $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,uni}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,bi}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,uni}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} \neq \emptyset$.

Proof. By Lemma 3.4, we have $\mathcal{C}^*_{\mathsf{uni,uni}} \subseteq \mathcal{C}^*_{\mathsf{uni,bi}}$. By Lemma 3.5, we have $\mathcal{C}^*_{\mathsf{uni,bi}} \subseteq \mathcal{C}^*_{\mathsf{bi,bi}}$. Hence, $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,uni}} \subseteq \mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,bi}} \subseteq \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}}$. By Lemma 3.6, we have $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,uni}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,bi}} \neq \emptyset$.

Similarly, we have $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,uni}} \overset{\text{Lemma 3.5}}{\subseteq} \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,uni}} \overset{\text{Lemma 3.4}}{\subseteq} \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}}$ and therefore $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,uni}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,uni}} \neq \emptyset$. $\square$

Remark 3.5. If the trivial win condition "$\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$" is never triggered in the uni- or bi-directional update setting, then by Corollary 3.1 we have $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \emptyset$. By Lemma 3.1, we have $\{0, \ldots, l\} \setminus \mathcal{K}^*_{\mathsf{bi}} = \mathcal{IR}$. Therefore, $\mathcal{C}^*_{\mathsf{uni,uni}} \subseteq \mathcal{C}^*_{\mathsf{bi,bi}} \subseteq \{0, \ldots, l\} \setminus \mathcal{K}^*_{\mathsf{bi}} = \mathcal{IR}$. The relationship among the sets $\mathcal{C}^*_{\mathsf{uni,uni}}, \mathcal{C}^*_{\mathsf{bi,bi}}, \mathcal{IR}, \mathcal{K}^*_{\mathsf{uni}}, \mathcal{K}^*_{\mathsf{bi}}$ is shown in Fig. 15.

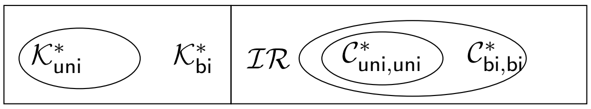
$\mathcal{K}^*_{\mathsf{uni}}$
$\mathcal{K}^*_{\mathsf{bi}}$
$\mathcal{IR}$
$\mathcal{C}^*_{\mathsf{uni,uni}}$
$\mathcal{C}^*_{\mathsf{bi,bi}}$

Figure 15: The relationship among the sets $\mathcal{C}^*_{\mathsf{uni,uni}}, \mathcal{C}^*_{\mathsf{bi,bi}}, \mathcal{IR}, \mathcal{K}^*_{\mathsf{uni}}, \mathcal{K}^*_{\mathsf{bi}}$ if the trivial win condition "$\mathcal{K}^*_{\mathsf{kk}} \cap \mathcal{C}^*_{\mathsf{kk,cc}} \neq \emptyset$" is never triggered for any $\mathsf{kk}, \mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$.

Equivalence for Trivial Win Condition "$\tilde{\mathsf{e}} \in \mathcal{T}^*$ or $\mathcal{O}.\mathsf{Upd}(\bar{\mathsf{c}})$ is queried". The event "$\mathcal{O}.\mathsf{Upd}(\bar{\mathsf{c}})$ is queried" is independent of the key and ciphertext updates, so this trivial win condition is either triggered or not triggered in all variants of a security notion. The following Lemma shows that if the challenge token is known to the adversary in the bi-directional key update setting, then it is also known to the adversary in the uni-directional key update setting.

Lemma 3.7. For any $\mathcal{K}, \mathcal{T}, \mathcal{C}$. Suppose $\mathcal{K}^*_{\mathsf{kk}} \cap \mathcal{C}^*_{\mathsf{kk,cc}} = \emptyset$, where $\mathsf{kk}, \mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$, then $\tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{no}} \iff \tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{uni}} \iff \tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{bi}}$

Proof. We know that the challenge epoch $\tilde{\mathsf{e}} \in \mathcal{C}$, so $\tilde{\mathsf{e}} \notin \mathcal{K}^*_{\mathsf{kk}}$ for any kk-key updates, where $\mathsf{kk} \in \{\mathsf{uni}, \mathsf{bi}\}$. Since the adversary does not know the key $\mathsf{k}_{\tilde{\mathsf{e}}}$, which is needed to infer the update token $\Delta_{\tilde{\mathsf{e}}}$, so token $\Delta_{\tilde{\mathsf{e}}}$ cannot be inferred by the adversary. Therefore, $\tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{kk}}$ if and only if $\tilde{\mathsf{e}} \in \mathcal{T}$. Hence $\tilde{\mathsf{e}} \in \mathcal{T} \iff \tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{no}} \iff \tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{uni}} \iff \tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{bi}}$. $\square$

From now on until the end of this section, we assume the adversary queries a decryption oracle $\mathcal{O}.\mathsf{Dec}(\mathbf{c})$ or a try oracle $\mathcal{O}.\mathsf{Try}(\mathbf{c})$ in $\underline{\text{epoch } \mathsf{e}}$. We consider trivial win conditions which are checked in these oracles.

Equivalence for Trivial Win Condition " $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*$**".**

Lemma 3.8. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \dots, \mathsf{e}\}$. Suppose $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \emptyset$, then $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{uni,uni}} \iff (\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{bi,bi}}$.

Proof. By Remark 3.5 we have $\mathcal{C}^*_{\mathsf{uni,uni}} \subseteq \mathcal{C}^*_{\mathsf{bi,bi}} \subseteq \mathcal{IR}$. By Remark 2.1 we have $(\tilde{\mathbf{c}}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{L}}^* \iff \mathsf{e} \in \mathcal{C}^*$. Therefore, if $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{uni,uni}}$ we have $\mathsf{e} \in \mathcal{C}^*_{\mathsf{uni,uni}} \subseteq \mathcal{C}^*_{\mathsf{bi,bi}}$ and $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{bi,bi}}$.

If $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{bi,bi}}$, then $\mathsf{e} \in \mathcal{C}^*_{\mathsf{bi,bi}} \subseteq \mathcal{IR}$. Suppose $\{\mathsf{fwl}, \dots, \mathsf{e}\}$ is the last insulated region. If the adversary never asks for any challenge-equal ciphertext in this region, then $\{\mathsf{fwl}, \dots, \mathsf{e}\} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \emptyset$, which contradicts with $\mathsf{e} \in \mathcal{C}^*_{\mathsf{bi,bi}} \cap \{\mathsf{fwl}, \dots, \mathsf{e}\}$. Hence, $\{\mathsf{fwl}, \dots, \mathsf{e}\} \cap \mathcal{C} \neq \emptyset$, and we can assume $\mathsf{e}' \in \{\mathsf{fwl}, \dots, \mathsf{e}\} \cap \mathcal{C}$. By the definition of insulated region we have $\{\mathsf{fwl} + 1, \dots, \mathsf{e}\} \subseteq \mathcal{T}$, and the adversary can update the challenge-equal ciphertext $\tilde{\mathbf{c}}_{\mathsf{e}'}$ from epoch e' to epoch e to know $\tilde{\mathbf{c}}_{\mathsf{e}}$, i.e. $\mathsf{e} \in \mathcal{C}^*_{\mathsf{uni,uni}}$. Therefore, $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{uni,uni}}$ as well. $\square$

As a corollary of Lemma 3.4, Lemma 3.5 and Lemma 3.8, we have the following result. The proof is similar to the proof of Corollary 3.1.

Corollary 3.2. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \dots, \mathsf{e}\}$. Suppose $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \emptyset$, then $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{uni,uni}} \iff (\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{uni,bi}} \iff (\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{bi,uni}} \iff (\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{bi,bi}}$.

Equivalence for Trivial Win Condition " $(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*$**".**

Lemma 3.9. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \dots, \mathsf{e}\}$. Suppose $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \emptyset$, then $(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{uni,uni}} \iff (\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{bi,bi}}$.

Proof. The proof is similar to the proof of Lemma 3.8. We use the property that $(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^* \iff \mathsf{e} \in \mathcal{C}^*$. $\square$

As a corollary of Lemma 3.4, Lemma 3.5 and Lemma 3.9, we have the following result. The proof is similar to the proof of Corollary 3.1.

Corollary 3.3. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \dots, \mathsf{e}\}$. Suppose $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \emptyset$, then $(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{uni,uni}} \iff (\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{uni,bi}} \iff (\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{bi,uni}} \iff (\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{bi,bi}}$.

Equivalence for Trivial Win Condition " $\mathsf{e} \in \mathcal{K}^*$".

Lemma 3.10. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \dots, \mathsf{e}\}$, we have $\mathsf{e} \in \mathcal{K}^*_{\mathsf{uni}} \iff \mathsf{e} \in \mathcal{K}^*_{\mathsf{bi}}$.

Proof. The adversary never knows any information in the future, that is, the adversary does not know a key in an epoch $\hat{\mathsf{e}} > \mathsf{e}$. If the adversary knows the current epoch key $\mathbf{k}_{\mathsf{e}}$, then it is either a corrupted key or a key inferred from prior epoch key, thus $\mathsf{e} \in \mathcal{K}^*_{\mathsf{uni}} \iff \mathsf{e} \in \mathcal{K}^*_{\mathsf{bi}}$. □

Equivalence for Trivial Win Condition " $(\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*$".

Lemma 3.11. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \dots, \mathsf{e}\}$. Suppose $\mathsf{e} \notin \mathcal{K}^*_{\mathsf{bi}}$, then $(\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{uni,uni}} \iff (\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{bi,bi}}$.

Proof. By assumption and Lemma 3.10 the current epoch $\mathsf{e} \notin \mathcal{K}^*_{\mathsf{kk}}$ for any $\mathsf{kk} \in \{\mathsf{uni}, \mathsf{bi}\}$. We know that, by Remark 3.4, e is located in an insulated region, assume it is in $\{\mathsf{fwl}, \dots, \mathsf{e}\}$. Thus tokens $\Delta_{\mathsf{fwl}+1}, \dots, \Delta_{\mathsf{e}}$ are known to the adversary in any update setting, that is, $\{\mathsf{fwl}+1, \dots, \mathsf{e}\} \subseteq \mathcal{T} \subseteq \mathcal{T}^*_{\mathsf{uni}} \subseteq \mathcal{T}^*_{\mathsf{bi}}$. If the adversary never asks for any ciphertext in this region, then there is no ciphertext in epoch e located in the set $\mathcal{L}^*_{\mathsf{kk,cc}}$ for any $(\mathsf{kk}, \mathsf{cc})$. For all ciphertexts the adversary learns in an epoch i with $i \in \{\mathsf{fwl}, \dots, \mathsf{e}\}$, the adversary can update them to epoch e using tokens. Hence, we have $(\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{uni,uni}} \iff (\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{bi,bi}}$. □

As a corollary of Lemma 3.4, Lemma 3.5 and Lemma 3.11, we have the following result. The proof is similar to the proof of Corollary 3.1.

Corollary 3.4. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \dots, \mathsf{e}\}$. Suppose $\mathsf{e} \notin \mathcal{K}^*_{\mathsf{bi}}$, then $(\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{uni,uni}} \iff (\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{uni,bi}} \iff (\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{bi,uni}} \iff (\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{bi,bi}}$.

Equivalence for Trivial Win Condition " $(\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*$".

Lemma 3.12. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \ldots, \mathsf{e}\}$. Suppose $\mathsf{e} \notin \mathcal{K}^*_{\mathsf{bi}}$, then $(\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*_{\mathsf{uni,uni}} \iff (\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*_{\mathsf{bi,bi}}$.

Proof. The proof is similar to the proof of Lemma 3.11. As $\mathsf{e} \notin \mathcal{K}^*_{\mathsf{kk}}$ for any $\mathsf{kk} \in \{\mathsf{uni}, \mathsf{bi}\}$, we know that e is located in an insulated region. Assume it is in $\{\mathsf{fwl}, \ldots, \mathsf{e}\}$, then the adversary has corrupted the tokens $\Delta_{\mathsf{fwl}+1}, \ldots, \Delta_{\mathsf{e}}$. If the adversary never asks for any ciphertext with the underlying message $\mathbf{m}'$ in this region, then $(\mathbf{m}', \mathsf{e}) \notin \mathcal{Q}^*_{\mathsf{kk,cc}}$ for any $(\mathsf{kk}, \mathsf{cc})$. Otherwise, suppose $(\cdot, \mathbf{c}_i, i; \mathbf{m}') \in \mathcal{L}$ with $i \in \{\mathsf{fwl}, \ldots, \mathsf{e}\}$, then the adversary can update $\mathbf{c}_i$, via tokens $\Delta_{i+1}, \ldots, \Delta_{\mathsf{e}}$, to a ciphertext in epoch e with the underlying message $\mathbf{m}'$ and we have $(\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*_{\mathsf{kk,cc}}$ for any $(\mathsf{kk}, \mathsf{cc})$. □

As a corollary of Lemma 3.4, Lemma 3.5 and Lemma 3.12, we have the following result. The proof is similar to the proof of Corollary 3.1.

Corollary 3.5. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \ldots, \mathsf{e}\}$. Suppose $\mathsf{e} \notin \mathcal{K}^*_{\mathsf{bi}}$, then $(\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*_{\mathsf{uni,uni}} \iff (\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*_{\mathsf{uni,bi}} \iff (\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*_{\mathsf{bi,uni}} \iff (\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*_{\mathsf{bi,bi}}$.

3.2 Relations among Security Notions

In Fig. 16, Fig. 17 and Fig. 18, we show the relationship among six variants of the same security notion for UE schemes.

Fig. 16 demonstrates that the uni- and bi-directional update variants of the same security notion are equivalent, which means that the security notions (confidentiality and integrity) in the uni-directional update setting are not strictly stronger than the corresponding security notions in the bi-directional update setting. Hence, the security of a UE scheme is not influenced if the update setting is uni- or bi-directional. In terms of confidentiality and integrity, when we analyze the security of a UE scheme we can analyze the security based on the UE scheme with bi-directional updates.

The six variants of confidentiality notions have the relationship shown in Fig. 17, where we present that the $(\mathsf{no}, \mathsf{uni})$- variant of any confidentiality notion is strictly stronger than the other five variants of the corresponding confidentiality notion.

The six variants of integrity notions have the relationship shown in Fig. 18. No-directional key update variants of the same integrity notion is strictly stronger than the uni- or bi-directional key update variants. However, the two variants of no-directional key update notions are equivalent, that is, for the integrity notions uni- or bi-directional ciphertext update setting (with no-directional key updates) does not matter much.

It is ideal to construct an efficient UE scheme with no-directional key updates and uni-directional ciphertext updates. However, whether such a scheme exists is an open problem.

$$(\mathsf{bi},\mathsf{bi})\text{-}\mathsf{notion} \overset{3.1}{\Leftrightarrow} (\mathsf{bi},\mathsf{uni})\text{-}\mathsf{notion} \overset{3.1}{\Leftrightarrow} (\mathsf{uni},\mathsf{bi})\text{-}\mathsf{notion} \overset{3.1}{\Leftrightarrow} (\mathsf{uni},\mathsf{uni})\text{-}\mathsf{notion}$$

Figure 16: Relations among the uni- and bi-directional update variants of the same security notion, where notion $\in$ {detIND-UE-CPA, randIND-UE-CPA, detIND-UE-CCA, randIND-UE-CCA, INT-CTXT, INT-PTXT}.

$$\begin{tikzcd}
(\mathsf{no},\mathsf{uni})\text{-}\mathsf{notion} \arrow[r, Rightarrow, shift left, "3.14"] & (\mathsf{no},\mathsf{bi})\text{-}\mathsf{notion} \arrow[l, Rightarrow, shift left, "3.3", "/" marking] \arrow[r, Rightarrow, shift left, "3.13"] & (\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion} \arrow[l, Rightarrow, shift left, "3.2"]
\end{tikzcd}$$

Figure 17: Relations among the six variants of the same confidentiality notion, where notion $\in$ {detIND-UE-CPA, randIND-UE-CPA, detIND-UE-CCA, randIND-UE-CCA} and kk, cc $\in$ {uni, bi}.

$$\begin{tikzcd}
(\mathsf{no},\mathsf{uni})\text{-}\mathsf{notion} \arrow[r, Rightarrow, shift left, "3.14"] & (\mathsf{no},\mathsf{bi})\text{-}\mathsf{notion} \arrow[l, Rightarrow, shift left, "3.4"] \arrow[r, Rightarrow, shift left, "3.13"] & (\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion} \arrow[l, Rightarrow, shift left, "3.5", "/" marking]
\end{tikzcd}$$

Figure 18: Relations among the six variants of the same integrity notion, where notion $\in$ {INT-CTXT, INT-PTXT} and kk, cc $\in$ {uni, bi}.

Remark 3.6 (Informal intuition of these relations)**.** Consider the following confidentiality game, where we have an adversary against some variant of the confidentiality game for a UE scheme. The adversary corrupts a key $\mathbf{k}_1$ and a token Δ_2, and asks for a challenge ciphertext in epoch 2. For both uni- and bi-directional key update settings, the adversary can move the key $\mathbf{k}_1$ to epoch 2 and decrypt the challenge ciphertext to trivially win the confidentiality game. If the UE scheme has no-directional key updates and bi-directional ciphertext updates, the adversary can move the challenge ciphertext back to epoch 1 and decrypt it to trivially win the confidentiality game. However, if the UE scheme has no-directional key updates and uni-directional ciphertext updates, the adversary cannot trivially win the confidentiality game in this action.

Similarly, we consider the following integrity game, where we have an adversary against some variant of the integrity game for a UE scheme. The adversary corrupts a key $\mathbf{k}_1$ and a token Δ_2, and queries a try oracle in epoch 2. For both uni- and bi-directional key update settings, the adversary can move the key $\mathbf{k}_1$ to epoch 2 and provide forgeries in epoch 2 to trivially win the integrity game. However, if the UE scheme has no-directional key updates the adversary does not know $\mathbf{k}_2$, and cannot trivially win the integrity game.

Remark 3.7. We can define six variants of the $\mathsf{IND\text{-}ENC}$ and the $\mathsf{IND\text{-}UPD}$ notions[3] and can extend the result in Fig. 16 and Fig. 17 for notion $\in \{\mathsf{IND\text{-}ENC}, \mathsf{IND\text{-}UPD}\}$.

3.2.1 Relations between the Uni- and Bi-Directional Update Variants of Security Notions

We will only provide Theorem 3.1 with a fully detailed proof, since we will use similar proof techniques for Lemmas 3.13, 3.14 and Theorems 3.2, 3.4.

The following Theorem shows that for any $\mathsf{kk}, \mathsf{cc}, \mathsf{kk}', \mathsf{cc}' \in \{\mathsf{uni}, \mathsf{bi}\}$, $(\mathsf{kk}', \mathsf{cc}')$-notion implies $(\mathsf{kk}, \mathsf{cc})$-notion. Consequently, all four uni- and bi-directional update variants of the same notion are equivalent.

Theorem 3.1. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme and notion $\in \{\mathsf{INT\text{-}CTXT}, \mathsf{INT\text{-}PTXT}, \mathsf{detIND\text{-}UE\text{-}CPA}, \mathsf{randIND\text{-}UE\text{-}CPA}, \mathsf{detIND\text{-}UE\text{-}CCA}, \mathsf{randIND\text{-}UE\text{-}CCA}\}$. For any $\mathsf{kk}, \mathsf{cc}, \mathsf{kk}', \mathsf{cc}' \in \{\mathsf{uni}, \mathsf{bi}\}$ and any $(\mathsf{kk}, \mathsf{cc})$-notion adversary $\mathcal{A}$ against UE, there exists a $(\mathsf{kk}', \mathsf{cc}')$-notion adversary $\mathcal{B}_{3.1}$ against UE such that

$$\mathbf{Adv}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}}_{\mathsf{UE},\,\mathcal{A}}(1^\lambda) = \mathbf{Adv}^{(\mathsf{kk}',\mathsf{cc}')\text{-}\mathsf{notion}}_{\mathsf{UE},\,\mathcal{B}_{3.1}}(1^\lambda).$$

[3]Note that $\mathsf{IND\text{-}ENC}$ and $\mathsf{IND\text{-}UPD}$ are introduced in [LT18]

Proof. We construct a reduction $\mathcal{B}_{3.1}$ running the $(\mathsf{kk}', \mathsf{cc}')$-notion experiment which simulates the responses of queries made by the $(\mathsf{kk}, \mathsf{cc})$-notion adversary $\mathcal{A}$. The reduction will send all queries received from $\mathcal{A}$ to its $(\mathsf{kk}', \mathsf{cc}')$-notion challenger, and forwarding the responses to $\mathcal{A}$. Eventually, the reduction receives a guess from $\mathcal{A}$ and forwards it to its own challenger. In the end, the $(\mathsf{kk}', \mathsf{cc}')$-notion challenger evaluates whether or not the reduction wins, if a trivial win condition was triggered the reduction is considered as losing the game. This final win evaluation will be passed to the adversary $\mathcal{A}$.

By the analysis of trivial win equivalences in Section 3.1.2 (Corollary 3.1 to 3.5, Lemma 3.7 and Lemma 3.10), we have that if $\mathcal{A}$ does not trigger the trivial win conditions in the $(\mathsf{kk}, \mathsf{cc})$-notion game, then the reduction will not trigger the trivial win conditions in the $(\mathsf{kk}', \mathsf{cc}')$-notion game either. Similarly, if $\mathcal{A}$ does trigger the trivial win conditions in the $(\mathsf{kk}, \mathsf{cc})$-notion game, then the reduction will also trigger the trivial win conditions in the $(\mathsf{kk}', \mathsf{cc}')$-notion game. Hence, the reduction perfectly simulates the $(\mathsf{kk}, \mathsf{cc})$-notion game to adversary $\mathcal{A}$. And we have

$$\mathbf{Adv}^{(\mathsf{kk}',\mathsf{cc}')\text{-}\mathsf{notion}}_{\mathsf{UE},\,\mathcal{B}_{3.1}}(1^\lambda) = \mathbf{Adv}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}}_{\mathsf{UE},\,\mathcal{A}}(1^\lambda). \qquad \square$$

Remark 3.8. For any security notion $\mathsf{notion} \in \{\mathsf{INT\text{-}CTXT}, \mathsf{INT\text{-}PTXT}, \mathsf{detIND\text{-}UE\text{-}CPA}, \mathsf{randIND\text{-}UE\text{-}CPA}, \mathsf{detIND\text{-}UE\text{-}CCA}, \mathsf{randIND\text{-}UE\text{-}CCA}\}$, all four uni- and bi-directional update variants of the same notion are equivalent. We will use the $(\mathsf{bi}, \mathsf{bi})$-notion variant to prove notion security for a specific UE scheme. For simplicity, we will denote the notion $(\mathsf{bi}, \mathsf{bi})$-notion as notion.

3.2.2 Relations between the No-Directional and the Directional Variants of Security Notions

General Relations. We prove $(\mathsf{no}, \mathsf{bi})$-notion implies $(\mathsf{bi}, \mathsf{bi})$-notion in Lemma 3.13, combining this result with Theorem 3.1 we have that $(\mathsf{no}, \mathsf{bi})$-notion implies any $(\mathsf{kk}, \mathsf{cc})$-notion, where $\mathsf{kk}, \mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$.

Lemma 3.13. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme and $\mathsf{notion} \in \{\mathsf{INT\text{-}CTXT}, \mathsf{INT\text{-}PTXT}, \mathsf{detIND\text{-}UE\text{-}CPA}, \mathsf{randIND\text{-}UE\text{-}CPA}, \mathsf{detIND\text{-}UE\text{-}CCA}, \mathsf{randIND\text{-}UE\text{-}CCA}\}$.

For any $(\mathsf{bi},\mathsf{bi})$-notion adversary $\mathcal{A}$ against UE, there exists a $(\mathsf{no},\mathsf{bi})$-notion adversary $\mathcal{B}_{3.13}$ against UE such that

$$\mathbf{Adv}^{(\mathsf{bi},\mathsf{bi})\text{-}\mathsf{notion}}_{\mathsf{UE},\ \mathcal{A}}(1^\lambda) \leq \mathbf{Adv}^{(\mathsf{no},\mathsf{bi})\text{-}\mathsf{notion}}_{\mathsf{UE},\ \mathcal{B}_{3.13}}(1^\lambda).$$

Proof. The proof is similar to the proof of Theorem 3.1, we construct a reduction $\mathcal{B}_{3.13}$ running the $(\mathsf{no},\mathsf{bi})$-notion experiment which will simulate the responses of queries made by the $(\mathsf{bi},\mathsf{bi})$-notion adversary $\mathcal{A}$. Consider the leakage sets. For bi-directional ciphertext updates, no-directional key updates has less leakage than bi-directional key updates.

If $\mathcal{A}$ does not trigger the trivial win conditions in the $(\mathsf{bi},\mathsf{bi})$-notion game, then the reduction will not trigger the trivial win conditions in the $(\mathsf{no},\mathsf{bi})$-notion game. Hence the reduction has as least $\mathbf{Adv}^{(\mathsf{bi},\mathsf{bi})\text{-}\mathsf{notion}}_{\mathsf{UE},\ \mathcal{A}}(1^\lambda)$ advantage to win its $(\mathsf{no},\mathsf{bi})$-notion game. □

The following lemma shows that $(\mathsf{no},\mathsf{uni})$-notion implies $(\mathsf{no},\mathsf{bi})$-notion.

Lemma 3.14. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme and $\mathsf{notion} \in \{\mathsf{INT\text{-}CTXT}, \mathsf{INT\text{-}PTXT}, \mathsf{detIND\text{-}UE\text{-}CPA}, \mathsf{randIND\text{-}UE\text{-}CPA}, \mathsf{detIND\text{-}UE\text{-}CCA}, \mathsf{randIND\text{-}UE\text{-}CCA}\}$. For any $(\mathsf{no},\mathsf{bi})$-notion adversary $\mathcal{A}$ against UE, there exists an $(\mathsf{no},\mathsf{uni})$-notion adversary $\mathcal{B}_{3.14}$ against UE such that

$$\mathbf{Adv}^{(\mathsf{no},\mathsf{bi})\text{-}\mathsf{notion}}_{\mathsf{UE},\ \mathcal{A}}(1^\lambda) \leq \mathbf{Adv}^{(\mathsf{no},\mathsf{uni})\text{-}\mathsf{notion}}_{\mathsf{UE},\ \mathcal{B}_{3.14}}(1^\lambda).$$

Proof. The proof is similar to the proof of Theorem 3.1, we construct a reduction $\mathcal{B}_{3.14}$ running the $(\mathsf{no},\mathsf{uni})$-notion experiment which will simulate the responses of queries made by the $(\mathsf{no},\mathsf{bi})$-notion adversary $\mathcal{A}$. Consider the leakage sets. For no-directional key updates, uni-directional ciphertext updates has less leakage than bi-directional ciphertext updates.

If $\mathcal{A}$ does not trigger the trivial win conditions in the $(\mathsf{no},\mathsf{bi})$-notion game, then the reduction will not trigger the trivial win conditions in the $(\mathsf{no},\mathsf{uni})$-notion game. Hence the reduction has as least $\mathbf{Adv}^{(\mathsf{no},\mathsf{bi})\text{-}\mathsf{notion}}_{\mathsf{UE},\ \mathcal{A}}(1^\lambda)$ advantage to win its $(\mathsf{no},\mathsf{uni})$-notion game. □

Relations among Confidentiality Notions. Before we start to prove any relation in this section, we consider the equivalences for the following trivial win conditions "$\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$", "$\tilde{\mathsf{e}} \in \mathcal{T}^*$ **or** $\mathcal{O}.\mathsf{Upd}(\bar{\mathbf{c}})$ is queried" (already discussed on page), " $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*$" and " $(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*$". These results will be used to prove Theorem 3.2.

Lemma 3.15. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \ldots, l\}$, we have $\mathcal{K}^*_{\mathsf{no}} \cap \mathcal{C}^*_{\mathsf{no,bi}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,bi}} \neq \emptyset$.

Proof. Similar to the proof of Lemma 3.6, it suffices to prove

$$\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,bi}} \neq \emptyset \Rightarrow \mathcal{K}^*_{\mathsf{no}} \cap \mathcal{C}^*_{\mathsf{no,bi}} \neq \emptyset.$$

Suppose $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,bi}} \neq \emptyset$, as the analysis of Lemma 3.6, there exists an epoch $\mathsf{e}' \in \{0, \ldots, l\} \backslash \mathcal{IR}$ such that the adversary has asked for a challenge-equal ciphertext in this epoch, that is $\mathsf{e}' \in \mathcal{C}$. By Lemma 3.3, we know that e' is located in an epoch set which is either type 1 or type 2. Suppose $\mathsf{e}' \in \{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}}\}$ is the biggest such set around epoch e'.

If e' is located in a type 1 epoch set, then $\mathbf{k}_{\mathsf{e}_{\mathsf{end}}}$ and $\Delta_{\mathsf{e}'+1}, \ldots, \Delta_{\mathsf{e}_{\mathsf{end}}}$ are corrupted. Hence, the adversary can update the challenge-equal ciphertext $\tilde{\mathbf{c}}_{\mathsf{e}'}$ from epoch e' to epoch $\mathsf{e}_{\mathsf{end}}$ to know $\tilde{\mathbf{c}}_{\mathsf{e}_{\mathsf{end}}}$. Which means $\mathsf{e}_{\mathsf{end}} \in \mathcal{K}^*_{\mathsf{no}} \cap \mathcal{C}^*_{\mathsf{no,bi}}$.

If e' is located in a type 2 epoch set, we claim that any epoch $\mathsf{e} \in \{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}}\}$ is either in $\mathcal{K}$ or $\mathcal{T}$ or both, because revealed epoch keys in the uni-directional update setting is either a corrupted key or a key inferred from prior epoch via a corrupted update token. Furthermore, we claim that $\mathsf{e}_{\mathsf{start}} \in \mathcal{K}$. Otherwise, we have $\mathsf{e}_{\mathsf{start}} \in \mathcal{T}$ and $\mathsf{e}_{\mathsf{start}} - 1 \in \mathcal{K}^*_{\mathsf{uni}}$ (in this situation, $\mathbf{k}_{\mathsf{e}_{\mathsf{start}}}$ is inferred from $\mathbf{k}_{\mathsf{e}_{\mathsf{start}}-1}$) and find a bigger type 2 set, which is contradict with the assumption that $\{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}}\}$ is the biggest type 2 set around epoch e'. Then we discuss the joint set $\mathcal{K}^*_{\mathsf{no}} \cap \mathcal{C}^*_{\mathsf{no,bi}}$. If $\mathsf{e}' \in \mathcal{K}$, then we have $\mathsf{e}' \in \mathcal{K}^*_{\mathsf{no}} \cap \mathcal{C}^*_{\mathsf{no,bi}}$. If $\mathsf{e}' \notin \mathcal{K}$, then $\mathsf{e}' \in \mathcal{T}$. Iteratively, we can find an epoch $\mathsf{e}_s < \mathsf{e}'$ such that $\mathsf{e}_s \in \mathcal{K}$ and $\mathsf{e}_s + 1, \ldots, \mathsf{e}' \in \mathcal{T}$. Hence, the adversary can reversely update the challenge-equal ciphertext $\tilde{\mathbf{c}}_{\mathsf{e}'}$ from epoch e' to epoch e_s to know $\tilde{\mathbf{c}}_{\mathsf{e}_s}$. Then we have $\mathsf{e}_s \in \mathcal{K}^*_{\mathsf{no}} \cap \mathcal{C}^*_{\mathsf{no,bi}}$. □

The proof of the following lemma is similar to the proof of Lemma 3.8.

Lemma 3.16. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \ldots, \mathsf{e}\}$. Suppose $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \emptyset$, then $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{uni,bi}} \iff (\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{no,bi}}$.

The proof of the following lemma is similar to the proof of Lemma 3.9.

Lemma 3.17. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \dots, \mathsf{e}\}$. Suppose $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \emptyset$, then $(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{uni,bi}} \iff (\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{no,bi}}$.

The following theorem shows that the $(\mathsf{uni}, \mathsf{bi})$- variant of confidentiality notions implies the $(\mathsf{no}, \mathsf{bi})$- variant of the corresponding confidentiality notions. Combining this result with Theorem 3.1 we have that any $(\mathsf{kk}, \mathsf{cc})$-variant of confidentiality notions implies the $(\mathsf{no}, \mathsf{bi})$- variant of the corresponding confidentiality notions, where $\mathsf{kk}, \mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$.

Theorem 3.2. Let UE be an updatable encryption scheme and $\mathsf{notion} \in \{\mathsf{detIND\text{-}UE\text{-}CPA}, \mathsf{randIND\text{-}UE\text{-}CPA}, \mathsf{detIND\text{-}UE\text{-}CCA}, \mathsf{randIND\text{-}UE\text{-}CCA}\}$ For any $(\mathsf{no}, \mathsf{bi})$-notion adversary $\mathcal{A}$ against UE, there exists a $(\mathsf{uni}, \mathsf{bi})$-notion adversary $\mathcal{B}_{3.2}$ against UE such that

$$\mathbf{Adv}^{(\mathsf{no,bi})\text{-}\mathsf{notion}}_{\mathsf{UE},\ \mathcal{A}}(1^\lambda) = \mathbf{Adv}^{(\mathsf{uni,bi})\text{-}\mathsf{notion}}_{\mathsf{UE},\ \mathcal{B}_{3.2}}(1^\lambda).$$

Proof. The proof is similar to the proof of Theorem 3.1, we construct a reduction $\mathcal{B}_{3.2}$ running the $(\mathsf{uni}, \mathsf{bi})$-notion experiment which will simulate the responses of queries made by the $(\mathsf{no}, \mathsf{bi})$-notion adversary $\mathcal{A}$.

By the analysis of trivial win equivalences (Lemmas 3.15, 3.7, 3.16 and 3.17), we have that the trivial win results are the same in both the $(\mathsf{no}, \mathsf{bi})$-notion game and the $(\mathsf{uni}, \mathsf{bi})$-notion game. Hence, the reduction perfectly simulates the $(\mathsf{no}, \mathsf{bi})$-notion game to adversary $\mathcal{A}$. And we have $\mathbf{Adv}^{(\mathsf{no,bi})\text{-}\mathsf{notion}}_{\mathsf{UE},\ \mathcal{A}}(1^\lambda) = \mathbf{Adv}^{(\mathsf{uni,bi})\text{-}\mathsf{notion}}_{\mathsf{UE},\ \mathcal{B}_{3.2}}(1^\lambda)$. □

The following Theorem states that the $(\mathsf{no}, \mathsf{bi})$- variant of confidentiality notions do not imply the $(\mathsf{no}, \mathsf{uni})$- variant of the corresponding confidentiality notions.

Theorem 3.3. Let UE be an updatable encryption scheme and $\mathsf{notion} \in \{\mathsf{detIND\text{-}UE\text{-}CPA}, \mathsf{randIND\text{-}UE\text{-}CPA}, \mathsf{detIND\text{-}UE\text{-}CCA}, \mathsf{randIND\text{-}UE\text{-}CCA}\}$. Let α_{notion} be the $(\mathsf{no}, \mathsf{bi})$-notion advantage of an adversary $\mathcal{A}$ against UE, Then there exists a modified scheme $\mathsf{UE}^{\mathsf{new}}_1$, detailed in Fig. 19, such that the $(\mathsf{no}, \mathsf{bi})$-notion advantage of $\mathcal{A}$ against $\mathsf{UE}^{\mathsf{new}}_1$ is $\alpha_{\mathsf{notion}} + l\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{SKE}}$, and there exists an $(\mathsf{no}, \mathsf{uni})$-notion adversary $\mathcal{B}_{3.3}$ against $\mathsf{UE}^{\mathsf{new}}_1$ that wins with probability 1.

Proof technique of Theorem 3.3. For a confidentiality notion notion, we use a $(\mathsf{no}, \mathsf{bi})$-notion secure UE scheme UE to construct a new UE scheme $\mathsf{UE}_1^{\mathsf{new}}$, which is still $(\mathsf{no}, \mathsf{bi})$-notion secure but not $(\mathsf{no}, \mathsf{uni})$-notion secure. As a result, $(\mathsf{no}, \mathsf{uni})$-notion is strictly stronger than $(\mathsf{no}, \mathsf{bi})$-notion.

```
$\underline{\mathsf{UE}_1^{\mathsf{new}}.\mathsf{KG}(1^\lambda):}$
  $\mathbf{k} \xleftarrow{\$} \mathsf{UE.KG}(1^\lambda)$
  return $\mathbf{k}$

$\underline{\mathsf{UE}_1^{\mathsf{new}}.\mathsf{TG}(\mathbf{k}_\mathsf{e}, \mathbf{k}_{\mathsf{e}+1}):}$
  $\Delta_{\mathsf{e}+1}^1 \leftarrow \mathsf{UE.TG}(\mathbf{k}_\mathsf{e}, \mathbf{k}_{\mathsf{e}+1})$
  $sk_{\mathsf{e}+1} \xleftarrow{\$} \mathsf{SKE.KG}$
  return $(\Delta_{\mathsf{e}+1}^1, sk_{\mathsf{e}+1})$

$\underline{\mathsf{UE}_1^{\mathsf{new}}.\mathsf{Enc}(\mathbf{k}_\mathsf{e}, \mathbf{m}):}$
  $\mathbf{c}_\mathsf{e}^1 \xleftarrow{\$} \mathsf{UE.Enc}(\mathbf{k}_\mathsf{e}, \mathbf{m})$
  $\mathbf{c}_\mathsf{e}^2 \xleftarrow{\$} \mathcal{CS}$
  return $(\mathbf{c}_\mathsf{e}^1, \mathbf{c}_\mathsf{e}^2)$

$\underline{\mathsf{UE}_1^{\mathsf{new}}.\mathsf{Dec}(\mathbf{k}_\mathsf{e}, \mathbf{c}_\mathsf{e}):}$
  parse $\mathbf{c}_\mathsf{e} = (\mathbf{c}_\mathsf{e}^1, \mathbf{c}_\mathsf{e}^2)$
  $\mathbf{m}'$ or $\perp \leftarrow \mathsf{UE.Dec}(\mathbf{k}_\mathsf{e}, \mathbf{c}_\mathsf{e}^1)$
  return $\mathbf{m}'$

$\underline{\mathsf{UE}_1^{\mathsf{new}}.\mathsf{Upd}((\Delta_{\mathsf{e}+1}^1, sk_{\mathsf{e}+1}), \mathbf{c}_\mathsf{e}):}$
  parse $\mathbf{c}_\mathsf{e} = (\mathbf{c}_\mathsf{e}^1, \mathbf{c}_\mathsf{e}^2)$
  $\mathbf{c}_{\mathsf{e}+1}^1 \leftarrow \mathsf{UE.Upd}(\Delta_{\mathsf{e}+1}^1, \mathbf{c}_\mathsf{e}^1)$
  $\mathbf{c}_{\mathsf{e}+1}^2 \leftarrow \mathsf{SKE.Enc}(sk_{\mathsf{e}+1}, \mathbf{c}_\mathsf{e}^1)$
  return $(\mathbf{c}_{\mathsf{e}+1}^1, \mathbf{c}_{\mathsf{e}+1}^2)$
```

Figure 19: Updatable encryption scheme $\mathsf{UE}_1^{\mathsf{new}}$ for proof of Theorem 3.3, built from IND\$-CPA-secure SKE SKE and updatable encryption scheme UE.

Proof. $\underline{\mathsf{UE}_1^{\mathsf{new}} \text{ is not } (\mathsf{no}, \mathsf{uni})\text{-notion secure.}}$ If a challenge-equal ciphertext $\tilde{\mathbf{c}}_{\mathsf{e}+1}$ and the corresponding epoch update token $(\Delta_{\mathsf{e}+1}^1, sk_{\mathsf{e}+1})$ are corrupted then the adversary can compute the prior challenge-equal ciphertext $\tilde{\mathbf{c}}_\mathsf{e}^1$ by using $sk_{\mathsf{e}+1}$ to decrypt $\tilde{\mathbf{c}}_{\mathsf{e}+1}^2$. Then the adversary can corrupt a key $\mathbf{k}_\mathsf{e}$ to win the $(\mathsf{no}, \mathsf{uni})$-notion game without trigger the trivial win conditions.

$\underline{\mathsf{UE}_1^{\mathsf{new}} \text{ is } (\mathsf{no}, \mathsf{bi})\text{-notion secure.}}$ All algorithms for $\mathsf{UE}_1^{\mathsf{new}}$ are the same as for UE, except for the encryption algorithm $\mathsf{UE}_1^{\mathsf{new}}.\mathsf{Enc}$ and the update algorithm $\mathsf{UE}_1^{\mathsf{new}}.\mathsf{Upd}$, where we use a SKE scheme SKE to make the $\mathsf{UE}_1^{\mathsf{new}}$ scheme has bi-directional ciphertext updates. This does not affect an adversary's ability to win the $(\mathsf{no}, \mathsf{bi})$-notion game. The additional advantage term $l\mathbf{Adv}_{\mathsf{SKE}}^{\mathsf{IND\$\text{-}CPA}}$ is because if the adversary knows a challenge-equal ciphertext, but not a token, then the second challenge-equal ciphertext term will not leak the prior challenge-equal ciphertext. □

Relations among Integrity Notions. The following Theorem states that the $(\mathsf{no}, \mathsf{bi})$- variant of an integrity notion implies the $(\mathsf{no}, \mathsf{uni})$- variant of the same integrity notion.

Theorem 3.4. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. For any $(\mathsf{no}, \mathsf{uni})$-notion adversary $\mathcal{A}$ against UE, where notion $\in \{\mathsf{INT\text{-}CTXT}, \mathsf{INT\text{-}PTXT}\}$, there exists an $(\mathsf{no}, \mathsf{bi})$-notion adversary $\mathcal{B}_{3.4}$ against UE such that

$$\mathbf{Adv}^{(\mathsf{no},\mathsf{uni})\text{-}\mathsf{notion}}_{\mathsf{UE},\ \mathcal{A}}(1^\lambda) \leq \mathbf{Adv}^{(\mathsf{no},\mathsf{bi})\text{-}\mathsf{notion}}_{\mathsf{UE},\ \mathcal{B}_{3.4}}(1^\lambda).$$

Proof. In the no-directional key update setting, $\mathcal{K}^*_{\mathsf{no}} = \mathcal{K}$. The proof is similar to the proof of Theorem 3.1, we construct a reduction $\mathcal{B}_{3.4}$ running the $(\mathsf{no}, \mathsf{bi})$-notion experiment which will simulate the responses of queries made by the $(\mathsf{no}, \mathsf{uni})$-notion adversary $\mathcal{A}$. Suppose $\mathcal{A}$ queries a $\mathcal{O}.\mathsf{Try}$ oracle in epoch e. Notice that for any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \ldots, \mathsf{e}\}$, $(\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{no},\mathsf{uni}} \iff (\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{no},\mathsf{bi}}$ and $(\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*_{\mathsf{no},\mathsf{uni}} \iff (\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*_{\mathsf{no},\mathsf{bi}}$, because such ciphertexts or plaintexts cannot be inferred from future ciphertexts or plaintexts. Hence, the trivial win results are the same in both the the $(\mathsf{no}, \mathsf{uni})$-notion game and the $(\mathsf{no}, \mathsf{bi})$-notion game. □

However, $(\mathsf{uni}, \mathsf{bi})$- variant of integrity notions do not imply the $(\mathsf{no}, \mathsf{bi})$- variant of the corresponding integrity notions. Combining this result with Theorem 3.1 we have that the $(\mathsf{no}, \mathsf{bi})$- variant of integrity notions are strictly stronger than any $(\mathsf{kk}, \mathsf{cc})$- variant of the corresponding integrity notions, where $\mathsf{kk}, \mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$.

Theorem 3.5. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme and notion $\in \{\mathsf{INT\text{-}CTXT}, \mathsf{INT\text{-}PTXT}\}$. Let α_{notion} be the $(\mathsf{uni}, \mathsf{bi})$-notion advantage of an adversary $\mathcal{A}$ against UE, then there exists a modified scheme $\mathsf{UE}^{\mathsf{new}}_2$, detailed in Fig. 20, such that the $(\mathsf{uni}, \mathsf{bi})$-notion advantage of $\mathcal{A}$ against $\mathsf{UE}^{\mathsf{new}}_2$ is $\alpha_{\mathsf{notion}} + l\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{SKE}}$, and there exists an $(\mathsf{no}, \mathsf{bi})$-notion adversary $\mathcal{B}_{3.5}$ against $\mathsf{UE}^{\mathsf{new}}_2$ that wins with probability 1.

Proof technique of Theorem 3.2. We use a $(\mathsf{uni}, \mathsf{bi})$-notion secure updatable encryption scheme UE to construct a new updatable encryption scheme $\mathsf{UE}_2^{\mathsf{new}}$, which is still $(\mathsf{uni}, \mathsf{bi})$-notion secure but not $(\mathsf{no}, \mathsf{bi})$-notion secure. As a result, $(\mathsf{no}, \mathsf{bi})$-notion notion is strictly stronger than $(\mathsf{uni}, \mathsf{bi})$-notion notion.

```
$\underline{\mathsf{UE}_2^{\mathsf{new}}.\mathsf{KG}(1^\lambda):}$
  $\mathbf{k} \xleftarrow{\$} \mathsf{UE}.\mathsf{KG}(1^\lambda)$
  return $\mathbf{k}$

$\underline{\mathsf{UE}_2^{\mathsf{new}}.\mathsf{TG}(\mathbf{k}_\mathsf{e}, \mathbf{k}_{\mathsf{e}+1}):}$
  $\Delta_{\mathsf{e}+1}^t \leftarrow \mathsf{UE}.\mathsf{TG}(\mathbf{k}_\mathsf{e}, \mathbf{k}_{\mathsf{e}+1})$
  $\Delta_{\mathsf{e}+1}^c \leftarrow \mathsf{SKE}.\mathsf{Enc}(\mathbf{k}_\mathsf{e}, \mathbf{k}_{\mathsf{e}+1})$
  $\Delta_{\mathsf{e}+1} \leftarrow (\Delta_{\mathsf{e}+1}^t, \Delta_{\mathsf{e}+1}^c)$
  return $\Delta_{\mathsf{e}+1}$

$\underline{\mathsf{UE}_2^{\mathsf{new}}.\mathsf{Enc}(\mathbf{k}_\mathsf{e}, \mathbf{m}):}$
  $\mathbf{c}_\mathsf{e} \xleftarrow{\$} \mathsf{UE}.\mathsf{Enc}(\mathbf{k}_\mathsf{e}, \mathbf{m})$
  return $\mathbf{c}_\mathsf{e}$

$\underline{\mathsf{UE}_2^{\mathsf{new}}.\mathsf{Dec}(\mathbf{k}_\mathsf{e}, \mathbf{c}_\mathsf{e}):}$
  $\mathbf{m}'$ or $\perp \leftarrow \mathsf{UE}.\mathsf{Dec}(\mathbf{k}_\mathsf{e}, \mathbf{c}_\mathsf{e})$
  return $\mathbf{m}'$

$\underline{\mathsf{UE}_2^{\mathsf{new}}.\mathsf{Upd}(\Delta_{\mathsf{e}+1}, \mathbf{c}_\mathsf{e}):}$
  parse $\Delta_{\mathsf{e}+1} = (\Delta_{\mathsf{e}+1}^t, \Delta_{\mathsf{e}+1}^c)$
  $\mathbf{c}_{\mathsf{e}+1} \leftarrow \mathsf{UE}.\mathsf{Upd}(\Delta_{\mathsf{e}+1}^t, \mathbf{c}_\mathsf{e})$
  return $\mathbf{c}_{\mathsf{e}+1}$
```

Figure 20: Updatable encryption scheme $\mathsf{UE}_2^{\mathsf{new}}$ for proof of Theorem 3.5, built from $\mathsf{IND\$\text{-}CPA}$-secure symmetric key encryption (SKE) scheme SKE and updatable encryption scheme UE.

Proof. $\underline{\mathsf{UE}_2^{\mathsf{new}} \text{ is not } (\mathsf{no}, \mathsf{bi})\text{-notion secure.}}$ If a key $\mathbf{k}_\mathsf{e}$ and the next epoch update token $\Delta_{\mathsf{e}+1}$ are corrupted then the adversary can compute the next epoch key $\mathbf{k}_{\mathsf{e}+1}$ by decrypting $\Delta_{\mathsf{e}+1}^c$ using $\mathbf{k}_\mathsf{e}$. The adversary can compute a forgery $\mathbf{c}_{\mathsf{e}+1}$ in an integrity game to win the security game $(\mathsf{no}, \mathsf{bi})$-notion without trigger the trivial win conditions.

$\underline{\mathsf{UE}_2^{\mathsf{new}} \text{ is } (\mathsf{uni}, \mathsf{bi})\text{-notion secure.}}$ All algorithms for $\mathsf{UE}_2^{\mathsf{new}}$ are the same as for UE, except for the token generation algorithm $\mathsf{UE}_2^{\mathsf{new}}.\mathsf{TG}$, where we use a SKE scheme SKE to make the $\mathsf{UE}_2^{\mathsf{new}}$ scheme has uni-directional key updates. This does not affect an adversary's ability to win the $(\mathsf{uni}, \mathsf{bi})$-notion game. The additional advantage term $l\mathbf{Adv}_{\mathsf{SKE}}^{\mathsf{IND\$\text{-}CPA}}$ is because if the adversary knows a token, but not the prior epoch key, then the second token value $\Delta_{\mathsf{e}+1}^c$ is computationally indistinguishable from a random element.

Which makes sure that the second token term will not leak information that would help the adversary win this security game. □

4 LWE-based PKE Scheme

In this section, we look at an LWE-based PKE scheme LWEPKE, which is detailed in Fig. 21. We prove that LWEPKE is IND$-CPA-secure, if the underlying LWE problem is hard. We will later use this PKE scheme to construct an updatable encryption scheme in Section 5.

```
LWEPKE.Setup(1^λ):
  $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$

LWEPKE.KG(1^λ):
  $\mathbf{s} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$
  $\mathbf{e} \leftarrow D_{\mathbb{Z},\alpha}^m$
  $\mathbf{p} \leftarrow \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \mod q$
  return $(\mathbf{s}, \mathbf{p})$

LWEPKE.Enc(p, m):
  $\mathbf{R} \leftarrow \mathcal{D}_r^t$
  $\mathbf{e}' \leftarrow D_{\mathbb{Z},\beta}^{1 \times t}$
  $\mathbf{C}_1 \leftarrow \mathbf{A}^\intercal \cdot \mathbf{R}$
  $\mathbf{c}_2 \leftarrow \mathbf{p}^\intercal \cdot \mathbf{R} + \mathbf{e}' + \frac{q}{2}\mathbf{m} \mod q$
  return $(\mathbf{C}_1, \mathbf{c}_2)$

LWEPKE.Dec(s, c):
  parse $\mathbf{c} = (\mathbf{C}_1, \mathbf{c}_2)$
  $\mathbf{d} \leftarrow \mathbf{c}_2 - \mathbf{s}^\intercal \cdot \mathbf{C}_1$
  parse $\mathbf{d} = (d_1, ..., d_t)$
  for $i \in \{1, 2, ..., t\}$ do
    if $d_i \in (\frac{3q}{8}, \frac{5q}{8})$ then
      $m'_i \leftarrow 1$
    else if $d_i \in (-\frac{q}{8}, \frac{q}{8})$ then
      $m'_i \leftarrow 0$
    else
      return $\perp$
  $\mathbf{m}' \leftarrow (m'_1, ..., m'_t)$
  return $\mathbf{m}'$
```

Figure 21: The algorithms of the LWE-based LWEPKE scheme. The randomness distribution $\mathcal{D}_r$ is defined over $\mathbb{Z}_q^m$. $D_{\mathbb{Z},\alpha}, D_{\mathbb{Z},\beta}$ are discrete Gaussian distributions. The message $\mathbf{m}$ lies in $\{0,1\}^{1\times t}$.

4.1 PKE Construction

In the setup phase, the scheme LWEPKE randomly chooses a matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m\times n}$. The key generation algorithm samples a secret $\mathbf{s}$ from the uniform distribution $\mathcal{U}(\mathbb{Z}_q^n)$ and computes $\mathbf{p} = \mathbf{A}\cdot\mathbf{s}+\mathbf{e}$, where the error $\mathbf{e}$ is chosen from the discrete Gaussian distribution $D_{\mathbb{Z},\alpha}^m$. The matrix $\mathbf{A}$ and the vector

$\mathbf{p}$ form the public key. Encryption takes a bit string $\mathbf{m} \in \{0,1\}^{1\times t}$ as input, and outputs a ciphertext $(\mathbf{A}^\intercal \cdot \mathbf{R}, \mathbf{p}^\intercal \cdot \mathbf{R} + \mathbf{e}' + \frac{q}{2}\mathbf{m} \mod q)$. Decryption is performed by computing $\mathbf{d} = \mathbf{c}_2 - \mathbf{s}^\intercal \cdot \mathbf{C}_1$. For each entry d_i of $\mathbf{d}$, the decryption algorithm outputs 0 if d_i is close to $0 \mod q$, and outputs 1 if d_i is close to $\frac{q}{2} \mod q$.

Parameter Setting. The parameter setting of the scheme LWEPKE is as follows:

- $n = \lambda$ is the security parameter,
- $q = q(n) \geq 2$ be a prime,
- $m = \mathsf{poly}(n)$ and $t = \mathsf{poly}(n)$ be two integers,
- $\mathcal{D}_r$ be a distribution over $\mathbb{Z}_q^m$ with min-entropy k such that $n \leq (k - 2\log(1/\epsilon) - O(1))/\log(q)$ for negligible $\epsilon > 0$, the infinite norm of the vector outputted by this distribution is at most $B = \mathsf{poly}(n)$ with overwhelming probability,
- $\alpha, \beta > 0$ be two numbers such that $\beta \leq \frac{q}{8}$ and $\alpha B/\beta = \mathsf{negl}(n)$.
- $D_{\mathbb{Z},\alpha}$ and $D_{\mathbb{Z},\beta}$ be two discrete Gaussian distributions.

Remark 4.1. We specify that all operations in this paper are done in field $\mathbb{Z}_q$, and stop writing $\mod q$ for the rest of this paper.

4.2 Correctness and Security

Correctness. We claim that LWEPKE.Dec decrypts correctly with overwhelming probability. The decryption algorithm computes $\mathbf{d} = \mathbf{c}_2 - \mathbf{s}^\intercal \cdot \mathbf{C}_1 = \mathbf{e}^\intercal \cdot \mathbf{R} + \mathbf{e}' + \frac{q}{2}\mathbf{m}$, and outputs $\mathbf{m}$ if $\mathbf{e}^\intercal \cdot \mathbf{R} + \mathbf{e}'$ has distance at most $\frac{q}{8}$ from $\mathbf{0} \mod q$.

Let $\mathbf{R} = [\mathbf{r}_1, \ldots, \mathbf{r}_t]$. By Lemma A.1, we have that each entry of $\mathbf{e}$ have size at most α with overwhelming probability. As each entry of $\mathbf{R}$ has size at most B, then each entry of $\mathbf{e}^\intercal \cdot \mathbf{R}$ has size at most $\alpha \cdot B$ with overwhelming probability. That is, for all $1 \leq j \leq t$, $|\mathbf{e}^\intercal \cdot \mathbf{r}_j| \leq \alpha \cdot B$. Hence, $|\mathbf{e}^\intercal \cdot \mathbf{r}_j|/\beta \leq \alpha \cdot B/\beta = \mathsf{negl}(n)$ is negligible, and by Lemma A.2 we have that $D_{\mathbb{Z},\beta}$ and $D_{\mathbb{Z},\beta} + \mathbf{e}^\intercal \cdot \mathbf{r}_j$ are statistically close. Therefore, $D_{\mathbb{Z},\beta}^{1\times t}$

is statistically close to $D_{\mathbb{Z},\beta}^{1\times t} + \mathbf{e}^\intercal \cdot \mathbf{R}$, so $\mathbf{e}^\intercal \cdot \mathbf{R} + \mathbf{e}' \overset{s}{\approx} \mathbf{e}'$. By Lemma A.1, we have that each entry of $\mathbf{e}'$ has size at most β with overwhelming probability. Since $\beta \leq \frac{q}{8}$, we have the desired result.

Security. We now show that LWEPKE is IND\$-CPA-secure under the assumption that the $\mathsf{DLWE}_{n,q,\alpha}$ problem is hard.

Theorem 4.1. Let LWEPKE be the public key encryption described in Fig. 21, using the parameter setting described in Section 4.1. Then for any adversary IND\$-CPA $\mathcal{A}$ against LWEPKE, there exists an adversary $\mathcal{B}$ against $\mathsf{DLWE}_{n,q,\alpha}$ such that

$$\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{LWEPKE},\,\mathcal{A}}(1^\lambda) \leq t\epsilon + \mathbf{Adv}^{\mathsf{DLWE}}_{n,q,\alpha}(\mathcal{B}) + \mathsf{negl}(n).$$

Proof. The proof of theorem 4.1 consists of a sequence of games. Denote E_i be the event that the adversary's guess $\mathrm{b}' = 1$ in game i.

Game 0

The first game is the experiment $\mathbf{Exp}^{\mathsf{IND\$\text{-}CPA\text{-}0}}_{\mathsf{LWEPKE},\,\mathcal{A}}$, given in Fig. 1. Then we have

$$\mathbf{Pr}[E_0] = \mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\$\text{-}CPA\text{-}0}}_{\mathsf{LWEPKE},\,\mathcal{A}} = 1].$$

Game 1

We consider a modified game, which is the same as Game 0 except for the second component of the challenge ciphertext ($\mathbf{c}_2$) is generated as $\mathbf{s}^\intercal \cdot \mathbf{C}_1 + \mathbf{e}' + \frac{q}{2}\mathbf{m}$ instead of $\mathbf{p}^\intercal \cdot \mathbf{R} + \mathbf{e}' + \frac{q}{2}\mathbf{m}$.

In game 0, $\mathbf{c}_2 - \frac{q}{2}\mathbf{m} = \mathbf{p}^\intercal \cdot \mathbf{R} + \mathbf{e}' = (\mathbf{A} \cdot \mathbf{s} + \mathbf{e})^\intercal \cdot \mathbf{R} + \mathbf{e}' = \mathbf{s}^\intercal \cdot \mathbf{A}^\intercal \cdot \mathbf{R} + \mathbf{e}^\intercal \cdot \mathbf{R} + \mathbf{e}' = \mathbf{s}^\intercal \cdot \mathbf{A}^\intercal \cdot \mathbf{R} + \mathbf{e}^\intercal \cdot \mathbf{R} + \mathbf{e}' = \mathbf{s}^\intercal \cdot \mathbf{C}_1 + \mathbf{e}^\intercal \cdot \mathbf{R} + \mathbf{e}'$. Similar to the correctness analysis, we know by Lemma A.1 and Lemma A.2 that $\mathbf{e}^\intercal \cdot \mathbf{R} + \mathbf{e}' \overset{s}{\approx} \mathbf{e}'$. Thus,

$$|\mathbf{Pr}[E_0] - \mathbf{Pr}[E_1]| \leq \mathsf{negl}(n).$$

Game 2

We consider a modified game that is the same as Game 1, except that the first component of the challenge ciphertext ($\mathbf{C}_1$) is sampled from the uniform distribution over $\mathbb{Z}_q^{n\times t}$.

From the leftover hash lemma (Lemma. A.6) we know that the joint distribution of $(\mathbf{A}, \mathbf{A}^\intercal \cdot \mathbf{R})$ is $t\epsilon$-close to the uniform distribution over $\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{n\times t}$, since the min-entropy of $\mathbf{r}_i$ is at least $n\log(q) + 2\log(1/\epsilon) + O(1)$. That is, $(\mathbf{A}, \mathbf{C}_1) \overset{c}{\approx} (\mathbf{A}, \mathbf{U}_1)$, for a uniformly random matrix $\mathbf{U}_1$, and we have

$$|\mathbf{Pr}[E_1] - \mathbf{Pr}[E_2]| \leq t\epsilon.$$

Game 3

We consider a modified game that is the same as Game 2, except that the second component of the ciphertext ($\mathbf{c}_2$) is sampled from the uniform distribution over $\mathbb{Z}_q^{1\times t}$.

Since the $\mathsf{DLWE}_{n,q,\alpha}$ problem is hard, the $\mathsf{DLWE}_{n,q,\beta}$ problem is also hard by Lemma A.4. Therefore, the joint distribution of $(\mathbf{C}_1, \mathbf{s}^\intercal \cdot \mathbf{C}_1 + \mathbf{e}')$ is computationally indistinguishable from the uniform distribution over $\mathbb{Z}_q^{n\times t} \times \mathbb{Z}_q^{1\times t}$, under the $\mathsf{DLWE}_{n,q,\beta}$ problem is hard. That is, $(\mathbf{C}_1, \mathbf{s}^\intercal \cdot \mathbf{C}_1 + \mathbf{e}') \overset{c}{\approx} (\mathbf{C}_1, \mathbf{u}_2)$, for a uniformly random vector $\mathbf{u}_2$, and we have

$$|\mathbf{Pr}[E_2] - \mathbf{Pr}[E_3]| \leq \mathbf{Adv}^{\mathsf{DLWE}}_{n,q,\alpha}(\mathcal{B}) + \mathsf{negl}(n).$$

We know that $\mathbf{Pr}[E_3] = \mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\$\text{-}CPA\text{-}1}}_{\mathsf{LWEPKE},\ \mathcal{A}} = 1]$. By Definition 1, we have $\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{LWEPKE},\ \mathcal{A}}(1^\lambda) = |\Pr[E_3] - \Pr[E_0]|$, which concludes the proof. □

5 LWE-based Updatable Encryption Scheme

We construct an LWE-based updatable encryption scheme LWEUE and prove that it is $\mathsf{randIND\text{-}UE\text{-}CPA}$ secure if the underlying LWE problem is hard.

5.1 UE Construction

We now introduce our updatable encryption scheme LWEUE, which is parameterized by an LWE-based PKE scheme LWEPKE (see Fig. 21). LWEUE uses algorithms from LWEPKE to do key generation, encryption and decryption. To generate a new key from an old key in the next algorithm, our UE scheme uses the homomorphic property of the LWE pairs. In particular, suppose the old key is $(\mathbf{s}_\mathsf{e}, \mathbf{p}_\mathsf{e})$, $\mathsf{LWEUE.KG}$ samples a new pair of LWE pairs $(\Delta^{\mathbf{s}}_{\mathsf{e}+1}, \Delta^{\mathbf{p}}_{\mathsf{e}+1})$ and sets $(\mathbf{s}_\mathsf{e} + \Delta^{\mathbf{s}}_{\mathsf{e}+1}, \mathbf{p}_\mathsf{e} + \Delta^{\mathbf{p}}_{\mathsf{e}+1})$ as the new epoch key, where $(\Delta^{\mathbf{s}}_{\mathsf{e}+1}, \mathbf{p}_\mathsf{e} + \Delta^{\mathbf{p}}_{\mathsf{e}+1})$ is the update token. To update ciphertexts, LWEUE uses the re-randomization idea that was similar to the idea from RISE in the work by Lehmann and Tackmann [LT18]. As the ciphertext can be re-randomized by the update token, the update algorithm uses the update token to update ciphertext from an old one to a new one. More precisely, the scheme LWEUE is described in Fig. 22.

Parameter Setting. We use the parameter setting of the scheme LWEPKE, described in Section 4.1. Additionally, we require $\beta \leq \frac{q}{8\sqrt{l}}$, where $l = \mathsf{poly}(n)$ is an upper bound on the last epoch.

5.2 Construction Challenges in LWE-based UE Schemes

In this section, we discuss leakage from tokens due to bad UE construction and show how to solve this leakage problems.

Secret Key Distribution. We first state that a binary secret does not work in the UE scheme, as an update token might reveal the secret information. Suppose an entry of the update token $\Delta^{\mathbf{s}}_{\mathsf{e}+1} (= \mathbf{s}_{\mathsf{e}+1} - \mathbf{s}_\mathsf{e})$ is -1 (1, resp.), then we can conclude the corresponding entry of the previous secret $\mathbf{s}_\mathsf{e}$ is 1 (0, resp.) and the corresponding entry of the new secret $\mathbf{s}_{\mathsf{e}+1}$ is 0 (1, resp.).

We choose that secret keys and update tokens are sampled from the uniform distribution over $\mathbb{Z}_q^n$, which ensures that any corrupted token will not reveal any information about the relevant secret keys.

Epoch Key Generation. Intuitively, it is natural to consider generating the epoch keys by sampling a secret $\mathbf{s}_i \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$ and setting the public key

```
$\underline{\mathsf{Setup}(1^\lambda):}$
    $\mathbf{A} \leftarrow \mathsf{LWEPKE.Setup}(1^\lambda)$

$\underline{\mathsf{LWEUE.KG}(1^\lambda):}$
    if $\mathsf{e} = 0$ then
        $(\mathbf{s}_0, \mathbf{p}_0) \leftarrow \mathsf{LWEPKE.KG}(1^\lambda)$
    else
        parse $\mathbf{k}_{\mathsf{e}-1} = (\mathbf{s}_{\mathsf{e}-1}, \mathbf{p}_{\mathsf{e}-1})$
        $(\Delta^{\mathbf{s}}_{\mathsf{e}}, \Delta^{\mathbf{p}}_{\mathsf{e}}) \leftarrow \mathsf{LWEPKE.KG}(1^\lambda)$
        $\mathbf{s}_{\mathsf{e}} \leftarrow \mathbf{s}_{\mathsf{e}-1} + \Delta^{\mathbf{s}}_{\mathsf{e}}$
        $\mathbf{p}_{\mathsf{e}} \leftarrow \mathbf{p}_{\mathsf{e}-1} + \Delta^{\mathbf{p}}_{\mathsf{e}}$
        $\mathbf{k}_{\mathsf{e}} \leftarrow (\mathbf{s}_{\mathsf{e}}, \mathbf{p}_{\mathsf{e}})$
    return $\mathbf{k}_{\mathsf{e}}$

$\underline{\mathsf{LWEUE.TG}(\mathbf{k}_{\mathsf{e}}, \mathbf{k}_{\mathsf{e}+1}):}$
    parse $\mathbf{k}_{\mathsf{e}} = (\mathbf{s}_{\mathsf{e}}, \mathbf{p}_{\mathsf{e}})$
    parse $\mathbf{k}_{\mathsf{e}+1} = (\mathbf{s}_{\mathsf{e}+1}, \mathbf{p}_{\mathsf{e}+1})$
    $\Delta^{\mathbf{s}}_{\mathsf{e}+1} \leftarrow \mathbf{s}_{\mathsf{e}+1} - \mathbf{s}_{\mathsf{e}}$
    $\Delta_{\mathsf{e}+1} \leftarrow (\Delta^{\mathbf{s}}_{\mathsf{e}+1}, \mathbf{p}_{\mathsf{e}+1})$
    return $\Delta_{\mathsf{e}+1}$

$\underline{\mathsf{LWEUE.Enc}(\mathbf{k}_{\mathsf{e}}, \mathbf{m}):}$
    parse $\mathbf{k}_{\mathsf{e}} = (\mathbf{s}_{\mathsf{e}}, \mathbf{p}_{\mathsf{e}})$
    $\mathbf{c}_{\mathsf{e}} \leftarrow \mathsf{LWEPKE.Enc}(\mathbf{p}_{\mathsf{e}}, \mathbf{m})$
    return $\mathbf{c}_{\mathsf{e}}$

$\underline{\mathsf{LWEUE.Dec}(\mathbf{k}_{\mathsf{e}}, \mathbf{c}_{\mathsf{e}}):}$
    parse $\mathbf{k}_{\mathsf{e}} = (\mathbf{s}_{\mathsf{e}}, \mathbf{p}_{\mathsf{e}})$
    $\mathbf{m}' \leftarrow \mathsf{LWEPKE.Dec}(\mathbf{s}_{\mathsf{e}}, \mathbf{c}_{\mathsf{e}})$
    return $\mathbf{m}'$

$\underline{\mathsf{LWEUE.Upd}(\Delta_{\mathsf{e}+1}, \mathbf{c}_{\mathsf{e}}):}$
    parse $\Delta_{\mathsf{e}+1} = (\Delta^{\mathbf{s}}_{\mathsf{e}+1}, \mathbf{p}_{\mathsf{e}+1})$
    parse $\mathbf{c}_{\mathsf{e}} = (\mathbf{C}^1_{\mathsf{e}}, \mathbf{c}^2_{\mathsf{e}})$
    $(\mathbf{C}^1, \mathbf{c}^2) \xleftarrow{\$} \mathsf{LWEPKE.Enc}(\mathbf{p}_{\mathsf{e}+1}, \mathbf{0})$
    $\mathbf{C}^1_{\mathsf{e}+1} \leftarrow \mathbf{C}^1_{\mathsf{e}} + \mathbf{C}^1$
    $\mathbf{c}^2_{\mathsf{e}+1} \leftarrow \mathbf{c}^2_{\mathsf{e}} + (\Delta^{\mathbf{s}}_{\mathsf{e}+1})^{\mathsf{T}} \cdot \mathbf{C}^1_{\mathsf{e}} + \mathbf{c}^2$
    $\mathbf{c}_{\mathsf{e}+1} \leftarrow (\mathbf{C}^1_{\mathsf{e}+1}, \mathbf{c}^2_{\mathsf{e}+1})$
    return $\mathbf{c}_{\mathsf{e}+1}$
```

Figure 22: The algorithms of LWE-based updatable encryption scheme LWEUE, which is parameterized by an LWE-based PKE scheme LWEPKE.

to be $\mathbf{p}_i = \mathbf{A} \cdot \mathbf{s}_i + \mathbf{e}_i$, where $\mathbf{e}_i \leftarrow D^m_{\mathbb{Z},\alpha}$. Then the update token is set as $\Delta_i = (\mathbf{s}_i - \mathbf{s}_{i-1}, \mathbf{p}_i)$.

In a confidentiality game for such UE schemes, suppose the adversary knows two consecutive tokens Δ_{i-1} and Δ_i. Using these tokens the adversary can compute $\mathbf{p}_i - \mathbf{p}_{i-1} - \mathbf{A} \cdot \Delta^{\mathbf{s}}_i = \mathbf{e}_i - \mathbf{e}_{i-1}$, and knows $\mathbf{e}_i - \mathbf{e}_{i-1}$. Which means if the adversary knows a set of consecutive tokens $\Delta_i, \Delta_{i+1}, \ldots, \Delta_{i+j}$ then it will also knows $\{\mathbf{e}_{i+1} - \mathbf{e}_i, \mathbf{e}_{i+2} - \mathbf{e}_i, \ldots, \mathbf{e}_{i+j} - \mathbf{e}_i\}$, the values in this set are sampled from a discrete Gaussian distribution centered at $\mathbf{e}_i$. Through evaluating these errors the adversary can possibly find the error value $\mathbf{e}_i$ and therefore knows the secret value $\mathbf{s}_i$. Furthermore, the adversary is allowed to ask for a challenge-equal ciphertext in epoch i, which will not trigger the trivial win condition, and can therefore break this confidentiality game. The above attack shows that this epoch key generation approach is not safe, it might leak the secret epoch key information.

We choose to generate a fresh pair $(\Delta^{\mathsf{s}}_{\mathsf{e}+1}, \Delta^{\mathbf{p}}_{\mathsf{e}+1})$ to compute the new epoch key and the update token, which makes sure the update token $\Delta_{\mathsf{e}+1} = (\Delta^{\mathsf{s}}_{\mathsf{e}+1}, \mathbf{p}_{\mathsf{e}+1})$ is independent from the previous epoch key. Additionally, this pair is computationally indistinguishable from a uniformly random pair as long as the underlying LWE problem is hard.

5.3 Correctness

Errors in updated ciphertexts increase when they are updated. Since the total number of epoch is bounded with a comparatively small integer l, the UE scheme supports a limited number of ciphertext updates. As a result, errors in updated ciphertexts will not grow too big and the decryption will be correct with overwhelming probability for some parameter setting.

Following the correctness analysis of the underlying PKE scheme, encrypted ciphertexts decrypt to the correct message with overwhelming probability. So, we only need to consider if updated ciphertexts will decrypt to the correct message.

First, assume a ciphertext is encrypted in epoch e and that it will be updated from epoch e to epoch $\mathsf{e}+1$. For any $\mathbf{m} \in \mathcal{M}$, suppose $\mathbf{p}_{\mathsf{e}} = \mathbf{A} \cdot \mathbf{s}_{\mathsf{e}} + \sum_{i=0}^{\mathsf{e}} \mathbf{e}_i$, $\mathbf{p}_{\mathsf{e}+1} = \mathbf{A} \cdot \mathbf{s}_{\mathsf{e}+1} + \sum_{i=0}^{\mathsf{e}+1} \mathbf{e}_i$, $(\mathbf{C}^1_{\mathsf{e}}, \mathbf{c}^2_{\mathsf{e}}) = (\mathbf{A}^{\mathsf{T}} \cdot \mathbf{R}_{\mathsf{e}}, \mathbf{p}^{\mathsf{T}}_{\mathsf{e}} \cdot \mathbf{R}_{\mathsf{e}} + \mathbf{e}'_{\mathsf{e}} + \frac{q}{2}\mathbf{m})$ and $(\mathbf{C}^1, \mathbf{c}^2) = (\mathbf{A}^{\mathsf{T}} \cdot \mathbf{R}_{\mathsf{e}+1}, \mathbf{p}^{\mathsf{T}}_{\mathsf{e}+1} \cdot \mathbf{R}_{\mathsf{e}+1} + \mathbf{e}'_{\mathsf{e}+1})$. Then

$$\begin{aligned}\mathbf{C}^1_{\mathsf{e}+1} &= \mathbf{C}^1_{\mathsf{e}} + \mathbf{C}^1 \qquad (6)\\ &= \mathbf{A}^{\mathsf{T}} \cdot (\mathbf{R}_{\mathsf{e}} + \mathbf{R}_{\mathsf{e}+1}),\end{aligned}$$

and

$$\begin{aligned}\mathbf{c}^2_{\mathsf{e}+1} &= \mathbf{c}^2_{\mathsf{e}} + (\Delta^{\mathsf{s}}_{\mathsf{e}+1})^{\mathsf{T}} \cdot \mathbf{C}^1_{\mathsf{e}} + \mathbf{c}^2\\ &= \mathbf{p}^{\mathsf{T}}_{\mathsf{e}} \cdot \mathbf{R}_{\mathsf{e}} + \mathbf{p}^{\mathsf{T}}_{\mathsf{e}+1} \cdot \mathbf{R}_{\mathsf{e}+1} + (\mathbf{s}_{\mathsf{e}+1} - \mathbf{s}_{\mathsf{e}})^{\mathsf{T}} \cdot \mathbf{A}^{\mathsf{T}} \cdot \mathbf{R}_{\mathsf{e}} + \mathbf{e}'_{\mathsf{e}} + \mathbf{e}'_{\mathsf{e}+1} + \frac{q}{2}\mathbf{m}\\ &= \mathbf{p}^{\mathsf{T}}_{\mathsf{e}+1} \cdot (\mathbf{R}_{\mathsf{e}+1} + \mathbf{R}_{\mathsf{e}}) - \mathbf{e}^{\mathsf{T}}_{\mathsf{e}+1} \cdot \mathbf{R}_{\mathsf{e}} + \mathbf{e}'_{\mathsf{e}} + \mathbf{e}'_{\mathsf{e}+1} + \frac{q}{2}\mathbf{m}\\ &\overset{s}{\approx} \mathbf{p}^{\mathsf{T}}_{\mathsf{e}+1} \cdot (\mathbf{R}_{\mathsf{e}+1} + \mathbf{R}_{\mathsf{e}}) + (\mathbf{e}'_{\mathsf{e}} + \mathbf{e}'_{\mathsf{e}+1}) + \frac{q}{2}\mathbf{m} \qquad (7)\\ &= (\mathbf{p}^{\mathsf{T}}_{\mathsf{e}+1} \cdot \mathbf{R}_{\mathsf{e}} + \mathbf{e}'_{\mathsf{e}}) + \mathbf{c}^2 + \frac{q}{2}\mathbf{m}. \qquad (8)\end{aligned}$$

Note that by Lemma A.1 and Lemma A.2, Equation (7) holds. Notice that the updated ciphertext is of the same shape as the encrypted ciphertext with the new randomness $(\mathbf{R}_{\mathsf{e}}+\mathbf{R}_{\mathsf{e}+1})$ and new error $(\mathbf{e}'_{\mathsf{e}}+\mathbf{e}'_{\mathsf{e}+1})$. Which means when we update a ciphertext multiple times the updated ciphertext will keep the same shape as well, only with a bigger randomness and bigger error.

Iteratively, we consider ciphertext $\mathbf{c}_{\mathsf{e}'}$ to be an updated ciphertext updated from epoch e to epoch e', where $0 \leq \mathsf{e}' - \mathsf{e} < l$. Assume $\mathbf{c}_{\mathsf{e}}$ is the original encryption. For $0 \leq i \leq \mathsf{e}' - \mathsf{e}$, let the re-randomization performed in epoch $\mathsf{e} + i$ outputs $(\mathbf{A}^{\mathsf{T}} \cdot \mathbf{R}_{\mathsf{e}+i}, \mathbf{p}^{\mathsf{T}}_{\mathsf{e}+1} \cdot \mathbf{R}_{\mathsf{e}+i} + \mathbf{e}'_{\mathsf{e}+i})$. Then

$$\mathbf{C}^1_{\mathsf{e}'} = \mathbf{A}^{\mathsf{T}} \cdot \sum_{i=0}^{\mathsf{e}'-\mathsf{e}} \mathbf{R}_{\mathsf{e}+i},$$

and

$$\begin{aligned}
\mathbf{c}^2_{\mathsf{e}'} &= \mathbf{p}^{\mathsf{T}}_{\mathsf{e}'} \cdot \sum_{i=0}^{\mathsf{e}'-\mathsf{e}} \mathbf{R}_{\mathsf{e}+i} + \sum_{i=0}^{\mathsf{e}'-\mathsf{e}} \mathbf{e}'_{\mathsf{e}+i} + \frac{q}{2}\mathbf{m} - (\sum_{i=1}^{\mathsf{e}'-\mathsf{e}} \mathbf{e}^{\mathsf{T}}_{\mathsf{e}+i} \cdot (\sum_{j=0}^{i-1} \mathbf{R}_{\mathsf{e}+j})) \\
&\overset{s}{\approx} \mathbf{p}^{\mathsf{T}}_{\mathsf{e}'} \cdot \sum_{i=0}^{\mathsf{e}'-\mathsf{e}} \mathbf{R}_{\mathsf{e}+i} + \sum_{i=0}^{\mathsf{e}'-\mathsf{e}} \mathbf{e}'_{\mathsf{e}+i} + \frac{q}{2}\mathbf{m}. \qquad (9)
\end{aligned}$$

By Lemma A.1 and Lemma A.2, Equation (9) holds. Then we have $\mathbf{c}^2_{\mathsf{e}'} - \mathbf{s}^{\mathsf{T}}_{\mathsf{e}'} \cdot \mathbf{C}^1_{\mathsf{e}'} \overset{s}{\approx} \sum_{i=0}^{\mathsf{e}'} \mathbf{e}^{\mathsf{T}}_i \cdot \sum_{i=0}^{\mathsf{e}'-\mathsf{e}} \mathbf{R}_{\mathsf{e}+i} + \sum_{i=0}^{\mathsf{e}'-\mathsf{e}} \mathbf{e}'_{\mathsf{e}+i} + \frac{q}{2}\mathbf{m} \overset{s}{\approx} \sum_{i=0}^{\mathsf{e}'-\mathsf{e}} \mathbf{e}'_{\mathsf{e}+i} + \frac{q}{2}\mathbf{m}$. By Lemma A.3, each entry of $\sum_{i=0}^{\mathsf{e}'-\mathsf{e}} \mathbf{e}'_{\mathsf{e}+i}$ is of the size at most $\sqrt{\mathsf{e}' - \mathsf{e}} \cdot \beta$ with overwhelming probability, which has distance at most $\frac{q}{8}$ from $0 \mod q$ because $\beta \leq \frac{q}{8\sqrt{l}}$ and $\mathsf{e}' - \mathsf{e} < l$. In summary, $\mathsf{LWEUE.Dec}$ decrypts correctly with overwhelming probability.

5.4 Challenges of the Security Proof in LWE-based UE Schemes

In this section we highlight the difficulties when proving that LWEUE is a secure UE scheme, specifically, our UE scheme has a randomized update algorithm. Lehmann and Tackmann [LT18] and Klooß et al. [KLR19] both described a method, similar to each other, to prove that updatable encryption schemes with randomized update algorithms are secure. Their technique can be seen when they prove that RISE and NYUE (NYUAE) are secure, resp. However, this method can not be directly used to prove that LWEUE

is secure. The method introduced requires that UE schemes have perfect re-encryption, which means the distribution of updated ciphertexts has the same distribution as fresh encryptions. In their proof, they replace updated ciphertexts by fresh encryptions of the underlying messages. However, in the LWEUE scheme, we cannot simply replace updated ciphertexts by a fresh encryption because the randomness terms and the error terms grow while updating and an updated ciphertext does not have the same distribution as a fresh encryption.

5.5 Security

If LWEPKE is IND\$-CPA-secure then the output of the encryption algorithm is computationally indistinguishable from a pair of uniformly random elements. Hence, the fresh encryption in the LWEUE scheme is computationally indistinguishable from a pair of uniformly random elements as well. Furthermore, the update algorithm LWEUE.Upd runs the encryption algorithm of LWEPKE to re-randomize the old ciphertext to a new ciphertext, therefore, the updated ciphertext is also computationally indistinguishable from a pair of uniformly random elements. So, a fresh encryption is computationally indistinguishable from an updated ciphertext and LWEUE is randIND-UE-CPA secure (see Definition 5). This provides the underlying intuition for the security proof.

5.5.1 Technical Simulations in the Proof

Simulate Public Keys. Suppose $\mathbf{p}_0 = \mathbf{A} \cdot \mathbf{s}_0 + \mathbf{e}_0$ and $(\Delta_i^{\mathsf{s}}, \Delta_i^{\mathbf{p}}) = (\Delta_i^{\mathsf{s}}, \mathbf{A} \cdot \Delta_i^{\mathsf{s}} + \mathbf{e}_i)$ for $i > 0$, where $\mathbf{s}_0, \Delta_i^{\mathsf{s}} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$ and $\mathbf{e}_i \leftarrow D_{\mathbb{Z},\alpha}^m$. Then

$$\begin{aligned} \mathbf{p}_{\mathsf{e}} &= \mathbf{p}_{\mathsf{e}-1} + \mathbf{A} \cdot \Delta_{\mathsf{e}}^{\mathsf{s}} + \mathbf{e}_{\mathsf{e}} \\ &= \mathbf{p}_j + \mathbf{A} \cdot \Big(\sum_{i=j+1}^{\mathsf{e}} \Delta_i^{\mathsf{s}} \Big) + \sum_{i=j+1}^{\mathsf{e}} \mathbf{e}_i \qquad (10) \\ &= \mathbf{A} \cdot \mathbf{s}_{\mathsf{e}} + \sum_{i=0}^{\mathsf{e}} \mathbf{e}_i . \qquad (11) \end{aligned}$$

Equation (10) shows that any public key can be simulated by any other public key and tokens. Equation (11) shows that any public key can be simulated by the corresponding secret key.

Simulate Updated Ciphertexts. Then we consider how to simulate updated ciphertexts by only using public keys. We will have a bookkeeping in set $\mathcal{L}$ to track the randomness $\mathbf{R}$ and the big error $\mathbf{e}'$. Specifically, we record $\mathcal{L}$ as $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathrm{c}, \mathbf{c}_{\mathsf{e}}, \mathsf{e}; \mathbf{R}_{\mathsf{e}}, \mathbf{e}'_{\mathsf{e}}, \mathbf{m})\}$.

To simulate an updated ciphertext from epoch e to epoch $\mathsf{e}+1$, suppose $(\mathrm{c}, \mathbf{c}_{\mathsf{e}}, \mathsf{e}; \mathbf{R}_{\mathsf{e}}, \mathbf{e}'_{\mathsf{e}}, \mathbf{m}) \in \mathcal{L}$, and the new public key is $\mathbf{p}_{\mathsf{e}+1}$. We use the new public key to perform the re-randomization, to produce $(\mathbf{C}^1, \mathbf{c}^2)$. Suppose $(\mathbf{C}^1, \mathbf{c}^2) = (\mathbf{A}^\intercal \cdot \mathbf{R}, \mathbf{p}^\intercal_{\mathsf{e}+1} \cdot \mathbf{R} + \mathbf{e}')$, then we use Equations (6) and (8) to simulate the updated ciphertext $\mathbf{c}_{\mathsf{e}+1} = (\mathbf{C}^1_{\mathsf{e}+1}, \mathbf{c}^2_{\mathsf{e}+1})$, where $\mathbf{C}^1_{\mathsf{e}+1} = \mathbf{C}^1_{\mathsf{e}} + \mathbf{C}^1$ and $\mathbf{c}^2_{\mathsf{e}+1} = (\mathbf{p}^\intercal_{\mathsf{e}+1} \cdot \mathbf{R}_{\mathsf{e}} + \mathbf{e}'_{\mathsf{e}} + \frac{q}{2}\mathbf{m}) + \mathbf{c}^2$. After simulation, the set $\mathcal{L}$ will store the updated ciphertext with the updated randomness $\mathbf{R}_{\mathsf{e}} + \mathbf{R}$ and the updated error $\mathbf{e}'_{\mathsf{e}} + \mathbf{e}'$, i.e. $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathrm{c}, \mathbf{c}_{\mathsf{e}+1}, \mathsf{e}+1; \mathbf{R}_{\mathsf{e}} + \mathbf{R}, \mathbf{e}'_{\mathsf{e}} + \mathbf{e}', \mathbf{m})\}$. We use the re-randomization algorithm reR described in Fig. 23 to simulate updated ciphertexts.

```
$\underline{\mathsf{reR}(\mathbf{p}_{\mathsf{e}+1}, \mathbf{c}_{\mathsf{e}} = (\mathbf{C}^1_{\mathsf{e}}, \mathbf{c}^2_{\mathsf{e}}); \mathbf{R}_{\mathsf{e}}, \mathbf{e}'_{\mathsf{e}}, \mathbf{m}):}$
  $\mathbf{R} \leftarrow \mathcal{D}^t_r$
  $\mathbf{e}' \leftarrow D^{1\times t}_{\mathbb{Z},\beta}$
  $\mathbf{C}^1 \leftarrow \mathbf{A}^\intercal \cdot \mathbf{R}$
  $\mathbf{c}^2 \leftarrow \mathbf{p}^\intercal_{\mathsf{e}+1} \cdot \mathbf{R} + \mathbf{e}'$
  $\mathbf{C}^1_{\mathsf{e}+1} \leftarrow \mathbf{C}^1_{\mathsf{e}} + \mathbf{C}^1$
  $\mathbf{c}^2_{\mathsf{e}+1} \leftarrow (\mathbf{p}^\intercal_{\mathsf{e}+1} \cdot \mathbf{R}_{\mathsf{e}} + \mathbf{e}'_{\mathsf{e}} + \frac{q}{2}\mathbf{m}) + \mathbf{c}^2$
  $\mathbf{R}_{\mathsf{e}+1} \leftarrow \mathbf{R}_{\mathsf{e}} + \mathbf{R}$
  $\mathbf{e}'_{\mathsf{e}+1} \leftarrow \mathbf{e}'_{\mathsf{e}} + \mathbf{e}'$
  **return** $(\mathbf{c}_{\mathsf{e}+1}; \mathbf{R}_{\mathsf{e}+1}, \mathbf{e}'_{\mathsf{e}+1})$
```

Figure 23: Algorithm reR, used to simulate updated ciphertexts.

5.5.2 LWEUE is randIND-UE-CPA

Theorem 5.1 (LWEUE is randIND-UE-CPA)**.** Let LWEUE be the updatable encryption scheme described in Fig. 22, using parameter setting described in Section 5.1. For any randIND-UE-CPA adversary $\mathcal{A}$ against LWEUE, there exists an adversary $\mathcal{B}_{5.1}$ against $\mathsf{DLWE}_{n,q,\alpha}$ such that

$$\mathbf{Adv}^{\mathsf{randIND\text{-}UE\text{-}CPA}}_{\mathsf{LWEUE},\,\mathcal{A}}(1^\lambda) \leq 2(l+1)^3 \cdot \big(t\epsilon + 3\mathbf{Adv}^{\mathsf{DLWE}}_{n,q,\alpha}(\mathcal{B}_{5.1}) + \mathsf{negl}(n)\big).$$

Overview of the Security Proof. We now explain how we bound the advantage of any adversary playing the randIND-UE-CPA game for LWEUE by the DLWE advantage.

To prove the security, we play a hybrid game over epochs, where the reduction constructs one hybrid for each epoch. In hybrid i, to the left of epoch i the game returns real challenge-equal ciphertexts, to the right of epoch i the game returns random ciphertexts. This means we have one hybrid for each epoch, moving real-to-random across the epoch space.

In each hybrid, we play an intermediate sequence of games. In the first game, we apply the firewall technique to set up a modified hybrid game. This modification ensures that the reduction (we will construct this reduction later) can simulate the modified hybrid game because it can provide valid keys and tokens to the adversary and easily check the trivial win conditions. In the second game, we change how the update algorithm runs by only using the public key without the update token (see the technique in Section 5.5.1), which helps the reduction to simulate updated ciphertexts. It is impossible to simulate updated ciphertext across firewalls, if the token is needed when updating, as the reduction does not know tokens in both firewalls. Our scheme makes it possible to generate updated ciphertexts using only the public key. In the third game, we change how public keys are generated, which makes sure that the reduction can simulate valid public keys. Eventually, we construct a reduction playing the IND\$-CPA (the IND\$-CPA advantage is upper bounded by the DLWE advantage) game by simulating the third game to an adversary. The IND\$-CPA game moves the real challenge-equal ciphertext in epoch i to a random ciphertext, which means the fresh encryption is computationally indistinguishable from the updated ciphertext.

Proof. In this proof, we use three steps to reach our desired goal. In the first step, we use a sequence of hybrid games H_i to bound the randIND-UE-CPA advantage. In the second step, we bound the advantage of each hybrid game to the advantage of a modified hybrid game $\mathcal{G}_i$. In the third step, we bound the advantage of the modified hybrid game $\mathcal{G}_i$ to the DLWE advantage, using an intermediate sequence of games.

Step 1. For $\mathrm{b} \in \{0, 1\}$, we construct a sequence of hybrid games $H_0^{\mathrm{b}}, \ldots, H_{l+1}^{\mathrm{b}}$. In game H_i^{b}, if the adversary asks for a challenge-equal ciphertext by the $\mathcal{O}.\mathsf{Chall}$ query or a $\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$ query, with challenge input $(\bar{\mathbf{m}}, \bar{\mathbf{c}})$, in epoch j:

- if $j < i$, then return a real challenge-equal ciphertext, which is (updated from) the encryption of $\bar{\mathbf{m}}$ if $\mathrm{b} = 0$ or an updated ciphertext of $\bar{\mathbf{c}}$ if $\mathrm{b} = 1$,
- if $j \geq i$, return a random ciphertext.

Thus H_{l+1}^{b}, $i = l + 1$, is $\mathbf{Exp}_{\mathsf{LWEUE},\mathcal{A}}^{\mathsf{randIND\text{-}UE\text{-}CPA\text{-}b}}$, i.e. all challenge responses are real challenge-equal ciphertexts. Notice that $H_0^0 = H_0^1$, $i = 0$, because all challenge responses are random ciphertexts. We have

$$\begin{aligned}\mathbf{Adv}_{\mathsf{LWEUE},\,\mathcal{A}}^{\mathsf{randIND\text{-}UE\text{-}CPA}}(1^\lambda) =& \left|\mathbf{Pr}[H_{l+1}^1 = 1] - \mathbf{Pr}[H_{l+1}^0 = 1]\right| \\ \leq& \sum_{i=1}^{l+1} \left|\mathbf{Pr}[H_i^1 = 1] - \mathbf{Pr}[H_{i-1}^1 = 1]\right| \\ &+ \sum_{i=1}^{l+1} \left|\mathbf{Pr}[H_i^0 = 1] - \mathbf{Pr}[H_{i-1}^0 = 1]\right|.\end{aligned}$$

Step 2. In Hybrid i, for $\mathrm{b} \in \{0, 1\}$, let $\mathcal{A}_i'$ be an adversary trying to distinguish game H_i^{b} from game H_{i-1}^{b}. For all queries concerning epochs other than i the responses will be equal in either game, so we assume that $\mathcal{A}_i'$ asks for a challenge-equal ciphertext in epoch i, therefore there exist two epochs[4] (denote $\mathsf{fwl}, \mathsf{fwr}$) around the epoch i such that no key in the sequence of epochs $(\mathsf{fwl}, \ldots, \mathsf{fwr})$ and no token in epochs fwl and $\mathsf{fwr} + 1$ are corrupted.

[4]This observation was introduced in the work of Lehmann and Tackmann [LT18], Klooß et al. [KLR19] provided an extended description of this *key insulation* technique, and Boyd et al. [BDGJ20] formally defined it as *firewall technique*.

Furthermore, $H_j^{\mathrm{b}} = H_0^{\mathrm{b}}$, for all $j < \tilde{\mathsf{e}}$, since the adversary never asks for a challenge-equal ciphertext before the challenge epoch ($\tilde{\mathsf{e}}$). So we assume $i \geq \tilde{\mathsf{e}}$.

Define a new game $\mathcal{G}_i^{\mathrm{b}}$ that is the same as game H_i^{b}, except for the game randomly picks two numbers $\mathsf{fwl}, \mathsf{fwr} \xleftarrow{\$} \{0, \dots, l\}$. If the adversary corrupts a key in the sequence of epochs $(\mathsf{fwl}, \dots, \mathsf{fwr})$ or a token in epochs fwl and $\mathsf{fwr} + 1$, the game aborts. This loss is upper bounded by $(l + 1)^2$. Then we have

$$|\mathbf{Pr}[H_i^{\mathrm{b}} = 1] - \mathbf{Pr}[H_{i-1}^{\mathrm{b}} = 1]| \leq (l + 1)^2 |\mathbf{Pr}[\mathcal{G}_i^{\mathrm{b}} = 1] - \mathbf{Pr}[\mathcal{G}_{i-1}^{\mathrm{b}} = 1]|.$$

Step 3. In this step, we will prove that $|\mathbf{Pr}[\mathcal{G}_i^{\mathrm{b}} = 1] - \mathbf{Pr}[\mathcal{G}_{i-1}^{\mathrm{b}} = 1]|$ is upper bounded by the DLWE advantage.

Claim 5.1.1. For any $\mathrm{b} \in \{0, 1\}$, there exists an adversary $\mathcal{B}_{5.1}$ against $\mathsf{DLWE}_{n,q,\alpha}$ such that

$$|\mathbf{Pr}[\mathcal{G}_i^{\mathrm{b}} = 1] - \mathbf{Pr}[\mathcal{G}_{i-1}^{\mathrm{b}} = 1]| \leq t\epsilon + 3\mathbf{Adv}_{n,q,\alpha}^{\mathsf{DLWE}}(\mathcal{B}_{5.1}) + \mathsf{negl}(n).$$

Proof of Claim 5.1.1. Assume $\mathcal{A}_i$ is an adversary trying to distinguish $\mathcal{G}_i^{\mathrm{b}}$ from $\mathcal{G}_{i-1}^{\mathrm{b}}$, and $\mathcal{A}_i$ asks for a challenge-equal ciphertext in epoch i ($i \geq \tilde{\mathsf{e}}$). The proof of Claim 5.1.1 consists of the following sequence of games.

Game 0

The game flips a coin $d \xleftarrow{\$} \{0, 1\}$, if $d = 0$ it plays game $\mathcal{G}_{i-1}^{\mathrm{b}}$ (responses a real challenge-equal ciphertext in epoch i) to $\mathcal{A}_i$, if $d = 1$ it plays game $\mathcal{G}_i^{\mathrm{b}}$ (responses a random ciphertext in epoch i) to $\mathcal{A}_i$. Denote E_i be the event that the adversary succeeds in guessing d in game i.

Then we have

$$\mathbf{Pr}[E_0] = |\mathbf{Pr}[\mathcal{G}_i^{\mathrm{b}} = 1] - \mathbf{Pr}[\mathcal{G}_{i-1}^{\mathrm{b}} = 1]|.$$

Game 1

We consider a modified game that is the same as Game 0, except for the updated ciphertext is generated by using only public key without the secret token. More precisely, the game running the update algorithm as follows.

```
$\underline{\mathcal{O}.\mathsf{Upd}(\mathbf{c}_{\mathsf{e}-1}):}$
  **if** $(\cdot, \mathbf{c}_{\mathsf{e}-1}, \mathsf{e}-1; \mathbf{R}_{\mathsf{e}-1}, \mathbf{e'}_{\mathsf{e}-1}, \mathbf{m}) \notin \mathcal{L}$ **then**
    **return** $\perp$
  $(\mathbf{c}_{\mathsf{e}}; \mathbf{R}_{\mathsf{e}}, \mathbf{e'}_{\mathsf{e}}) \leftarrow \mathsf{reR}(\mathbf{p}_{\mathsf{e}}, \mathbf{c}_{\mathsf{e}-1}; \mathbf{R}_{\mathsf{e}-1}, \mathbf{e'}_{\mathsf{e}-1}, \mathbf{m})$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathbf{c}_{\mathsf{e}}, \mathsf{e}; \mathbf{R}_{\mathsf{e}}, \mathbf{e'}_{\mathsf{e}}, \mathbf{m})\}$
  **return** $\mathbf{c}_{\mathsf{e}}$
```

The re-randomization algorithm reR is described in Fig. 23. By the analysis in Section 5.5.1, we have

$$|\mathbf{Pr}[E_1] - \mathbf{Pr}[E_0]| \leq \mathsf{negl}(n).$$

Game 2

We consider a modified game that is the same as Game 1, except for the public key in epoch fwl is generated with one additional error and the public key in epoch $\mathsf{fwr}+1$ is generated with one error less. In particular, let $\Delta^{\mathbf{s}}_{\mathsf{fwl}}, \Delta^{\mathbf{s}}_{\mathsf{fwr}+1} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$ and $\mathbf{e}_{\mathsf{fwl}}, \mathbf{e} \leftarrow D^m_{\mathbb{Z},\alpha}$, these public keys are generated as follows.

$$\mathbf{p}_{\mathsf{fwl}} = \mathbf{p}_{\mathsf{fwl}-1} + \mathbf{A} \cdot \Delta^{\mathbf{s}}_{\mathsf{fwl}} + \mathbf{e}_{\mathsf{fwl}} + \mathbf{e},$$

$$\mathbf{p}_{\mathsf{fwr}+1} = \mathbf{p}_{\mathsf{fwr}} + \mathbf{A} \cdot \Delta^{\mathbf{s}}_{\mathsf{fwr}+1} - \mathbf{e}.$$

We claim that, in Game 2, all public key distributions except for $\mathbf{p}_{\mathsf{fwl}}$ are equal to the corresponding real public key distributions. Furthermore, the public key $\mathbf{p}_{\mathsf{fwl}}$ produced in Game 2 is computationally indistinguishable from the real public key $\mathbf{p}_{\mathsf{fwl}}$.

Any public key that is in an epoch smaller than fwl, or greater than $\mathsf{fwr}+1$, are generated by algorithm LWEUE.KG in both games. So we only consider public keys from epoch fwl to epoch $\mathsf{fwr}+1$. In Game 2, the public keys from epoch fwl to epoch $\mathsf{fwr}+1$ are computed as follows.

$$\mathbf{p}_{\mathsf{fwl}} = \mathbf{p}_{\mathsf{fwl}-1} + \mathbf{A} \cdot \Delta^{\mathbf{s}}_{\mathsf{fwl}} + \mathbf{e}_{\mathsf{fwl}} + \mathbf{e}, \tag{12}$$

$$\cdots$$

$$\mathbf{p}_i = \mathbf{p}_{i-1} + \mathbf{A} \cdot \Delta^{\mathbf{s}}_i + \mathbf{e}_i, \tag{13}$$

$$\cdots$$

$$\mathbf{p}_{\mathsf{fwr}} = \mathbf{p}_{\mathsf{fwr}-1} + \mathbf{A} \cdot \Delta^{\mathbf{s}}_{\mathsf{fwr}} + \mathbf{e}_{\mathsf{fwr}}, \tag{14}$$

$$\mathbf{p}_{\mathsf{fwr}+1} = \mathbf{p}_{\mathsf{fwr}} + \mathbf{A} \cdot \Delta^{\mathbf{s}}_{\mathsf{fwr}+1} - \mathbf{e}. \tag{15}$$

Based on Equation (13), (14), and (15), we know that public keys $\mathbf{p}_{\mathsf{fwl}+1}$, …, $\mathbf{p}_{\mathsf{fwr}}, \mathbf{p}_{\mathsf{fwr}+1}$ are generated as real public keys by running algorithm LWEUE.KG. Furthermore, note that $\mathbf{A} \cdot \Delta^{\mathbf{s}}_{\mathsf{fwl}} + \mathbf{e}_{\mathsf{fwl}}$ is computationally indistinguishable from a uniform random element in $\mathbb{Z}_q^m$. Therefore, the public value $\mathbf{p}_{\mathsf{fwl}}$ in both games are computationally indistinguishable from a uniform random element in $\mathbb{Z}_q^m$, under the $\mathsf{DLWE}_{n,q,\alpha}$ problem. Then we have

$$|\mathbf{Pr}[E_2] - \mathbf{Pr}[E_1]| \le 2 \cdot \mathbf{Adv}^{\mathsf{DLWE}}_{n,q,\alpha}.$$

Now we are almost finished, all we need to prove now is that $\mathbf{Pr}[E_2] \le \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{LWEPKE},\, \tilde{\mathcal{B}}_{5.1}}(1^\lambda) + \mathsf{negl}(n)$, which we claim is true in Claim 5.1.2. Finally, by Theorem 4.1, we get $\mathbf{Pr}[E_2] \le t\epsilon + \mathbf{Adv}^{\mathsf{DLWE}}_{n,q,\alpha} + \mathsf{negl}(n)$, which concludes the proof of Claim 5.1.1.

Claim 5.1.2. $\mathbf{Pr}[E_2] \le \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{LWEPKE},\, \tilde{\mathcal{B}}_{5.1}}(1^\lambda) + \mathsf{negl}(n)$.

Proof of Claim 5.1.2. Before we start constructing a reduction to simulate Game 2 we need the following equations, describing the public key in an epoch j:

- For $\mathsf{fwl} < j \le \mathsf{fwr}$, we have

$$\begin{aligned}
\mathbf{p}_j &= \mathbf{p}_{j-1} + \mathbf{A} \cdot \Delta^{\mathbf{s}}_j + \mathbf{e}_j \\
&= \mathbf{p}_{\mathsf{fwl}} + \mathbf{A} \cdot \Big(\sum_{k=\mathsf{fwl}+1}^{j} \Delta^{\mathbf{s}}_k \Big) + \sum_{k=\mathsf{fwl}+1}^{j} \mathbf{e}_k \\
&= (\mathbf{p}_{\mathsf{fwl}-1} + \mathbf{A} \cdot \Delta^{\mathbf{s}}_{\mathsf{fwl}} + \mathbf{e}_{\mathsf{fwl}} + \mathbf{e}) + \mathbf{A} \cdot \Big(\sum_{k=\mathsf{fwl}+1}^{j} \Delta^{\mathbf{s}}_k \Big) + \sum_{k=\mathsf{fwl}+1}^{j} \mathbf{e}_k \\
&= \mathbf{e} + \mathbf{A} \cdot (\mathbf{s}_0 + \sum_{k=1}^{j} \Delta^{\mathbf{s}}_k) + \sum_{k=0}^{j} \mathbf{e}_k \\
&= (\mathbf{A} \cdot \mathbf{s}_j + \mathbf{e}) + \sum_{k=0}^{j} \mathbf{e}_k, \qquad (16)
\end{aligned}$$

Equation (16) also holds for $j = \mathsf{fwl}$, it follows from Equation (11) and (12). This equation will be used to simulate the public key in epoch i.

- For $i < j \leq \mathsf{fwr}$, we have

$$\mathbf{p}_j = \mathbf{p}_i + \big(\mathbf{A} \cdot (\sum_{k=i+1}^{j} \Delta_k^{\mathbf{s}}) + \sum_{k=i+1}^{j} \mathbf{e}_k\big). \tag{17}$$

- Similarly, for $\mathsf{fwl} \leq j < i$, we have

$$\mathbf{p}_j = \mathbf{p}_i - \big(\mathbf{A} \cdot (\sum_{k=j+1}^{i} \Delta_k^{\mathbf{s}}) + \sum_{k=j+1}^{i} \mathbf{e}_k\big). \tag{18}$$

- For $j = \mathsf{fwr} + 1$, we have

$$\begin{aligned}\mathbf{p}_{\mathsf{fwr}+1} &= \mathbf{p}_{\mathsf{fwr}} + \mathbf{A} \cdot \Delta_{\mathsf{fwr}+1}^{\mathbf{s}} - \mathbf{e} \\ &= (\mathbf{A} \cdot \mathbf{s}_{\mathsf{fwr}} + \mathbf{e} + \sum_{k=0}^{\mathsf{fwr}} \mathbf{e}_k) + \mathbf{A} \cdot \Delta_{\mathsf{fwr}+1}^{\mathbf{s}} - \mathbf{e} \\ &= \mathbf{A} \cdot \mathbf{s}_{\mathsf{fwr}+1} + \sum_{j=0}^{\mathsf{fwr}} \mathbf{e}_j. \end{aligned} \tag{19}$$

We construct a reduction $\tilde{\mathcal{B}}_{5.1}$, detailed in Fig. 24, that is playing the IND\$-CPA game and will simulate the responses of queries made by adversary $\mathcal{A}_i$ in game 2. Initially, the reduction guesses two numbers fwl, fwr. If $\mathcal{A}_i$ corrupts $\mathbf{k}_{\mathsf{fwl}}, \ldots, \mathbf{k}_{\mathsf{fwr}}$, Δ_{fwl}, or $\Delta_{\mathsf{fwr}+1}$ the reduction aborts the game.

A summary of the technical simulations are as follows.

- In the setup phrase,
  - $\tilde{\mathcal{B}}_{5.1}$ samples errors and tokens as follows.
    - For $j \in \{0, \ldots, \mathsf{fwr}\} \cup \{\mathsf{fwr}+2, \ldots, n\}$, $\tilde{\mathcal{B}}_{5.1}$ samples each error as $\mathbf{e}_j \leftarrow D_{\mathbb{Z},\alpha}^m$,
    - For $j \in \{\mathsf{fwl}+1, \ldots, \mathsf{fwr}\}$, $\tilde{\mathcal{B}}_{5.1}$ samples each token as $\Delta_j^{\mathbf{s}} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$.
  - $\tilde{\mathcal{B}}_{5.1}$ generates all keys and tokens, except for $\mathbf{k}_{\mathsf{fwl}}, \ldots, \mathbf{k}_{\mathsf{fwr}}$, Δ_{fwl}, $\Delta_{\mathsf{fwr}+1}$, as follows.

```
For b $\in \{0,1\}$, $\tilde{\mathcal{B}}_{5.1}$ plays
IND\$-CPA game by running $\mathcal{A}_i$ :
  receive $(\mathbf{A}, \mathbf{p})$
  do Setup
  $\mathrm{b}' \leftarrow \mathcal{A}_i^{oracles}(1^\lambda)$
  if ABORT occurred or $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$
   or $i \notin \{\mathsf{fwl}, \ldots, \mathsf{fwr}\}$ or $i < \tilde{\mathsf{e}}$ then
    $\mathrm{b}' \xleftarrow{\$} \{0,1\}$
    return $\mathrm{b}'$
  if $\mathrm{b}' = \mathrm{b}$ then
    return 0
  else
    return 1

Setup$(1^\lambda)$
  $\Delta_0 \leftarrow \perp$;  $\mathsf{e} \leftarrow 0$;  $\mathsf{phase}, \mathsf{twf} \leftarrow 0$;
  $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$
  $\mathsf{fwl}, \mathsf{fwr} \xleftarrow{\$} \{0, \ldots, l\}$
  for $j \in \{0, \ldots, n\}^{\dagger\dagger}$ do
    $\mathbf{e}_j \leftarrow D_{\mathbb{Z},\alpha}^m$
  for $j \in \{\mathsf{fwl}+1, \ldots, \mathsf{fwr}\}$ do
    $\Delta_j^{\mathbf{s}} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$
  for $j \in \{0, \ldots, \mathsf{fwl}\text{-}1\}$
   or $j \in \{\mathsf{fwr}+1, \ldots, n\}$ do
    $\mathbf{s}_j \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$; $\Delta_j^{\mathbf{s}} \leftarrow \mathbf{s}_j\text{-}\mathbf{s}_{j\text{-}1}^{\dagger}$;
    $\mathbf{p}_j \leftarrow \mathbf{A} \cdot \mathbf{s}_j + \sum_{k=0}^{j} \mathbf{e}_k^{\dagger\dagger}$
  $\mathbf{p}_i \leftarrow \mathbf{p} + \sum_{k=0}^{i} \mathbf{e}_k$
  for $j \in \{\mathsf{fwl}, \ldots, i\text{-}1\}$ do
    $\mathbf{p}_j \leftarrow \mathbf{p}\text{-}\mathbf{A}\sum_{k=j+1}^{i} \Delta_k^{\mathbf{s}} + \sum_{k=0}^{j} \mathbf{e}_k$
  for $j \in \{i+1, \ldots, \mathsf{fwr}\}$ do
    $\mathbf{p}_j \leftarrow \mathbf{p}+\mathbf{A}\sum_{k=i+1}^{j} \Delta_k^{\mathbf{s}} + \sum_{k=0}^{j} \mathbf{e}_k$
```

```
$\mathcal{O}$.Enc$(\mathbf{m})$ :
  $\mathrm{c} \leftarrow \mathrm{c}+1$
  $\mathbf{c}_{\mathsf{e}} \leftarrow \mathsf{LWEUE.Enc}(\mathbf{p}_{\mathsf{e}}, \mathbf{m})$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathrm{c}, \mathbf{c}_{\mathsf{e}}, \mathsf{e}; \mathbf{R}_{\mathsf{e}}, \mathbf{e}'_{\mathsf{e}}, \mathbf{m})\}$
  return $\mathbf{c}_{\mathsf{e}}$

$\mathcal{O}$.Next :
  $\mathsf{e} \leftarrow \mathsf{e}+1$

$\mathcal{O}$.Corr$(\mathsf{inp}, \hat{\mathsf{e}})$ :
  if $\hat{\mathsf{e}} > \mathsf{e}$ then
    return $\perp$
  if $\mathsf{inp} = \mathsf{key}$ then
    if $\hat{\mathsf{e}} \in \{\mathsf{fwl}, \ldots, \mathsf{fwr}\}$ then
      ABORT
    else
      $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
      return $\mathbf{k}_{\hat{\mathsf{e}}}$
  if $\mathsf{inp} = \mathsf{token}$ then
    if $\hat{\mathsf{e}} \in \{\mathsf{fwl}, \mathsf{fwr}+1\}$ then
      ABORT
    else
      $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
      return $\Delta_{\hat{\mathsf{e}}}$

$\mathcal{O}$.Upd$\tilde{\mathsf{C}}$ :
  if $\mathsf{phase} \neq 1$ then
    return $\perp$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
  $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{\mathbf{c}}_{\mathsf{e}}, \mathsf{e})\}$
  return $\tilde{\mathbf{c}}_{\mathsf{e}}$
```

Figure 24: Part 1. Reduction $\tilde{\mathcal{B}}_{5.1}$ for proof of Claim 5.1.2. † indicates Δ_0 and $\Delta_{\mathsf{fwr}+1}$ are skipped in the computation. †† indicates $\mathbf{e}_{\mathsf{fwr}+1}$ is skipped in the computation.

```
$\underline{\mathcal{O}.\mathsf{Upd}(\mathbf{c}_{\mathsf{e}\text{-}1}):}$
  if $(\cdot, \mathbf{c}_{\mathsf{e}\text{-}1}, \mathsf{e}\text{-}1; \mathbf{R}_{\mathsf{e}\text{-}1}, \mathbf{e}'_{\mathsf{e}\text{-}1}, \mathbf{m}) \notin \mathcal{L}$ then
    return $\perp$
  $(\mathbf{c}_{\mathsf{e}}; \mathbf{R}_{\mathsf{e}}, \mathbf{e}'_{\mathsf{e}}) \leftarrow \mathsf{reR}(\mathbf{p}_{\mathsf{e}}, \mathbf{c}_{\mathsf{e}\text{-}1}; \mathbf{R}_{\mathsf{e}\text{-}1}, \mathbf{e}'_{\mathsf{e}\text{-}1}, \mathbf{m})$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathbf{c}_{\mathsf{e}}, \mathsf{e}; \mathbf{R}_{\mathsf{e}}, \mathbf{e}'_{\mathsf{e}}, \mathbf{m})\}$
  return $\mathbf{c}_{\mathsf{e}}$

$\underline{\mathcal{O}.\mathsf{Chall}(\bar{\mathbf{m}}_0, \bar{\mathbf{c}}):}$
  if $\mathsf{phase} = 1$ then
    return $\perp$
  $\mathsf{phase} \leftarrow 1; \tilde{\mathsf{e}} \leftarrow \mathsf{e}$
  if $(\cdot, \bar{\mathbf{c}} = (\bar{\mathbf{C}}^1, \bar{\mathbf{c}}^2), \tilde{\mathsf{e}}\text{-}1; \mathbf{R}_{\tilde{\mathsf{e}}\text{-}1}, \mathbf{e}'_{\tilde{\mathsf{e}}\text{-}1}, \bar{\mathbf{m}}_1) \notin \mathcal{L}$ then
    return $\perp$
  Send $\bar{\mathbf{m}}_{\mathrm{b}}$ to the $\mathsf{IND\$\text{-}CPA}$ challenger, get $(\mathbf{X}, \mathbf{y})$
  if $\tilde{\mathsf{e}} = i$ then
    if $\mathrm{b} = 0$ then
      $\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}} \leftarrow (\mathbf{X}, \mathbf{y})$
    else
      $\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}} \leftarrow (\mathbf{X} + \bar{\mathbf{C}}^1, \mathbf{y} + \mathbf{p}_{\tilde{\mathsf{e}}}^{\intercal} \cdot \mathbf{R}_{\tilde{\mathsf{e}}\text{-}1} + \mathbf{e}'_{\tilde{\mathsf{e}}\text{-}1})$
  else
    if $\mathrm{b} = 0$ then
      $\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}} \leftarrow \mathsf{LWEUE.Enc}(\mathbf{p}_{\tilde{\mathsf{e}}}, \bar{\mathbf{m}}_0)$
    else
      $(\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}}; \mathbf{R}_{\tilde{\mathsf{e}}}, \mathbf{e}'_{\tilde{\mathsf{e}}}) \leftarrow \mathsf{reR}(\mathbf{p}_{\tilde{\mathsf{e}}}, \bar{\mathbf{c}}; \mathbf{R}_{\tilde{\mathsf{e}}\text{-}1}, \mathbf{e}'_{\tilde{\mathsf{e}}\text{-}1}, \bar{\mathbf{m}}_1)$
  for $j \in \{\tilde{\mathsf{e}}+1, \ldots, i\text{-}1\}$ do
    $(\tilde{\mathbf{c}}_j; \mathbf{R}_j, \mathbf{e}'_j) \leftarrow \mathsf{reR}(\mathbf{p}_j, \tilde{\mathbf{c}}_{j\text{-}1}; \mathbf{R}_{j\text{-}1}, \mathbf{e}'_{j\text{-}1}, \bar{\mathbf{m}}_{\mathrm{b}})$
  $\tilde{\mathbf{c}}_i \leftarrow (\mathbf{X} + \tilde{\mathbf{C}}^1_{i\text{-}1}, \mathbf{y} + \mathbf{p}_i^{\intercal} \cdot \mathbf{R}_{i\text{-}1} + \mathbf{e}'_{i\text{-}1})$
  for $j \in \{i+1, \ldots, n\}$ do
    $\tilde{\mathbf{c}}_j \xleftarrow{\$} \mathcal{CS}$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\tilde{\mathsf{e}}\}$
  $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}}, \tilde{\mathsf{e}})\}$
  return $\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}}$
```

Figure 24: Part 2. Reduction $\tilde{\mathcal{B}}_{5.1}$ for proof of Claim 5.1.2.

  - $*$ $\tilde{\mathcal{B}}_{5.1}$ generates the initial key by running the key generation algorithm. In particular, $\mathbf{k}_0 \leftarrow \mathsf{LWEUE.KG}(1^\lambda)$ is computed as: randomly sample a secret key $\mathbf{s}_0 \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$, set $\mathbf{k}_0 \leftarrow (\mathbf{s}_0, \mathbf{A} \cdot \mathbf{s}_0 + \mathbf{e}_0)$.
  - $*$ For $j \in \{1, \ldots, \mathsf{fwl} - 1\} \cup \{\mathsf{fwr} + 2, \ldots, n\}$, $\tilde{\mathcal{B}}_{5.1}$ generates the epoch key and update token by running the key generation algorithm and the token generation algorithm. In particular, $\mathbf{k}_j \leftarrow \mathsf{LWEUE.KG}(1^\lambda)$ and $\Delta_j \leftarrow \mathsf{LWEUE.TG}(\mathbf{k}_{j-1}, \mathbf{k}_j)$ are computed as: randomly sample a secret key $\mathbf{s}_j \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$, set token $\Delta_j^{\mathbf{s}} \leftarrow \mathbf{s}_j - \mathbf{s}_{j-1}$ and public key $\mathbf{p}_j \leftarrow \mathbf{A} \cdot \mathbf{s}_j + \sum_{k=0}^{j} \mathbf{e}_k$[5].
  - $*$ $\tilde{\mathcal{B}}_{5.1}$ generates $\mathbf{k}_{\mathsf{fwr}+1}$ by using Equation (19): randomly sample a secret key $\mathbf{s}_{\mathsf{fwr}+1} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$, set public key $\mathbf{p}_{\mathsf{fwr}+1} \leftarrow \mathbf{A} \cdot \mathbf{s}_{\mathsf{fwr}+1} + \sum_{k=0}^{\mathsf{fwr}} \mathbf{e}_k$.

- $\tilde{\mathcal{B}}_{5.1}$ generates public keys $\mathbf{p}_{\mathsf{fwl}}, \ldots, \mathbf{p}_{\mathsf{fwr}}$ as follows.
  - $*$ $\tilde{\mathcal{B}}_{5.1}$ generates $\mathbf{p}_i$ by using Equation (16): embed the public key $\mathbf{p}$ received from its IND\$-CPA challenger to $\mathbf{p}_i$, set $\mathbf{p}_i \leftarrow \mathbf{p} + \sum_{k=0}^{i} \mathbf{e}_k$[6].
  - $*$ For $\mathsf{fwl} \leq j < i$, $\mathbf{p}_j$ is generated by using Equations (16,18): set public key $\mathbf{p}_j \leftarrow \mathbf{p} - \mathbf{A} \cdot \sum_{k=j+1}^{i} \Delta_j^{\mathbf{s}} + \sum_{k=0}^{j} \mathbf{e}_k$.
  - $*$ For $i < j \leq \mathsf{fwr}$, $\mathbf{p}_j$ is generated by using Equations (16,17): set public key $\mathbf{p}_j \leftarrow \mathbf{p} + \mathbf{A} \cdot \sum_{k=i+1}^{j} \Delta_j^{\mathbf{s}} + \sum_{k=0}^{j} \mathbf{e}_k$.

- To simulate non-challenge ciphertexts: $\tilde{\mathcal{B}}_{5.1}$ uses public keys to simulate encrypted ciphertexts and updated ciphertexts. In particular, $\tilde{\mathcal{B}}_{5.1}$ runs the re-randomization algorithm reR to simulate updated ciphertexts.

- To simulate challenge-equal ciphertexts in an epoch that is:

[5] If $j > \mathsf{fwr} + 1$, skip $\mathbf{e}_{\mathsf{fwr}+1}$ in the computation. This computation is based on Equations (11,19)

[6] set $\mathbf{A} \cdot \mathbf{s}_i + \mathbf{e} \leftarrow \mathbf{p}$

- to the left of epoch i: $\tilde{\mathcal{B}}_{5.1}$ uses public keys to simulate encryption and updating. When $\tilde{\mathcal{B}}_{5.1}$ runs update algorithm, it performs re-randomization using algorithm reR, which is described in Fig. 23.
- epoch i: $\tilde{\mathcal{B}}_{5.1}$ embeds the challenge ciphertext $(\mathbf{X}, \mathbf{y})$ received from its $\mathsf{IND\$\text{-}CPA}$ challenger to the challenge-equal ciphertext in epoch i. Using $(\mathbf{X}, \mathbf{y})$ as the re-randomization component or the random encryption component. More precisely, suppose $\tilde{\mathcal{B}}_{5.1}$ receives a challenge query $\mathcal{O}.\mathsf{Chall}$ with input $(\bar{\mathbf{m}}_0, \bar{\mathbf{c}})$ in challenge epoch $\tilde{\mathsf{e}}$, where the underlying message of $\bar{\mathbf{c}}$ is $\bar{\mathbf{m}}_1$. $\tilde{\mathcal{B}}_{5.1}$ sends $\bar{\mathbf{m}}_{\mathrm{b}}$ to its $\mathsf{IND\$\text{-}CPA}$ challenger and obtains $(\mathbf{X}, \mathbf{y})$. Hence, $(\mathbf{X}, \mathbf{y})$ is either an output of $\mathsf{LWEPKE.Enc}(\mathbf{p}, \bar{\mathbf{m}}_{\mathrm{b}})$[7] or a random ciphertext.

  For $\mathrm{b} = 1$. In the former case, based on Equations (6, 8), we know that $(\tilde{\mathbf{C}}_i^1, \tilde{\mathbf{c}}_i^2) = (\tilde{\mathbf{C}}_{i-1}^1 + \mathbf{X}, (\mathbf{p}_i^\mathsf{T} \cdot \mathbf{R}_{i-1} + \mathbf{e}'_{i-1}) + \mathbf{y})$ is a valid simulation of the challenge-equal ciphertext in epoch i. In the latter case, $(\tilde{\mathbf{C}}_i^1, \tilde{\mathbf{c}}_i^2) = (\tilde{\mathbf{C}}_{i-1}^1 + \mathbf{X}, (\mathbf{p}_i^\mathsf{T} \cdot \mathbf{R}_{i-1} + \mathbf{e}'_{i-1}) + \mathbf{y})$ is a random ciphertext.

  For $\mathrm{b} = 0$. The analysis is similar to the discussion when $\mathrm{b} = 1$.
- to the right of epoch i: $\tilde{\mathcal{B}}_{5.1}$ samples a random ciphertext.

Eventually, $\tilde{\mathcal{B}}_{5.1}$ receives the output bit from $\mathcal{A}_i$. If $\mathcal{A}_i$ guesses that it receives a real challenge-equal ciphertext in epoch i, then $\tilde{\mathcal{B}}_{5.1}$ guesses that it has seen the real encryption (returns 0 to its $\mathsf{IND\$\text{-}CPA}$ challenger). Otherwise, $\tilde{\mathcal{B}}_{5.1}$ guess it has seen a random ciphertext (returns 1 to its $\mathsf{IND\$\text{-}CPA}$ challenger).

Note that $\tilde{\mathcal{B}}_{5.1}$ perfectly simulates the responses of queries made by adversary $\mathcal{A}_i$ in game 2 except for a negligible probability. Then we have that $\mathbf{Pr}[E_2] \leq \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{LWEPKE},\, \tilde{\mathcal{B}}_{5.1}}(1^\lambda) + \mathsf{negl}(n)$. □

Remark 5.1. Klooß et al. [KLR19] introduced a generic construction of transforming CPA-secure UE schemes to UE schemes with PTXT and RCCA security. The main idea is to use the extended Naor-Yung (NY) CCA-transform [NY90] (for public-key schemes). The NY approach is to encrypt a message under two (public) keys of a CPA-secure encryption scheme. The

[7]$\mathbf{y}$ is statistically close to the second component of $\mathsf{LWEPKE.Enc}(\mathbf{p}_i, \bar{\mathbf{m}}_{\mathrm{b}})$ in this case.

extended NY approach additionally includes a proof that shows the owner knows a valid signature that contains the NY ciphertext pair and the underlying message. A potential future work would be to incorporate LWEUE to their construction to create a UE scheme that achieves PTXT and RCCA security.

Acknowledgements. We would like to thank Gareth T. Davies, Herman Galteland and Kristian Gjøsteen for fruitful discussions, and the anonymous reviewers for a number of valuable suggestions.

References

[ABD$^+$a] Erdem Alkim, Joppe W. Bos, Léo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Chris Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM: Learning With Errors Key Encapsulation. https://frodokem.org/files/FrodoKEM-specification-20190330.pdf. Submission to the NIST Post-Quantum Standardization project, round 2.

[ABD$^+$b] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Kyber (version 2.0). https://pq-crystals.org/kyber/data/kyber-specification-round2.pdf. Submission to the NIST Post-Quantum Standardization project, round 2.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum Key Exchange - A New Hope. In *USENIX Security Symposium*, pages 327–343. USENIX Association, 2016.

[BCLvV17] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU Prime: Reducing attack surface at low cost. In *SAC*, volume 10719 of *Lecture Notes in Computer Science*, pages 235–260. Springer, 2017.

[BDGJ20] Colin Boyd, Gareth T. Davies, Kristian Gjøsteen, and Yao Jiang. Fast and secure updatable encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 464–493. Springer, 2020.

[BEKS20] Dan Boneh, Saba Eskandarian, Sam Kim, and Maurice Shih. Improving Speed and Security in Updatable Encryption Schemes. *IACR Cryptology ePrint Archive*, 2020:222, 2020.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 410–428. Springer, 2013.

[BLMR15] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. *IACR Cryptology ePrint Archive*, 2015:220, 2015.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011.

[CDH$^+$] Cong Chen, Oussama Danba, Jeffrey Hoffstein, Andreas Hülsing, Joost Rijneveld, John M. Schanck, Peter Schwabe, William Whyte, and Zhenfei Zhang. NTRU. https://

ntru.org/f/ntru-20190330.pdf. Submission to the NIST Post-Quantum Standardization project, round 2.

[DKRV18] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure KEM. In Antoine Joux, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *Progress in Cryptology - AFRICACRYPT 2018 - 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7-9, 2018, Proceedings*, volume 10831 of *Lecture Notes in Computer Science*, pages 282–305. Springer, 2018.

[EPRS17] Adam Everspaugh, Kenneth G. Paterson, Thomas Ristenpart, and Samuel Scott. Key rotation for authenticated encryption. In *Proceedings of CRYPTO 2017 III*, volume 10403 of *LNCS*, pages 98–129. Springer, 2017.

[Gen09] Craig Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford, CA, USA, 2009. AAI3382729.

[GKPV10] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In Andrew Chi-Chih Yao, editor, *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 230–240. Tsinghua University Press, 2010.

[Ham] Mike Hamburg. Three Bears. https://sourceforge.net/projects/threebears/. Submission to the NIST Post-Quantum Standardization project, round 2.

[KLR19] Michael Klooß, Anja Lehmann, and Andy Rupp. (R)CCA secure updatable encryption with integrity protection. In *Proceedings of EUROCRYPT 2019 I*, volume 11476 of *LNCS*, pages 68–99. Springer, 2019.

[LLJ$^+$] Xianhui Lu, Yamin Liu, Dingding Jia, Haiyang Xue, Jingnan He, Zhenfei Zhang, Zhe Liu, Hao Yang, Bao Li, and Kunpeng

Wang. LAC Lattice-based Cryptosystems. Submission to the NIST Post-Quantum Standardization project, round 2.

[LT18] Anja Lehmann and Björn Tackmann. Updatable encryption with post-compromise security. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 685–716. Springer, 2018.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 427–437. ACM, 1990.

[OZS$^+$] Garcia-Morchon Oscar, Zhang Zhenfei, Bhattacharya Sauvik, Rietman Ronald, Tolhuizen Ludo, Torre-Arce Jose-Luis, Baan Hayo, Saarinen Markku-Juhani O., Fluhrer Scott, Laarhoven Thijs, Player Rachel, Cheon Jung, Hee, and Son Yongha. Round5. https://round5.org. Submission to the NIST Post-Quantum Standardization project, round 2.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David A. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2008.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005.

A Lattice Background

A.1 Hardness Assumptions – Learning With Errors

Regev [Reg05] introduced the learning with error (LWE) problem, where is as follows. For a secret element s and an error distribution $\mathcal{X}$, the problem is to distinguish m pairs $(\mathbf{a}_i, \mathbf{a}_i \cdot \mathbf{s} + e_i)$ from uniformly random pairs, where $\mathbf{a}_i$ are uniformly random elements and the e_is are sampled from the error distribution $\mathcal{X}$.

Definition 12. [Learning With Errors] Let n, $q = q(n) \geq 2$, $m = \text{poly}(n)$ be positive integers, $\mathcal{X} = \mathcal{X}(n)$ be an error distribution over $\mathbb{Z}$. The *decision learning with errors problem* $\mathsf{DLWE}_{n,q,\mathcal{X}}$ is to distinguish between the following pairs of distributions: $\{\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \mod q\}$ and $\{\mathbf{A}, \mathbf{u}\}$, where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$, $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \mathcal{X}^m$ and $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^m$. The advantage of an adversary $\mathcal{A}$ against $\mathsf{DLWE}_{n,q,\mathcal{X}}$ is defined as

$$\begin{aligned}
&\mathbf{Adv}_{n,q,\mathcal{X}}^{\mathsf{DLWE}}(\mathcal{A}) = \\
&\quad \big|\mathbf{Pr}[\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n, \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m\times n}, \mathbf{e} \leftarrow \mathcal{X}^m : \mathcal{A}(\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \mod q) = 1] \\
&\quad - \mathbf{Pr}[\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m\times n}, \mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^m : \mathcal{A}(\mathbf{A}, \mathbf{u}) = 1]\big|.
\end{aligned}$$

We say that the $\mathsf{DLWE}_{n,q,\mathcal{X}}$ problem is hard if $\mathbf{Adv}_{n,q,\mathcal{X}}^{\mathsf{DLWE}}(\mathcal{A})$ is negligible for any PPT adversary $\mathcal{A}$.

Regev [Reg05] showed that, for some parameter setting, LWE problem is as hard as some hard lattice problem, namely the worst-case SIVP and GapSVP, under a quantum reduction.

Another variant of the LWE problem is described in the work of Peikert et al. [PVW08], where secret key $\mathbf{s}$ is not a vector but a randomly chosen matrix. Specifically, $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m\times n}$, $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_q^{n\times l}$ and $\mathbf{E} \leftarrow \mathcal{X}^{m\times l}$, the problem is to distinguish $\{\mathbf{A}, \mathbf{A} \cdot \mathbf{S} + \mathbf{E}\}$ from $\{\mathbf{A}, \mathbf{U}\}$, where $\mathbf{U} \xleftarrow{\$} \mathbb{Z}_q^{m\times l}$. We denote this problem as $\mathsf{DLWE}_{n,l,q,\mathcal{X}}$. Peikert et al. [PVW08, Lemma 7.3] proved that $\mathsf{DLWE}_{n,l,q,\mathcal{X}}$ is hard if $\mathsf{DLWE}_{n,q,\mathcal{X}}$ is hard.

Gaussian Distribution. Let $D_{\mathbb{Z}^n,\alpha}$ denote the n-dimensional discrete Gaussian distribution with standard deviation α. We let $\mathsf{DLWE}_{n,q,\alpha}$ denote the

LWE problem $\mathsf{DLWE}_{n,q,D_{\mathbb{Z}^n,\alpha}}$ using a (discrete) Gaussian distribution as the error distribution.

The following three lemmas regarding discrete Gaussian distribution we will frequently use in this paper. The first lemma states that the discrete Gaussian distribution is spherical. The second lemma shows the discrete Gaussian distribution is statistical close to its (short distance) translated discrete Gaussian distribution. The third lemma describes the sum of two discrete Gaussian distributions is statistical close to another discrete Gaussian distribution.

Lemma A.1. [Brakerski and Vaikuntanathan [BV11]] Let $n \in \mathbb{N}$. For any real number $\alpha = \omega(\sqrt{\log n})$, we have $\mathbf{Pr}[\|\mathbf{x}\| > \alpha \cdot \sqrt{n} \mid \mathbf{x} \leftarrow D_{\mathbb{Z}^n,\alpha}] \leq 2^{-n+1}$.

Lemma A.2. [Brakerski and Vaikuntanathan [BV11]] Let $n \in \mathbb{N}$. For any real number $\alpha = \omega(\sqrt{\log n})$, any $\mathbf{c} \in \mathbb{Z}^n$, then the statistical distance between $D_{\mathbb{Z}^n,\alpha}$ and $D_{\mathbb{Z}^n,\alpha,\mathbf{c}}$ is at most $\|\mathbf{c}\|/\alpha$.

Lemma A.3. [Gentry [Gen09]] Let $n \in \mathbb{N}$. For any real number $\alpha, \beta > 0$ satisfies $(\alpha\beta)/\sqrt{\alpha^2+\beta^2} = \omega(\sqrt{\log n})$, then $D_{\mathbb{Z}^n,\alpha} + D_{\mathbb{Z}^n,\beta}$ (and $D_{\mathbb{Z}^n,\alpha} - D_{\mathbb{Z}^n,\beta}$) is statistical to $D_{\mathbb{Z}^n,\sqrt{\alpha^2+\beta^2}}$.

Next we show a lemma that states that if an LWE problem with small standard deviation is hard, then the corresponding LWE with a big standard deviation is hard.

Lemma A.4. Let $n \in \mathbb{N}$. For any real number $\alpha, \beta > 0$ satisfies $\alpha/\beta = \mathsf{negl}(n)$ then for any adversary $\mathcal{A}$ against $\mathsf{DLWE}_{n,q,\beta}$, there exists an adversary $\mathcal{B}$ against $\mathsf{DLWE}_{n,q,\alpha}$ such that

$$\mathbf{Adv}^{\mathsf{DLWE}}_{n,q,\beta}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{DLWE}}_{n,q,\alpha}(\mathcal{B}) + \mathsf{negl}(n).$$

Proof. We construct a reduction $\mathcal{B}$ plays $\mathsf{DLWE}_{n,q,\alpha}$ game by running adversary $\mathcal{A}$. When the reduction receives a $\mathsf{DLWE}_{n,q,\alpha}$ pair $(\mathbf{A}, \mathbf{p})$, it samples a big error $\mathbf{e}' \leftarrow D^m_{\mathbb{Z},\beta}$ and sends $(\mathbf{A}, \mathbf{p}+\mathbf{e}' \mod q)$ to $\mathcal{A}$. If $(\mathbf{A}, \mathbf{p})$ is a real $\mathsf{DLWE}_{n,q,\alpha}$ sample, by Lemma A.2, we have $(\mathbf{A}, \mathbf{p}+\mathbf{e}' \mod q)$ is statistical close to a real $\mathsf{DLWE}_{n,q,\beta}$ sample. If $(\mathbf{A}, \mathbf{p})$ is a random $\mathsf{DLWE}_{n,q,\alpha}$ sample, then $(\mathbf{A}, \mathbf{p}+\mathbf{e}' \mod q)$ is a random $\mathsf{DLWE}_{n,q,\beta}$ sample as well. So the reduction $\mathcal{B}$ perfectly simulate $\mathsf{DLWE}_{n,q,\beta}$ game to $\mathcal{A}$ except for a negligible probability. □

A.2 Leftover Hash Lemma

We use a variant of Leftover Hash Lemma [GKPV10]. We use this lemma to prove our LWE-based PKE scheme is IND\$-CPA-secure in Section 4.

Lemma A.5. [Leftover Hash Lemma] Let $\mathcal{D}$ be a distribution over $\mathbb{Z}_q^n$ with min-entropy k. For any $\epsilon > 0$ and $l \le (k - 2\log(1/\epsilon) - O(1))/\log(q)$, the joint distribution of $(\mathbf{C}, \mathbf{Cs})$ is ϵ-close to the uniform distribution over $\mathbb{Z}_q^{l\times n} \times \mathbb{Z}_q^l$, where $\mathbf{C} \xleftarrow{\$} \mathbb{Z}_q^{l\times n}$ and $\mathbf{s} \leftarrow \mathcal{D}$.

We prove that when we reuse $\mathbf{C}$ to generate t samples, the above result is still hold.

Lemma A.6. [Matrix Variant of Leftover Hash Lemma] Let $\mathcal{D}$ be a distribution over $\mathbb{Z}_q^n$ with min-entropy k. For any $\epsilon > 0$ and $l \le (k-2\log(1/\epsilon)-O(1))/\log(q)$, the joint distribution of $(\mathbf{C}, \mathbf{C}\cdot\mathbf{S})$ is $t\epsilon$-close to the uniform distribution over $\mathbb{Z}_q^{l\times n} \times \mathbb{Z}_q^{l\times t}$, where $\mathbf{C} \xleftarrow{\$} \mathbb{Z}_q^{l\times n}$ and $\mathbf{S} \leftarrow \mathcal{D}^t$.

Proof. The proof method is the same as the proof of Lemma 7.3 stated by Peikert et al. [PVW08]. Consider hybrid distributions $H_0, H_1, \ldots, H_t$ for the pair $(\mathbf{C}, \mathbf{B})$: in H_i the entire matrix $\mathbf{C}$ and the first i columns of $\mathbf{B}$ are all uniformly random. Then H_0 is the joint distribution of $(\mathbf{C}, \mathbf{C}\cdot\mathbf{S})$, H_t is the uniform distribution over $\mathbb{Z}_q^{l\times n} \times \mathbb{Z}_q^{l\times t}$.

Hence,

$$\left|\mathbf{Pr}[(\mathbf{C},\mathbf{B}) \leftarrow H_0 : \mathcal{A}(\mathbf{C},\mathbf{B}) = 1] - \mathbf{Pr}[(\mathbf{C},\mathbf{B}) \leftarrow H_t : \mathcal{A}(\mathbf{C},\mathbf{B}) = 1]\right|$$

$$\le \sum_{1\le i\le t} \left|\mathbf{Pr}[(\mathbf{C},\mathbf{B}) \leftarrow H_{i-1} : \mathcal{A}(\mathbf{C},\mathbf{B}) = 1] - \mathbf{Pr}[(\mathbf{C},\mathbf{B}) \leftarrow H_i : \mathcal{A}(\mathbf{C},\mathbf{B}) = 1]\right|.$$

We construct a reduction $\mathcal{B}$ trying to simulate distribution H_{i-1} or H_i by contacting with an oracle $\mathcal{O}$. The oracle $\mathcal{O}$ returns a sample either from the distribution of $(\mathbf{C}, \mathbf{C}\cdot\mathbf{s})$ or the uniform distribution over $\mathbb{Z}_q^{l\times n} \times \mathbb{Z}_q^l$, where $\mathbf{s}$ is chosen from $\mathcal{D}$. $\mathcal{B}$ simulates a distribution in the following way:

1. $\mathcal{B}$ queries oracle $\mathcal{O}$ to obtain a sample $(\mathbf{C}, \mathbf{p})$, sets $\mathbf{p}_i \leftarrow \mathbf{p}$.
2. For $j > i$, $\mathcal{B}$ samples $\mathbf{s}_j \leftarrow \mathcal{D}$ and computes $\mathbf{p}_j \leftarrow \mathbf{C}\cdot\mathbf{s}_j$.
3. For $j < i$, $\mathcal{B}$ samples $\mathbf{p}_j \xleftarrow{\$} \mathbb{Z}_q^l$.

4. $\mathcal{B}$ sets $\mathbf{p}_j$ as the j-th column of $\mathbf{B}$.

5. $\mathcal{B}$ outputs $(\mathbf{C}, \mathbf{B})$.

Note that $\mathcal{B}$ can perfectly simulate the distribution H_{i-1} if it receives a sample from the distribution of $(\mathbf{C}, \mathbf{C} \cdot \mathbf{s})$, that is $\mathbf{Pr}[(\mathbf{C}, \mathbf{B}) \leftarrow H_{i-1} : \mathcal{A}(\mathbf{C}, \mathbf{B}) = 1] = \mathbf{Pr}[(\mathbf{C}, \mathbf{C} \cdot \mathbf{s}) \leftarrow \mathcal{O} : \mathcal{B}(\mathbf{C}, \mathbf{C} \cdot \mathbf{s}) = 1]$. Otherwise, $\mathcal{B}$ can perfectly simulate the distribution H_i, that is $\mathbf{Pr}[(\mathbf{C}, \mathbf{B}) \leftarrow H_i : \mathcal{A}(\mathbf{C}, \mathbf{B}) = 1] = \mathbf{Pr}[(\mathbf{C}, \mathbf{u}) \leftarrow \mathcal{O} : \mathcal{B}(\mathbf{C}, \mathbf{u}) = 1]$. $\square$