ISBN 978-82-326-4580-0 (printed version)
ISBN 978-82-326-4581-7 (electronic version)
ISSN 1503-8181

Herman Galteland

Doctoral Thesis

Doctoral theses at NTNU, 2020:116

Herman Galteland

Malicious cryptography

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

NTNU
Norwegian University of
Science and Technology

Doctoral theses at NTNU, 2020:116

NTNU

NTNU
Norwegian University of
Science and Technology

Herman Galteland

Malicious cryptography

Thesis for the degree of Philosophiae Doctor

Trondheim, April 2020

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 978-82-326-4580-0 (printed version)
ISBN 978-82-326-4581-7 (electronic version)
ISSN 1503-8181

Doctoral theses at NTNU, 2020:116

Printed by Skipnes Kommunikasjon as

Acknowledgments

This thesis marks the end of my PhD studies and I would like to express my gratitude to everyone that has helped me along the way.

First and foremost I would like to thank my advisor Kristian Gjøsteen for giving me the support I needed, answering all of my questions, guiding me, and always being calm and reassuring.

I would like to thank my co-authors: Gareth, Kristian, Stig Frode, Tjerand, Ruxandra, and Yao for writing papers and working with me. I would get stuck working all by myself and I would overlook many details. The discussions we had made the papers and the process better. I would also like to give an extra thanks to the post-doctors, Gareth Davies and Ruxandra Olimid, for showing me good research habits.

There are ups and downs in a PhD's life, just like anywhere else, and I would like to give my gratitude to my family for being encouraging when I shared my successes and for being supportive when I encountered difficulties. Last but not least, I would like to express my greatest appreciation to Yao for brightening my days, answering my silly questions, supporting me in my endeavors, and working together with me.

Herman Galteland
Trondheim, January 2020

Introduction

Historically, cryptography has been a tool for the defender. In this thesis we discuss malicious uses of cryptography and countermeasures to such use. That is, we study how attackers could use cryptography to make the defender's work harder, and how such techniques can be countered by the defender. The goal of this thesis is not to develop new attacks, but to understand possible future threats. This is important, because to prepare for future attacks we must know what to defend against. Obviously there is an ethical dilemma here, we have tried to balance this by only considering theoretical studies. We have not developed attack code. We also note that some tools and techniques studied in this thesis are dual-use, technologies that can be used by both a defender and an attacker. The Tor network, and other anonymity networks, is used to avoid surveillance and censorship, but can also be used to hide attackers. Similar for subliminal channels.

Malicious cryptography started with Young and Yung and their submission to the 1996 IEEE Security & Privacy conference, where they proposed malware that encrypts the local files on the infected computer and holds them for ransom [13]. The malicious code generates a symmetric encryption key on the infected computer and uses it to encrypt files. The symmetric key is then encrypted using a public encryption key stored in the code, where the malware author has the secret decryption key. The owner of the infected computer is notified and to recover the symmetric key, and the encrypted files, the owner has to pay the malware author and send the encrypted symmetric key. Using the secret decryption key the malware author decrypts the symmetric key and sends it back, so that the owner can recover their

files. This attack was called *cryptoviral extortion* by Young and Yung. Today such malware is commonly known as ransomware. With the paper Young and Yung showed how malware can use cryptography and by publishing the paper they gave the security community time to prepare. Young and Yung continued this line of work and called it *cryptovirology* [15].

This thesis consists of six papers and they appear in logical order, not chronological order. In Section 1 we introduce the two principal actors of this thesis, and discuss the techniques we have studied that the attacker use. In Section 2 we discuss the countermeasures we have studied that the defender use.

1 Malicious cryptography

The two principal actors of this thesis is an attacker, the *Malware Author*, and an defender, the *Opponent*. The Malware Author aims to attack the Opponent's computers for personal gain, and the Opponent aims to defend their network and expose the Malware Author.

The Malware Author writes and uses *malware* to attack computers. Malware is software that is maliciously installed on a computer and is designed to give functionality and behavior desired by the Malware Author, but not by the legitimate computer owner. The Malware Author's goals are to hide their identity and intentions. We discuss how the Malware Author can improve their malware using encryption, communicate with their malware anonymously or discretely, and send encrypted commands to their malware. We will not discuss the malicious code itself or what it does.

The Opponent protects a network of computers against malware attacks. The Opponent monitors their network and collects malware samples found on their computers, samples are analyzed and checked if they pose a threat to the Opponent's network. The Opponent wants to discover the intentions and identity of the Malware Author. We discuss possible countermeasures the Opponent can use when defending against attacks from the Malware Author. We show how the Opponent can break anonymity networks to discover the Malware Author's identity and how to prevent subliminal channels.

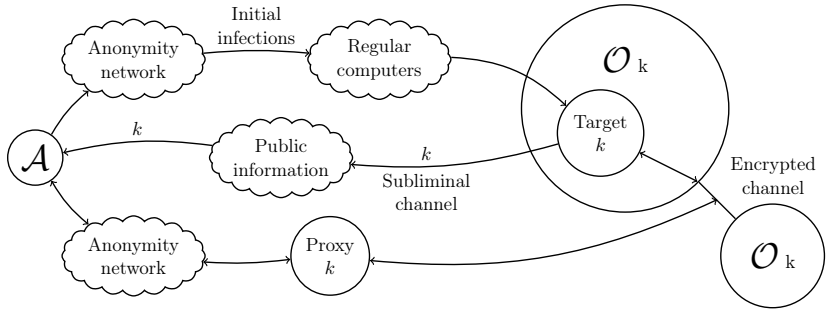


Figure 1: The Malware Author $\mathcal{A}$ wants access to the network of the Opponent $\mathcal{O}$. $\mathcal{A}$ infects regular computers with malware, which propagates to the target computer. Keying material k is stolen by malware and sent out of $\mathcal{O}$'s network via a subliminal channel, where it becomes public information and can be retrieved by $\mathcal{A}$. The key is uploaded to a proxy, where it is used by $\mathcal{A}$ to gain access $\mathcal{O}$'s encrypted channel and network.

1.1 Attack plan

Cryptography can be used by the Malware Author in different ways and below we describe how the techniques studied in this thesis can be used as part of an attack, see Figure 1.

The Malware Author has a piece of malicious software and wants to use it to gain access to the Opponent's network, where malware is used to steal keying material, for an encrypted channel, from a specific target located inside the Opponent's network. To protect the malicious code the Malware Author encrypts it before it is released. The encryption scheme is designed such that encrypted malware will only be decrypted if it infects the intended target computer, anywhere else the Opponent would find unintelligible malware samples that do not reveal their purpose. The Malware Author infects a few regular computers, that is not in the Opponent's network, and encrypted malware starts propagating. Let's assume malware successfully propagates into the Opponent's network and infects the target computer, where encrypted malware is decrypted and the malicious code is re-

leased. Its first task is to collect keying material for an encrypted channel used by the Opponent and send it through a subliminal channel. The keying material is encoded into a digital signature, produced by the infected host, and sent out of the Opponent's network, where it becomes public information that the Malware Author can retrieve. The signature is decoded and the Malware Author recovers the keying material. The key is stored on a file storage server, which is used as a proxy to prevent direct communication. The key gives the Malware Author access to the Opponent's encrypted channel and network.

1.2 Malware encryption

The Malware Author wants to hide their identity and their intentions, sending a readable, unprotected malware into the Opponent's network makes it vulnerable to analysis. By reading and testing the malicious code the Opponent can find the intentions of the Malware Author, which can also make it easier to expose their identity. We want the Opponent to do as much work as possible when analyzing malware, and encrypting the malicious code protects it from analysis [8, 9].

The Author encrypts malware using some encryption key, where the encrypted malware consists of an encrypted *payload*, the malicious code, and an *un*encrypted *loader* that manages the encrypted malware. The loader needs the correct decryption key before the payload can be run. It is not always possible for the Malware Author to send the key to the loader, for instance in an air gap network. If the key is hidden inside the loader the Opponent could discover it. We need a different method for generating the decryption key.

Environmental keys [11] are generated from locally available data on a computer, this information could be IP addresses, PATH variables, or any arbitrary time and date. With environmental keys the loader scans the infected host for environmental data, hashes it, and generates several potential decryption keys. Each generated key is checked if it decrypts the payload. The Malware Author gathers, or guesses, this information to generate the encryption key that is used to encrypt the payload. The environmental data specifies the target the Malware Author wants to attack, where the correct decryption key can only be generated on the target computer.

To ensure that no computer in the Opponent's network is the target their only option is to find or guess the decryption key. The loader tells what kinds of environmental data that might generate the correct decryption key, however, to find this data the Opponent needs to scan all of his computers. We assume the Opponent has access to all computers in his network. If no computer has the environmental data that generates the correct decryption key then the malware sample is most likely not targeting the Opponent's computers. If no decryption key can be found then the malicious code can never be analyzed and the intentions of the Malware Author stay protected.

Paper I *Malware encryption schemes – rerandomizable ciphertexts encrypted using environmental key.* We present malware encryption schemes that use environmental encryption keys, used to encrypt malicious software, and rerandomization techniques, to make different-looking copies of encrypted malware.

An earlier version of this paper was published at Mycrypt'16, the version in this thesis is an extended version and includes an additional method for encryption malware called the Path scheme. Instead of using one encryption key, specifying the target, we use several keys to specify a path of computers that needs to be infected before the encrypted payload can be decrypted at the target. By using more keys we reduce the Opponent's chances of successfully decrypting the payload.

1.3 Anonymity networks

Suppose the Malware Author wants to infect computers with their malware, or wants to retrieve stolen information stored on the proxy server. If the Malware Author communicates directly with the computers, or the proxy, their identity can be discovered by the Opponent and they need a method to communicate anonymously.

Anonymity networks are designed to hide the connection between the sender and receiver of a message. If the Malware Author uses an anonymity network when infecting computers, or when communicating with the proxy, their identity stays hidden even if the Opponent observes the messages sent to the infected computers, or the

proxy. Therefore, the Malware Author should use an anonymity network whenever they communicate with computers.

Anonymity networks are tools the Malware Author uses. We will talk more about mix networks, specifically the cMix protocol, and the Tor network later in this introduction.

1.4 Subliminal channels

A *subliminal channel* is a covert communication channel that is inserted into an overt communication channel, for example a digital signature scheme. The sender and receiver of the messages sent over a subliminal channel will typically share secret information before starting the communication such that they, and only they, can encode and decode subliminal messages in the overt communication.

The subliminal channel is a solution to Simmons' *Prisoners' Problem* [12]. Two criminals are arrested and sent to jail, imprisoned the two accomplices want to communicate and plan their escape. The warden of the prison allows them to communicate if, and only if, the prisoners send plaintext messages, such that the warden can learn their escape plan. The prisoners agree on sending plaintext message if they can sign them, such that they can verify that the received messages was sent from a prisoner and not from the warden. The problem of the prisoners is to find a method to communicate covertly over the overt communication channel controlled by the warden. If there is a subliminal channel in the digital signature scheme used to sign the messages then it might be possible for the prisoners to communicate covertly by encoding subliminal messages into the signatures.

Subliminal channels are designed to hide the communication channel, and for the Malware Author this is a tool similar to anonymity networks. Messages sent from malware can be disguised as the Opponent's computers' normal network communication. Information is sent over a subliminal channel by encoding it into the infected computer's digital signatures that are sent to any server or website located outside of the Opponent's network. Subliminal channels give malware additional protection, because no malware would be analyzed if the Opponent does not notice any suspicious activity.

The bandwidth of the subliminal channel depends on the signature scheme.

Paper II *Subliminal channels in post-quantum digital signature schemes.* We look for subliminal channels in the digital signatures schemes submitted to NIST's Post Quantum Cryptography Standardization Project.

All proposed digital signatures schemes accepted into NIST's second round have a subliminal channel. If the Opponent wants to use a (soon to be) standardized post-quantum secure digital signatures scheme for their network then it will contain a subliminal channel.

1.5 Command server

Suppose malware has reached its target and has performed some predetermined task, where the Malware Author becomes informed of its status and wishes to send a command. Alternatively, suppose malware is stealing information from a computer and wants to send it to the Malware Author. To send commands to or retrieve information the Malware Author can use a *command server* as a proxy to prevent direct communication with malware.

The Malware Author sends commands to the server and information is uploaded to the command server from malware. This makes it harder to connect the Malware Author with their malicious software, as there is no direct communication between them. However, the Opponent might find the command server, seize it, gather any information stored, and potentially reveal the Malware Author's intentions and identity from the information. In other words, the command server cannot be trusted and any information stored on it should be encrypted. To give malware access to the encrypted commands the Malware Author should use a group key exchange protocol to distribute keys, and they have two requirements of the protocol.

The first requirement is that the protocol should be noninteractive. When a file is to be encrypted and shared on the command server neither the Malware Author nor malicious software wish to wait for all participants to be online for the key exchange protocol. The malicious code mostly stays encrypted and while encrypted it should not

perform a key exchange. Nor should the unencrypted loader, since the Opponent could discover any keying material it has stored. Instead, the malicious code should do the key exchange once it has been decrypted. Similar for the Malware Author, information might be uploaded at any time by malware and while it waits for the Malware Author to respond the Opponent might discover the infection and prevent data being sent.

The second requirement is that the protocol should provide forward secrecy. If the Opponent manages to discover a decrypted malicious software then any file encryption- and decryption keys become compromised, this should not jeopardize all of the encrypted files on the server. By periodically updating the file encryption keys with respect to a master private key, i.e. session keys and a long-term key, the Malware Author can separate each uploaded file into sessions, where one compromised session key will not jeopardize files encrypted under a different session key. This is not enough to protect the files if the master private key is compromised, where a key exchange protocol with forward secrecy is needed.

To sum up, the Malware Author wants a noninteractive group key exchange protocol that provides forward secrecy, and the Offline Assisted Group Key Exchange (OAGKE) protocol of Boyd et al. [3] satisfies both of the Malware Author's requirements. Below we describe an adapted version of the user scenario of the OAGKE protocol. For simplicity we use the Malware Author as the initiator that shares a command, and malware as the responders that receives the command, the file sharing process is identical when malware initiates.

Suppose malware has reached its target, decrypted the payload, and possibly done some malicious act. The Malware Author has been notified of the status and wants to issue a new command. Before the command is uploaded to the proxy it is encrypted, where the OAGKE protocol utilizes the command server to assist in the group key exchange protocol. File encryption keys are shared using a modified key encapsulation mechanism (KEM) scheme denoted as *blinded* KEM (BKEM), a new primitive introduced by Boyd et al. [3]. The server generates keys for the BKEM scheme and sends the encapsulation key to the Malware Author, the server keeps the decapsulation key until all

recipients have responded and completed the protocol. The Malware Author encapsulates the file encryption key and the encapsulation is uploaded to the server together with the encrypted command. Note that in the original user scenario of the OAGKE protocol the encapsulation is directly sent to the recipient, however, we do not want the Malware Author to directly communicate with their malicious software and encapsulations are therefore sent to the server. This is not a problem as the encapsulation can be encrypted using a public encryption (PKE) scheme. The malicious software retrieves the encrypted command and the encapsulation from the server, the BKEM scheme has the added ability to blind encapsulations such that when malware asks the server to decapsulate the *blinded* encapsulation nothing can be learned about the file encryption key, as it is also blinded when the encapsulation is. The malicious code receives a blinded key from the server, which is unblinded using the BKEM scheme and the command can be decrypted. Once this process is finished the command server deletes its keys for the session, such that forward secrecy is achieved.

Paper III *Cloud-assisted Asynchronous Key Transport with Post-Quantum Security.* The paper presents a generic method of constructing post-quantum secure blinded KEMs, a primitive used to construct an OAGKE protocol.

If the Opponent gains access to the malicious code they get some session keys and a long-term key. The session key can be used to decrypt files on the command server for that session. The long-term key should not be able to decrypt any files from previous sessions, since the OAGKE protocol provides forward secrecy. However, the Opponent can use the long-term key to generate new session keys, upload some encrypted file, and trick the Malware Author into interacting with the command server. If the Opponent is monitoring the network traffic around the command server they might be able to discover and identify the Malware Author. Therefore an anonymity network must be used by the Malware Author to protect their identity.

2 Opponent's countermeasures

We have showed how the Malware Author can use cryptography to improve their attack. Below we describe how the remaining techniques studied in this thesis can be used by the Opponent to counter the problems caused by the Malware Author.

The Opponent has noticed encrypted malware samples on multiple computers in their network and is currently unable to analyze the malicious code. During this stage all the Opponent can do is to guess or find the decryption key, and prevent malware from propagating further. Let's assume the Opponent successfully discovers the decryption key and analyzes the malicious code in a safe environment. The Opponent notices malware is looking for keying material, it wants to gain access to the Opponent's network, and gives it some random number hoping that the malicious code sends it to the Malware Author. However, it attempts to encode the randomness into a digital signature and send it over a subliminal channel. The Opponent sets all of their computers to use a subliminal-free digital signature scheme to prevent any information leakage. All signatures generated by the Opponent's computers are checked by the Opponent and they can stop any stolen information before it is sent out of the network.

By further analyzing the malicious code the Opponent discovers a long-term key that is used to generate sessions keys for a file storage sever. The Opponent seizes the server and uncovers a few encrypted files, however, the long-term cannot be used to decrypt them. Instead, the Opponent generates a fresh session key, uploads some encrypted file, and waits for someone to communicate with the server. The file is retrieved by an unknown entity using an anonymity network and to reveal their identity the Opponent attacks the anonymity network.

2.1 Attacking anonymity networks

Anonymity networks are designed to hide the connection between the sender and receiver of a message. Suppose the Opponent attempts to discover the Malware Author's identity by monitoring a server the Malware Author is communicating with. However, the Opponent only notices messages being from someone using an anonymity network and

is unable to connect the messages back to the sender. To discover the identity of the sender the Opponent can attack the anonymity network and potentially discover the Malware Author.

2.2 Mix network

Mix networks [4] are anonymous communication networks that use a chain of servers, called *mix nodes*. A mix network collects a specified number of messages before they are sent though the network as a batch. Each mix node permutes and does some cryptographic computation on the messages such that it is hard to connect the input and the output messages.

A mix network usually has high latency because a batch of messages must be gathered before it can start mixing, and it performs expensive computation during mixing. Chaum et al. have proposed the mix network called *cMix* [6] that has an offline phase, that do most of the expensive computations, and an efficient online phase. The offline precomputation phase can be performed while a the servers collect messages. This reduces latency.

Paper IV *Attacks on the Basic cMix Design: On the Necessity of Commitments and Randomized Partial Checking.* We analyze the cMix protocol for weaknesses and present two attacks on the basic description of the protocol.

Note that we analyzed an earlier version of the protocol [5] and the attacks mentioned in the paper do no longer apply [6].

The attacks presented in the paper make it possible for the Opponent to connect the sender and receiver of a message, breaking the *relationship anonymity* [10] of the cMix protocol. Suppose the Opponent has corrupted the required mix nodes to perform these attacks, while the Malware Author communicates over the cMix network the Opponent can use these attacks to connect the messages back to the Malware Author and reveal their identity.

Protocol analysis is common practice to ensure secure implementations, any attacks discovered by the Opponent will eventually become common knowledge and fixed. To have a permanent attack on an anonymity network the Opponent can design and implement their

own protocol with a backdoor that only they know. A backdoor into an anonymity network could make it possible for the Opponent to connect the sender and receiver of any message. Implementing such systems is known as *kleptography* [14].

2.3 Onion routing

The onion routing network Tor [7] consists of servers, called *onion routers*, that relay messages sent to and from the users. When a user wants to communicate with a website it makes a chain of onion routers, three onion routers is the default choice, and establishes a symmetric key with each onion router. This chain is called a *Tor circuit*. The first node of a circuit is called a *guard node*, each user has their own small set of guard nodes that they use when creating circuits. The last node is an *exit node*, only trusted onion routers are marked as an exit. The middle node of a circuit is called a *relay*. Each node in the circuit knows each of its neighboring nodes and none of the nodes knows both the user and the sender, making a single onion router unable to link the sender and the receiver. The user encrypts its messages three times using its three symmetric keys before sending it to the website over the circuit, where each onion router removes one layer of encryption before passing it to the next node. Similarly, when the website sends a message back each onion router adds one layer of encryption, which the user can remove once the message has been received.

The Tor network is a tool the Malware Author uses to hide their connection with malware. Suppose the Opponent observes messages sent to the proxy from someone using the Tor network, to find the sender of these messages the Opponent needs to break the relationship anonymity of the Tor protocol. All onion routers are public and we assume the Opponent can recognize messages sent to the Tor network.

Paper V *Jurisdictional adversaries monitoring and reconstructing the Tor network.* We introduce an adversary against the Tor network that monitors Tor traffic crossing the borders of a jurisdiction the adversary controls, and show that a coalition of such adversaries is able to break the relationship anonymity of

the Tor network.

The Opponent protecting a network of computers would generally see any messages sent inside their jurisdiction, the jurisdictional adversaries do not. This does not mean that the results of the paper cannot be used by the Opponent, they would only have more information to aid them in breaking the relationship anonymity of the Tor network.

We achieve a 40%–60% reconstruction of the Tor network. There is a considerable chance the Opponent will know parts of the Malware Author's circuits, and whenever a new circuit is created the Opponent will gather more information that might help in discovering the Malware Author's identity.

The weakness of the jurisdictional adversaries is that they do not see any Tor traffic sent inside their jurisdiction. If the user chooses its Tor circuits such that the traffic that would cross a border is sent between onion routers then no jurisdiction would be able to see the identity of the website or the user. Note that this only avoids detection by the jurisdictional adversaries, an adversary corrupting onion routers would still have the capability to detect the user or the website. The Opponent could have their own corrupted onion routers inside their network, however, this would only allow them to see where malware is located inside their network and not the Malware Author. The Opponent needs to corrupt onion routers inside the Malware Author's jurisdiction to be able to see their identity.

2.4 Subliminal-free digital signature schemes

A subliminal channel in a digital signature scheme makes it possible for the Malware Author to communicate covertly with their malicious software and send information out of the Opponent's network unnoticed. To prevent subliminal channels the Opponent can choose a digital signature scheme with a small subliminal bandwidth. This would give them more time to detect malware and possibly prevent some information flow between the malicious code and the Malware Author, however, a small bandwidth will not prevent subliminal channels.

The Opponent needs a subliminal-free digital signature scheme to

prevent malware using subliminal channels. Subliminal messages typically replace any random values used to generate the signatures, and by controlling how the randomness is generated the Opponent can remove subliminal channels. The RSA-FDH signature scheme [1] is deterministic and therefore subliminal-free, but not always practical. Bohli et al. [2] proposed a digital signature scheme that is subliminal-free. They use the pseudorandom number generator of Naor to generate deterministic randomness and use this output to produce digital signatures. To prove a signature is subliminal-free Bohli et al. use non-interactive zero-knowledge proofs to show that the generated random number was deterministic and used to produce the signature. The proofs are sent to a warden who checks if the signature is generated honestly and is free of subliminal channels. Bohli et al. used these techniques to construct a subliminal-free variant of ECDSA. Hence, for classical signatures subliminal-free schemes exists.

Paper VI *Verifiable Random Secrets and Subliminal-Free Digital Signatures.* We introduce the notion of verifiable random secrets and use it with Schnorr-like digital signature schemes to make a post-quantum secure subliminal-free digital signature scheme.

We use a verifiable random secret scheme to produce random numbers and, similar to Bohli et al., we use proofs to show that the random number and the signature was honestly generated and, therefore, subliminal-free. Signature-proof pairs are sent to a warden, who verifies the proofs and forwards the signature to the recipient if they are valid. The Opponent plays the role of the warden and checks all signatures sent from computers in their network. If malware tries to encode a subliminal message in a signature the Opponent will notice and can stop the communication immediately, before any information has been lost.

dfgd

References

[1] Mihir Bellare and Phillip Rogaway. The Exact Security of Digital Signatures-how to Sign with RSA and Rabin. In *Proceedings*

of the 15th Annual International Conference on Theory and Application of Cryptographic Techniques, EUROCRYPT'96, pages 399–416, Berlin, Heidelberg, 1996. Springer-Verlag.

[2] Jens-Matthias Bohli, María Isabel González Vasco, and Rainer Steinwandt. A Subliminal-Free Variant of ECDSA. In Jan L. Camenisch, Christian S. Collberg, Neil F. Johnson, and Phil Sallee, editors, *Information Hiding*, pages 375–387, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[3] Colin Boyd, Gareth T. Davies, Kristian Gjøsteen, and Yao Jiang. Offline Assisted Group Key Exchange. In Liqun Chen, Mark Manulis, and Steve Schneider, editors, *Information Security*, pages 268–285, Cham, 2018. Springer International Publishing.

[4] David Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM*, 24(2):84–90, February 1981.

[5] David Chaum, Debajyoti Das, Farid Javani, Aniket Kate, Anna Krasnova, Joeri de Ruiter, and Alan T. Sherman. cMix: Mixing with Minimal Real-Time Asymmetric Cryptographic Operations. Cryptology ePrint Archive, Report 2016/008, 2016. `https://eprint.iacr.org/2016/008`.

[6] David Chaum, Debajyoti Das, Farid Javani, Aniket Kate, Anna Krasnova, Joeri De Ruiter, and Alan T. Sherman. cMix: Mixing with Minimal Real-Time Asymmetric Cryptographic Operations. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *Applied Cryptography and Network Security*, pages 557–578, Cham, 2017. Springer International Publishing.

[7] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-generation Onion Router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[8] Eric Filiol. Strong Cryptography Armoured Computer Viruses Forbidding Code Analysis: the bradley virus. Research Report RR-5250, INRIA, 2004.

[9] Eric Filiol. Malicious cryptography techniques for unreversable (malicious or not) binaries. *CoRR*, abs/1009.4000, 2010.

[10] Andreas Pfitzmann and Marit Hansen. A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management, 2010.

[11] James Riordan and Bruce Schneier. Environmental Key Generation Towards Clueless Agents. In Giovanni Vigna, editor, *Mobile Agents and Security*, volume 1419 of *Lecture Notes in Computer Science*, pages 15–24. Springer Berlin Heidelberg, 1998.

[12] Gustavus J. Simmons. The Prisoners' Problem and the Subliminal Channel. *Advances in Cryptology: Proceedings of Crypto 83*, pages 51–67, 1984.

[13] Adam Young and Moti Yung. Cryptovirology: extortion-based security threats and countermeasures. In *Proceedings 1996 IEEE Symposium on Security and Privacy*, pages 129–140, May 1996.

[14] Adam Young and Moti Yung. The Dark Side of "Black-Box" Cryptography or: Should We Trust Capstone? In Neal Koblitz, editor, *Advances in Cryptology — CRYPTO '96*, pages 89–103, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[15] Adam Young and Moti Yung. *Malicious Cryptography: Exposing Cryptovirology*. John Wiley & Sons, 2004.

Paper I

Malware encryption schemes – rerandomizable ciphertexts encrypted using environmental key

Herman Galteland and Kristian Gjøsteen

Published in Paradigms in Cryptology – Mycrypt 2016. Malicious and Exploratory Cryptology, full version available on ePrint 2017/1007

Malware encryption schemes – rerandomizable ciphertexts encrypted using environmental keys *

Herman Galteland† and Kristian Gjøsteen

Department of Mathematical Sciences,
NTNU – Norwegian University of Science and Technology
{herman.galteland, kristian.gjosteen}@ntnu.no

Abstract

It has been shown that encrypting malware prevents an opponent, defending a network of computers, from analyzing the malicious code and identifying the intentions of the malware author. We discuss malware encryption schemes that use environmental encryption keys, generated from computers the malware author intends to attack, and use rerandomization techniques to make each malware sample in the network indistinguishable. We are interested in hiding the intentions and identity of the malware author, not in hiding the existence of malware.

Keywords. Malicious cryptography, environmental keys, rerandomization, provable security.

1 Introduction

Malware is software maliciously installed on a computer designed to give functionality and behavior desired by the *malware author*, but not by the legitimate computer owner.

*This is an extended version of [7]

†This work is funded by Nasjonal sikkerhetsmyndighet (NSM), www.nsm.stat.no

This work extends on our previous work [7], where we have included an additional encryption scheme. Our goal is to study malware propagation and how to protect propagating malware from analysis. We do not study the construction of computer viruses or any other types of malware, but rather how to construct schemes designed to encrypt malware that hides the intentions and the identity of the malware author. In the previous work we presented two schemes as proofs of concepts, whereas in this work we include a third scheme that extends on the first two constructions and uses several keys to encrypt malware.

Each encryption scheme gives us insight on how a malware author could use cryptography maliciously. For each scheme, we describe how encrypted malware behaves, the technical details, and their strengths and weaknesses.

1.1 Real world examples

BurnEye [13] is a tool designed to defend binary files and is an example on how to protect malware. The tool adds three layers of protection to a file: obfuscation, encryption, and a fingerprint layer. The fingerprint layer ensures that the file can only be executed on a specific computer that has the specifications stated by the fingerprint. The encryption layer uses a user-chosen password as the encryption key such that the file can only be executed (or analyzed) by someone who knows the password.

Gauss [10] is a sophisticated malware that uses encryption to protect certain payloads. Gauss uses *environmental keys* to decrypt these payloads, where an environmental key is generated from locally available data. The malware gathers data on the infected computer and hashes it to create decryption keys, where the string of data that results in the correct key is selected by the malware author. The malicious code can only be executed when the correct key is produced, that is, when the malware infects the intended target. To our knowledge the contents of the encrypted payloads of Gauss are still unknown.

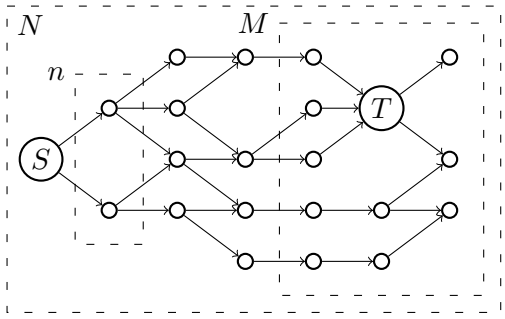


Figure 1: Malware propagating into an opponent's network. The source S infects n initial nodes. The opponent protects a network of M nodes. There are at most N malware samples.

1.2 Malware propagation

Consider a malware author whose objective is to attack a specific location (or locations). The malware author's goals is to hide his intentions and identity. The malware author's opponent is an *opponent* observing and defending a network of nodes, which contains one (or more) of the malware author's target(s). The goal of the opponent is to detect malware targeting any node in the network he is protecting and to discover the intentions and identity of the malware author. Hiding the mere existence of malware from the opponent is a distinct problem and not one we consider in the current work.

1.2.1 Setup

We use the following model to describe malware propagation, see Figure 1. The source S, the malware author, infects n initial nodes with (different variations of) his malware. Released malware infects subsequent nodes by making similar copies of themselves and propagates throughout the network.

The opponent protects M nodes in the network from malware threat. We assume he has full knowledge of the environment he is protecting. By observing the wider network the opponent can find at most N malware samples.

Every direct link to the malware author increases the opponent's chance to discover the author's identity. Therefore, to avoid identification, the malware author should perform as few initial infections as possible and use indirect paths to the target node.

1.2.2 Encrypting malware payload

The malware author encrypts the malware payload to increase the opponent's workload. Encrypting the payload prevents an opponent reverse engineering the malicious code and hides the intentions of the malware author [4, 5]. We use encryption keys derived from environmental parameters, network triggers, or a combination of these [11]. The environmental information is gathered from the target node and could consist of, for example, IP address, PATH variables, and/or any arbitrary time and date. This requires the information about the target node to be guessed, or to be general information (to target several nodes).

An encrypted malware consists of the *encrypted payload*, containing the malicious code, and a cleartext *loader* that gathers environmental parameters to generate decryption keys.

To initialize an encrypted malware the author chooses environmental data identifying the target node(s), hashes the data to create an encryption key, encrypts the payload using the key, and releases the encrypted malware. When malware infects a new node the cleartext loader determines the environmental data of the infected node, hashes the data to derive $K \geq 1$ keys, and attempts to decrypt the payload using the K derived keys. If the decryption is a success then the malicious code can be executed. Otherwise the loader creates copies of the malware and infects new nodes in the network.

The malware author can initialize at most n distinct encrypted malware, one for each initial infection, which encrypts the same malicious code. Hence, there are at most n distinct encrypted payloads among the samples collected by the opponent. Each sample is encrypted and has an unknown target. If the opponent wants to guarantee that none of these n samples would attack a node in his network then he needs to do roughly K trial decryptions for each of his M nodes. Hence, the opponent's workload is at most nMK.

Malware consist of a cleartext loader and an encrypted payload. When malware arrives on a new host the loader is executed and preforms the following steps:

1. The loader scans the host environment and determines the environmental data.
2. The loader hashes the environmental data to produce one or more keys.
3. The loader tries to decrypt the encrypted payload with each key.
4. If the decryption succeeds, the decrypted payload is executed.
5. The malware may also attempt to infect some other host, in which case the encrypted payload is rerandomized before it is transmitted to the new host.

Note that the malware author will certainly use some polymorphic engine and other standard malware techniques in order to provide a basic level of protection for the cleartext loader and the encrypted payload.

Figure 2: The malware attack process.

1.2.3 Rerandomizable encrypted payload

Instead of making exact copies of the malware we want the loader to *rerandomize* [2, 8] the encrypted payload. The rerandomization process takes as input an encrypted payload and some uniformly random values to produce a new ciphertext that encrypts the same malicious code. Hence, the loader can produce several different-looking encrypted malware to infect nodes without needing the knowledge of the secret key. The process is described in Figure 2.

We want to rerandomize the encrypted payloads in such a way that any two malware samples are indistinguishable. If the opponent is unable to distinguish between malware samples then, essentially, there are N unique variations of the malware in the network. This means that the opponent need to do K trial decryptions for N samples for M different nodes to ensure that none of the malware samples are targeting any of his nodes. This will increase the opponent's workload to NMK.

Since the malware creates different variations of itself, the malware author can choose n to be small and, possibly, significantly reducing the risk of unveiling his identity.

1.2.4 Path variation

Instead of using a single encryption key we can choose to use several keys, derived from different nodes in the network, describing a path

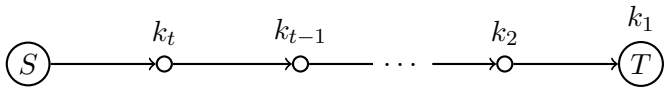


Figure 3: Location of keys on the path towards the target

towards the target (see Figure 3). The last key k_1 is derived from environmental data identifying the target node, just as before. The remaining keys, called *default keys*, are derived from environmental data that is available on all nodes in the path. This requires the malware author to investigate and gather the environmental data of each node in the path towards the target before malware can be encrypted. The difference between the path and the single key variations is that the path scheme will *always* try to decrypt the payload using a key (a default key in most cases).

The path variation encrypted malware can only be decrypted if malware infect the nodes in the correct path toward the target. There is no difference between malware samples in the correct path and malware samples in a wrong path, hence, all samples needs to be treated as if they are both in the path and not in the path by the opponent. The only difference is when the malicious code has been executed and the opponent notices.

If the opponent only wants to know if a node in his network is targeted he has to check all possible paths between a node with a sample to all other nodes in his network and see if the sample decrypt correctly. If the opponent finds the target node, either by analysis or by noticing that the malicious code was executed, then he can trace the correct path back towards the source, by using the algorithms used by the malware, and possibly find information that could identify the malware author. However, this requires knowledge of the full network and the opponent only has knowledge over his network – unless the source is inside the opponent's network. This is an unwanted trait, but it seems to be unavoidable if we also wish to include the rerandomization. Without knowledge of the target node the opponent cannot trace back to the source by looking at the algorithms alone.

1.2.5 Limitations

The limitation of our schemes is that the opponent can always guess, or predict, the target of the malware author. Also, if the malware reaches its target, the payload will be decrypted and executed. If the opponent notices the attack he will be able to deduce the environmental key and thus be able to decrypt the payload. This seems impossible to avoid.

Once an opponent discovers the key used for one sample, he can easily discover all other samples corresponding to that key. However, the malware author will hope that different opponents are unwilling to reveal that they are under attack, they somehow consider this fact sensitive, and that they therefore do not share discovered keys. This means that one opponent's success may not make the other opponent's work easier.

For the path variant, if the opponent discover the target node then it is possible to find the source. If the opponent finds the correct path back to the source he can discover all the encryption keys. However, this is not enough to determine if another malware sample is encrypted under the same keys, especially if that sample has infected a node in a wrong path. The opponent has to find the sample's infection path back to the source first, remove any decryption keys used (to restore the sample to its initial condition), and then check if the keys decrypt the payload. This would require knowledge of the full network.

1.2.6 The potential threat

Assuming there is more than one malware author, an opponent cannot be certain of whether every new encrypted malware sample corresponds to one he has previously determined is no threat or a genuinely new piece of malware. That is, all malware samples created by different malware authors looks like the same encrypted malware. This requires all malware authors to agree on a malware encryption standard, they all have malware that have the same size, use the same loader, and otherwise create encrypted malware that looks the same. This seems unlikely. If malware authors do not agree on a standard

Path variation malware consist of a cleartext loader and an encrypted payload. When path variation malware arrives on a new host the loader is executed and preforms the following steps:

1. The loader scans the host environment and determines the environmental data.
2. The loader hashes the environmental data to produce one or more keys and a default key.
3. The loader check each key if they can decrypt the encrypted payload.
   - If a key is found use it to decrypt, if not use the default key to decrypt.
4. The loader checks if the decrypted payload could be executed.
5. The malware may also attempt to infect some other host, in which case the encrypted payload is rerandomized before it is transmitted to the new host.

Note that the malware author will certainly use some polymorphic engine and other standard malware techniques in order to provide a basic level of protection for the cleartext loader and the encrypted payload.

Figure 4: Path variation malware attack process.

then the opponents can use these pieces of information to classify samples.

1.3 Related work

Traditionally cryptography has been developed and used as a defense against attackers. However, it is clear that cryptography can also be of use to the attackers.

Young and Yung where the first to raise the concern about malicious use of cryptography (cryptovirology) [15] and have several works related to malware construction and propagation, where we will mention three related papers. First, Young and Yung designed a virus capable of encrypting files on the victim's computer and hold them for ransom [14]. Second, they describe how to utilize a mix network to mix programs and propagate malware [15]. Third, they designed a mobile program that carries a rerandomizable ciphertext, which enables anonymous communication, where the program takes random walks through a network and rerandomizes the ciphertext at each node, using a system called Feralcore [16].

The mix network and the mobile program, by Young and Yung, use the idea of universal re-encryption, by Golle et al. [8], to re-encrypt ciphertexts. The re-encryption process transforms the ci-

phertexts into a new ciphertext that encrypts the same message and does not require knowledge about the public key. Similar to universal re-encryption is the notion of rerandomization by Canetti et al. [2].

Filiol showed that by encrypting malware payload [4, 5] one can prevent anyone from analyzing the code and reverse engineer it, possibly using the environmental keys of Riordan and Schneier [11] as encryption keys. Similar to Riordan and Schneier's environmental keys, secure triggers [6, 9] are also used to keep certain content private until a particular event occurs.

1.4 Overview

The rest of this paper contains the technical details of our schemes. The general cryptosystem designed to encrypt and rerandomize malware payload is described in Section 2.1. The basic scheme, in Section 2.2, shows that malware encryption described in the introduction is possible in theory, however, the scheme is not practical because it can encrypt short messages. The extended scheme, in Section 2.3, is based on the basic scheme and can encrypt longer messages, making it more practical. The basic and extended schemes use the malware attack process described in Figure 2. The path scheme, in Section 2.4, is the path variant of both the basic and the extended scheme and uses several encryption keys instead of one. The malware attack process for the path variation is described in Figure 4. For each scheme we show that they are secure using games, where the opponent is asked to distinguish between ciphertexts encrypting the same message and ciphertexts encrypting two different messages. That is, we will simulate whether an opponent is able to distinguish malware samples. The security proof of the basic scheme is in Section 2.2.1, the security proof of the extended scheme is in Section 2.3.1, and the security proof of the path scheme is in Section 2.4.3. All three proofs are similar.

2 Rerandomizable encryption schemes

In this section we present three encryption schemes designed to encrypt and rerandomize malware payload. The first scheme is a basic proof of concept and the second is an extension of the basic scheme capable of encrypting longer payloads. The third scheme is the path variant of the first two. Further, we show that it is hard to distinguish between encrypted payload samples by using games.

As a simplification we denote payload as messages, encrypted payload as ciphertexts, replication of malware as rerandomization of ciphertexts, and environmentally derived keys as keys.

2.1 Preliminary

In each scheme we have an algorithm $\mathcal{E}$ encrypting messages, an algorithm $\mathcal{D}$ decrypting ciphertexts, and an algorithm $\mathcal{R}$ rerandomizing ciphertexts. In the path variant of the extended scheme (in Section 2.4) we add a padding functionality to the rerandomize algorithm and rename it to a padding algorithm $\mathcal{P}$.

Encryption For a message m and a key k the encryption algorithm $\mathcal{E}(k, m)$ outputs a ciphertext c.

Decryption For a ciphertext c and a key k the decryption algorithm $\mathcal{D}(k, c)$ either outputs a message m or a special symbol $\perp$ indicating decryption failure.

Rerandomization For a ciphertext c, encrypting a message m, the rerandomize algorithm $\mathcal{R}(c)$ outputs a ciphertext c' encrypting the same message m.

The output distribution of the rerandomize algorithm should be computationally indistinguishable from the output distribution of the encryption algorithm. That is, it should be hard to determine if two different ciphertexts encrypts the same message or not.

The systems should be correct, we should almost always be able to decrypt all ciphertexts output by the encryption algorithm and any rerandomized ciphertexts output by the rerandomize algorithm.

Correctness If c was output from $\mathcal{E}(k, m)$ then $\mathcal{D}(k, c)$ will always output m except with negligible probability.

Rerandomization If c was output by $\mathcal{E}(k, m)$ then the output distribution of $\mathcal{R}(c)$ should be computationally indistinguishable from the output distribution of $\mathcal{E}(k, m)$. Furthermore, if c' was output from $\mathcal{R}^n(\mathcal{E}(k, m))$, for any $n \geq 1$ then $\mathcal{D}(k, c')$ will always output m except with negligible probability.

We will not always be able to apply an arbitrary number of rerandomizations to a ciphertext without getting decryption errors, which we will see is the case in Section 2.3 and in Section 2.4.

The security requirements of our cryptosystems reflect the intentions of the malware author. It should be difficult to guess the malware author's target, and it should be hard to determine if two ciphertexts are the encryption of the same message or not.

Key indistinguishability It should be hard to say something about which key a ciphertext is encrypted under.

Ciphertext indistinguishability It should be hard to decide if two ciphertexts, encrypted under the same key, decrypt to the same message or not.

2.2 Basic scheme

The basic scheme is based on the ElGamal cryptosystem over a group G of prime order p generated by g. This scheme is essentially the same as the encryption scheme proposed by Golle et al [8]. The key used in the algorithms is generated by the loader using environmental data.

Encryption For a message $m \in G$ and a key $k \in \mathbb{Z}_p^*$, sample $r \xleftarrow{r} \mathbb{Z}_p^*$ and $s \xleftarrow{r} \mathbb{Z}_p$, and output

$$c = (x, y, z, w) = (g^r, g^{kr}, g^s, g^{ks}m).$$

Decryption For a ciphertext $c = (x, y, z, w)$ and a key $k \in \mathbb{Z}_p^*$ check if $x^k = y$. If it is then output

$$m = z^{-k}w.$$

If not output $\perp$.

Rerandomize For a ciphertext $c = (x, y, z, w)$, sample $r' \xleftarrow{r} \mathbb{Z}_p^*$ and $s' \xleftarrow{r} \mathbb{Z}_p$, and output

$$c' = (x', y', z', w') = (x^{r'}, y^{r'}, zx^{s'}, wy^{s'}).$$

Correctness If $c = (x, y, z, w)$ was output by the encryption algorithm then there exists parameters r, s, k, and a message m such that

$$c = (x, y, z, w) = (g^r, g^{kr}, g^s, g^{ks}m).$$

With input c the rerandomize algorithm will output $c' = (x', y', z', w')$ where

$$\begin{aligned} x' &= x^{r'} = g^{rr'}, \\ y' &= y^{r'} = g^{krr'}, \\ z' &= zx^{s'} = g^s g^{rs'} = g^{s+rs'}, \\ w' &= wy^{s'} = g^{ks} g^{krs'} m = g^{k(s+rs')} m. \end{aligned}$$

That is, $c' = (g^{rr'}, g^{krr'}, g^{s+rs'}, g^{k(s+rs')}m)$. Since $r \neq 0$, we get that $s + rs'$ can take any value in $\mathbb{Z}_p$ except s and all values are equally probable. Hence, the output distribution of the encryption and rerandomize algorithms are computationally indistinguishable and has the same structure, that is,

$$c' = (g^{\hat{r}}, g^{k\hat{r}}, g^{\hat{s}}, g^{k\hat{s}}m)$$

for some parameters $\hat{r}$, $\hat{s}$, k, and a message m.

For a ciphertext $c = (x, y, z, w)$, output by the encryption, and the correct key k we have that $x^k = (g^r)^k = g^{kr} = y$. The message m is retrieved by computing

$$z^{-k} w = (g^s)^{-k} g^{ks} m = g^{-ks+ks} m = m.$$

Similar for a ciphertext c' output by the rerandomization algorithm.

We will have decryption errors if the sampled values s, in the encryption algorithm, or s', in the rerandomization algorithm, is equal to zero. This can be made negligible for large values of p, hence, the decryption algorithm is almost always correct.

Longer messages The limitation of the basic scheme is that the message size is relatively small. One option is to encrypt several messages under the same key. That is, a set of messages $\{m_1, m_2, \ldots, m_n\}$ can be encrypted as

$$(g^r, g^{kr}, g^{s_1}, g^{ks_1}m_1, g^{s_2}, g^{ks_2}m_2, \ldots, g^{s_n}, g^{ks_n}m_n)$$

for some parameters $s_1, s_2, \ldots, s_n$, r, and key k. However, this is inefficient. In Section 2.3 we construct the extended scheme where we use techniques from symmetric cryptography to encrypt longer messages.

2.2.1 Security of the basic scheme

The decryption key is derived from environmental data sampled by the loader from the infected computer. From the opponent's perspective the collection of sampled data types can be considered as a probability space of possible decryption keys. We will denote this space by D. If the size of D is large then the opponent is less likely to guess the correct decryption key, where the size of D is determined by the number of keys generated by the loader.

We show that the opponent $\mathcal{O}$ is unable to distinguish between ciphertexts and that his advantage is determined by D, that is, the probability of the opponent guessing the correct key. To prove this we use games [12]. In our games we start by simulating Experiment 1 where we ask the opponent to differentiate between the two cases; two ciphertexts encrypting different messages, and two ciphertexts encrypting the same message.

Experiment 1 Given two ciphertext c_1 and c_2, decide if

$$\begin{array}{l} c_1 = \mathcal{E}(k_1, m_1) \\ c_2 = \mathcal{E}(k_2, m_2) \end{array} \quad \text{or} \quad \begin{array}{l} c_1 = \mathcal{E}(k_1, m_1) \\ c_2 = \mathcal{R}(c_1) \end{array}$$

for some messages m_1, m_2 and keys k_1, k_2.

We show that the security of the scheme can be based on the hardness of the Decisional Diffie-Hellman (DDH) problem [1] in the random oracle model. The DDH problem is to distinguish tuples

Algorithm 1 Game 0 simulating Experiment 1

$u_1, u_2 \xleftarrow{r} D$, $k_1 \leftarrow H(u_1)$, $k_2 \leftarrow H(u_2)$, $b \xleftarrow{r} \{0,1\}$
Get m_1, m_2 from $\mathcal{O}$
if $b = 0$ **then**
    $c_1 \leftarrow \mathcal{E}(k_1, m_1)$
    $c_2 \leftarrow \mathcal{E}(k_2, m_2)$
    Send c_1, c_2 to $\mathcal{O}$
if $b = 1$ **then**
    $c_1 \leftarrow \mathcal{E}(k_1, m_1)$
    $c_2 \leftarrow \mathcal{R}(c_1)$
    Send c_1, c_2 to $\mathcal{O}$
Get b' from $\mathcal{O}$

of the form (g, g^a, g^b, g^{ab}) and tuples of the form (g, g^a, g^b, g^c), for some $a, b, c \in \mathbb{Z}_p^*$. Where the DDH assumption states that the DDH problem is hard to solve.

To create the encryption keys we use an oracle to hash elements drawn from the probability space D. We denote the oracle by H, where it should be impossible to get any information about the input of the oracle by looking at its output.

Game 0 Simulate Experiment 1. See Algorithm 1 for the detailed procedure. Let E_0 be the event that $b = b'$ in Game 0.

Game 1 Stop the game if the opponent queries either u_1 or u_2 (guessed the correct key). The random oracle H outputs $b' \xleftarrow{r} \{0,1\}$. We denote this event by F_1.

Let E_1 be the event that $b = b'$ in Game 1. Unless event F_1 occurs Game 1 behaves just like Game 0. Thus $E_0 \wedge \neg F_1 \iff E_1 \wedge \neg F_1$ and by the difference lemma we have

$$|\Pr[E_0] - \Pr[E_1]| \leq \Pr[F_1].$$

Game 2 Draw $k_1, k_2 \xleftarrow{r} \mathbb{Z}_p^*$ and stop querying the oracle. The opponent can still query the oracle, hence, we need to draw samples from D to check if the opponent is guessing the correct key(s).

Let E_2 be the event that $b = b'$ in Game 2. The output of H is indistinguishable from uniform samples of $\mathbb{Z}_p^*$, hence, $\Pr[E_1] = \Pr[E_2]$.

Game 3 For uniform $s, s' \xleftarrow{r} \mathbb{Z}_p^*$ and keys k_1, k_2 precompute

$$(x, y, z, w) = (g, g^{k_1}, g^s, g^{k_1 s}), \text{ and } (x', y', z', w') = (g, g^{k_2}, g^{s'}, g^{k_2 s'})$$

before receiving message m_1 and m_2.

Let E_3 be the event that $b = b'$ in Game 3. After encrypting both messages, or encrypting one and rerandomizing it, we get that the output of the encryption, and rerandomize, algorithms are exactly the same in Game 2 and Game 3. Thus, $\Pr[E_2] = \Pr[E_3]$.

Game 4 Let $(x, y, z, w) = (g, g^{k_1}, g^s, g^{k_1 s})$ be the first tuple and

$$\begin{aligned}(x', y', z', w') &= (x, x^a y^c, z x^b, w^c z^a y^{cb} x^{ab}) \\ &= (g, g^{a+ck_1}, g^{b+s}, g^{(a+ck_1)(b+s)})\end{aligned}$$

be the second, for some $a, b, c \xleftarrow{r} \mathbb{Z}_p^*$.

Let E_4 be the event that $b = b'$ in Game 4. Since the tuples of Game 4 results in the same output space as the tuples of Game 3 we get that $\Pr[E_3] = \Pr[E_4]$.

Game 5 The output of the rerandomize algorithm is of the form

$$(g^{rr'}, g^{k_1 rr'}, g^{s+rs'}, g^{k_1(s+rs')} m)$$

for some r, r', s and s', where $s + rs' \neq s$ since the variables used in the algorithm cannot be zero. This gives us a statistical difference of $1/p$ between the output distributions. Change the rerandomization algorithm such that the second ciphertext (in case $b = 1$) is computed as

$$(g^{rr'}, g^{k_1 rr'}, g^{s+rs'+\tilde{s}}, g^{k_1(s+rs'+\tilde{s})} m_1),$$

where $\tilde{s} \xleftarrow{r} \mathbb{Z}_p$. The new sum $s + rs' + \tilde{s}$ can be any value in $\mathbb{Z}_p$ and all values are equally probable.

Let F_5 be the event that $s+rs'+\tilde{s} = s$ and let E_5 be the event that $b = b'$ in Game 5. Unless F_5 occurs, Game 4 and Game 5 behaves the same, that is, $E_4 \wedge \neg F_5 \iff E_5 \wedge \neg F_5$ and by the difference lemma we get that

$$|\Pr[E_4] - \Pr[E_5]| \leq \Pr[F_5] = \frac{1}{p}.$$

Algorithm 2 Input: (x, y, z, w)

$u_1, u_2 \xleftarrow{r} D$, $b \xleftarrow{r} \{0, 1\}$
$a, b, c \xleftarrow{r} \mathbb{Z}_p^*$
$(x', y', z', w') = (x, x^a y^c, z x^b, w^c z^a y^{cb} x^{ab})$
Get m_1, m_2 from $\mathcal{O}$
if $b = 0$ **then**
    $r, r' \xleftarrow{r} \mathbb{Z}_p^*$
    $c_1 \leftarrow (x^r, y^r, z, w m_1)$
    $c_2 \leftarrow (x'^{r'}, y'^{r'}, z', w' m_2)$
    Send c_1, c_2 to $\mathcal{O}$
if $b = 1$ **then**
    $r, r', s', \tilde{s} \xleftarrow{r} \mathbb{Z}_p^*$
    $c_1 \leftarrow (x^r, y^r, z, w m_1)$
    $c_2 \leftarrow (x^{rr'}, y^{rr'}, z x^{rs'+\tilde{s}}, w y^{rs'+\tilde{s}} m_1)$
    Send c_1, c_2 to $\mathcal{O}$
Get b' from $\mathcal{O}$

Game 6 Change the first tuple into the form $(g, g^{a'}, g^{b'}, g^{c'})$, for uniform elements $a', b', c' \in \mathbb{Z}_p^*$. The second tuple will then look like

$$(g, g^{a+a'c}, g^{b+b'}, g^{ab+ab'+a'bc+cc'}).$$

Let E_6 be the event that $b = b'$ in Game 6. Since we are using uniform elements in the tuples the encryption and rerandomization algorithms are, essentially, one-time pads. Hence, $\Pr[E_6] = 1/2$.

We claim that $|\Pr[E_5] - \Pr[E_6]|$ is equal to the advantage of an adversary $\mathcal{A}$, solving the DDH problem, with access to Algorithm 2. We let $\mathrm{Adv}_{\mathrm{ddh}}(\mathcal{A})$ denote the advantage of $\mathcal{A}$. Algorithm 2 has (g, g^a, g^b, g^c) as input, for some a, b, and c, where c can be equal to ab. The algorithm simulates Game 5 if the input is on the form (g, g^a, g^b, g^{ab}) and

$$\Pr[A_2(g, g^a, g^b, g^{ab}) = 1 \mid a, b \xleftarrow{r} \mathbb{Z}_p^*] = \Pr[E_5].$$

If the input is on the form (g, g^a, g^b, g^c) the algorithm proceed as in Game 6 and

$$\Pr[A_2(g, g^a, g^b, g^c) = 1 \mid a, b, c \xleftarrow{r} \mathbb{Z}_p^*] = \Pr[E_6],$$

where the DDH advantage of $\mathcal{A}$ is equal to $|\Pr[E_5] - \Pr[E_6]|$.

Summary From the games we bound the advantage of the opponent $\mathcal{O}$.

$$\begin{aligned}
\mathrm{Adv}(\mathcal{O}) &= |\Pr[E_0] - 1/2| \\
&= |\Pr[E_0] - \Pr[E_1] + \Pr[E_1] - \Pr[E_2] + \Pr[E_2] - \Pr[E_3] \\
&\qquad + \Pr[E_3] - \Pr[E_4] + \Pr[E_4] - \Pr[E_5] + \Pr[E_5] \\
&\qquad - \Pr[E_6] + \Pr[E_6] - 1/2| \\
&\leq |\Pr[E_0] - \Pr[E_1]| + |\Pr[E_4] - \Pr[E_5]| + |\Pr[E_5] - \Pr[E_6]| \\
&\leq \Pr[F_1] + \frac{1}{p} + \mathrm{Adv}_{\mathrm{ddh}}(\mathcal{A}).
\end{aligned}$$

By the DDH assumption the DDH advantage of $\mathcal{A}$ is negligible and, for large enough p, we get that the advantage of our opponent is determined by the probability that $\mathcal{O}$ guesses or predicts the correct key, that is, determined by the probability space D.

2.3 Extended scheme

In the extended scheme we represent messages as bit strings to encrypt longer messages. However, this change reduces the number of rerandomizations we can perform on a ciphertext and we need to relax the requirements of the cryptosystem.

Correctness If c was produced by iteratively applying $\mathcal{R}$ to the output of $\mathcal{E}(k, m)$ at most n times then $\mathcal{D}(k, c)$ will output m except with negligible probability.

We use a pseudorandom function $f : G \to \{0,1\}^N$ mapping group elements to bit strings of length N, for some large $N \in \mathbb{N}$. We let f_L denote the truncation of the output to L bits, for $L < N$. We assume that group elements can be encoded as bit strings of length at most $l/2$. The construction in this section is very similar to the hybrid scheme by Golle et al [8]. The key used in the algorithms is generated by the loader using environmental data.

Encryption For a message $m \in \{0,1\}^L$ and a key $k \in \mathbb{Z}_p^*$, sample $r \xleftarrow{r} \mathbb{Z}_p^*$, $s \xleftarrow{r} \mathbb{Z}_p$, $\gamma \xleftarrow{r} G$ and output

$$c = g^r || g^{kr} || g^s || g^{ks}\gamma || \left(f_{L+l(n+1)+1}(\gamma) \oplus (m||1||0^{l(n+1)}) \right).$$

Decryption For a ciphertext $c = x||y||b'_0$ and a key $k \in \mathbb{Z}_p^*$ check if $x^k = y$. If not output $\perp$. If it is let $b'_0 = z_0||w_0||b_0$ and compute

$$b'_1 = f_{|b_0|}(z_0^{-k} w_0) \oplus b_0.$$

If the result b'_1 ends in $l' \geq l$ zeros then the message is the result minus the tail of zeros and exactly one 1. Otherwise interpret b'_1 as $z_1||w_1||b_1$ and repeat the procedure. If this procedure is repeated $n+1$ times output $\perp$.

Rerandomization For a ciphertext $c = x||y||b_m||b_l$, where b_l is the last l bits. Sample $r' \xleftarrow{r} \mathbb{Z}_p^*$, $s' \xleftarrow{r} \mathbb{Z}_p$, $\gamma' \xleftarrow{r} G$, and output

$$c' = x^{r'}||y^{r'}||x^{s'}||y^{s'}\gamma'||\left(f_{|b_m|}(\gamma') \oplus b_m\right).$$

Correctness Before applying the rerandomize algorithm, b_m looks like

$$g^s||g^{ks}\gamma||\left(f_{L+ln+1}(\gamma) \oplus (m||1||0^{ln})\right)$$

for $s \in \mathbb{Z}_p^*$, key k, and $\gamma \in G$. The l last bits we discard, b_l, is an "encryption" of l zeros. We can therefore only perform n rerandomizations on a ciphertext before we get decryption failure, that is, there would be no tail of zeros left for the decryption algorithm to detect.

If $c = x||y||b'_0$ was output from the encryption algorithm, we have that $x^k = g^{kr} = y$. Hence, we can write b'_0 as $z||w||b_0$, and compute

$$\begin{aligned} f_{|b_0|}(z^{-k} w) \oplus b_0 &= f_{L+l(n+1)+1}(\gamma) \oplus f_{L+l(n+1)+1}(\gamma) \oplus (m||1||0^{l(n+1)}) \\ &= (m||1||0^{l(n+1)}). \end{aligned}$$

The result ends with a tail of $l' \geq l$ zeros and the output message is m.

Let c be a ciphertext that was produced by iteratively applying the rerandomize algorithm to the output of $\mathcal{E}(k, m)$ t times, where $1 \leq t \leq n$. Write c as $x||y||b'_t$, where $x = g^{r_1 \cdots r_{t+1}}$, $y = g^{k(r_1 \cdots r_{t+1})}$. Note b'_t has the form

$$(g^{r_1 \cdots r_t})^{s'}||(g^{k(r_1 \cdots r_t)})^{s'}\gamma_t||\left(f_{L+l(n+1-t)+1}(\gamma_t) \oplus b'_{t-1}\right)$$

for $s', r_1, \ldots, r_{t+1} \in \mathbb{Z}_p^*$, key k, and group element $\gamma_t \in G$. Observe that for all $1 \leq t \leq n$ we have that $x^k = y$. Thus we can write $b'_t = z_t||w_t||b_t$ and compute

$$\begin{aligned} f_{|b_t|}(z_t^{-k} w_t) \oplus b_t &= f_{L+l(n+1-t)+1}(\gamma_t) \oplus f_{L+l(n+1-t)+1}(\gamma_t) \oplus b'_{t-1} \\ &= b'_{t-1} \end{aligned}$$

where b'_{t-1} does not end with a tail of $l' \geq l$ zeros (except with negligible probability) since the ciphertext is also encrypted once using with the encryption algorithm (in addition to the t rerandomizations). Let $b'_{t-1} = z_{t-1}||w_{t-1}||b_{t-1}$ and repeat the process t more times. In the last iteration we perform the decryption on the bit string $z_0||w_0||b_0$, where b_0 looks like

$$f_{L+l(n+1-t)+1}(\gamma_0) \oplus (m||1||0^{l(n+1-t)}).$$

We know this decrypts to the message m.

We will have decryption errors if the sampled values s, in the encryption algorithm, or s', in the rerandomization algorithm, is equal to zero. This can be made negligible for large values of p, hence, the decryption algorithm is almost always correct.

2.3.1 Security of the extended scheme

Similar to the security proof of the basic scheme, we show that the opponent is unable to distinguish between encrypted ciphertexts and that his advantage is determined by D, the probability of guessing the correct key. As in the proof of the basic scheme, we use games to simulate Experiment 1.

Game 0 Simulate Experiment 1. The full procedure can be seen in Algorithm 1. Let E_0 be the event that $b = b'$ in Game 0.

Game 1 Similar to the basic Game 1, where we get $|\Pr[E_0] - \Pr[E_1]| \leq \Pr[F_1]$.

Game 2 Similar to the basic Game 2, where we get $\Pr[E_1] = \Pr[E_2]$.

Algorithm 3 Input: (x, y, z, w)

$u_1, u_2 \xleftarrow{r} D$, $b \xleftarrow{r} \{0,1\}$
$a, b, c \xleftarrow{r} \mathbb{Z}_p^*$
$(x', y', z', w') = (x, x^a y^c, z x^b, w^c z^a y^{cb} x^{ab})$
Get m_1, m_2 from $\mathcal{O}$
if $b = 0$ **then**
    $r, r'\gamma, \gamma' \xleftarrow{r} \mathbb{Z}_p^*$
    $c_1 \leftarrow x^r || y^r || z || w\gamma || \left(f_{L+l(n+1)+1}(\gamma) \oplus (m_1 || 1 || 0^{l(n+1)})\right)$
    $c_2 \leftarrow x'^{r'} || y'^{r'} || z' || w'\gamma' || \left(f_{L+l(n+1)+1}(\gamma') \oplus (m_2 || 1 || 0^{l(n+1)})\right)$
    Send c_1, c_2 to $\mathcal{O}$
if $b = 1$ **then**
    $r, r's', \gamma, \gamma' \xleftarrow{r} \mathbb{Z}_p^*$
    $c_1 \leftarrow x^r || y^r || z || w\gamma || \left(f_{L+l(n+1)+1}(\gamma) \oplus (m_1 || 1 || 0^{l(n+1)})\right)$
    Let $c_1 = x^r || y^r || b_m || b_l$, where b_l is the last l bits
    $c_2 \leftarrow x^{rr'} || y^{rr'} || x^{rs'} || y^{rs'} \gamma' || \left(f_{|b_m|}(\gamma') \oplus b_m\right)$
    Send c_1, c_2 to $\mathcal{O}$
Get b' from $\mathcal{O}$

Game 3 Similar to the basic Game 3, where we get $\Pr[E_2] = \Pr[E_3]$.

Game 4 Similar to the basic Game 4, where we get $\Pr[E_3] = \Pr[E_4]$.

Game 5 Similar to the basic Game 6 except we use Algorithm 3 instead. We get that $|\Pr[E_4] - \Pr[E_5]| = \mathrm{Adv}_{\mathrm{ddh}}(\mathcal{A})$.

Game 6 Sample a function h from a family Γ of all functions from G to $\{0,1\}^N$ instead of using the function f. We denote h_L as the truncation of the output of h to L bits. The pseudorandom function (PRF) advantage of an adversary $\mathcal{B}$ is its ability to distinguishing f from any function h sampled from Γ. The PRF-advantage of $\mathcal{B}$ is negligible if f is pseudorandom. We let $\mathrm{Adv}_{\mathrm{prf}}(\mathcal{B})$ denote the advantage of $\mathcal{B}$.

Let E_6 be the event $b = b'$ in Game 6. We use an arbitrary function h, with a random group element γ, to encrypt the message m, hence, the output ciphertexts of the encryption and rerandomization algorithms can be any random bit string. Thus $\Pr[E_6] = 1/2$.

We claim that $|\Pr[E_5] - \Pr[E_6]|$ is equal to the PRF-advantage, where we use Algorithm 4. The algorithm draws a function h from

the family Γ, which may be equal to f. The PRF-advantage is

$$\left|\Pr[A_4(x,y,z,w)=1 \mid A_4 \leftarrow f] - \Pr[A_4(x,y,z,w)=1 \mid h \leftarrow \Gamma, A_4 \leftarrow h]\right|$$

which is equal to $|\Pr[E_5] - \Pr[E_6]|$.

Summary From the games we bound the advantage of the opponent $\mathcal{O}$.

$$\begin{aligned}
\mathrm{Adv}(\mathcal{O}) &= |\Pr[E_0] - 1/2| \\
&= |\Pr[E_0] - \Pr[E_1] + \Pr[E_1] - \Pr[E_2] + \Pr[E_2] - \Pr[E_3] \\
&\qquad + \Pr[E_3] - \Pr[E_4] + \Pr[E_4] - \Pr[E_5] + \Pr[E_5] \\
&\qquad - \Pr[E_6] + \Pr[E_6] - 1/2| \\
&\leq |\Pr[E_0] - \Pr[E_1]| + |\Pr[E_4] - \Pr[E_5]| + |\Pr[E_5] - \Pr[E_6]| \\
&\leq \Pr[F_1] + \mathrm{Adv}_{\mathrm{ddh}}(\mathcal{A}) + \mathrm{Adv}_{\mathrm{prf}}(\mathcal{B}).
\end{aligned}$$

By the DDH assumption the DDH advantage of $\mathcal{A}$ is negligible, and assuming f is pseudorandom the PRF advantage of $\mathcal{B}$ is negligible. Therefore, the advantage of the opponent is determined by the probability that the opponent guesses or predicts the correct key, that is, determined by the probability space D.

2.4 Path scheme

The path variation scheme encrypts malware under several keys and encrypted malware will be correctly decrypted if it travels on the correct path in the network toward the target. If a malware sample infects a node not in the path then it can no longer be correctly decrypted, except with negligible probability.

The scheme uses several encryption keys, where each key is generated from environmental data gathered from a node in the path. The malware author selects the path and, hence, needs knowledge about each node in the path to generate the encryption keys.

When encrypted malware infects a node the loader samples local environmental data to generate a set of keys and a *default key*, checks if one of the non-default keys can be used for decryption, if not it will use the default key in the decryption algorithm. If the node is in the

Algorithm 4 Input: (x, y, z, w)

$u_1, u_2 \xleftarrow{r} \mathcal{D}$, $b \xleftarrow{r} \{0,1\}$, $h \leftarrow \Gamma$
$a, b, c \xleftarrow{r} \mathbb{Z}_p^*$
$(x', y', z', w') = (x, x^a y^c, z x^b, w^c z^a y^{cb} x^{ab})$
Get m_1, m_2 from $\mathcal{O}$
if $b = 0$ **then**
    $r, r', \gamma, \gamma' \xleftarrow{r} \mathbb{Z}_p^*$
    $c_1 \leftarrow x^r || y^r || z || w\gamma || \left(h_{L+l(n+1)+1}(\gamma) \oplus (m_1 || 1 || 0^{l(n+1)})\right)$
    $c_2 \leftarrow x'^{r'} || y'^{r'} || z' || w'\gamma' || \left(h_{L+l(n+1)+1}(\gamma') \oplus (m_2 || 1 || 0^{l(n+1)})\right)$
    Send c_1, c_2 to $\mathcal{O}$
if $b = 1$ **then**
    $r, r', s', \gamma, \gamma' \xleftarrow{r} \mathbb{Z}_p^*$
    $c_1 \leftarrow x^r || y^r || z || w\gamma || \left(h_{L+l(n+1)+1}(\gamma) \oplus (m_1 || 1 || 0^{l(n+1)})\right)$
    Let $c_1 = x^r || y^r || b_m || b_l$, where b_l is the last l bits
    $c_2 \leftarrow x^{rr'} || y^{rr'} || x^{rs'} || y^{rs'} \gamma' || \left(h_{|b_m|}(\gamma') \oplus b_m\right)$
    Send c_1, c_2 to $\mathcal{O}$
Get b' from $\mathcal{O}$

correct path a key will be removed from the encrypted malware. If it is not in the correct path then the attempted decryption introduces a new random value to the ciphertext, which will not be removed on any subsequent node except for some negligible probability.

Malware payload will be encrypted using the keys $k_t, \ldots, k_2, k_1$, where $k_t, \ldots, k_2$ are default keys associated to a node in the path and k_1 is a specific key identifying the target node, see Figure 3.

2.4.1 Based on the basic scheme

Creating a path variant of the basic scheme is straight forward. We use the same algorithms except for the following. The encryption algorithm encrypts the malware under the sum of the keys, instead of a single key, and the decryption algorithm will always try to decrypt using a key. The algorithms are as follows.

Encryption For a message $m \in G$ and keys $k_1, k_2, \ldots, k_t \in \mathbb{Z}_p^*$, sample $r \xleftarrow{r} \mathbb{Z}_p^*$ and $s \xleftarrow{r} \mathbb{Z}_p$, and output

$$c = (x, y, z, w) = (g^r, g^{r(k_1+k_2+\cdots+k_t)}, g^s, g^{s(k_1+k_2+\cdots+k_t)} m).$$

Decryption For a ciphertext $c = (x, y, z, w)$ check if any of the non-

Algorithm 5 Extended path variation attack process.

```
1: compute c_t ← E((k_1, ..., k_t), m)
2: for i = 1, ..., n do
3:     compute c'_t ← P(c_t) and infect a computer
4: while node is infected with c'_i do
5:     compute c_i ← D(k, c'_i)
6:     if result is executable then
7:         run malware
8:     else
9:         compute c'_{i-1} ← P(c_i) and use it to infect a new node
```

default keys $\hat{k}$ satisfies $x^{\hat{k}} = y$,[1] if so let $k = \hat{k}$ if not let k be the default key. Output

$$c' = (x, y', z, w') = (x, x^{-k}y, z, z^{-k}w).$$

Rerandomize For a ciphertext $c = (x, y, z, w)$, sample $r' \xleftarrow{r} \mathbb{Z}_p^*$ and $s' \xleftarrow{r} \mathbb{Z}_p$, and output

$$c' = (x', y', z', w') = (x^{r'}, y^{r'}, zx^{s'}, wy^{s'}).$$

2.4.2 Based on the extended scheme

The path variant of the extended scheme utilizes an onion type encryption [3], where each onion layer is encrypted under one of the keys. The rerandomize algorithm hides keys inside the ciphertext using a PRF and locks it using a group element. Thus, we need to completely remove a key from the ciphertext when we use it in a decryption, hence, if we encrypt each layer using one key we can remove a key completely from the ciphertext.

Since we encrypt in layers we need a padding algorithm, $\mathcal{P}$, to pad the ciphertexts such they have the same default length, denoted L_D. The scheme pads the ciphertext after an encryption or a decryption. Note that the encryption algorithm no longer adds any zeros when encrypting. The padding algorithm also rerandomizes the ciphertexts, hence, it replaces the rerandomize algorithm.

[1] This will be true if the malware infects the target node after traveling on the correct path, then there will only be one non-default key remaining encryption the ciphertext.

Padding For a ciphertext c, encrypting a message m, the padding algorithm $\mathcal{P}(c)$ outputs a ciphertext c', encrypting the same message m, with a defined length.

See Algorithm 5 for the malware algorithm, which is specific for the extended path variation. Note that the decryption algorithm will be correct only if the malware attack process is performed as showed in the algorithm. Thus we need a specific correctness requirement for the path version of the extended scheme. We also need a requirement for the padding algorithm, since it replaces the rerandomization algorithm.

Correctness If c_i' was output from $\mathcal{P}(\mathcal{E}(k_i, c_{i-1}))$ or $\mathcal{P}(\mathcal{D}(k_{i+1}, c_{i+1}))$ then $\mathcal{D}(k_i, c_i')$ will always output c_{i-1} except with negligible probability.

Padding If c was output by $\mathcal{E}(k, m)$ then the output distribution of $\mathcal{P}(c)$ should be computationally indistinguishable from the output distribution of $\mathcal{E}(k, m)$.

The algorithms of the extended path scheme are as follows.

Encryption For a message $m \in \{0,1\}^L$, keys $k_1, k_2, \ldots, k_t \in \mathbb{Z}_p^*$, $r_i \xleftarrow{r} \mathbb{Z}_p^*$, $s_i \xleftarrow{r} \mathbb{Z}_p$, and $\gamma_i \xleftarrow{r} G$, for $i = 1, \ldots, t$. Encrypt the message in layers where

$$c_1 = g^{r_1} || g^{r_1 k_1} || g^{s_1} || g^{s_1 k_1} \gamma_1 || \left(f_L(\gamma_1) \oplus m\right)$$

and

$$c_i = g^{r_i} || g^{r_i k_i} || g^{s_i} || g^{s_i k_i} \gamma_i || \left(f_{L+2(i-1)l}(\gamma_i) \oplus c_{i-1}\right)$$

for $i \in \{2, \ldots, t\}$. Pad c_t such that it has default length L_D.

Padding For a ciphertext c. If $|\, c \,| = L_D$ let $c = x||y||b_m||b_l$, where b_l is the last l bits, otherwise let $c = x||y||b_m$.

$$\mathcal{P}(c) = x^{r'} || y^{r'} || x^{s'} || y^{s'} \gamma' || \left(f_{|b_m|+N+1}(\gamma') \oplus (b_m||1||0^N\right).$$

where $N = L_D - |b_m| - 2l - 1$.[2]

[2] N is a multiple of l.

Decryption For a ciphertext $c = x_0||y_0||b'_0$ check if any of the non-default keys $\hat{k}$ satisfies $x_0^{\hat{k}} = y_0$, if so let $k = \hat{k}$ if not let k be the default key. Let $b'_0 = z_0||w_0||b_0$ and compute

$$b'_1 = f_{|b_0|}(z_0^{-k} w_0) \oplus b_0,$$

interpret b'_1 as $z_1||w_1||b_1||1||0^{N'}$, where $N' \in \mathbb{N}$.[3] Compute

$$b'_2 = f_{|b_1|}(z_1^{-k} w_1) \oplus b_1$$

and check if b'_2 is an executable malware. If it is then the attack was successful, if not pad b'_2.

2.4.3 Security of the path scheme

Based on the basic scheme The proof is as the basic proof, given in Section 2.2.1, except that:

- the probability space D is replaced with $D_1 \times \cdots \times D_t$,
- use vectors of samples $\mathbf{u}_1$ and $\mathbf{u}_2$ instead of u_1 and u_2, and
- use vectors of keys $\mathbf{k}_1$ and $\mathbf{k}_2$ instead of k_1 and k_2.

The opponent's advantage is bounded by

$$\mathrm{Adv}(\mathcal{O}) \leq \Pr[F_1] + \frac{1}{p} + \mathrm{Adv}_{\mathrm{ddh}}(\mathcal{A}).$$

That is, the advantage is bounded by the opponent's ability to guess or predict the correct keys and is determined by the probability space $D_1 \times \cdots \times D_t$.

Based on the extended scheme The proof is as the extended proof, given in Section 2.3.1, except that:

- replace the rerandomize algorithm with the padding algorithm in Experiment 1,

[3] N' will be a multiple of l zeros if the correct key is used in the decryption algorithm.

- the probability space D is replaced with $D_1 \times \cdots \times D_t$,
- use vectors of samples $\mathbf{u}_1$ and $\mathbf{u}_2$ instead of u_1 and u_2,
- use vectors of keys $\mathbf{k}_1$ and $\mathbf{k}_2$ instead of k_1 and k_2,
- precompute $2t$ tuples of the form

$$\{(x_j, y_j, z_j, w_j)\}_{j=1}^{t} = \left\{\left(g, g^{k_{1,j}}, g^{s_j}, g^{s_j k_{1,j}}\right)\right\}_{j=1}^{t}$$

and

$$\left\{\left(x'_j, y'_j, z'_j, w'_j\right)\right\}_{j=1}^{t} = \left\{\left(g, g^{k_{2,j}}, g^{s'_j}, g^{s'_j k_{2,j}}\right)\right\}_{j=1}^{t}$$

instead of precomputing two tuples, and

- from the first tuple, (x, y, z, w), create an additional $2t-1$ tuples, instead of one, by uniformly sample $\{a_j, b_j, c_j\}_{j=1}^{2t-1}$ and compute

$$\left\{\left(x, x^{a_j} y^{c_j}, z x^{b_j}, w^{c_j} z^{a_j} y^{c_j b_j} x^{a_j b_j}\right)\right\}_{j=1}^{2t-1}.$$

The opponent's advantage is bounded by

$$\mathrm{Adv}(\mathcal{O}) \leq \Pr[F_1] + \mathrm{Adv}_{\mathrm{ddh}}(\mathcal{A}) + \mathrm{Adv}_{\mathrm{prf}}(\mathcal{B}).$$

That is, the advantage is bounded by the opponent's ability to guess or predict the correct keys and is determined by the probability space $D_1 \times \cdots \times D_t$.

Acknowledgments

We would like to thank Adam Young for valuable discussions and we would like to thank the anonymous reviewers for helpful comments.

References

[1] Dan Boneh. *Algorithmic Number Theory: Third International Symposiun, ANTS-III Portland, Oregon, USA, June 21–25, 1998 Proceedings*, chapter The Decision Diffie-Hellman problem, pages 48–63. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[2] Ran Canetti, Hugo Krawczyk, and Jesper B. Nielsen. Relaxing chosen-ciphertext security. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, pages 565–582, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[3] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[4] Eric Filiol. Strong Cryptography Armoured Computer Viruses Forbidding Code Analysis: the bradley virus. Research Report RR-5250, INRIA, 2004.

[5] Eric Filiol. Malicious cryptography techniques for unreversable (malicious or not) binaries. *CoRR*, abs/1009.4000, 2010.

[6] Ariel Futoransky, Emiliano Kargieman, Carlos Sarraute, and Ariel Waissbein. Foundations and applications for secure triggers. Cryptology ePrint Archive, Report 2005/284, 2005. http://eprint.iacr.org/.

[7] Herman Galteland and Kristian Gjøsteen. *Malware, Encryption, and Rerandomization – Everything Is Under Attack*, pages 233–251. Springer International Publishing, Cham, 2017.

[8] Philippe Golle, Markus Jakobsson, Ari Juels, and Paul Syverson. *Universal Re-encryption for Mixnets*, pages 163–178. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[9] Fritz Hohl. Time limited blackbox security: Protecting mobile agents from malicious hosts, 1998.

[10] Kaspersky Lab Global Research and Analysis Team. Gauss: Abnormal distribution. In-depth research analysis report, KasperSky Lab, August 9th 2012. securelist.com/en/analysis/204792238/gauss_abnormal_distribution.

[11] James Riordan and Bruce Schneier. Environmental key generation towards clueless agents. In Giovanni Vigna, editor, *Mobile Agents and Security*, volume 1419 of *Lecture Notes in Computer Science*, pages 15–24. Springer Berlin Heidelberg, 1998.

[12] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004.

[13] Ed Skoudis and Lenny Zeltser. *Malware: Fighting Malicious Code*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2003.

[14] Adam Young and Moti Yung. Cryptovirology: extortion-based security threats and countermeasures. In *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, pages 129–140, May 1996.

[15] Adam Young and Moti Yung. *Malicious Cryptography: Exposing Cryptovirology*. John Wiley & Sons, 2004.

[16] Adam Young and Moti Yung. The drunk motorcyclist protocol for anonymous communication. In *Communications and Network Security (CNS), 2014 IEEE Conference on*, pages 157–165, Oct 2014.

Paper II

Subliminal channels in post-quantum digital signature schemes

Herman Galteland and Kristian Gjøsteen

ePrint 2019/574

Subliminal channels in post-quantum digital signature schemes

Herman Galteland* and Kristian Gjøsteen

Department of Mathematical Sciences,
NTNU – Norwegian University of Science and Technology
{herman.galteland, kristian.gjosteen}@ntnu.no

Abstract

We analyze digital signatures schemes submitted to NIST's Post-Quantum Cryptography Standardization Project in search for subliminal channels.

Keywords. Subliminal channels, post-quantum, digital signatures.

1 Introduction

The National Institute of Standards and Technology's (NIST) current *Post-Quantum Cryptography Standardization Project* (PQCSP) [19] is analyzing and testing post quantum secure public key encryption, key encapsulation mechanism, and digital signature schemes submitted by the cryptographic research community. NIST's goal is to have a selection of standardized schemes for future use.

In this paper we analyze the proposed digital signature schemes submitted to PQCSP for *subliminal channels*, covert communication channels that uses existing cryptographic protocols to send information, called *subliminal messages*, the protocols were not intended for.

*This work is funded by Nasjonal sikkerhetsmyndighet (NSM), www.nsm.stat.no

Knowledge of subliminal channels is important for those who want to use them and for those who want avoid them. Privacy concerned users could use subliminal channels to bypass censorship or avoid surveillance. When constructing secure systems developers might avoid schemes with subliminal channels to prevent, for example, information leakage or malicious actors secretly breaching their system. It is likely that some of the digital signatures schemes submitted to PQCSP will be used. It is therefore useful to know about any subliminal channels present in these schemes.

We use the following model when we look for subliminal channels, where we try to fulfill the goals of the sender and receiver of the subliminal messages.

Subliminal channel model The *subliminal sender* wants to communicate discreetly with a *subliminal receiver* over a channel controlled by an adversary. We shall assume that the subliminal sender is sending message-signature pairs that the subliminal receiver later can observe.

Hiding information directly in the message is steganography, which is not our current topic of interest. Therefore we shall assume that the message part of the message-signature pairs will not be under the subliminal senders' control. Instead, the subliminal sender generates the signatures and will try to embed subliminal messages in the signatures.

We assume that the subliminal sender and the subliminal receiver share a key for a suitable symmetric cryptosystem (that may even be stateful). In particular, this means that the subliminal message to be embedded will be indistinguishable from random bits.

We may allow the subliminal receiver to know the signing key, but it is better if the receiver does not need the signing key. The subliminal sender may also be the one generating the key, and in this case we may allow cheating during key generation.

The subliminal sender's goal is to send as much information as possible without detection. That is, except for the subliminal receiver, anyone who inspects the generated signatures should not be able to decide if they contain subliminal messages or were generated honestly.

This should hold even if the one who inspects the signatures also chooses the messages to be signed.

We note that even when the subliminal channel is used, the signature scheme should remain secure in the ordinary sense. (It may seem like this follows from indistinguishability, since if the scheme is insecure when the subliminal channel is used, a distinguisher would try to break the scheme and distinguish in that way. However, since checking for subliminal channels is something that we want to do regularly, any subliminal channel distinguisher must be fast. Which means that it may not have time to run an attack against the signature scheme.)

1.1 Examples of subliminal channels

One would expect a deterministic signature scheme to be free from subliminal channels. Signatures of the RSA-FDH scheme [7] have the form $\text{Sign}_{RSA-FDH} = (H_{FDH}(M))^d \mod n$, for an RSA decryption key d and modulus n. There are no random value to exploit, however, it is possible to use it to send (short) subliminal messages. Using the halting strategy [22] we can in some sense make a 1 bit subliminal channel. However, this is a generic attack.

Any sufficiently non-deterministic digital signature scheme allows a generic subliminal channel. For example, to send n bits, generate signatures until the first n bits of the signature matches the message. We can generalize the approach using the leftover hash lemma [35], where we generate signatures until the hash of the signature matches the message to be sent. This holds for any signature scheme. Of course, the computational cost of this approach is exponential in n, so this will be a low-bandwidth channel.

It is possible to *derandomize* signature schemes by replacing the random bits with the output of a pseudo-random function applied to the message. In general this will not prevent subliminal channels [12]. In most construction, it is hard to verify that the signature generation is deterministic without knowledge of the secret key. (Another option is to sign the same message twice. We assume the subliminal sender keeps track of messages sent.) It is, however, possible to check if there was a subliminal channel present assuming that the secret value becomes public at a later point.

The PSS message encoding scheme, used with the RSA or Rabin primitive to make a signature scheme [7], has a subliminal channel that only requires *public* values to recover the message. Taking RSA as an example, we have $\text{Sign}_{PSS-RSA} = (0||w||r^*||M)^d \mod n$, for a RSA private key d, modulus n, hash value w, random value r^*, and message M. Using the RSA public key e we can recover the randomness r^*. Replacing the random value r^* with a subliminal message m_s we can make a subliminal channel. The Rabin primitive works in the same way.

A subliminal channel using Schnorr Signatures [49] requires the *secret* values to recover the message, which has to be shared with the subliminal receiver. A Schnorr signature has the form $\text{Sign}_{Schnorr} = (y, e) = (r + se, e)$, where s is the secret key, e is a hash value, and r is a random value. We can use s to recover r by computing $r = y - se$. It is then trivial to replace the randomness with a subliminal message m_s to make a subliminal channel.

1.2 Our contributions

We modify the proposed digital signature schemes submitted to the PQCSP to reliably send subliminal messages, typically by exploiting any random values, while keeping the signatures valid. We show how to insert and recover subliminal messages and give an overview of the subliminal bandwidth of each channel found.

1.3 Related work

Simmons motivated his work on the subliminal channel with the prisoners' problem [50], where two prisoners wish to plan their escape and the warden allows them to send signed message if he can read the content of the messages. The problem for the prisoners is to communicate covertly using their monitored communication channel, to plan their escape. The problem of the warden is to discover and prevent the existence of subliminal channels, to prevent prisoners escaping. Using digital signatures Simmons showed that such channels exist [51–53], and since then more have been found in various digital signature schemes [4, 12, 38, 57]. Of subliminal channels in post-

quantum secure digital signature schemes: Hartl et al. showed how to insert subliminal messages in signature schemes based on the multivariate quadratic polynomial problem [31]. Kwant et al. constructed a backdoor (Kleptography [56]) in the NTRU and the pqNTRUsign digital signature scheme, which can be used as a subliminal channel, and a subliminal channel in the NTRU scheme [37]. The subliminal channel of the pqNTRUsign scheme can send a few bits per signature.

Desmedt made the first attempt of making a subliminal-free authentication and signature scheme [21], and since then more schemes have been constructed to prevent subliminal channels [11, 12, 25, 53, 54]. Divertible protocols [10, 14, 15, 43] are separable by an unnoticeable third party (a warden) sitting in between two communicating principals (the prisoners), where the third party can rerandomize messages to remove any subliminal channel. Cryptographic reverse firewalls [17, 42] can, in theory, be used to prevent subliminal channels and existing constructions are based on Decisional Diffie–Hellman and use rerandomization techniques to remove subliminal messages, which are not suitable for post-quantum secure schemes.

1.4 Notation

We try to follow the notation given in each signature scheme as much as we can, where we describe each scheme's notation when they are introduced. We denote the subliminal message in plain m_s or in bold $\mathbf{m}_s$ to fit with the notation of each scheme. Let $\operatorname{sgn}(x)$ denote the sign of x and is equal to 1 if $x \geq 0$ or equal to -1 if $x < 0$. When the distribution of random elements are not specified, other than that they are random, we assume it is uniform.

1.5 Overview

In Section 2 we briefly describe each scheme submitted to NIST's Post-Quantum Cryptography Standardization Project and show the subliminal channel(s) we have found. In Section 3 we summarize and give a table of each channel's subliminal bandwidth.

2 Proposed digital signature schemes

2.1 Ideas

While we analyze many different schemes, many ideas are reused for many different schemes and we discuss some of the theory here.

In addition to the *public* and *secret* channels described in Section 1.1 we also note the following subliminal channels. Some schemes use random values that are *included in the clear*, which can be used to embed encrypted subliminal messages. A few lattice based schemes use a large random value to hide a small secret value, where we can insert information in the *higher order digits* of the random value.

Some signature schemes require the signatures to have a specific form. By using probabilistic encryption we can simply encrypt the subliminal messages many times until we get a signature of the desired form. This will reduce the available bandwidth, but the cost is usually just a few bits, especially if stateful encryption is used.

The random elements used in the signature scheme are sampled according to some distribution. For non-uniform distributions we can sometimes do rejection sampling on the randomness used to encrypt the subliminal message such that the output is close to the desired distribution.

2.2 CRYSTALS – Dilithium

The Dilithium digital signature scheme is based on Fiat-Shamir with Aborts [39] where the security is based on the shortest vector problem. The scheme uses the polynomial rings $\mathcal{R} = \mathbb{Z}[X]/\langle X^n + 1\rangle$ and $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$, for an integer n. An element $x \in \mathcal{R}$, and $x \in \mathcal{R}_q$, is denoted in plain, and vectors and matrices are denoted in bold. An element in $B_{60} \subset \mathcal{R}$ has 60 coefficients that are either -1 or 1 and the rest are 0.

Signatures have the form

$$\mathrm{Sign}_{Dilithium} = (\mathbf{z}, \mathbf{h}, c),$$

where $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$, $\mathbf{y} \in \mathcal{R}_q^l$ is uniformly random, $\mathbf{s}_1 \in \mathcal{R}_q^l$ is part of the secret key, $\mathbf{h}$ is a Boolean vector used to recover high-order bits,

Table 1: CRYSTALS – Dilithium parameters and subliminal message bounds for high order digits subliminal channel.

Parameter	Description	Parameter set I	II	III	IV
n	degree	256	256	256	256
l	dimension	2	3	4	5
γ_1	randomness bound	523776	523776	523776	523776
β	bound	375	325	275	175
$\|\mathbf{m}_s\|_\infty$	message bound	523	523	523	523
$\|r'_i\|$	masking value	$[375, 624]$	$[325, 674]$	$[275, 724]$	$[175, 824]$

and $c \in B_{60}$ is the hash of the message digest and the higher-order bits of a public matrix multiplied by the vector $\mathbf{y}$. The vector $\mathbf{z}$ has max norm $\|\mathbf{z}\|_\infty \leq \gamma_1 - \beta - 1$.

2.2.1 Subliminal channel using secret values

Assume that the sender and receiver both know the secret key $sk = (\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$. The sender can replace the random value $\mathbf{y}$ with a subliminal message $\mathbf{m}_s$, that is, let $\text{Sign}_{Dilithium} = (\mathbf{z}, \mathbf{h}, c)$, where $\mathbf{z} = \mathbf{m}_s + c\mathbf{s}_1$. With the secret key the receiver can retrieve the subliminal message as $\mathbf{m}_s = \mathbf{z} - c\mathbf{s}_1$. We can encrypt the subliminal messages again, using fresh randomness, if the produced signature is not a valid signature.

2.2.2 Subliminal channel using high order digits

The scheme requires that $\|c\mathbf{s}_1\|_\infty \leq \beta$ and $\|\mathbf{z}\|_\infty \leq \gamma_1 - \beta - 1$. We can insert a subliminal message in the high order digits of $\mathbf{y}$ if β is small compared to the coefficients of $\mathbf{y}$.

Proof of concept For the proposed parameters, the coefficients of the random value have at most six digits and the coefficients of the error term $\|c\mathbf{s}_1\|_\infty$ have at most three digits. By bounding the randomness $\|\mathbf{y}\|_\infty \leq \gamma_1 - 2\beta - 1$ we should be able to send subliminal messages with three digit coefficients. Let $\mathbf{y} = 1000\mathbf{m}_s + \mathbf{r}'$ and

$$\mathbf{z}' = 1000\mathbf{m}_s + \mathbf{r}' + c\mathbf{s}_1,$$

where $\mathbf{m}_s \in \mathcal{R}_q^l$ and $\mathbf{r}' \in \mathcal{R}_q^l$. The bounds on the coefficients of $\mathbf{m}_s$ and the coefficient interval of the masking randomness $\mathbf{r}'$ are in Table 1. Note that each coefficient of $\mathbf{r}'$ need to have the same sign as the subliminal message, $\operatorname{sgn}(r'_i) = \operatorname{sgn}(m_{s_i})$ for all $i \in \{1, \ldots, n\}$, for the message recovery to be correct. The subliminal message $\mathbf{m}_s$ is bounded such that $\|\mathbf{z}'\|_\infty \leq \gamma_1 - \beta - 1$, and the coefficients of the masking value $\mathbf{r}'$ are chosen such that $0 \leq \|\mathbf{r}' + c\mathbf{s}_1\|_\infty \leq 999$. The receiver computes

$$\left\lfloor \frac{\mathbf{z}'}{1000} \right\rfloor$$

to recover the subliminal message $\mathbf{m}_s$. When we encrypt the subliminal message we can encrypt it one more time if the produced signature does not meet the requirements of the signature algorithm. We should also pick the masking values $\mathbf{r}'$ such that $\|\mathbf{z}'\|_\infty \leq \gamma_1 - \beta - 1$.

2.3 DME

The DME signature scheme is based on multivariate polynomials. The scheme uses the field $\mathbb{F}_p$ where vectors are denoted in plain lowercase.

Signatures have the form

$$\operatorname{Sign}_{DME} = (x, z_0, h_1),$$

where $z_0 \in \mathbb{F}_p^{N_1}$ is a message, $h_1 : \{1, \ldots, N_1\} \to \{1, \ldots, e \cdot n \cdot m\}$ is a map used to add a random padding to the messages, and $x = F^{-1}(z_0) \in \mathbb{F}_p^{N_1}$, where F is the public multivariate polynomial. If z_0 is not in the image of F, pad z_0, using h_1, to get $z \in \operatorname{Im}(F)$ and set $x = F^{-1}(z)$. The values N_1, e, n, and m are parameters. A verifier computes $z = F(x)$, discards the padding using h_1 to get z'_0, and verifies that $z'_0 = z_0$.

2.3.1 Subliminal channel using public values

Anyone with the public key can recover $z = F(x)$ and, specifically, the random padding which can be replaced with a subliminal message. The scheme offers two parameter sets where up to 16 and 32 bits are

added to messages, respectively. If the padded message z is not in the image of F, for a chosen padding, we can encrypt the subliminal message again, using fresh randomness, until we find a z which is.

2.4 DRS

The DRS signature scheme is a variation of the scheme by Plantard et al. [47], where both schemes are based on GGH [30]. The security of the scheme is based on the guaranteed distance decoding problem. The scheme uses a diagonal dominant lattice $\mathcal{L}(P)$, with public basis P that has large coefficients. An element $x \in \mathcal{L}(P)$ is an integer vector.

Signatures have the form

$$\text{Sign}_{DRS} = (k, v, w),$$

where k satisfies $kP = v - w$, $v \in \mathbb{Z}^n$ is the hash of the message, and $w \in \mathbb{Z}^n$ is a reduced vector of v that satisfies $w \equiv v \mod \mathcal{L}(S)$ and $\|w\|_\infty < D$. The public basis P has big coefficients and is constructed from the secret basis S using unimodular matrices U, and D is a max norm bound.

2.4.1 No channel found

We were unable to find a subliminal channel in the DRS signature scheme.

2.5 DualModeMS

The DualModeMS scheme has two layers, an inner and an outer. The inner layer is a Matsumoto-Imai multivariate construction [40] based on FHEv, and the outer layer modifies the output of the inner layer by using the method of Szepieniec, Beullens, and Preneel [55]. The scheme uses the fields $\mathbb{F}_2$ and $\mathbb{F}_2^n$, where vectors are denoted in bold lower case, matrices are denoted in bold uppercase and polynomials are denoted in capital plain letters.

The output of the *inner* algorithm is

$$\mathbf{s} = (\phi(Z), \mathbf{v}) \times \mathbf{S}^{-1},$$

where $\phi(Z)$ is a root represented as a binary vector, $\mathbf{S}$ is an invertible $n+\nu \times n+\nu$ matrix, and $\mathbf{v} \in \mathbb{F}_2^\nu$ are the random vinegar variables. A signature of the DualModeMS scheme has the form

$$\mathrm{Sign}_{DualModeMS} = (\mathbf{s}_1, \ldots, \mathbf{s}_\sigma, \mathbf{h}, \texttt{openpaths}),$$

where the $\mathbf{s}_i$'s are output of the inner algorithm, $\mathbf{h}$ is a random set of linear combination of the public key, and `openpaths` is a set of Merkle tree paths used to compute the public key.

2.5.1 Subliminal channel using public values

For a chosen vector $\mathbf{s}'$, that contains a subliminal message in the last ν bits, we can find a vector $\mathbf{v} = \mathbf{s}' \times \mathbf{S}$ such that $\mathbf{s} = (\phi(Z), \mathbf{v}) \times \mathbf{S}^{-1}$ contains an encrypted subliminal message in the last ν bits. If $\mathbf{s}'$ is not in the image of $\mathbf{S}^{-1}$ we can encrypt the subliminal message again, using fresh randomness.

The subliminal message $\mathbf{m}_i \in \mathbb{F}_2^\nu$ is inserted in the last ν bits of $\mathbf{s}_i \in \mathbb{F}_2^{n+\nu}$, for $i = 1, \ldots, \sigma$. The scheme provides three parameter sets. The dimension ν is 11, 18, and 32, respectively. The number of inner signatures σ is 64, 96, and 256, respectively. If, for a chosen subliminal message, the inner algorithm does not terminate we can encrypt the subliminal message again, using fresh random values, and try again.

2.6 Falcon

The Falcon signature scheme is a combination of the GPV framework [27], over the NTRU lattice, with fast Fourier sampling. The scheme uses the polynomial ring $\mathbb{Z}[x]/\langle\phi\rangle$, where $\phi \in \mathbb{Z}[x]$ is a monic and irreducible cyclotomic polynomial of degree n. Vectors are denoted in lowercase bold and matrices in uppercase bold.

Signatures have the form

$$\mathrm{Sign}_{Falcon} = (\mathbf{r}, \mathbf{s}),$$

where $\mathbf{r} \in \{0,1\}^{320}$ is a uniformly sampled salt, and is used in a hash together with the message to get a point $c \in \mathbb{Z}_q[x]/\langle\phi\rangle$. A preimage

$\mathbf{t} = c\mathbf{B}^{-1}$ is computed, for a secret basis $\mathbf{B}$, and is used to find two short polynomials $s_1, s_2 \in \mathbb{Z}_q[x]/\langle\phi\rangle$ such that $s_1 + s_2 h \equiv c \mod q$, for a NTRU public key $h = gf^{-1} \mod q$. The polynomial s_2 is encoded as the bit string $\mathbf{s}$.

2.6.1 Random values included in the clear

The random value $\mathbf{r} \in \{0,1\}^{320}$ can be replaced by a subliminal message $\mathbf{m}_s \in \{0,1\}^{320}$.

2.7 G*e*MSS

The G*e*MSS scheme is based on the digital signature scheme called `Quartz` [45], modified using the ideas of the `Gui` digital signature scheme [24]. The scheme uses the fields $\mathbb{F}_2$ and $\mathbb{F}_2^n$. Vectors are denoted in bold lowercase, matrices in bold uppercase and polynomials in capital plain letters.

The signature algorithm of the G*e*MSS scheme uses a subroutine called G*e*MSS inversion, which output the $n + \nu$ bit vector

$$\mathbf{s} = (\phi(Z), \mathbf{v}) \times \mathbf{S}^{-1},$$

where $\phi(Z)$ is a root represented as a binary coefficient vector, $\mathbf{S}$ is a secret invertible $n+\nu \times n+\nu$ matrix, and $\mathbf{v} \in \mathbb{F}_2^{\nu}$ are random vinegar variables. A signature of G*e*MSS has the form

$$\text{Sign}_{GeMSS} = (\mathbf{S}_{nb\_ite}, \mathbf{X}_{nb\_ite}, \ldots, \mathbf{X}_1),$$

where $(\mathbf{S}_i, \mathbf{X}_i)$ is the output $\mathbf{s}_i$ of the inversion algorithm discussed above. The first component $\mathbf{S}_i \in \mathbb{F}_2^m$ is the first m bits of the output $\mathbf{s}_i$ and the second component $\mathbf{X}_i \in \mathbb{F}_2^{n+\nu-m}$ is the remaining $n+\nu-m$ bits. The last ν bits of $\mathbf{s}_i$ is contained in the second component $\mathbf{X}_i$.

2.7.1 Subliminal channel using public values

For a chosen vector $\mathbf{s}'$, that contains a subliminal message in the last ν bits, we can find a vector $\mathbf{v} = \mathbf{s}' \times \mathbf{S}$ such that $\mathbf{s} = (\phi(Z), \mathbf{v}) \times \mathbf{S}^{-1}$ contains an encrypted subliminal message in the last ν bits. If $\mathbf{s}'$ is

not in the image of $\mathbf{S}^{-1}$ we can encrypt the subliminal message again, using fresh randomness.

The subliminal message $\mathbf{m}_i \in \mathbb{F}_2^{\nu}$ is inserted in the last ν bits of $\mathbf{s}_i \in \mathbb{F}_2^{n+\nu}$, for $i = 1, \ldots, nb\_ite$. The scheme provides three parameter sets. The dimension ν is 12, 20, and 33, respectively, and the number nb_ite is 4 for all sets. If the inversion algorithm does not terminate, for a chosen subliminal message, we can encrypt the message again, using fresh randomness, and try again.

2.8 Gravity – SPHINCS

Gravity–SPHINCS is an extension of Goldreich's construction of a stateless hash based signature scheme [29], and share many similarities with SPHINCS [8]. The scheme outputs bit strings, where a binary number $v \in \{0,1\}^n$ is denoted in plain. The scheme uses four types of trees; hyper tree, subtrees (Merkle trees [41]), WOTS public key compression trees, and PORST public key compression trees. The signature is composed of a PORST signature (improved version of HORST few times signatures [8]), Winternitz one time signatures [34], and Merkle authentication paths.

Signatures have the form

$$\mathrm{Sign}_{Gravity-SPHINCS} = (s, \sigma_d, oct, \sigma_{d-1}, A_{d-1}, \ldots, \sigma_0, A_0, A_c),$$

where s is a hash and a public salt, σ_d is a PORST signature, oct is an authentication value for the PORST signatures, $\sigma_{d-1}, \ldots, \sigma_0$ are Winternitz signatures, and $A_{d-1}, \ldots, A_0, A_c$ are Merkle authentication paths.

2.8.1 Random values included in the clear

The public salt s is constructed using a hash function with a secret salt and the message as input, where the verifier cannot verify that the random value s was computed using this hash function without the secret salt. The sender can replace the random value s with a subliminal message m_s.

The PORST signature $\sigma_d = (s_{x_1}, s_{x_2}, \ldots, s_{x_k})$, where the x_i's are indices and the s_i are generated using a pseudorandom function G

with a secret seed and an address in the hyper tree as input. We can replace the random values with a subliminal message. The security of the scheme will be reduced as we can now make a collision in the PORST signatures with probability 2^{-128}, which is acceptable.

The submission proposes tree parameter sets. The binary string of length n is 256 for all parameter sets, and the dimension k is 24, 32, and 28, respectively.

2.9 Gui

The Gui scheme is based on the Hidden Field Equations cryptosystem using the minus and vinegar modification (`HFEv-`) [45]. The scheme uses a finite field $\mathbb{F}_q$ and the field extension $\mathbb{F}_{q^n}$. Field elements are denoted in lowercase plain. Vectors are denoted in lowercase bold or in uppercase plain. The affine transformations and maps are denoted in uppercase plain and their inverses are denoted with the prefix Inv.

The signature algorithm of the Gui scheme uses a subroutine called HFEv- inversion that outputs

$$\mathbf{z} = InvT \cdot ((\mathbf{y}||v_1|| \dots ||v_\nu) - c_T),$$

where $invT$ is the inverse of the affine transformation $\mathcal{T}$, $\mathbf{y}$ is a root of a polynomial Y, $v_1, \dots, v_\nu$ are random vinegar elements, and c_T is a random masking value. The output of the signature scheme is

$$\text{Sign}_{Gui} = (S_k||X_k|| \dots ||X_1||r),$$

where r is a random bit string and (S_i, X_i) is the output of the HFEv- inversion algorithm. S_i is the first $n - a$ elements and X_i is the last $a + \nu$ elements, where the last ν elements are the random vinegar values.

2.9.1 Random values included in the clear

The random value $r \in \{0,1\}^{\bar{l}}$ can be replaced by a subliminal message $m_s \in \{0,1\}^{\bar{l}}$, where $\bar{l} = 128$ for all parameter settings.

2.9.2 Subliminal channel using secret values

Using the verification algorithm we can recover all S_i's, from the first and the X_i's. Then we can recover the random values $v_1, \ldots, v_\nu$, in each (S_i, X_i), using the secret key, which contains $\mathcal{T}$ and c_T.

In each X_j, for $j = 1, \ldots, k$, the subliminal message $m_{s_i} \in \mathbb{F}_q$ replaces the random values v_i, for $i = 1, \ldots, \nu$. The scheme offers three parameter sets. The number of random elements ν is 16, 20 and 28, respectively. The number k is 2 for all sets and the group size q is 2 for all sets. If the signing algorithm does not terminate, for a chosen subliminal message, we can encrypt the subliminal message again, using fresh randomness.

2.10 HiMQ – 3

The HiMQ – 3 is based on multivariate quadratic equations. The scheme uses a finite field $\mathbb{F}_q$. The affine transformations and maps are denoted in uppercase plain and vectors in lowercase bold.

Signatures have the form

$$\mathrm{Sign}_{HiMQ-3} = T^{-1}(\mathbf{s}),$$

where $T^{-1} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ is an invertible affine or linear map, depending on the parameter set, and $\mathbf{s} = (s_1, \ldots, s_\nu, s_{\nu+1}, \ldots, s_n)$ is a solution of the central map $\mathcal{F}$, where the first ν elements of $\mathbf{s}$ are random.

2.10.1 Subliminal channel using secret values

The random elements of $\mathbf{s}$ can be replaced with a subliminal message. The message can be recovered using the secret map T.

The subliminal message $m_{s_i} \in \mathbb{F}_q$ replaces the first ν random values s_i, in $\mathbf{s}$, for $i = 1, \ldots, \nu$. The authors offer three variation the scheme, HiMQ – 3, HiMQ – 3F and HiMQ – 3P, where each variation has one parameter set. The number of group elements q is 2^8, for all variations. The number of random values ν is 31, 24, and 31, respectively. The variations HiMQ – 3F and HiMQ – 3P has a different central map, but the output of the signature algorithms is the same as HiMQ – 3. If the signing algorithm does not terminate we can encrypt the chosen subliminal message again, using fresh randomness.

2.11 LUOV

The LUOV, Lifted Unbalanced Oil and Vinegar, scheme is based on the Unbalanced Oil and Vinegar and is a modification of the Oil and Vinegar scheme by Patarin [44]. The scheme uses the finite field $\mathbb{F}_2$ and the extension $\mathbb{F}_{2^r}$. Vectors are denoted in lowercase bold and matrices in uppercase bold.

Signatures have the form

$$\text{Sign}_{LUOV} = \begin{pmatrix} \mathbf{1}_\nu & -\mathbf{T} \\ \mathbf{0} & \mathbf{1}_m \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{o} \end{pmatrix},$$

where $\mathbf{1}_\nu$ and $\mathbf{1}_m$ are identity matrices, $-\mathbf{T}$ is a $\nu \times m$ binary matrix, $\mathbf{v}$ is a vector of random elements, and $\mathbf{o}$ is a unique solution of the central map $\mathcal{F}$.

2.11.1 Subliminal channel using secret values

Let $\mathbf{A} = \begin{pmatrix} \mathbf{1}_\nu & -\mathbf{T} \\ \mathbf{0} & \mathbf{1}_m \end{pmatrix}$, then $\mathbf{A}$ is an upper triangular binary $(\nu+m) \times (\nu+m)$ matrix and is invertible. Using $\mathbf{A}$ we can recover the vector $\mathbf{v}$ by

$$\mathbf{A}^{-1}\text{Sign}_{LUOV} = \begin{pmatrix} \mathbf{v} \\ \mathbf{o} \end{pmatrix}.$$

The vector $\mathbf{v}$ is generated by squeezing a vinegar-sponge, where the sponge is generated using the message and a private seed. Only the sender, which has the private seed, can verify that the vinegar-sponge, and the random values $\mathbf{v}$, is computed according to the protocol. We can replace the random vector $\mathbf{v} \in \mathbb{F}_{2^r}^\nu$ with a subliminal message $\mathbf{m}_s \in \mathbb{F}_{2^r}^\nu$. The scheme offers six parameter sets. The number of random elements ν is 256, 351, 404, 242, 330, and 399, respectively. The field extension r is 8, 8, 8, 48, 64, and 80, respectively. We can encrypt the subliminal message again, using fresh randomness, if the signing algorithm rejects the produced signature.

2.12 MQDSS

The MQDSS signature scheme is built from the SSH 5-pass identification scheme [48] using the Fiat-Shamir transform. The scheme

uses the finite field $\mathbb{F}_q$ for a prime, or a prime power, q. Vectors are denoted in lowercase bold and matrices in uppercase bold.

Signatures have the form

$$\text{Sign}_{MQDSS} = (R, \sigma_0, \sigma_1, \sigma_2),$$

where R is a random value, σ_0 is a digest of the commitments of $(\mathbf{r}_0^{(j)}, \mathbf{t}_0^{(j)}, \mathbf{e}_0^{(j)})$ and of $(\mathbf{r}_1^{(j)}, \mathbf{F}(\mathbf{t}_0^{(j)} + \mathbf{r}_1^{(j)}) - \mathbf{F}(\mathbf{t}_0^{(j)}) - \mathbf{F}(\mathbf{r}_1^{(j)}) + \mathbf{e}_0^{(j)})$, σ_1 contains r responses of the form $resp_1^{(j)} = (\alpha^{(j)}\mathbf{r}_0^{(j)} - \mathbf{t}_0^{(j)}, \alpha^{(j)}\mathbf{F}(\mathbf{r}_0^{(j)}) - \mathbf{e}_0^{(j)})$, and σ_2 contains r responses of the form $resp_2^{(j)} = \mathbf{r}_{b^{(j)}}^{(j)}$, for $b^{(j)} \in \{0, 1\}$, and r of the commitments in σ_0. The random elements $\mathbf{r}_0^{(1)}, \ldots, \mathbf{r}_0^{(r)}, \mathbf{t}_0^{(1)}, \ldots, \mathbf{t}_0^{(r)}, \mathbf{e}_0^{(1)}, \ldots, \mathbf{e}_0^{(r)}$ are sampled from $\mathbb{F}_q$, generated using a pseudo random generator $PRG_{\mathbf{rte}}$. The values $\mathbf{r}_1^{(j)} = \mathbf{s} - \mathbf{r}_0^{(j)}$, for $j \in \{1, \ldots, r\}$. The first challenge $ch_1 = (\alpha^{(0)}, \ldots, \alpha^{(r)})$ is computed as $H_1(\mathcal{H}(pk||R||M), \sigma_0)$, for hash functions H_1 and $\mathcal{H}$. The second challenge $ch_2 = (b^{(0)}, \ldots, b^{(r)})$ is the output of the hash function H_2 and is computed as $H_2(\mathcal{H}(pk||R||M), \sigma_0, ch_1, \sigma_1)$. The multivariate system $\mathbf{F}$ is generated as $XOF_{\mathbf{F}}(S_{\mathbf{F}})$, for an extendable output function XOF, and $S_{\mathbf{F}}$ is the output of a pseudorandom generator $PRG_{sk}(sk)$. The secret key $\mathbf{s}$ is generated as $PRG_{\mathbf{s}}(S_{\mathbf{s}})$, for a pseudorandom generator $PRG_{\mathbf{s}}$, and $S_{\mathbf{s}}$ is the output of a pseudorandom generator $PRG_{sk}(sk)$.

2.12.1 Random values included in the clear

The random value R is generated using a hash function with the message and the secret key as input. Only the sender, which has the secret key, can verify that the random value R is computed according to the protocol. We can replace the random vector with a subliminal message $m_s \in \{0, 1\}^k$. The scheme offers two parameter options. The security parameter k is 256 and 384, respectively.

2.12.2 Subliminal channel using secret values

Using the secret key sk the subliminal receiver can recover $\mathbf{r}_0^{(j)}$, $\mathbf{t}_0^{(j)}$, and $\mathbf{e}_0^{(j)}$, for $j \in \{1, \ldots, r\}$, by doing the following.

Generate the second challenge $ch_2 = (b^{(0)}, \ldots, b^{(r)})$ using pk, R and M, and $\mathbf{s}$ using sk. These are used to recover the random values $\mathbf{r}_0^{(j)}$'s from $\sigma_2 = (\mathbf{r}_{b^{(1)}}^{(1)}, \ldots, \mathbf{r}_{b^{(r)}}^{(r)})$. If $b^{(j)} = 0$ then $\mathbf{r}_0^{(j)} = \mathbf{r}_{b^{(j)}}^{(j)}$, if $b^{(j)} = 1$ then $\mathbf{r}_0^{(j)} = \mathbf{s} - \mathbf{r}_{b^{(j)}}^{(j)}$, for $j \in \{1, \ldots, r\}$.

Generate the first challenge $ch_1 = (\alpha^{(0)}, \ldots, \alpha^{(r)})$ using pk, R and M, and the multivariate system $\mathbf{F}$ using sk. These, together with the $\mathbf{r}_0^{(j)}$'s, are used to recover the $\mathbf{t}_0^{(j)}$'s and $\mathbf{e}_0^{(j)}$'s from $\sigma_1 = \{(\alpha^{(j)}\mathbf{r}_0^{(j)} - \mathbf{t}_0^{(j)}, \alpha^{(j)}\mathbf{F}(\mathbf{r}_0^{(j)}) - \mathbf{e}_0^{(j)})\}_{j \in \{1,\ldots,r\}}$. Use $\alpha^{(j)}$ and $\mathbf{r}_0^{(j)}$ to recover $\mathbf{t}_0^{(j)}$, and use $\alpha^{(j)}$ and $\mathbf{F}(\mathbf{r}_0^{(j)})$ to recover $\mathbf{e}_0^{(j)}$, for $j \in \{1, \ldots, r\}$.

Replace the random values with a subliminal message $\mathbf{m}_s$ with a combined bit length of $3rn\lceil \log_2 q \rceil$. The submission offer two parameter sets. The integer r is 269 and 403, respectively, the integer n is 48 and 64, respectively, and the field order q is 31 for all sets.

2.13 Picnic

The two building blocks of Picnic [16] are a hash function and a block cipher. The hash function is used in a zero knowledge proof and the block cipher is used to generate the public key. The public key is an encryption of a random value, using the secret key as encryption key, and the signature is a proof of knowledge of the secret key using the message as a nonce. The scheme uses a zero knowledge proof to prevent information about the secret key to leak. The scheme specifically use ZKB++, an optimized version of ZKBoo [28], for the zero-knowledge proofs and the block cipher LowMC [2]. The scheme uses bytes, byte arrays, integers, integer vectors, and bits, where everything is denoted in plain.

Signatures have the form

$$(e, b_0, \ldots, b_{T-1}, z_0, \ldots, z_{T-1}),$$

where e is a hash of all values used in the computation and the message, the b_i's are parts of the commitments used in the scheme, and the z_i's are parts of the randomness used.

2.13.1 Random values included in the clear

In total there are T triples of randomness used. Each z_i consists of two of the three random values of a triplet, where the sender does not know which two is selected before the signature is generated as the choice depends on the randomness used. Set two of the three random values to be subliminal messages and, using techniques from secret sharing, the third is chosen such that all three values sums to zero. This makes a two out of three threshold scheme and the receiver can recover all three random values from any two values.

Each triplet in the random value are S bits long and there are T triples, hence the subliminal message has length $m_s \in \{0,1\}^{2ST}$. The scheme proposes three security levels. For each security level the value S is 128, 192, and 256, respectively, and the value T is 219, 329, and 438, respectively.

2.14 pqNTRUsign

The pqNTRUsign signature scheme [32, 33] uses the NTRU lattice: the polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle x^N \pm 1\rangle$, for an integer N. An element $\mathbf{x} \in \mathcal{R}_q$ is denoted in bold.

Signatures have the form

$$\mathrm{Sign}_{pqNTRUsign} = \mathbf{r} + (-1)^b \mathbf{af}.$$

The random value $\mathbf{r}$ is either sampled from a uniform or a Gaussian distribution, the bit b is either zero or one, the value $\mathbf{a}$ is used to adjust a signature such that it meets a congruence requirement, and the value $\mathbf{f}$ is part of the secret key.

2.14.1 Subliminal channel using high order digits

For a valid signature we know that $\|\mathbf{af}\|_2 \leq B_s$, for a norm bound B_s, and $\|\mathbf{af}\|_\infty \leq \|\mathbf{af}\|_2 \leq B_s$. We can insert a subliminal message in the high order digits of $\mathbf{r}$ if B_s is small compared to the coefficients of $\mathbf{r}$.

Table 2: pqNTRUsign parameters and subliminal message bounds for high orders digits subliminal channel.

Parameter	Description	Parameter set	
		Uniform–1024	Gaussian–1024
N	dimension	1024	1024
B_s	norm bound	98	500
$\|\mathbf{r}\|_\infty$	randomness bound	16384	—
σ	standard deviation	—	250
b	bit	0	$\{0, 1\}$
$\|\mathbf{m}_s\|_\infty$	message space	16	1
$\|r'_i\|$	masking value	$[98, 901]$	—

Proof of concept – uniform distribution The pqNTRUsign scheme with uniform distribution has the parameters $b = 0$, $B_s = 98$, and $\|\mathbf{r}\|_\infty \leq 16384$, see Table 2. The coefficients of $\mathbf{af}$ is at most three digits long, which leaves us with two digits for the message space. Set the random value $\mathbf{r} = 1000\mathbf{m}_s + \mathbf{r}'$ and

$$\text{Sign}'_{pqNTRUsign} = 1000\mathbf{m}_s + \mathbf{r}' + \mathbf{af},$$

where the bounds on $\mathbf{m}_s$ and coefficient interval of $\mathbf{r}'$ are in Table 2. Note that each coefficient in the masking randomness $\mathbf{r}'$ needs to have the same sign as the subliminal message, that is, $\text{sgn}(r'_i) = \text{sgn}(\mathbf{m}_{s_i})$ for all $i \in \{1, \ldots, N\}$, for the message recovery to be correct. The subliminal message $\mathbf{m}_s$ is bounded such that $\|\text{Sign}'_{pqNTRUsign}\|_\infty \leq \|\text{Sign}_{pqNTRUsign}\|_\infty$, for a valid signature $\text{Sign}_{pqNTRUsign}$, and the coefficients of the masking randomness are chosen such that $0 \leq \|\mathbf{r}' + \mathbf{af}\|_\infty \leq 999$. To recover the subliminal message, the receiver computes

$$\left\lfloor \frac{\text{Sign}'_{pqNTRUsign}}{1000} \right\rceil$$

to remove the error term and retrieve the subliminal message $\mathbf{m}_s$. The signature is bounded, the max norm should not be too big, and has to meet a congruence requirement. We can encrypt the subliminal messages again, using fresh randomness, such that these requirements will be met. Resample $\mathbf{r}'$ if $\|1000\mathbf{m}_s + \mathbf{r}'\|_\infty > \|\mathbf{r}\|_\infty$.

Proof of concept – Gaussian distribution The pqNTRUsign scheme with discrete Gaussian distribution has the parameter setting $p = 2$, $B_s = 500$, and $\mathbf{r}$ is sampled from a discrete Gaussian with standard deviation $\sigma = 250$, see Table 2. It is expected that 95 % of the sampled values from the discrete Gaussian are in the interval $(-500, 500)$, and it is reasonable that 5 % will be outside the interval. If a sampled value $r_i > 500$ then $r_i + a_i f_i > 0$ since $\|\mathbf{af}\|_\infty \leq 500$. Similarly, if $r_i < -500$ then $r_i + a_i f_i < 0$. In other words, to send the bit $\mathbf{m}_{s_i} = 1$ pick a $r_i > 500$ and to send the bit $\mathbf{m}_{s_i} = 0$ pick a $r_i < -500$.

Five percent of $N = 1024$ is 51.2, hence it is reasonable to send 51 bits in the subliminal message. The placement of these 51 bits has to be specified before the signature is sent, say, index set $\{i_1, i_2, \ldots i_{51}\} \subset \{1, 2, \ldots, 1024\}$. The random value is sampled according to the scheme and permuted such that the values at index $i \in \{i_1, i_2, \ldots i_{51}\}$ are larger than 500 or smaller than -500 to send $\mathbf{m}_{s_i} = 1$ and $\mathbf{m}_{s_i} = 0$, respectively. If there not enough large values we can resample the random vector. The signature is required to be bounded, the max norm should not be too big, and meet a congruence. If the produced signature is does not meet theses requirements sample a new random value which will be modified again.

2.15 pqRSA

Post-Quantum RSA is based on factorization and uses large parameters to make it secure in the post quantum setting. Values are represented as byte strings and the computation is over the integers in little endian form.

Signatures have the form

$$\mathrm{Sign}_{pqRSA} = (R, \overline{X}),$$

where R is a uniform random value and $\overline{X}$ is an encoding of $H(R, M)^d \bmod N$, for a RSA private key d, modulus N, and message M.

2.15.1 Random values included in the clear

Replace the 32 byte long random string R with a subliminal message m_s.

2.16 pqsigRM

The pqsigRM is a signature scheme based on a punctured Reed-Muller code with random insertion and on CFS [18]. All elements are denoted in plain.

Signatures have the form

$$\text{Sign}_{pqsigRM} = (M, e, i_r),$$

where i_r is a random value generated using AES, M is the message, and $e^T = Q^{-1}e'^T$, where e' is a punctured error vector with weight w and Q is a permutation matrix. The error vector is generated by first computing a syndrome s, which is decoded to find a (nonpunctured) error vector and then punctured to get e'.

2.16.1 Random values included in the clear

Replace the random value i_r with a subliminal message m_s. The exact size of the value is not mentioned in the submission. Looking at the signature size it seems to be 16 bytes long.

2.17 qTesla

The qTesla signature scheme is a variant of the TESLA signature schemes [1, 3, 6], which is based on the scheme by Bai and Galbraith [5]. The hardness of the scheme is based on the decisional ring learning with error problem. The scheme uses the polynomial rings $\mathcal{R} = \mathbb{Z}[x]/\langle x^n + 1\rangle$ and $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$, for a dimension n and integer q. An element in $x \in \mathcal{R}$ is denoted in plain.

Signatures have the form

$$\text{Sign}_{qTesla} = (c', z),$$

where $z = y + sc$ and c' is a hash value. The value $y \in \mathcal{R}_q$ is a uniformly random value. The value $s \in \mathcal{R}_q$ is a secret key sampled from

a Gaussian distribution over $\mathcal{R}_q$ with standard deviation σ. The value $c \in \mathcal{R}_q$ is a polynomial with coefficients in $\{-1, 0, 1\}$. It is required for a valid signature that $\|z\|_\infty \leq B - L_S$, for bound parameters B and L_S. Using the signature and the public values a and t the verifier can compute $w = az - t\text{Enc}(c')$, where the value w should be bounded by $q/2 - L_E$ and the L least significant bits of $ay - ec$ is bounded by $2^L - L_E$ for the signature to be valid.

2.17.1 Subliminal channel using secret values

The receiver can recover the random value y with the secret key $sk = (s, e, seed_y, seed_a)$. The sender replaces the random value y with a subliminal message m_s, that is, let $\text{Sign}_{qTesla} = (c', z)$, where $z = m_s + sc$. The receiver computes $sc = sEnc(c')$ and $m_s = z - sc$ to retrieve the subliminal message. If the signature is not valid we can encrypt the subliminal message again, using fresh randomness, and try again.

2.17.2 Subliminal channel using high order digits

For a valid signature we know that $\|sc\|_\infty \leq L_S$, where L_S is a norm bound parameter. We can insert a subliminal message in the high order digits of y if L_S is small compared to the coefficients of y.

Proof of concept In all three parameter settings sc has at most four digits and y has up to seven digits, see Table 3. Set $y = 10000m_s + r'$ and

$$z' = 10000m_s + r' + sc,$$

where the bounds on m_s and r' are given in Table 3. Note that each coefficient in the masking randomness r' needs to have the same sign as the subliminal message, that is, $\text{sgn}(r'_i) = \text{sgn}(m_{s_i})$ for all $i \in \{1, \ldots, n\}$, for the message recovery to be correct. The subliminal message m_s is bounded such that $\|z'\|_\infty \leq B$, and the coefficients of the masking randomness are chosen such that $0 \leq \|r' + sc\|_\infty \leq 9999$. The receiver computes

$$\left\lfloor \frac{z'}{10000} \right\rfloor$$

Table 3: qTesla parameters and subliminal message bounds for high orders digits subliminal channel.

Parameter	Description	Parameter set qTesla–128	qTesla–192	qTesla–256
n	dimension	1024	2048	2048
L_S	norm bound	758	1138	1516
B	randomness bound	$2^{20}-1$	$2^{21}-1$	$2^{22}-1$
$\|m_s\|_\infty$	message space	104	209	419
$\|r'_i\|$	masking value	$[758, 9241]$	$[1138, 8861]$	$[1516, 8483]$

to remove the error term and retrieve the subliminal message m_s. If the signature is not valid we can encrypt the subliminal message again, using fresh randomness. Resample masking values r' such that $\|10000m_s + r'\|_\infty \leq B - L_S$.

2.18 RaCoSS

The RaCoSS signature scheme uses random codes and is based on CFS [18] and KKS [36]. The scheme uses binary vectors and matrices, where vectors are denoted in lowercase plain and a matrices in uppercase plain.

Signatures have the form

$$\text{Sign}_{RaCoSS} = (z, c),$$

where c is a bit string with hamming weight w, $z = S^t c + y$, S^t is a secret key, and y is a random value. The vector $y \in \{0,1\}^n$ is sampled from the Bernoulli distribution, where $y_i = 1$ with probability ρ and $y_i = 0$ with probability $1 - \rho$ for all indices i. The value $\rho = 0.057$ and y is a sparse vector.

2.18.1 Subliminal channel using secret values

A subliminal receiver with the secret key S^t can recover the random value by computing $y = z - S^t c$. The scheme proposes one parameter set, where $n = 2400$.

To encode a subliminal message in the sparse vector y we divide it into blocks, where a blocks consisting of only zeroes is sending the

bit 0 and a block consisting of at east one 1 is sending the bit 1. It is reasonable to see $2400 \cdot 0.057 \approx 137$ ones in a Bernoulli sampled vector, and, assuming messages consists of an almost equal number of ones and zeroes, we can have a block length of 10 we can send 240 bits of information. Sample each block according to the Bernoulli distribution and reject any sample that does not produce the block we want. Concatenate the blocks to make a vector y' and permute its coefficients to produce y. The permutation could be shared together with the secret key or be produced using a hash function with the secret and a counter as input.

2.19 Rainbow

The Rainbow signature scheme [23] is a generalization of the Oil and Vinegar structure. The scheme uses a finite field $\mathbb{F}_q$, where vectors are denoted in lowercase bold. The affine transformations and maps are denoted in uppercase plain letters and their inverse is denoted with the prefix Inv. Field elements are denoted in lowercase plain letters.

Signatures have the form

$$\text{Sign}_{Rainbow} = (InvT \cdot (\mathbf{y} - c_T), r),$$

where $invT$ is the inverse of the affine map $\mathcal{T}$, $\mathbf{y}$ is a solution under the central map $\mathcal{F}$ and consists of ν_1 random values, c_T is a random masking value and a part of the secret key, and $r \in \{0,1\}^l$ is a random salt.

2.19.1 Random values included in the clear

We can replace the random value $r \in \{0,1\}^l$ with a subliminal message, where l is 128 for all parameter sets.

2.19.2 Subliminal channel using secret values

We can insert a subliminal message $\mathbf{m}_s \in \mathbb{F}_q^{\nu_1}$ in the first ν_1 elements of $\mathbf{y}$ and the receiver can recover it by using the map T and masking value c_T, both contained in the secret key.

The scheme offers nine parameter sets. The number of random values ν_1 is 32, 36, 40, 64, 68, 56, 92, 76, and 84, respectively. The number of group elements q is 2^4, 31, 2^8, 31, 2^8, 2^4, 2^8, 2^4, and 31, respectively. The random elements are used to make a solvable system and if the chosen subliminal message does not produce such a system we can encrypt the message again, using fresh randomness.

2.20 RankSign

The RankSign signature scheme is based on the RankSign cryptosystem [26], this submission proposes a variation of the existing cryptosystem by adding a small random error in the signature. The scheme uses a finite field $\mathbb{F}_{q^m}$, where q is a power of a prime p and m is a positive integer. Vectors are denoted in bold.

Signatures have the form

$$\text{Sign}_{RankSign} = (\mathbf{e}, seed),$$

where *seed* is a counter and $\mathbf{e}$ is an error vector that satisfies $\mathbf{e}^T\mathbf{H}_{pub} = G(M, seed)$, for a public parity-check matrix $\mathbf{H}_{pub}$, hash function G, and message M.

2.20.1 Random values included in the clear

The seed is l bits, which can be replaced by a subliminal message m_s. The submission is unclear on the exact value of l, other than it is an integer input to their signature algorithm. From the implementation it seems that the seed uses the data type unsigned char, which is one byte. This implies that $l = 8$ for all parameter sets. If the corresponding syndrome (of the error vector $\mathbf{e}$) is not decodable we can encrypt the subliminal message again, using fresh randomness.

2.21 SPHINCS$^+$

SPHINCS$^+$ is based on SPHINCS [8], a stateless hash based signature scheme. The scheme outputs byte strings, where a byte string $\mathbf{b}$ is denoted in bold.

Signatures have the form

$$\text{Sign}_{SPHINCS^+} = (\mathbf{R}, \mathbf{SIG}_{FORS}, \mathbf{SIG_{HT}}),$$

where $\mathbf{R}$ is a random value generated using a pseudo random function, $\mathbf{SIG}_{FORS}$ is a FORS signature and $\mathbf{SIG_{HT}}$ is a hyper tree signature. The FORS signature (an improvement of the few times signature scheme HORST [8]) contains private key values, generated using a pseudorandom function, and their associated authentication paths. The hyper tree signature contains XMSS signatures [13]. A XMSS signature contains WOTS signatures [34] and their associated authentication paths, where the WOTS signature takes as input the messages and the public and secret keys.

2.21.1 Random values included in the clear

The random value $\mathbf{R}$ is constructed using a pseudorandom function with a salt, an optional value, and the message as input. The salt is part of the secret key and the verifier cannot verify that the random value was computed using the pseudorandom function. We can replace $\mathbf{R}$ with a subliminal message $\mathbf{m}_s$.

The private key values in $\mathbf{SIG}_{FORS}$ are generated with a pseudorandom function and can be used as a subliminal channel. The signature contains k random values, where each are n bytes.

The submission offers six parameter set. The security parameter n is 16, 24, or 32 bytes long, respectively, and the number of FORS trees k is 10, 30, 14, 33, 22, 30, respectively.

2.22 SRTPI

The SRTPI signature scheme is based on the Non-symmetric Simultaneous Algebraic Riccati Equations problem, which is showed to be NP hard [46]. The scheme works over a field $\mathbb{F}_q$, where matrices are denoted in plain uppercase.

Signatures have the form

$$\text{Sign}_{TPSig} = (m, X_m),$$

where m is the message and X_m is a matrix, where $X_m = X_0 + C^+M_m + (I - C^+C)U_0(I - CC^+)$. The matrices X_0 and U_0 are random and part of the secret key. The matrix C is random and part of the public key. The matrix C^+ is the Moor-Penrose pseudoinverse of C. The matrix

$$M_m = \begin{bmatrix} I_{n_1} & M_{1,2} & M_{1,3} \\ 0 & -I_{n_2} & L_{2,3} \\ 0 & 0 & 0_{n_3} \end{bmatrix},$$

where $M_{1,2}$ contains the hashed values of the message, in a permuted order, $M_{1,3} = -M_{1,2}L_{2,3}+L_{1,2}L_{2,3}+L_{1,3}$, and I is the identity matrix. The matrices $L_{1,2}$, $L_{1,3}$, and $L_{2,3}$ are random and part of the secret key.

2.22.1 No channel possible

The signature scheme is deterministic and the only difference between two signatures is the messages, and the adversary will notice if any of the secret random matrices is changed for a signature. Any changes in X_0 or U_0 will be detected by a verifier, as the public matrix Q depends them. It is impossible to recover L, M, or π since we need C and C^+ to be invertible, where both C and C^+ are noninvertible by construction. No (reliable) subliminal channel is possible in the SRTPI signature scheme.

2.23 WalnutDSA

The WalnutDSA scheme is based on the Reversing E-Multiplication problem over a braid group. The braid group B_N is defined by the set of Artin generators $\{b_1, b_2, \ldots, b_{N-1}\}$ and a braid $\beta \in B_N$ has the form $\beta = b_{i_1}^{\epsilon_1} b_{i_2}^{\epsilon_2} \cdots b_{i_k}^{\epsilon_k}$, where $i_j \in \{1, 2, \ldots, N-1\}$ and $\epsilon_j \in \{-1, 1\}$. The scheme uses the colored Burau representation,

$$\Pi_{CB} : B_N \to (GL(N, \mathbb{F}_q(t_1, t_1^{-1}, \ldots, t_N, t_N^{-1})) \times S_N),$$

which represents the braid as a $N \times N$ matrix over the ring of Laurent polynomials with N variables over the field $\mathbb{F}_q$ and a permutation $\sigma \in S_N$ of N letters. We denote the colored Burau representation

$(GL(N,\mathbb{F}) \times S_N)$ as $CB(\mathbb{F})$, for a field $\mathbb{F}$. E-Multiplication is an operation, denoted by $\star$, between a matrix-permutation pair and the colored Burau representation of a braid

$$\star : CB(F_q) \times CB(\mathbb{F}_q(t_1, t_1^{-1}, \ldots, t_N, t_N^{-1})) \to CB(F_q).$$

When we E-Multiply a colored Burau representation of a braid with a matrix-permutation pair we remove all information of the braid by inserting a set of chosen T-values in the Laurent polynomials.

A signature is a rewritten braid and a hash of the message

$$\mathrm{Sign}_{WalnutDSA} = (H(m), \mathcal{R}(v_1 \cdot w^{-1} \cdot v \cdot E(H(m)) \cdot w' \cdot v_2),$$

where (w, w') is the secret key, $E(H(m))$ is a braid encoding of the hashed message, v, v_1, and v_2 are random values called *cloaking elements*, and $\mathcal{R} : B_N \to B_N$ is a braid rewriting algorithm.

An element ν is a cloaking element for the matrix-permutation pair (M, σ) if $(M, \sigma) \star \Pi_{CB}(\nu) = (M, \sigma)$. The cloaking element is given by $\nu = w b_i^2 w^{-1}$, where b_i is a generator and the corresponding permutation of $w \in B_N$ satisfies $i \mapsto \sigma^{-1}(a)$ and $i+1 \mapsto \sigma^{-1}(b)$. The integers a, b are public values.

2.23.1 Rewritten random values

The scheme proposes three possible methods for rewriting braids: (1) Rewrite the given braid to the Birman–Ko–Lee (BKL) Normal Form [9] then shorten the braid using Dehornoy's FullHRed algorithm [20]; (2) Stochastic rewriting, that is, randomly rewrite the braid using lookup tables; and (3) Stochastic rewrite first then shorten the braid with FullHRed.

The BKL algorithm rewrites a braid, and every equivalent braid, to its unique normal form. This makes it hard to insert a subliminal message.

The FullHRed algorithm of Dehornoy rewrites the braid, to a fully reduced braid, by removing handles of the form $b_i^{\epsilon} \cdots b_i^{-\epsilon}$, for an Artin generator b_i and $\epsilon \in \{-1, 1\}$. Given a chosen braid $m_s\beta$ we *suspect* that we could use the FullHRed algorithm backward to change the

chosen braid to a cloaking element $v_1 = wb_i^2w^{-1}$. Assuming this is possible, then

$$m_s\beta' = \mathcal{R}(v_1 \cdot w^{-1} \cdot v \cdot E(H(m)) \cdot w' \cdot v_2),$$

where m_s is our chosen subliminal message, $\beta' \in B_N$, and $m_s\beta'$ is a fully reduced braid. We leave it as an open problem, to prove that it is possible to use Dehornoy's algorithm backwards, since the Walnut DSA signature scheme has not been accepted into NIST's second round.

The stochastic rewriting algorithm partitions the braid and replaces two consecutive generators $b_{i_j}^{\epsilon_j}b_{i_{j+1}}^{\epsilon_{j+1}}$ in each partition with a relation found in a lookup table. Nothing is changed if no relation was found. We can alter this algorithm such that our chosen subliminal message inserted into the cloaking element stays unchanged and, possibly, write additional messages into the signature.

The random braid w used to construct the cloaking elements has length L, which is 15 and 30 for each parameter set. For option 2 we can at least use three random braids of combined length $3L$ to insert a subliminal message.

3 Summary

In Table 4 and 5 we show the bandwidth of each subliminal channel, using the given parameter sets of the proposed signature schemes.

Table 4: The bandwidth of the subliminal channels found. *Public* channels require public values to recover the subliminal message and *Secret* channels needs secret values. A dash (—) denotes no channel possible, a blank space denotes no channel found, and a plus (+) denotes that both the public and secret channel can be used to send a single subliminal messages. The underlined schemes are accepted to the round 2 of NIST's PQCSP.

Signature scheme	Parameter set	Signature length (bits)	Subliminal bandwidth (bits) Public			Secret	
CRYSTAL–Dilithium	I	11096	4624	(41.67 %)		9726	(87.66 %)
	II	16352	6936	(42.41 %)		14590	(89.22 %)
	III	21608	9247	(42.80 %)		19453	(90.03 %)
	IV	26928	11559	(42.93 %)		24317	(90.30 %)
DME	(3,2,24)	144	16	(11.11 %)			
	(3,2,48)	288	32	(11.11 %)			
DRS	all sets						
DualModeMS	128	256016	704	(0.27 %)			
	192	635320	1728	(0.27 %)			
	256	1192232	8192	(0.69 %)			
Falcon	512	4939	320	(6.48 %)			
	768	7951	320	(4.02 %)			
	1024	9866	320	(3.24 %)			
GeMSS	128	384	48	(12.50 %)			
	192	704	80	(11.36 %)			
	256	832	132	(15.87 %)			
Gravity–SPHINCS	S	101120	6400	(6.33 %)			
	M	231432	8448	(3.65 %)			
	L	281344	7424	(1.64 %)			
Gui	184	360	128	(35.56 %)	+	32	(8.89 %)
	312	504	128	(25.40 %)	+	40	(7.94 %)
	448	664	128	(19.28 %)	+	56	(8.43 %)
HiMQ	3	600				248	(41.33 %)
	3F	536				192	(35.82 %)
	3P	536				248	(46.27 %)
LUOV	8-63-256	2552				2048	(80.25 %)
	8-90-351	3528				2808	(79.59 %)
	8-117-404	4168				3232	(77.54 %)
	48-49-242	13600				11616	(85.41 %)
	64-68-330	24800				21120	(85.16 %)
	80-86-399	37600				31920	(84.89 %)
MQDSS	31-48	263056	256	(0.10 %)	+	193680	(73.63 %)
	31-64	542400	384	(0.07 %)	+	386880	(71.33 %)
Picnic	L1FS	272000	56064	(20.61 %)			
	L1UR	431432	56064	(12.99 %)			
	L3FS	613920	126336	(20.58 %)			
	L3UR	974504	126336	(12.96 %)			
	L5FS	1062592	224256	(21.10 %)			
	L5UR	1675792	224256	(13.38 %)			

Table 5: The bandwidth of the subliminal channels found. *Public* channels require public values to recover the subliminal message and *Secret* channels needs secret values. A dash (—) denotes no channel possible, a blank space denotes no channel found, and a plus (+) denotes that both the public and secret channel can be used to send a single subliminal messages. The underlined schemes are accepted to the round 2 of NIST's PQCSP.

Signature scheme	Parameter set	Signature length (bits)	Subliminal bandwidth (bits) Public			Secret	
pqNTRUSig	Uniform	11264	4096	(36.36 %)			
	Gaussian	16384	51	(0.31 %)			
pqRSA	sign/pqrsa15	262400	256	(0.10 %)			
	sign/pqrsa20	8388864	256	(≈ 0 %)			
	sign/pqrsa25	268435712	256	(≈ 0 %)			
	sign/pqrsa30	8589934848	256	(≈ 0 %)			
pqsigRM	5-11	2080	128	(6.15 %)			
	6-12	4128	128	(3.10 %)			
	6-13	8224	128	(1.56 %)			
qTesla	128	21760	6861	(31.53 %)		20479	(94.11 %)
	192	45312	15785	(34.84 %)		43006	(94.91 %)
	256	47360	17840	(37.67 %)		45055	(95.13 %)
RaCoSS		4688				240	(5.12 %)
Rainbow	Ia	512	128	(25.00 %)	+	128	(25.00 %)
	Ib	624	128	(20.51 %)	+	178	(28.58 %)
	Ic	832	128	(15.38 %)	+	320	(38.46 %)
	IIIb	896	128	(14.29 %)	+	317	(35.39 %)
	IIIc	1248	128	(10.26 %)	+	544	(43.59 %)
	IVa	736	128	(17.39 %)	+	224	(30.43 %)
	Vc	1632	128	(7.84 %)	+	736	(45.10 %)
	VIa	944	128	(13.56 %)	+	304	(32.20 %)
	VIb	1176	128	(10.88 %)	+	416	(35.39 %)
RankSign	I	11016	8	(0.07 %)			
	II	12008	8	(0.07 %)			
	III	17288	8	(0.05 %)			
	IV	23432	8	(0.03 %)			
SPHINCS$^+$	128s	64640	1408	(2.18 %)			
	128f	135808	3968	(2.92 %)			
	192s	136512	2880	(2.11 %)			
	192f	285312	6528	(2.29 %)			
	256s	238336	5888	(2.47 %)			
	256f	393728	7936	(2.02 %)			
SRTPI	all sets			—			—
WalnutDSA	128 (Option 1)	5173					
	128 (Option 2)	11332	180	(1.59 %)			
	128 (Option 3)	5135					
	256 (Option 1)	9982					
	256 (Option 2)	21557	360	(1.67 %)			
	256 (Option 3)	9932					

References

[1] Sedat Akleylek, Nina Bindel, Johannes Buchmann, Juliane Krämer, and Giorgia Azzurra Marson. An efficient lattice-based signature scheme with provably secure instantiation. Cryptology ePrint Archive, Report 2016/030, 2016. https://eprint.iacr.org/2016/030.

[2] Martin Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. Cryptology ePrint Archive, Report 2016/687, 2016. https://eprint.iacr.org/2016/687.

[3] Erdem Alkim, Nina Bindel, Johannes Buchmann, Özgür Dagdelen, Edward Eaton, Gus Gutoski, Juliane Krämer, and Filip Pawlega. Revisiting TESLA in the quantum random oracle model. Cryptology ePrint Archive, Report 2015/755, 2015. https://eprint.iacr.org/2015/755.

[4] Ross Anderson, Serge Vaudenay, Bart Preneel, and Kaisa Nyberg. The newton channel. In Ross Anderson, editor, *Information Hiding*, pages 151–156, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[5] Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In Josh Benaloh, editor, *Topics in Cryptology – CT-RSA 2014*, pages 28–47, Cham, 2014. Springer International Publishing.

[6] Paulo S. L. M. Barreto, Patrick Longa, Michael Naehrig, Jefferson E. Ricardini, and Gustavo Zanon. Sharper ring-LWE signatures. Cryptology ePrint Archive, Report 2016/1026, 2016. https://eprint.iacr.org/2016/1026.

[7] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures-how to sign with rsa and rabin. In *Proceedings of the 15th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'96, pages 399–416, Berlin, Heidelberg, 1996. Springer-Verlag.

[8] Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. SPHINCS: Practical stateless hash-based signatures. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 368–397, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[9] J S Birman, K H Ko, and Lee Jae Sik. A new approach to the word and conjugacy problems in the braid groups. *Adv. Math.*, 139(math.GT/9712211):322–353. 31 p, Jul 1998.

[10] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT'98*, pages 127–144, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[11] Jens-Matthias Bohli, María Isabel González Vasco, and Rainer Steinwandt. A subliminal-free variant of ecdsa. In Jan L. Camenisch, Christian S. Collberg, Neil F. Johnson, and Phil Sallee, editors, *Information Hiding*, pages 375–387, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[12] Jens-Matthias Bohli and Rainer Steinwandt. On subliminal channels in deterministic signature schemes. In Choon-sik Park and Seongtaek Chee, editors, *Information Security and Cryptology – ICISC 2004*, pages 182–194, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[13] Johannes Buchmann, Erik Dahmen, and Andreas Hülsing. Xmss - a practical forward secure signature scheme based on minimal security assumptions. In Bo-Yin Yang, editor, *Post-Quantum Cryptography*, pages 117–129, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[14] Mike Burmester, Yvo G. Desmedt, Toshiya Itoh, Kouichi Sakurai, and Hiroki Shizuya. Divertible and subliminal-free zero-knowledge proofs for languages. *J. Cryptol.*, 12(3):197–223, June 1999.

[15] Mike V. D. Burmester and Yvo Desmedt. All languages in np have divertible zero-knowledge proofs and arguments under cryptographic assumptions. In Ivan Bjerre Damgård, editor, *Advances in Cryptology — EUROCRYPT '90*, pages 1–10, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[16] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1825–1842, 2017.

[17] Rongmao Chen, Yi Mu, Guomin Yang, Willy Susilo, Fuchun Guo, and Mingwu Zhang. Cryptographic reverse firewall via malleable smooth projective hash functions. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, pages 844–876, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[18] Nicolas T. Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a McEliece-based digital signature scheme. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, pages 157–174, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[19] Cryptographic Technology group at NIST. Post-quantum cryptography. `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`, 2016. Accessed: 2019-04-29.

[20] Patrick Dehornoy. A fast method for comparing braids. *Advances in Mathematics*, 125(2):200 – 235, 1997.

[21] Yvo Desmedt. Subliminal-free authentication and signature. In D. Barstow, W. Brauer, P. Brinch Hansen, D. Gries, D. Luckham, C. Moler, A. Pnueli, G. Seegmüller, J. Stoer, N. Wirth, and Christoph G. Günther, editors, *Advances in Cryptology — EU-*

ROCRYPT '88, pages 23–33, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[22] Yvo Desmedt. Simmons' protocol is not free of subliminal channels. In *Proceedings of the 9th IEEE Workshop on Computer Security Foundations*, CSFW '96, pages 170–, Washington, DC, USA, 1996. IEEE Computer Society.

[23] Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security*, pages 164–175, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[24] Jintai Ding and Bo-Yin Yang. Degree of regularity for HFEv and HFEv-. In Philippe Gaborit, editor, *Post-Quantum Cryptography*, pages 52–66, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[25] Qingkuan Dong and Guozhen Xiao. A subliminal-free variant of ecdsa using interactive protocol. In *2010 International Conference on E-Product E-Service and E-Entertainment*, pages 1–3, Nov 2010.

[26] Philippe Gaborit, Olivier Ruatta, Julien Schrek, and Gilles Zémor. RankSign : an efficient signature algorithm based on the rank metric. *CoRR*, abs/1606.00629, 2016.

[27] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 197–206, New York, NY, USA, 2008. ACM.

[28] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. ZKBoo: Faster zero-knowledge for boolean circuits. Cryptology ePrint Archive, Report 2016/163, 2016. `https://eprint.iacr.org/2016/163`.

[29] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004.

[30] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '97, pages 112–131, London, UK, UK, 1997. Springer-Verlag.

[31] Alexander Hartl, Robert Annessi, and Tanja Zseby. Subliminal channels in high-speed signatures. *JoWUA*, 9:30–53, 2018.

[32] Jeffrey Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, and William Whyte. Transcript secure signatures based on modular lattices. Cryptology ePrint Archive, Report 2014/457, 2014. `https://eprint.iacr.org/2014/457`.

[33] Jeffrey Hoffstein, Jill Pipher, William Whyte, and Zhenfei Zhang. A signature scheme from learning with truncation. Cryptology ePrint Archive, Report 2017/995, 2017. `https://eprint.iacr.org/2017/995`.

[34] Andreas Hülsing. W-ots+ – shorter signatures for hash-based signature schemes. In Amr Youssef, Abderrahmane Nitaj, and Aboul Ella Hassanien, editors, *Progress in Cryptology – AFRICACRYPT 2013*, pages 173–188, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[35] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, STOC '89, pages 12–24, New York, NY, USA, 1989. ACM.

[36] Gregory Kabatianskii, E. Krouk, and Ben Smeets. A digital signature scheme based on random error-correcting codes. In *Proceedings of the 6th IMA International Conference on Cryptography and Coding*, pages 161–167, Berlin, Heidelberg, 1997. Springer-Verlag.

[37] Robin Kwant, Tanja Lange, and Kimberley Thissen. Lattice klepto: Turning post-quantum crypto against itself. Cryptology ePrint Archive, Report 2017/1140, 2017. https://eprint.iacr.org/2017/1140.

[38] Dai-Rui Lin, Chih-I Wang, Zhi-Kai Zhang, and D. J. Guan. A digital signature with multiple subliminal channels and its applications. *Comput. Math. Appl.*, 60(2):276–284, July 2010.

[39] Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In *Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '09, pages 598–616, Berlin, Heidelberg, 2009. Springer-Verlag.

[40] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In D. Barstow, W. Brauer, P. Brinch Hansen, D. Gries, D. Luckham, C. Moler, A. Pnueli, G. Seegmüller, J. Stoer, N. Wirth, and Christoph G. Günther, editors, *Advances in Cryptology — EUROCRYPT '88*, pages 419–453, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[41] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 218–238, New York, NY, 1990. Springer New York.

[42] Ilya Mironov and Noah Stephens-Davidowitz. Cryptographic reverse firewalls. Cryptology ePrint Archive, Report 2014/758, 2014. https://eprint.iacr.org/2014/758.

[43] Tatsuaki Okamoto and Kazuo Ohta. Divertible zero knowledge interactive proofs and commutative random self-reducibility. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology — EUROCRYPT '89*, pages 134–149, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

[44] Jacques Patarin. The Oil and Vinegar signature scheme. presented at the Dagstuhl Workshop on Cryptography, 1997.

[45] Jacques Patarin, Nicolas Courtois, and Louis Goubin. QUARTZ, 128-bit long digital signatures. In David Naccache, editor, *Topics in Cryptology — CT-RSA 2001*, pages 282–297, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[46] Y. Peretz. On multivariable encryption schemes based on simultaneous algebraic riccati equations over finite fields. *Finite Fields Appl.*, 39(C):1–35, May 2016.

[47] Thomas Plantard, Willy Susilo, and Khin Than Win. A digital signature scheme based on $CVP^{\star}_{\infty}$. *International Workshop on Public Key Cryptography*, pages 288–307, 2008.

[48] Koichi Sakumoto, Taizo Shirai, and Harunaga Hiwatari. Public-key identification schemes based on multivariate quadratic polynomials. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, pages 706–723, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[49] C. P. Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 239–252, New York, NY, 1990. Springer New York.

[50] Gustavus J. Simmons. The prisoners' problem and the subliminal channel. *Advances in Cryptology: Proceedings of Crypto 83*, pages 51–67, 1984.

[51] Gustavus J. Simmons. The subliminal channel and digital signatures. In Thomas Beth, Norbert Cot, and Ingemar Ingemarsson, editors, *Advances in Cryptology*, pages 364–378, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.

[52] Gustavus J. Simmons. A secure subliminal channel (?). In Hugh C. Williams, editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 33–41, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

[53] Gustavus J. Simmons. An introduction to the mathematics of trust in security protocols. In *Computer Security Foundations*

Workshop VI, [1993] Computer Security Foundations Workshop VI, pages 121–127. IEEE Computer Society Press, 1993.

[54] Gustavus J. Simmons. Results concerning the bandwidth of subliminal channels. *IEEE Journal on Selected Areas in Communications*, 16(4):463–473, May 1998.

[55] Alan Szepieniec, Ward Beullens, and Bart Preneel. MQ signatures for PKI. Cryptology ePrint Archive, Report 2017/327, 2017. `https://eprint.iacr.org/2017/327`.

[56] Adam Young and Moti Yung. Kleptography: Using cryptography against cryptography. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, pages 62–74, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[57] Xianfeng Zhao and Ning Li. Reversible watermarking with subliminal channel. In Kaushal Solanki, Kenneth Sullivan, and Upamanyu Madhow, editors, *Information Hiding*, pages 118–131, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

Paper III

Cloud-assisted Asynchronous Key Transport with Post-Quantum Security

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

In submission, ePrint 2019/1409

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
davies@uni-wuppertal.de
[2]NTNU – Norwegian University of Science and Technology
{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 asynchronous 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, Post-quantum cryptography, Group Key Exchange, Blinded Key Encapsulation, NTRU, Forward Secrecy, Cloud Storage

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 proce-

dure was named by the authors as a *blinded KEM* (BKEM), and the complete protocol built from this was named as a cloud-assisted offline 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 their paper for full details [10]. In the protocol, the server runs the KG and Decap algorithms 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.

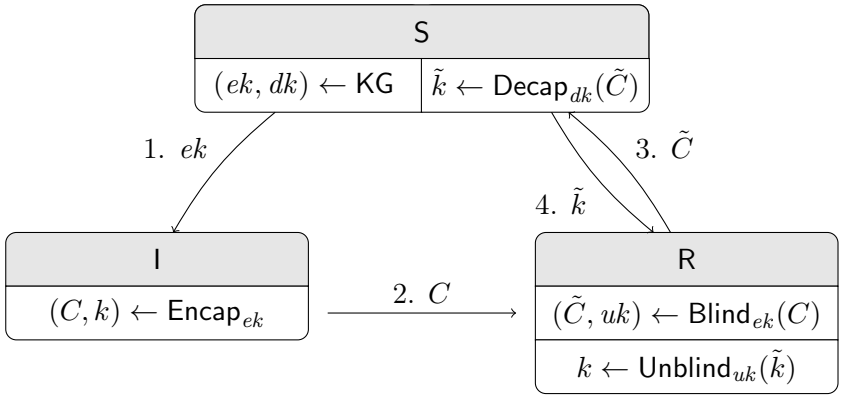


Figure 1: A simplified overview of an OAGKE protocol [10] between an initiator I, server S and potentially many recipients R (one is given here for ease of exposition), built using a BKEM. File encryption key k is used by I to encrypt one or more files. The numbered arrows indicate the order in which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

While the approach appears promising, their two 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

wish to construct 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 on constructing regular key encapsulation mechanisms (KEMs) [1, 18, 19, 21, 30, 32] that are post-quantum secure [8, 14, 31, 35, 36] (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 big 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 they provided two BKEM constructions, based on Diffie-Hellman and RSA. To our knowledge these are the only BKEM constructions in the literature.

Many works focused 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 in recent years [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].

After Gentry's breakthrough several FHE schemes where constructed following his framework [11,16,23,26], where 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 blinded KEM 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 construction 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 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.

$$\text{Hard problems} \xrightarrow[\text{or Lyubashevsky et al. [37]}]{\text{Quantum, Gentry [24]}} \begin{matrix}\mathsf{IND\text{-}CPA}\\ \mathsf{HE}\end{matrix} \xrightarrow{\text{This work}} \begin{matrix}\mathsf{IND\text{-}secure}\\ \mathsf{HE\text{-}BKEM}\end{matrix}$$

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 BKEM 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.

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. 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 $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \in \mathbb{R}^m$, 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 will 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 the half-open parallelepiped associated to the basis $\mathbf{B}$ be $\mathcal{P}(\mathbf{B}) = \{\sum_{i=1}^{n} x_i \mathbf{b}_i \mid x_i \in [-1/2, 1/2), \mathbf{b}_i \in \mathbf{B}\}$.

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, $\|\mathbf{B}\| = \max_{1 \le i \le n}(\|\mathbf{b}_i\|)$.

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 convenience 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

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

for all $\mathbf{x} \in L$.

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 any lattice L and real value $\epsilon > 0$, let the *smoothing parameter* $\eta_\epsilon(L)$ denote the smallest s such

that $\rho_{1/s}(L^* \setminus \{\mathbf{0}\}) \leq \epsilon$. 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 $\mathsf{GHE} = (\mathsf{KG}_{\mathsf{GHE}}, \mathsf{Enc}_{\mathsf{GHE}}, \mathsf{Dec}_{\mathsf{GHE}}, \mathsf{Add}_{\mathsf{GHE}})$ be an (additively) Homomorphic encryption scheme derived from ideal lattices, with algorithms as defined in Figure 2. The scheme is similar to Gentry's somewhat homomorphic scheme [24]. The parameters of the GHE scheme are chosen as follows.

- Choose a polynomial ring $R = \mathbb{Z}[x]/(f(x))$ according to a security parameter λ;
- Choose a basis $\mathbf{B}_I$ of the ideal $I \subseteq R$;
- $\mathsf{IdealGen}$ is an algorithm which 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 is an algorithm which 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.
  - $\mathsf{Samp}_1(\mathbf{B}_I, \mathbf{x}, s) = \mathbf{x} + D_{I,s,-\mathbf{x}}$;
  - $\mathsf{Samp}_2(\mathbf{B}_I, \mathbf{x}, s) = \mathbf{x} + D_{I,s,\mathbf{0}}$;
- The plaintext space $\mathcal{P} = R \mod \mathbf{B}_I$ is the set of distinguished representatives of cosets of I with respect to the basis $\mathbf{B}_I$.

$\underline{\mathsf{KG}_{\mathsf{GHE}}(R, \mathbf{B}_I):}$
$(\mathbf{B}_J^{pk}, \mathbf{B}_J^{sk}) \xleftarrow{\$} \mathsf{IdealGen}(R, \mathbf{B}_I)$
$\mathsf{pk} = (R, \mathbf{B}_I, \mathbf{B}_J^{pk}, \mathsf{Samp})$
$\mathsf{sk} = \mathbf{B}_J^{sk}$
return pk, sk

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

$\underline{\mathsf{Dec}_{\mathsf{GHE}}(\mathsf{sk}, \psi):}$
$\pi \leftarrow (\psi \mod \mathbf{B}_J^{sk}) \mod \mathbf{B}_I$
return π

$\underline{\mathsf{Add}_{\mathsf{GHE}}(\mathsf{pk}, \psi_1, \psi_2):}$
$\psi \leftarrow \psi_1 + \psi_2 \mod \mathbf{B}_J^{pk}$
return ψ

Figure 2: 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 $r_{\mathsf{Dec}} = 1/(2 \cdot \|((\mathbf{B}_J^{sk})^{-1})^T\|)$ [24].

Obviously, if $r_{\mathsf{Enc}} \leq r_{\mathsf{Dec}}$ then the encryption scheme is correct. For GHE, if $r_{\mathsf{Enc}} \leq r_{\mathsf{Dec}}$, the probability of decryption error is less than $\frac{1+\epsilon}{1-\epsilon} \cdot 2^{-n}$, which is negligible.

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. Choose the parameters of the scheme 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 $D_{\mathbb{Z}^n,s,\mathbf{0}}$.

The algorithms of the scheme are given in Figure 3.

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 Lemma 12 of [41]). 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

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

$\underline{\mathsf{KG}_{\mathsf{NTRU}}(n, q \in \mathbb{Z}, p \in R_q^\times, s > 0):}$
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 $(\mathsf{pk}, \mathsf{sk})$

$\underline{\mathsf{Enc}_{\mathsf{NTRU}}(\mathsf{pk} = h, s, \pi \in \mathcal{P}):}$
$e_1, e_2 \leftarrow D_{\mathbb{Z}^n,s,\mathbf{0}}$
$\psi \leftarrow \pi + pe_1 + he_2 \in R_q$
return ψ

$\underline{\mathsf{Dec}_{\mathsf{NTRU}}(\mathsf{sk} = f, \psi):}$
$\psi' = f \cdot \psi \in R_q$
$\pi \leftarrow \psi' \mod p$
return π

$\underline{\mathsf{Add}_{\mathsf{NTRU}}(\psi_1, \psi_2):}$
$\psi \leftarrow \psi_1 + \psi_2 \in R_q$
return ψ

Figure 3: The algorithms of the revised NTRU encryption scheme [41].

The standard deviation s is greater or equal to $\eta_\epsilon(\mathbb{Z}^n)$ and has to satisfy $\eta_\epsilon(\mathbb{Z}^n) \leq s$ and $8p_i^2 s^2 n < q/2$ for the decryption to be correct, with high probability.

2.5 Hard lattice problems

The following lattice problems, 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 the following parameters; a polynomial ring R, and a positive real $\gamma \geq 1$. Let $\mathbf{B}_I$ be a basis for an ideal lattice I of R. Given $\mathbf{B}_I$, and the 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 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* $s_{\mathsf{IVIP}} < (\sqrt{2}s\epsilon - 4n^2(\max\{\|\mathbf{B}_I\|\})^2)/(n^4\gamma_\times(R)\cdot \|f\|\max\{\|\mathbf{B}_I\|\})$*, 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 $s\in R_q$, an error distribution D over R_q, define $A_{s,D}$ to 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.

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

Theorem 4 (Lyubashevsky et al. [37]). *Let $\alpha < \sqrt{logn/n}$ 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* $\mathsf{R\text{-}LWE}_{q,D_s}$ *given only $l(\geq 1)$ samples, where $s = \alpha \cdot (nl/\log(nl))^{1/4}$.*

We will consider a different variant of the R-LWE problem, namely $\mathsf{R\text{-}LWE}^{\times}_{\mathsf{HNF}}$, which is the same as $\mathsf{R\text{-}LWE}_{q,\mathcal{D}}$ 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 $q = \Omega(n)$, $\mathsf{R\text{-}LWE}^{\times}_{\mathsf{HNF}}$ remains hard.

The security proof of NTRU encryption scheme is similar to the security proof of Lemma 3.8 provided by Stehlé and Steinfeld [41]. The proof technique relies on the uniformity of public key and $p \in R_q^{\times}$. However, we chose a slightly different error distribution for our construction in Section 5.5, but the 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* $\mathsf{R\text{-}LWE}^{\times}_{\mathsf{HNF}}$ *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 BKEM is parameterized by a key encapsulation mechanism $\mathsf{KEM} = (\mathsf{KG}, \mathsf{Encap}, \mathsf{Decap})$, a blinding algorithm Blind and an unblinding algorithm $\mathsf{Unblind}$; put together we have that $\mathsf{BKEM} = (\mathsf{KG}, \mathsf{Encap}, \mathsf{Blind}, \mathsf{Decap}, \mathsf{Unblind})$.

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)**.** Scheme BKEM has *correctness* if $\mathsf{Unblind}_{uk}(\tilde{k}) = k$, when $(ek, dk) \leftarrow \mathsf{KG}$, $(C, k) \leftarrow \mathsf{Encap}_{ek}$, $(\tilde{C}, uk) \leftarrow \mathsf{Blind}_{ek}(C)$ and $\tilde{k} \leftarrow \mathsf{Decap}_{dk}(\tilde{C})$.

(Note that the KEM scheme has *correctness* if $\mathsf{Decap}_{dk}(C) = k$, when $(ek, dk) \leftarrow \mathsf{KG}$ 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 4.

Definition 7 (PKE-based BKEM)**.** Let BKEM be a blinded KEM, where 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 in Figure 4. We call such a BKEM a *PKE-based BKEM*.

3.1 Security

We define indistinguishability under chosen-plaintext attack ($\mathsf{IND\text{-}CPA}$) for public-key encryption and indistinguishability (IND) for blinded KEMs, respectively.

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

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

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

Figure 4: KEM algorithms parameterized by a PKE scheme $\mathsf{PKE} = (\mathsf{KG}_{\mathsf{PKE}}, \mathsf{Enc}, \mathsf{Dec})$.

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 5 (left). We say that PKE is IND-CPA-secure if $\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A})$ is negligible for any probabilistic polynomial-time adversary $\mathcal{A}$.

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

$\underline{\mathbf{Exp}^{\mathsf{IND}}_{\mathsf{BKEM}}(\mathcal{A}, r):}$
$b \xleftarrow{\$} \{0,1\}$
$(ek, dk) \leftarrow \mathsf{KG}$
$(C, k_1) \leftarrow \mathsf{Encap}_{ek}$
$k_0 \xleftarrow{\$} \mathcal{K}_F$
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 $b' \stackrel{?}{=} b$

Figure 5: IND-CPA experiment $\mathbf{Exp}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A})$ for PKE scheme PKE (left). Indistinguishability experiment $\mathbf{Exp}^{\mathsf{IND}}_{\mathsf{BKEM}}(\mathcal{A}, r)$ for a BKEM scheme BKEM (right).

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 5 (right). We say that BKEM is IND-secure if $\mathbf{Adv}^{\mathsf{IND}}_{\mathsf{BKEM}}(\mathcal{A}, r)$ is negligible for any probabilistic polynomial-time adversary $\mathcal{A}$.

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 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 ciphertexts of k and k', resp., then $\mathsf{Dec}_{\mathsf{sk}}(C \oplus_1 C') = k \oplus_2 k'$, where $\oplus_1$ and $\oplus_2$ denote two group operations. We see two reasons to look at such PKE schemes.

The first reason is that in a BKEM scheme we want the blinding algorithm to alter the file encryption key k. Having a homomorphic encryption (HE) scheme makes this possible and we can construct a blinding algorithm. The second reason is that we want k' to hide k such that the adversary is unable to gain information about k even with knowledge of $\tilde{k} = k \oplus_2 k'$. With a homomorphic encryption scheme we can combine two independently random ciphertexts and make a third one.

We can construct blinding and unblinding algorithms, using this homomorphic property, to create a BKEM with 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 the decapsulation algorithm, that is, $\tilde{k} \leftarrow \mathsf{Decap}_{dk}(\tilde{C})$. To unblind $\tilde{k}$ the unblinding algorithm outputs $\tilde{k} \oplus_2 uk$, which is $(k \oplus_2 k') \oplus_2 (k'^{-1}) = k$, and so the BKEM scheme has 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 $\mathsf{HE} = (\mathsf{KG}_{\mathsf{HE}}, \mathsf{Enc}, \mathsf{Dec})$ such that for any $k, k' \in \mathcal{M}$ and any key pair $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KG}_{\mathsf{HE}}$ it holds that

$$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(k) \oplus_1 \mathsf{Enc}_{\mathsf{pk}}(k')) = k \oplus_2 k'$$

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 6. 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 6: 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, which are the following two sets: $\{\tilde{k}_i = k \oplus_2 k'_i\}_{i=1\ldots r}$ and $\{\tilde{C}_i = C \oplus_1 C'_i\}_{1,\ldots,r}$, in addition to an encapsulation C of the real file encryption key. 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 like a fresh output of the encapsulation algorithm, 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.

Eventually, we provide the main theorem in this paper stating how to achieve an IND secure BKEM scheme. Particularly, if the underlying HE scheme is post-quantum IND-CPA secure then the corresponding homomorphic-based BKEM scheme is post-quantum IND secure.

4.2.1 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 will be 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 the key 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 $\tilde{k}$, and 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* $\mathsf{S} := \{\tilde{k} \in \mathcal{K}_B \mid \forall k \in \mathcal{K}_F, \exists 1 k' \in \mathcal{K}_R \text{ such that } \tilde{k} = k \oplus_2 k'\}$: 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.

4.2.2 Fresh-looking blinded encapsulations

Blinded encapsulations are constructed from two encapsulations, one containing the file encryption key and one containing a blinded value, where we want it to look like a fresh encapsulation, containing the result of the two values with respect to $\oplus_2$. 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 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 HE-BKEM has ϵ-*blinded blinded encapsulation* if the statistical distance between the following distributions is at most ϵ:

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

The above property ensures that the output of the blinding algorithm looks like a fresh encapsulation expect for probability ϵ. Note that the BKEM constructions of Boyd et al. [10], DH-BKEM [10, Section 4.1] and RSA-BKEM [10, Section 4.2], both have 0-blinded blinded encapsulation.

It is well known that in a fully homomorphic encryption scheme the product of two ciphertexts is much larger 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 will use addition.

4.2.3 Indistinguishability of BKEM

Furthermore, if we want to achieve indistinguishability of blinded KEM. 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)**.** *Let* BKEM *be a homomorphic based BKEM designed as in Definition 10 from a 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*

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

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 5 (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

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

Game 1

We consider a modified game which is the same as Game 0 expect 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 the adversary $\mathcal{A}$ queries for 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 the adversary $\mathcal{A}$ 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 output in Game 0 is $(\epsilon_1 + \epsilon_2)$ statically close to the modified values output in Game 1.

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 0, 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 1. 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.

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

Note that in Equation 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, in the Equation 2 holds because the distributions Z and Y over the set S are equal.

For r samples we get

$$\Big|\Pr[E_1] - \Pr[E_0]\Big| \leq r(\epsilon_1 + \epsilon_2).$$

Step 2 We claim that there exists an adversary $\mathcal{B}$ against IND-CPA security of HE such that

$$2\Big|\Pr[E_1] - \frac{1}{2}\Big| = \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{HE}}(\mathcal{B}).$$

We construct a reduction $\mathcal{B}$ that plays the IND-CPA game by running $\mathcal{A}$, it simulates the responses of Game 1 to $\mathcal{A}$ as follows.

1. $\mathcal{B}$ flips a coin $b \xleftarrow{\$} \{0,1\}$.
2. $\mathcal{B}$ queries the 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 the Blind and Decap algorithms 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 $\mathcal{A}$ sends back a guess b', $\mathcal{B}$ sends b to the challenger if $b' = 1$ and $1 - b$ if $b' = 0$.

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 1 when the output of the key is a real key. Otherwise ($\mathcal{B}$ interacts with $\mathbf{Exp}^{\mathsf{IND\text{-}CPA}}_{\mathsf{HE}}(\mathcal{B})$), k_b is a random key to $\mathcal{A}$ and $\mathcal{B}$ perfectly simulate the inputs of $\mathcal{A}$ in Game 1 when the output of the key is a random key.

□

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 indistinguishibility 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 specific homomorphic-based BKEM constructions, based on Gentry's homomorphic encryption scheme (see Section 2.3) and the NTRU variant by Stehlé and Steinfeld (see Section 2.4). We show that our BKEM schemes are post-quantum secure, by Theorem 5, as long as the underlying HE schemes are post-quantum secure [24, 37, 41].

5.1 Two Homomorphic-based BKEM

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. Let L be any full-rank n-dimensional lattice, for any $\epsilon \in (0,1)$, $s \geq \eta_\epsilon(L)$, and $r \geq \frac{6\pi sn}{\epsilon}$. The abstract construction of HE-BKEM is in Figure 7. Suppose HE-BKEM has ϵ_2–blinded blinded keys, a detailed description of these designs follows in Section 5.2.

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

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

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

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

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

Figure 7: 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 the message space of the HE scheme 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 $\mathcal{K}_R = \{d_0 + d_1 x + \cdots + d_{n-1}x^{n-1} \mid d_i \in \mathbb{Z}_{\lfloor q/2 \rfloor}\}$. Notice that 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 $b_i = c_i - a_i \in \mathbb{Z}_{\lfloor q/2 \rfloor}$. In other words, 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 $i \in \{0, \ldots, n-1\}$

$$\begin{aligned}&\mathbf{Pr}[a_i + b_i \in \{\lfloor\sqrt{q/2}\rfloor, \ldots, \lfloor q/2 \rfloor\} \mid a_i \xleftarrow{\$} \mathbb{Z}_{\lfloor\sqrt{q/2}\rfloor}, b_i \xleftarrow{\$} \mathbb{Z}_{\lfloor q/2 \rfloor}] \\ &\quad = 1 - \frac{\lfloor\sqrt{q/2}\rfloor - 1}{\lfloor q/2 \rfloor}.\end{aligned}$$

Hence, the probability of a blinded key locates in the ϵ-blinded blinded set is

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

Using this construction we can have a HE-BKEM with ϵ-blinded blinded keys for $\epsilon = n/\lfloor\sqrt{q/2}\rfloor$. 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 we have

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

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

Remark 2. *Both of these constructions can be applied to our* HE-BKEM *schemes.*

5.3 Construction of fresh-looking blinded encapsulations

We claim that the above constructed HE-BKEM has 2ϵ-blinded blinded encapsulations. The idea is to take the small constant ciphertext and add a ciphertext with big errors and the resulting ciphertext should look like the big error ciphertext. The details are showed in the following lemma.

Lemma 4. *Let* HE-BKEM *be a homomorphic based BKEM with the underlying homomorphic encryption scheme, described in Section 2.3*

or Section 2.4. Let ek be any encapsulation key, recall that the encryption algorithm $\mathsf{Enc}_{\mathsf{HE}}(ek, s, \cdot)$ *uses the discrete Gaussian distribution* $D_{L,s,\mathbf{0}}$ *as the error distribution. Suppose* $C_0 = \mathsf{Enc}_{\mathsf{HE}}(ek, s, k_0)$ *is an encapsulation of the underlying file encryption key* k_0*. For any* $\epsilon \in (0,1)$*,* $s \geq \eta_\epsilon(L)$*, and* $r \geq \frac{s\sqrt{n}}{\epsilon}$ *the statistical distance of the following distributions is at most* 2ϵ

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

which means HE-BKEM *has* 2ϵ*-blinded blinded encapsulation.*

Proof. From Lemma 1 we have $\mathbf{Pr}[\mathbf{x} \notin \mathcal{B}(s\sqrt{n}) \mid \mathbf{x} \leftarrow D_{L,s,\mathbf{0}}] \leq \epsilon$, which means the size of the error output by the distribution $D_{L,s,\mathbf{0}}$ is upper bounded by $s\sqrt{n}$ expect for negligible probability ϵ.

For Gentry's scheme, suppose $C_0 = k_0 + e_0$, where $e_0 \leftarrow D_{L,s,\mathbf{0}}$. From Lemma 2 we know that for a small error $\|e_0\| \leq s\sqrt{n}$ and big randomness $r \geq \|e_0\|/\epsilon$ the statistical distance between $D_{L,r,\mathbf{0}}$ and $D_{L,r,\mathbf{e_0}}$ is at most ϵ. So the following approximation holds

$$\begin{aligned} C_0 \oplus_1 \mathsf{Enc}_{\mathsf{HE}}(ek, r, k') &= k_0 + e_0 + k' + D_{L,r,\mathbf{0}} \\ &\approx k_0 + k' + D_{L,r,\mathbf{0}} \\ &= \mathsf{Enc}_{\mathsf{HE}}(ek, r, k_0 \oplus_2 k'). \end{aligned}$$

The above result can be easily adapted to NTRU encryption scheme. □

5.4 Indistinguishability of GHE-BKEM

The following result says GHE-BKEM is an IND-secure BKEM with post-quantum security.

Corollary 1. *Let* GHE-BKEM *be a homomorphic-based BKEM described in Section 5.1. For negligible* $\epsilon_1 = \epsilon, \epsilon_2$*, choose parameters as in Lemma 4, Theorem 1 and Theorem 2. Suppose* GHE-BKEM *has* ϵ_2*-blinded blinded keys. Then* GHE-BKEM *has* $2\epsilon_1$*-blind blinded encapsulation. Furthermore, 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 we know GHE-BKEM has $2\epsilon_1$-blinded blinded encapsulation.

Theorem 5 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.

□

5.5 Indistinguishability of NTRU-BKEM

The following result says NTRU-BKEM is an IND-secure BKEM with post-quantum security.

Corollary 2. *Let* NTRU-BKEM *be a homomorphic based BKEM constructed in Section 5.1. For negligible $\epsilon_1 = \epsilon, \epsilon_2$, choose parameters as in Lemma 4, Lemma 3, and Theorem 4. Suppose* NTRU-BKEM *has ϵ_2-blinded blinded keys. Then* NTRU-BKEM *has $2\epsilon_1$-blinded blinded encapsulation. Furthermore, if there is an algorithm that breaks the indistinguishability of* NTRU-BKEM*, then there exists a quantum algorithm that solves $O(\sqrt{n}/\alpha)$-approximate* SIVP *(or* SVP*) on ideal lattices.*

Proof. By Lemma 4 we know NTRU-BKEM has $2\epsilon_1$-blinded blinded encapsulation.

Theorem 5 states 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. □

References

[1] Masayuki Abe, Rosario Gennaro, Kaoru Kurosawa, and Victor Shoup. Tag-KEM/DEM: A New Framework for Hybrid Encryp-

tion and A New Analysis of Kurosawa-Desmedt KEM. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, pages 128–146, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[2] Martin Albrecht, Shi Bai, and Léo Ducas. A Subfield Lattice Attack on Overstretched NTRU Assumptions. In *Proceedings, Part I, of the 36th Annual International Cryptology Conference on Advances in Cryptology — CRYPTO 2016 - Volume 9814*, pages 153–178, Berlin, Heidelberg, 2016. Springer-Verlag.

[3] 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.

[4] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum Key Exchange—A New Hope. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 327–343, Austin, TX, August 2016. USENIX Association.

[5] 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.

[6] Nimrod Aviram, Kai Gellert, and Tibor Jager. Session Resumption Protocols and Efficient Forward Security for TLS 1.3 0-RTT. 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 II*, volume 11477 of *Lecture Notes in Computer Science*, pages 117–150. Springer, 2019.

[7] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU Prime: reducing attack surface at low cost. `https://ntruprime.cr.yp.to/papers.html`.

[8] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange from LWE. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1006–1018. ACM, 2016.

[9] Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme. In *Proceedings of the 14th IMA International Conference on Cryptography and Coding - Volume 8308*, IMACC 2013, pages 45–64, Berlin, Heidelberg, 2013. Springer-Verlag.

[10] Colin Boyd, Gareth T. Davies, Kristian Gjøsteen, and Yao Jiang. Offline Assisted Group Key Exchange. In Liqun Chen, Mark Manulis, and Steve Schneider, editors, *Information Security - 21st International Conference, ISC 2018, Guildford, UK, September 9-12, 2018, Proceedings*, volume 11060 of *Lecture Notes in Computer Science*, pages 268–285. Springer, 2018.

[11] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) Fully Homomorphic Encryption Without Bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, New York, NY, USA, 2012. ACM.

[12] 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.

[13] 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.

[14] Jung Hee Cheon, Kyoohyung Han, Jinsu Kim, Changmin Lee, and Yongha Son. A Practical Post-Quantum Public-Key Cryptosystem Based on \textsf spLWE. In Seokhie Hong and Jong Hwan Park, editors, *Information Security and Cryptology - ICISC 2016 - 19th International Conference, Seoul, South Korea, November 30 - December 2, 2016, Revised Selected Papers*, volume 10157 of *Lecture Notes in Computer Science*, pages 51–74, 2016.

[15] Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. 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.

[16] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic Encryption for Arithmetic of Approximate Numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437, Cham, 2017. Springer International Publishing.

[17] Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. On Ends-to-Ends Encryption: Asynchronous Group Messaging with Strong Security Guarantees. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 1802–1819. ACM, 2018.

[18] Ronald Cramer and Victor Shoup. Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive

Chosen Ciphertext Attack. Cryptology ePrint Archive, Report 2001/108, 2001. https://eprint.iacr.org/2001/108.

[19] 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, January 2004.

[20] 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*, pages 282–305, Cham, 2018. Springer International Publishing.

[21] Alexander W. Dent. A Designer's Guide to KEMs. In Kenneth G. Paterson, editor, *Cryptography and Coding*, pages 133–151, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[22] 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.

[23] Junfeng Fan and Frederik Vercauteren. Somewhat Practical Fully Homomorphic Encryption. Cryptology ePrint Archive, Report 2012/144, 2012. https://eprint.iacr.org/2012/144.

[24] Craig Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford, CA, USA, 2009. AAI3382729.

[25] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for Hard Lattices and New Cryptographic Constructions.

In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 197–206, New York, NY, USA, 2008. ACM.

[26] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 75–92, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[27] Matthew D. Green and Ian Miers. Forward Secure Asynchronous Messaging from Puncturable Encryption. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 305–320. IEEE Computer Society, 2015.

[28] Felix Günther, Britta Hale, Tibor Jager, and Sebastian Lauer. 0-RTT Key Exchange with Full Forward Secrecy. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III*, volume 10212 of *Lecture Notes in Computer Science*, pages 519–548, 2017.

[29] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe Buhler, editor, *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.

[30] Dennis Hofheinz and Eike Kiltz. Secure Hybrid Encryption from Weakened Key Encapsulation. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, pages 553–571, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[31] Andreas Hülsing, Joost Rijneveld, John Schanck, and Peter Schwabe. High-speed key encapsulation from NTRU. In *Inter-*

national Conference on Cryptographic Hardware and Embedded Systems (CHES), pages 232–252. Springer, 2017.

[32] Kaoru Kurosawa and Yvo Desmedt. A New Paradigm of Hybrid Encryption Scheme. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, pages 426–442, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[33] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 1219–1234, New York, NY, USA, 2012. ACM.

[34] 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.

[35] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On Ideal Lattices and Learning with Errors over Rings. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010.

[36] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A Toolkit for Ring-LWE Cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 35–54. Springer, 2013.

[37] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On Ideal Lattices and Learning with Errors over Rings. *J. ACM*, 60(6):43:1–43:35, November 2013.

[38] Moxie Marlinspike and Trevor Perrin. The X3DH Key Agreement Protocol. https://signal.org/docs/specifications/x3dh/, November 2016.

[39] Daniele Micciancio and Oded Regev. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM J. Comput.*, 37(1):267–302, April 2007.

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

[41] Damien Stehlé and Ron Steinfeld. Making NTRU As Secure As Worst-case Problems over Ideal Lattices. In *Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT'11, pages 27–47, Berlin, Heidelberg, 2011. Springer-Verlag.

[42] The Messaging Layer Security (MLS) Protocol. Internet draft, in progress. https://datatracker.ietf.org/wg/mls/about. Accessed: 2019-11-25.

Paper IV

Attacks on the Basic cMix Design: On the Necessity of Commitments and Randomized Partial Checking

Herman Galteland, Stig F. Mjølsnes and Ruxandra F. Olimid

Published in Paradigms in Cryptology – Mycrypt 2016.
Malicious and Exploratory Cryptology

Attacks on the Basic cMix Design: On the Necessity of Commitments and Randomized Partial Checking*

Herman Galteland[1], Stig F. Mjølsnes[2], and Ruxandra F. Olimid[2]

[1]Department of Mathematical Sciences,
NTNU – Norwegian University of Science and Technology,
herman.galteland@ntnu.no

[2]Department of Information Security and Communication Technology,
NTNU – Norwegian University of Science and Technology
{sfm, ruxandra.olimid}@ntnu.no

Abstract

The cMix scheme was proposed by Chaum et al. in 2016 as the first practical set of cryptographic protocols that offer sender-recipient unlinkability at scale. The claim was that the cMix is secure unless all nodes collude. We argue that their assertion does not hold for the basic description of cMix, and we sustain our statement by two different types of attacks: a tagging attack and an insider attack. For each one, we discuss the settings that make the attack feasible, and then possible countermeasures. By this, we highlight the necessity of implementing additional commitments or mechanisms that have only been mentioned as additional features.

Keywords. cryptographic protocols, sender-recipient unlinkability, anonymity, mixnets, attacks

*The final publication is available at Springer via http://dx.doi.org/978-3-319-61273-7\protect_22

1 Introduction

1.1 cMix

The cMix protocol by Chaum et al. [6] is an improved mixing network [5] which aims to provide an anonymous communication tool for its users at large scales. The mixing should be such that no one is able to relate an output message to a user input message, that is, no one is able to link a sender with a recipient. An important advantage over its predecessors is that cMix performs expensive computations (like public key encryption) during a precomputation phase, keeping the real-time phase, which is in charge with actual message delivery, fast. The protocol is a part of a larger system, called Privategrity, but its authors describe cMix independently.

The authors of Ref. [6] claim that cMix is the first practical system that provides sender-recipient unlinkability, unless all nodes collude. We argue that their assertion does not hold for the basic description of the protocol (as given in [6, Section 4]) and we sustain our statement by two different types of attacks. Each of them has its own effect on the design of the original protocol. By this, we want to highlight *the necessity* of using additional commitment mechanisms, whereas Ref. [6] mentions this as additional features.

1.2 Related Work

The cMix system is designed to be resistant to most of the usual mix network attacks. This paper focuses on the cryptanalysis of cMix, and Subsection 2.2 introduces in detail the adversarial model from [6]. We present here a very brief survey of proposed general attacks on anonymous overlay networks.

Tagging attacks are a potential threat to all mix networks [14]. An adversary can put an identifier tag on an input message to the mix network and attempt to recognize the tag in the output messages. If successful, the adversary can break the anonymity of a specific sender. We show in Section 3 that cMix is vulnerable to a tagging attack.

Replay attacks are attacks in which an adversary retransmits a valid message several times, making it possible to analyze the outgoing

traffic [3]. We do not analyze replay attacks against cMix system, as they are eliminated by the adversarial model (see Subsection 2.2).

Intersection attacks and *statistical disclosure attacks* use information acquired by observing mix networks where the users can freely choose the mix node for their messages (free mix nodes) [3, 9, 10]. In such systems different batches can be distinguished since they come from different mix nodes. If a sender use the same mix nodes for every message then the adversary can separate the routes by analyzing the network flow.

Traffic analysis attacks is a family of attacks that observes the network traffic in order to deduce informational patterns in communication and targets connection-based systems. Unlike message-based systems like cMix, connection-based systems use free mix nodes that do not batch and permute messages. By *counting packets* [19] and *timing communication* [8] the adversary is able to distinguish between different paths in the (free) mix network. *Contextual attacks* [18] (or *traffic confirmation attacks* [20], or *intersection attacks* [2]) analyze the traffic when specific users and recipients use a protocol, their communication pattern, and how many messages they send and receive.

The authors of cMix recognize that their proposal is potentially vulnerable to attacks that make anonymous systems fail, like the broadband intersection attacks, contextual attacks, or DoS (Denial of Service) [6].

1.3 Results

We focus on the security analysis of the basic cMix description as described in [6, Section 4], and show that it is susceptible to two attacks, which differ by action type.

Tagging Attack. The cMix paper [6] introduces commitments [4] to overcome tagging attacks. The paper states that *"tagging attacks do not work before the exit node"*, and *"if a tagging attack is detected, at least the last node should be removed from the cascade"* [6, Section 4.3]. Therefore the authors might be aware of a possible attack that can be performed by the exit node. However, they do not consider any prevention for this. We introduce a simple tagging attack launched

by the exit node. Although a prevention mechanism is immediate (by adding an extra commitment) we consider it for completeness, as an example of a possible tagging attack against the system. In personal communications, the authors of cMix acknowledged that the actual design of the system adds the additional commitment we refer to as a countermeasure [11].

Insider Attack. The cMix paper [6] claims that attacks are unsuccessful unless all nodes collude. We contradict this by showing that the last node can break the unlinkability, essentially by creating a mix network consisting of itself only. The attack will succeed by the last node deviating from the protocol rules and choose its own output. We argue that this attack remains undetected in the original version of cMix, and becomes detectable only if additional checks like *Randomized Partial Checking* (RPC, see Subsection 2.3) are considered (suggested by the authors of [6] as a special feature). We show the necessity of using randomized partial checking. However, an inappropriate use of RPC could allow a coalition of nodes (all except one) to link a large fraction of the senders to their recipients.

1.4 Outline

Section 2 describes the cMix scheme and presents the adversarial model. The two following sections contain our results: Section 3 describes a simple tag attack similar to the tag attack described in the original cMix paper [6]. Section 4 presents the insider attack where the adversary controls the last node and makes the overall mixing process independent of the preceding nodes. Section 6 concludes and indicates possible future research directions.

2 Preliminaries

2.1 cMix Description

Figure 2 describes the cMix protocol from [6]. We ignore the return steps, since they are irrelevant for our attacks. Note that this does not restrict the applicability of our results, since the same permutation

U_j	user j
$\mathbf{M}$	a batch of β messages $\mathbf{M} = (M_1, \ldots, M_\beta)$, each M_i sent by a distinct user
N_i	node i from the set of n mix nodes $\{N_1, \ldots, N_n\}$
e_i	the share of node N_i of the secret key e
d	the public key of the system, where $d = \prod_{i=1}^{n} g^{e_i}$
$\mathcal{E}(\cdot)$	a multi-party group-homomorphic encryption under the system public key d
π_i	a random permutation on a batch, applied by node N_i
Π_i	the composed permutation performed by all nodes from N_1 to N_i
$k_{i,j}$	the derived secret key shared between node N_i and the sending user of slot j
$\mathbf{k}_i$	the vector of derived secret keys shared between node N_i and all users in a batch, $\mathbf{k}_i = (k_{i,1}, \ldots, k_{i,\beta})$
K_j	the product of all shared keys for the sending user of slot j, $K_j = \prod_{i=1}^{n} k_{i,j}$
$\mathbf{r}_i, \mathbf{s}_i$	random values of node N_i for the batch, where $\mathbf{r}_i = (r_{i,1}, \ldots, r_{i,\beta})$ and $\mathbf{s}_i = (s_{i,1}, \ldots, s_{i,\beta})$, resp.
$\mathbf{R}_i, \mathbf{S}_i$	the direct product of the first i values, $\mathbf{R}_i = \prod_{j=1}^{i} \mathbf{r}_j$ and $\mathbf{S}_i = \prod_{j=1}^{i} \mathbf{s}_j$, resp.

Figure 1: Notations

is used for both forward and return paths. Once the permutation is disclosed both directions of communication are compromised.

cMix has two phases: a *precomputation phase* and a *real-time phase*. By design, the heavy public key computations are performed in the precomputation phase, which can be performed on separate hardware (for each node). Since the precomputation phase does not require any input from the users it can be performed offline and while a batch is being filled up with messages.

The scheme consists of a sequence of n mix nodes that process β messages at a time (a *batch* of messages); made simple, each node performs a permutation on the input and blinds the output by multiplying it with a random value. The last node N_n makes an exception, as it usually behaves differently from the other nodes (see Figure 2).

Besides the last node there is another entity with a special role in the system – the *network handler* – that interacts both with the users and the whole set of nodes. The network handler receives messages from the users and arranges them into batches; once a batch is full it is sent to the first node in the mix network. After the last node performs its mixing it sends the batch back to the network handler, which can then forward or broadcast the messages to the destination. The mixing should be such that no one is able to relate an output message to a user input message, that is, no one is able to link a

Precomputation Phase

- Step 1 (preprocessing). Each node N_i, $1 \leq i \leq n$, selects a random $\mathbf{r}_i$, computes the encryption $\mathcal{E}(\mathbf{r}_i^{-1})$ and sends it to the network handler. The network handler computes the product of all the received values, produces $\mathcal{E}(\mathbf{R}_n^{-1}) = \prod_{i=1}^{n} \mathcal{E}(\mathbf{r}_i^{-1})$ and sends it to the first node.
- Step 2 (mixing). Each node N_i, $1 \leq i \leq n$, computes $\pi_i(\mathcal{E}(\Pi_{i-1}(\mathbf{R}_n^{-1}) \times \mathbf{S}_{i-1}^{-1})) \times \mathcal{E}(\mathbf{s}_i^{-1})$, where Π_0 is the identity permutation and $\mathbf{S}_0^{-1} = \mathbf{1}$. The last node sends the vector of random components (i.e. the first component) of the ciphertext $(\mathbf{x}, \mathbf{c}) = \mathcal{E}((\Pi_n(\mathbf{R}_n) \times \mathbf{S}_n)^{-1})$ to the other nodes and stores the vector of message components (i.e. the second component) locally for the real-time phase.
- Step 3 (postprocessing). Using the random component $\mathbf{x}$, each node N_i, $1 \leq i \leq n$, computes its individual decryption share for $(\mathbf{x}, \mathbf{c})$ as $\mathcal{D}_i(\mathbf{x}) = \mathbf{x}^{-e_i}$, stores it locally to use in the real-time phase and publicly commits to it.

Real-Time Phase

- Step 0. Each user constructs its message MK_j^{-1} (for slot j) by multiplying the message M_j with the inverse of the key K_j and it sends it to the network handler, which collects all messages and combines them to get a vector $\mathbf{M} \times \mathbf{K}^{-1}$.
- Step 1 (preprocessing). Each node N_i, $1 \leq i \leq n$, sends $\mathbf{k}_i \times \mathbf{r}_i$ to the network handler, which uses them to compute $\mathbf{M} \times \mathbf{R}_n = \mathbf{M} \times \mathbf{K}^{-1} \times \prod_{i=1}^{n} \mathbf{k}_i \times \mathbf{r}_i$ and sends the result to N_1.
- Step 2 (mixing). Each node N_i, $1 \leq i \leq n$, computes $\pi_i(\Pi_{i-1}(\mathbf{M} \times \mathbf{R}_n) \times \mathbf{S}_{i-1}) \times \mathbf{s}_i$, where Π_0 is the identity permutation and $\mathbf{S}_0 = \mathbf{1}$. The last node N_n sends a commitment to its message $\Pi_n(\mathbf{M} \times \mathbf{R}_n) \times \mathbf{S}_n$ to every other node.
- Step 3 (postprocessing). Each node N_i, $1 \leq i \leq n-1$, sends its precomputed decryption share for $(\mathbf{x}, \mathbf{c}) = \mathcal{E}((\Pi_n(\mathbf{R}_n) \times \mathbf{S}_n)^{-1})$ to the network handler, while the last node N_n sends its decryption share multiplied by the value in the previous step and the message component: $\Pi_n(\mathbf{M} \times \mathbf{R}_n) \times \mathbf{S}_n \times \mathcal{D}_n(\mathbf{x}) \times \mathbf{c}$. Finally, the network handler retrieves the permuted message as $\Pi_n(\mathbf{M}) = \Pi_n(\mathbf{M} \times \mathbf{R}_n) \times \mathbf{S}_n \times \prod_{i=1}^{n} \mathcal{D}_i(\mathbf{x}) \times \mathbf{c}$.

Figure 2: The cMix Protocol (forward path) [6]

sender with a recipient.

Before using the system each sender U_j must establish a private symmetric key with each of the nodes N_i, which they use as a seed in a pseudorandom generator to derive the secret keys $k_{i,j}$. To blind a message M_j before it is sent to the network handler, user U_j multiplies M_j with a key composed by the derived keys shared with each of the nodes $K_j = \prod_{i=1}^{n} k_{i,j}$. The network handler arranges messages into a batch and sends it through the mix network. Each node applies its permutation to the batch and the last node sends it back to the network handler. The output is a permuted batch of messages.

During the mixing step of the precomputation phase each node performs encryption under a public key of the system; the related private key is split across all nodes in the network. The encryption scheme suggested by the authors of [6] is the multi-party group-homomorphic encryption based on ElGamal [12] described by Benaloh [1]. Moreover, all computations of the protocol are performed in a prime order cyclic group G that satisfies the decisional Diffie-Hellman security assumption. We denote by G^* the set of nonidentity elements in G.

Refer to Figure 2 for the detailed self-contained description of the cMix process, using the notation defined in Figure 1.

2.2 Adversarial Model

The adversarial model in [6] assumes authenticated channels among the mix nodes and between the network handler and each mix node. This implies that the adversary can read, forward, and delete messages, but not modify, inject, or replay messages without detection. The adversary can compromise the users (up to all except two), and the mix nodes (up to all except one). Compromised nodes can behave malicious but cautious, since the attacker aims to remain undetected. Within this attacker model, the authors of cMix claim that the output is unlinkable to the input, even if the adversary knows the set of senders and the set of recipients for every batch of messages.

The security analysis in the Appendix A of the cMix paper assumes secure authenticated channels for which the adversary cannot read the content, only the length of a message. All our attacks hold under these stronger security assumptions.

2.3 Features and Extensions

The cMix paper [6] dedicates a section to special features and extensions of the system. It shortly discusses the utility of adding RPC (*Randomized Partial Checking*) to cMix, an integrity check mechanism introduced by Jacobsson, Juels and Rivest [15], and further analyzed and developed in Refs. [16, 17]. The usage of RPC in the cMix system is that each node commits to a randomly chosen per-

- **Goal:** Tag a message M_j belonging to user U_j and recognize it in the permuted batch of messages, linking the sender U_j to its recipient.
- Step 1. The corrupted node N_n creates a tag vector $\mathbf{t}$ which consists of $\beta - 1$ ones and one tag $t \in G^*$ in slot j (i.e. $\mathbf{t} = (1, \ldots, 1, t, 1, \ldots, 1)$), computes $\mathbf{k}_n \times \mathbf{r}_n \times \mathbf{t}$ and sends it to the network handler (Real-time Phase.Step 1).
- Step 2. The network handler sends the set of all decryption shares $\{\mathcal{D}_i(\mathbf{x}) | 1 \leq i < n\}$ to the last node (Real-time Phase.Step 3). Node N_n can retrieve the permuted messages as $\Pi_n(\mathbf{M} \times \mathbf{t}) = \Pi_n(\mathbf{M} \times \mathbf{R}_n \times \mathbf{t}) \times \mathbf{S}_n \times \prod_{i=1}^{n} \mathcal{D}_i(\mathbf{x}) \times \mathbf{c}$ and recognize the tagged message in slot j'.
- Step 3. The corrupted node N_n creates the inverse tag vector $\mathbf{t}^{-1}$, which consists of $\beta - 1$ ones and one tag $t^{-1} \in G^*$ in slot j', computes $\mathbf{c}' = \mathbf{c} \times \mathbf{t}^{-1}$, and sends $\Pi_n(\mathbf{M} \times \mathbf{R}_n) \times \mathbf{S}_n \times \mathcal{D}_n(\mathbf{x}) \times \mathbf{c}'$ to the network handler.

Figure 3: The Tagging Attack

mutation, publishes its input and output, and validates that it has followed the protocol correctly by revealing a (large) fraction of its secret input/output pairs, where these pairs are selected by the other nodes (or by a random oracle). The cMix system protects the user's privacy by putting nodes in pairs, such that each node belongs to only one pair. Nodes in a pair reveal their secret information such that none of the messages can be followed from the input of the first node to the output of the second node.

3 The Tagging Attack

Our first attack is similar to the tag attack described in the cMix paper [6], but it uses a different value to remove the tag. During the precomputation phase the nodes compute the value $(\mathbf{x}, \mathbf{c}) = \mathcal{E}((\Pi_n(\mathbf{R}_n) \times \mathbf{S}_n)^{-1})$, where the last node stores the vector of message components $\mathbf{c}$ locally and sends the vector of random components $\mathbf{x}$ to all other nodes. Each node computes its decryption share using $\mathbf{x}$ and commits to this value. Note that it is uncertain whether $\mathbf{c}$ is being committed to or not in the description of the basic cMix protocol.

The authors of cMix introduce commitments to detect potential tagging attacks and to expose any attempt of using the decryption shares to remove the tag. However, the commitments are independent of $\mathbf{c}$, so it is possible to perform a similar attack which uses $\mathbf{c}$ instead of $\mathcal{D}_n(\mathbf{x})$ to remove the tag. The downside is that the adversary needs

to corrupt the last node (which has access to $\mathbf{c}$) and the network handler (under the assumption of secure authorized channels). Figure 3 describes the tag attack.

For the tag attack to be successful we need to assume that it is possible to recognize valid messages in the output. To tag a message M_j the last node creates a tag vector $\mathbf{t} = (1, \ldots, t, \ldots, 1)$, where t is in position j, multiplies it with the keys and random values $\mathbf{k}_n \times \mathbf{r}_n \times \mathbf{t}$, and sends the result to the network handler. The tag goes through the mixnet attached to message M_j and arrives at the last node as $\Pi_{n-1}(\mathbf{M} \times \mathbf{R}_n \times \mathbf{t}) \times \mathbf{S}_{n-1}$. The last node will permute the batch and do the computations according to the protocol, and publish its commitment to the value $\Pi_n(\mathbf{M} \times \mathbf{R}_n \times \mathbf{t}) \times \mathbf{S}_n$. This triggers all other nodes to send their decryption share to the network handler, which forwards them to N_n. The last node can then retrieve the batch of permuted messages and find the invalid message $M_j t$ in slot j' of the permuted batch. The last node creates the inverse tag $\mathbf{t}^{-1}$, which has t^{-1} in slot j', and replaces the message components with the altered value $\mathbf{c}' = \mathbf{c} \times \mathbf{t}^{-1}$. The network handler computes

$$\Pi_n(\mathbf{M} \times \mathbf{R}_n \times \mathbf{t}) \times \mathbf{S}_n \times \mathbf{c}' \times \prod_{i=1}^{n} \mathcal{D}_i(\mathbf{x}) =$$
$$\Pi_n(\mathbf{M} \times \mathbf{R}_n \times \mathbf{t}) \times \mathbf{S}_n \times (\Pi_n(\mathbf{R}_n) \times \mathbf{S}_n)^{-1} \times \mathbf{t}^{-1} =$$
$$\Pi_n(\mathbf{M} \times \mathbf{t}) \times \mathbf{t}^{-1} = \Pi_n(\mathbf{M})$$

and delivers the permuted batch as normal. That is, the adversary has successfully linked a sender with a recipient without being detected.

To make this attack detectable, the last node should publish a commitment to the vector of message components $\mathbf{c}$ in the Precomputation Phase.Step 3, or the system should implement RPC as an integrity check mechanism. Although prevention can be simply achieved by natural solutions like the ones mentioned, we introduce the attack for completeness; it stands as an example of tagging attack performed by the last node, a type of attack the authors of cMix seem to be aware of (see [6], Section 4.2: *"tagging attacks do not work before the exit node"* and *"if a tagging attack is detected, at least the last node should be removed from the cascade"*).

At the time of writing, the authors of cMix acknowledged that the actual design of the system implements the countermeasure we refer to and commits to the vector of message components $\mathbf{c}$, as explained above [11].

4 The Insider Attack

Our second attack allows the last node to ignore all permutations introduced by the previous nodes and perform the overall mixing process by itself. Hence, the output of the real-time phase will be a batch of messages permuted with a known permutation making it easy to link all senders and recipients. To succeed, the adversary needs to corrupt the last node (which controls the output of the mixing process) and the network handler (which knows the content of the values $\mathcal{E}(\mathbf{R}_n^{-1})$ and $\mathbf{M} \times \mathbf{R}_n$, under the assumption of secure authenticated channels). Figure 4 describes the insider attack.

During Precomputation Phase.Step 1 the corrupted network handler computes and sends $\mathcal{E}(\mathbf{R}_n^{-1})$ to the first and the last nodes. The honest nodes operates as normal, where the last, dishonest, node discards the input it receives from the previous node and chooses its own output. The last node draws a random vector $\mathbf{A} = (A_1, \ldots, A_\beta)$, encrypts the inverted values, $\mathcal{E}(\mathbf{A}^{-1})$, and computes $\pi_n(\mathcal{E}(\mathbf{R}_n^{-1}) \times \mathcal{E}(\mathbf{A}^{-1})) = \pi_n(\mathcal{E}(\mathbf{R}_n^{-1} \times \mathbf{A}^{-1}))$. The last node publishes the random components, that is $\mathbf{x}$, of $\pi_n(\mathcal{E}(\mathbf{R}_n^{-1} \times \mathbf{A}^{-1})) = (\mathbf{x}, \mathbf{c})$ to the other nodes such that they can prepare their decryption shares.

In Real-Time Phase.Step 1 the network handler sends $\mathbf{M} \times \mathbf{R}_n$ to the first and the last node. In the mixing step the last node discards what it receives from the previous node, selects its output $\pi_n(\mathbf{M} \times \mathbf{R}_n \times \mathbf{A})$, commits to this batch of messages, and sends $\pi_n(\mathbf{M} \times \mathbf{R}_n \times \mathbf{A}) \times \mathbf{c} \times \mathcal{D}_n(\mathbf{x})$ to the network handler. As the network handler receives the decryption shares from the other nodes it can retrieve the permuted messages and forward them to the receivers:

$$\pi_n(\mathbf{M}\times\mathbf{R}_n\times\mathbf{A})\times\mathbf{c}\times\prod_{i=1}^{n}\mathcal{D}_i(\mathbf{x}) = \pi_n(\mathbf{M}\times\mathbf{R}_n\times\mathbf{A})\times\pi_n(\mathbf{R}_n^{-1}\times\mathbf{A}^{-1})$$
$$= \pi_n(\mathbf{M}).$$

- **Goal:** Perform the mixing process with only the last node using only a known permutation to permute the batch of messages.
- Step 1. The network handler computes and sends $\mathcal{E}(\mathbf{R}_n^{-1})$ to the first and last node (Precomputation Phase.Step 1). The last node discards the input it is given form the previous node and publishes the component of random elements of $\pi_n(\mathcal{E}(\mathbf{R}_n^{-1} \times \mathbf{A}^{-1}))$, for a random and invertible $\mathbf{A}$ (Precomputation Phase.Step 3).
- Step 2. The network handler computes and sends $\mathbf{M} \times \mathbf{R}_n$ to the first and last node (Real-Time Phase.Step 1). The last node discards the input it is given form the previous node, publishes a commitment to $\pi_n(\mathbf{M} \times \mathbf{R}_n \times \mathbf{A})$, and sends $\pi_n(\mathbf{M} \times \mathbf{R}_n \times \mathbf{A}) \times \mathbf{c} \times \mathcal{D}_n(\mathbf{x})$ to the network handler (Real-Time Phase.Step 3).
- Step 3. The network handler retrieves the permuted batch of messages as $\pi_n(\mathbf{M}) = \pi_n(\mathbf{M} \times \mathbf{R}_n \times \mathbf{A}) \times \pi_n(\mathbf{R}_n^{-1} \times \mathbf{A}^{-1})$ and publishes it. The adversary can recover $\mathbf{M}$ by applying ${\pi_n}^{-1}$.

Figure 4: The Insider Attack

Note that the output batch is only permuted with the permutation π_n, which is known to the last node. Hence, the adversary can easily deanonymize all of the senders by applying ${\pi_n}^{-1}$ to the output.

The RPC mechanism ensures with high probability that each node follows its instructions, hence, this will prevent the last node from deviating from the protocol. Since our insider attack changes the entire batch, RPC will detect the attack. This shows the necessity of implementing RPC in the cMix protocol.

Notes on the RPC mechanism.

The RPC mechanism makes the nodes reveal a (large) fraction of their secret information, which could break the anonymity of the users [17]. As an example, let's assume that each node performs only one permutation and proves the correctness of its output for this permutation. Further assume that an adversary corrupts all except one, honest, node and therefore only needs the permutation from this node to deanonymize the users. When using RPC, the honest node would reveal information about its permutation. Hence, the adversary can easily break the anonymity for a substantial portion of the users using the information made public by the RPC mechanism.

Even in the scenario where there are two honest nodes that are paired, the adversary can get some information about the senders and receivers [17]. Nodes in a pair reveal information such that no

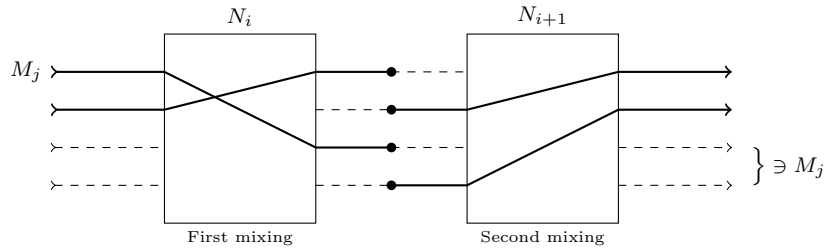


Figure 5: RPC: Two paired nodes revealing each separate half of their permutation. Continuous lines means information is revealed and dashed lines means information is not revealed

messages can be followed from the input of the first node to the output of the second node in a pair. This means that if a message, say M_j, is revealed by the first node, then it will not be revealed by the second node (see Figure 5). Given enough rounds of cMix, an adversary might eventually link senders and recipients that are frequently talking with each other. Therefore, two honest nodes (a single pair) are usually not enough to protect the anonymity of all users.

5 Rebuttal from the Authors of cMix

The authors of the original cMix paper [6] made the following rebuttal to the initial submission of this paper:

> [...] *Galteland, Mjølsnes, and Olimid propose a tagging attack and an insider attack against the cMix protocol, as described in the preliminary cMix eprint [6]. But security mechanisms specified in this preliminary cMix eprint prevent both attacks. In addition, alternative integrity mechanisms (e.g., trap messages) specified in the current cMix paper [7] provide additional ways to prevent these attacks. In particular, as presented in the preliminary cMix eprint, Random Partial Checking (RPC) [16] prevents both attacks.*[...]
>
> *The tagging attack does not work because RPS prevents*

> *it, as explained in the preliminary cMix eprint [6, Section 7.3]. In addition, cMix stops the attack by commitment: it commits to the value the Galteland et al. allege makes the tagging attack possible. Our system design and prototype implements this commitment, even though the original cMix eprint does not mention this detail. Before reading the paper by Galteland et al., we were aware of this attack and of similar ones. After coming across an earlier version of their paper [13], we contacted the authors to inform them that our design and implementation included commitments to prevent these types of attacks, which they do acknowledge.* [...]
>
> *Despite some interesting features, the insider attack does not work because RPC detects it, as Galteland et al. also acknowledge. The preliminary cMix eprint [6, Section 7.3] prescribes using RPC.* [...]
>
> *We disagree with their claim that we overlooked important details underlying these alleged attacks. The attacks proposed by Galteland et al. do not work: security mechanisms specified in the preliminary cMix eprint prevent both attacks.*

As a response to their remarks, both our attacks are valid under the basic protocol description given in [6, Section 4]. The commitment mechanism required to overcome the first attack is not used or referred to in the original paper, as acknowledged in the authors' response. Furthermore, the cMix paper describes RPC as an extension, therefore usage of RPC can hardly be understood as *necessary* [6]. We claim the necessity of RPC or an equivalent mechanism. RPC is not included in the formal analysis, hence it is left outside the security theorems and performance discussions. Whereas we find that RPC is crucial for the security of the system, and it might introduce a significant performance penalty. The response note informs us that proper security mechanisms protecting against the attacks we have presented are used in their prototype and explained in a new paper, but both of those are currently unavailable to us for inspection.

6 Conclusions

We demonstrate by examples that the cMix scheme, as it was initially defined in its basic settings, would allow linkability between senders and recipients, hence compromising the anonymity of the users. We describe the actions an adversary could follow to succeed for both types of attacks (the tagging attack and the insider attack). The attacks succeed in the secure authenticated channels settings, and under the assumption that the adversary can corrupt the network handler. This is a natural assumption that was also made by that the authors of cMix.

By discussing the attacks, we highlight the necessity of the use of commitments and the RPC integrity mechanisms, which have only been mentioned as additional features in cMix scheme, and where these mechanisms are not fully included in the security proofs. However, the authors of cMix have expressed that their demonstration software implements the commitment mechanism that prevents our tagging attack.

This paper is restricted to a theoretical exposure of some attacks against the cMix standalone set of cryptographic protocols. Future analysis work can include experimental activities for practical attacks on real-world cMix implementations. Of course, the scalability of performance, throughput, and latency are key issues. An enterprising theoretical work would be to analyze the cMix security within the context of the larger system Privategrity.

Acknowledgements.

Herman Galteland is funded by Nasjonal sikkerhetsmyndighet (NSM), `www.nsm.stat.no`.

References

[1] Josh Benaloh. Simple verifiable elections. In *Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop*, EVT'06, Berkeley, CA, USA, 2006. USENIX Association.

[2] Oliver Berthold and Heinrich Langos. Dummy traffic against long term intersection attacks. *Privacy Enhancing Technologies: Second International Workshop, PET 2002 San Francisco, CA, USA, April 14–15, 2002 Revised Papers*, pages 110–128, 2003.

[3] Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. The disadvantages of free mix routes and how to overcome them. *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability Berkeley, CA, USA, July 25–26, 2000 Proceedings*, pages 30–45, 2001.

[4] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, October 1988.

[5] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, February 1981.

[6] David Chaum, Debajyoti Das, Farid Javani, Aniket Kate, Anna Krasnova, Joeri de Ruiter, and Alan T. Sherman. cMix: Anonymization by high-performance scalable mixing. Cryptology ePrint Archive, Report 2016/008, 2016. `http://eprint.iacr.org/`, version 20160530:183553 from May, 30 2016.

[7] David Chaum, Debajyoti Das, Farid Javani, Aniket Kate, Anna Krasnova, Joeri de Ruiter, and Alan T. Sherman. cMix: Mixing with minimal real-time asymmetric cryptographic operations. submitted to Privacy Enhanced Technologies (PETS) 2016, 2016.

[8] George Danezis. The traffic analysis of continuous-time mixes. *Privacy Enhancing Technologies: 4th International Workshop, PET 2004, Toronto, Canada, May 26-28, 2004. Revised Selected Papers*, pages 35–50, 2005.

[9] George Danezis, Claudia Diaz, and Carmela Troncoso. Two-sided statistical disclosure attack. *Privacy Enhancing Technologies: 7th International Symposium, PET 2007 Ottawa, Canada, June 20-22, 2007 Revised Selected Papers*, pages 30–44, 2007.

[10] George Danezis and Andrei Serjantov. Statistical disclosure or intersection attacks on anonymity systems. *Information Hiding: 6th International Workshop, IH 2004, Toronto, Canada, May 23-25, 2004, Revised Selected Papers*, pages 293–308, 2005.

[11] Joeri de Ruiter. Personal communication in e-mail. from July, 28 2016.

[12] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in Cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[13] Herman Galteland, Stig F. Mjølsnes, and Ruxandra F. Olimid. Attacks on cmix - some small overlooked details. Cryptology ePrint Archive, Report 2016/729, 2016. `http://eprint.iacr.org/2016/729`.

[14] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding routing information. In R. Anderson, editor, *Proceedings of Information Hiding: First International Workshop*, pages 137–150. Springer-Verlag, LNCS 1174, May 1996.

[15] Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proceedings of the 11th USENIX Security Symposium*, pages 339–353, Berkeley, CA, USA, 2002. USENIX Association.

[16] Shahram Khazaei and Douglas Wikström. Randomized partial checking revisited. In *Proceedings of the 13th International Conference on Topics in Cryptology*, CT-RSA'13, pages 115–128, Berlin, Heidelberg, 2013. Springer-Verlag.

[17] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Formal analysis of chaumian mix nets with randomized partial checking. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, SP '14, pages 343–358, Washington, DC, USA, 2014. IEEE Computer Society.

[18] Jean-François Raymond. Traffic analysis: Protocols, attacks, design issues, and open problems. In *International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*, pages 10–29, New York, NY, USA, 2001. Springer-Verlag New York, Inc.

[19] Andrei Serjantov and Peter Sewell. Passive attack analysis for connection-based anonymity systems. *Computer Security – ESORICS 2003: 8th European Symposium on Research in Computer Security, Gjøvik, Norway, October 13-15, 2003. Proceedings*, pages 116–131, 2003.

[20] Paul F. Syverson, David M. Goldschlag, and Michael G. Reed. Anonymous connections and onion routing. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, SP '97, pages 44–54, Washington, DC, USA, 1997. IEEE Computer Society.

Paper V

Jurisdictional adversaries monitoring and reconstructing the Tor network

Herman Galteland and Kristian Gjøsteen

In submission, arXiv:1808.09237

Jurisdictional adversaries monitoring and reconstructing the Tor network

Herman Galteland* and Kristian Gjøsteen

Department of Mathematical Sciences,
Norwegian University of Science and Technology, NTNU
{herman.galteland, kristian.gjosteen}@ntnu.no

Abstract

We model and analyze passive adversaries that monitor Tor traffic crossing the border of a jurisdiction the adversary is controlling. We show by simulations that a single jurisdiction is able to connect incoming and outgoing traffic crossing its border, tracking the traffic, and that a coalition of jurisdictions is able to reconstruct parts of the Tor network, revealing user-website connections. We use two algorithms to estimate the capabilities of the adversaries, the first simulates a Tor network and the second analyzes data from the simulation and reconstructs the network.

Keywords. Onion routing, anonymity, simulations.

1 Introduction

The Onion Router (Tor) protocol [9] is a well-established onion routing system that tries to provide a low-latency communications channel while also defending against network-level adversaries trying to

*This work is funded by Nasjonal sikkerhetsmyndighet (NSM), www.nsm.stat.no.

reveal who is talking to whom. It is well understood how the Tor network behaves when an adversary compromises a fraction of the onion routers, and in particular that if the entire network is monitored little or no security is left.

In this paper we analyze the power of (coalitions of) less powerful adversaries who do not monitor onion router traffic directly, but instead partition the network into jurisdictions and monitor traffic crossing from one partition into another.

These kinds of adversaries are interesting because they are real, in particular of the form of programs to monitor *traffic crossing borders*. In 2008 the Swedish parliament passed a bill allowing the Swedish National Defence Radio Establishment (*Försvarets radioanstalt*) to monitor both wireless and cable signals passing the Swedish border [17].

In 2016, the Norwegian government appointed a group of experts to investigate whether or not the Norwegian Intelligence Service should be allowed access to communication crossing the Norwegian border, similar to the Swedish National Defence Radio Establishment. The investigating report concluded that the Norwegian Intelligence Service should be allowed to monitor the Norwegian border [11], however, this has not yet been put into effect.

Denmark [18], France [6], and the United Kingdom [29] have similar laws on how to gather and store digital information. It seems likely that other countries either have or plan to have similar programs.

1.1 Related work

Formal analysis of the Tor protocol comes in two variants. The first use an abstract model of the protocol and gives security bounds based on a worst case adversary [2, 13, 15, 16, 19, 20]. The second includes a detailed description of the protocols in their analysis when proving the security bound [3, 22, 37].

Adversaries that observe both ends of a Tor circuit can connect the user with the website it is communicating with [25, 28]. The literature has considered adversaries controlling: an *autonomous system* (AS) [14, 34], an *Internet exchange point* (IXP) [26], and several ASes and IXPs [22, 27]. It has been shown that ASes can observe both ends of Tor circuits [37]. Tor path selection algorithms has been proposed

to avoid detection from ASes [1, 12].

An adversary with access to timing, packet size, and direction of packets sent over an encrypted HTTP tunnel can reveal the identity of the server and user by *traffic analysis* attacks [4, 7, 21, 24, 33, 39]. Countermeasures to traffic analysis attacks includes *padding messages* [8] and *morphing Tor traffic* to mimic traffic associated to other servers [39]. Note that hiding the packet length is insufficient [10].

Tor network simulators [31, 35] makes it possible to analyze the effectiveness of adversaries versus the Tor protocol.

A *stepping stone* is an intermediate node used by an attacker to conceal his identity. Algorithms used to detect stepping stones analyses streams of traffic to confirm or reject the existence of intermediate nodes between the analyzed traffic streams [5, 38].

1.2 Our contribution

In this paper we model and discuss a specific adversary versus the Tor protocol. The *jurisdictional adversary* is similar to an adversary controlling AS(es) or IXP(s), however, ASes and IXPs are typically located inside a jurisdiction whereas we consider a passive adversary that only monitors traffic crossing the border of a jurisdiction. Further, an adversary controlling an AS or an IXP would see all traffic inside their network whereas an adversary monitoring jurisdictional borders would not.

We simulate a Tor network, which includes the adversaries monitoring and storing traffic crossing their border. A chosen coalition of jurisdictions is trying to reconstruct the simulated Tor network by analyzing the stored data using traffic analysis. We do not morph the Tor traffic since the adversaries are only interested in the existence of traffic and not what it looks like.

Algorithms used to detect stepping stones analyzes streams of traffic to find intermediate nodes between the streams. Similarly, our reconstruction algorithm attempts to connect stream of traffic between known onion routers to recreate circuits. The techniques used to detect stepping stones could be used to detect onion routers. The difference between the stepping stone literature and our work is the adversary we are modeling and analyzing, where we assume that

the location of all onion routers is already known and we want to connect Tor traffic to reconstruct Tor circuits.

1.3 Overview

The model for our overlay network of Tor is in Section 3, this model is used to classify the types of traffic, and connections, the jurisdictional adversaries can observe, and create. The simulation algorithm is described in Section 4 and the reconstruction algorithm in Section 5. In Section 6 we present the reconstruction test results. We conclude with a possible countermeasure against the adversaries and summarize the adversaries in Section 7. We include the parameters used for the reconstruction results in Appendix A.

2 Background

2.1 Tor

The Onion Router protocol is an anonymous communication protocol [9]. The Tor protocol uses intermediate nodes called *onion routers* to achieve anonymity. A user establishes a *circuit* of onion routers in the Tor network to communicate with a server, where each onion router only knows the identity of its neighboring nodes. The first onion router of a circuit is a *guard node* G and when a user creates a new circuit he picks the guard node from a small set of onion routers, the default Tor configuration is three guard nodes. The last onion router communicates with the server on behalf of the user and is called an *exit node* E. The Tor protocol does not ensure encryption between the exit and the server and a malicious actor could abuse the information. Only a few onion routers get marked as an exit and it is believed that the majority of the exit nodes are honest. The intermediate node, between the guard and exit, is a *relay node* R. We let *circuit node* refer to any of the nodes in a circuit. For simplicity we assume that all circuits consist of one user, three onion routers, and one server, which is the default Tor configuration. Restricting circuits to contain only three onion routers is not essential for our reconstruction algorithm.

The user establishes a secret key with each onion router in the circuit and encrypts data in layers when sending it to the server, where each onion router removes one layer of encryption before relaying the data to the next node. When the server sends data back to the user each onion router encrypts the data and adds a layer. Note that we are only interested in the flow of information and will not include any encryption in our simulation.

3 Modeling jurisdictional adversaries

We describe the overlay network of the Tor network and use this model to determine what types of traffic a jurisdiction could observe and reconstruct.

3.1 Overlay network

The *overlay network* of the Tor network describes how information is sent between circuit nodes. A node in the overlay network represents a jurisdiction and an edge represents a communication connection between two jurisdictions. A jurisdiction contains a number of circuit nodes, where we assume the jurisdictions know which node is located inside its border. Information is sent in the overlay network when circuit nodes communicate. If two communicating circuit nodes are located in the same jurisdiction then no information is sent, and if the two circuit nodes are located in two different jurisdictions a *network path* in the overlay network is chosen. This path shows how the traffic is sent between jurisdictions, from the first jurisdiction containing the sender circuit node to the last jurisdiction containing the receiver circuit node, and determines which jurisdiction is able to observe the traffic data sent between the two circuit nodes.

Note that we make a simplification. Traffic between two circuit nodes inside a jurisdiction could very well cross the jurisdiction's borders in the physical network. In fact, since routing is dynamic, it could cross borders one day and not cross borders the next. Hence, the adversaries get less information in our model than in the real world.

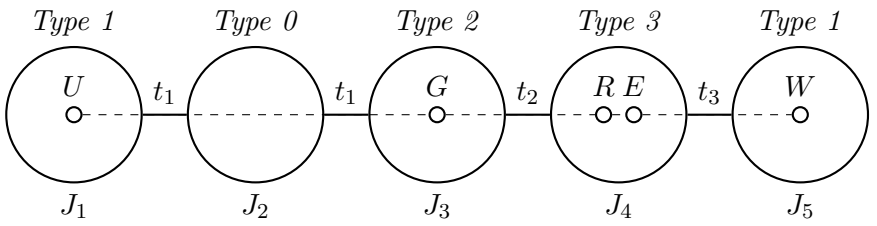


Figure 1: An example of the overlay network with a Tor circuit. U denotes a user, G a guard node, R a relay node, E an exit node, W a website, $J_1, \ldots, J_5$ denotes five distinct jurisdictions, and t_1, t_2, t_3 denotes timestamps for packets traveling between two circuit nodes. J_1 and J_5 observe *Type 1* traffic, an endpoint. J_2 observes *Type 0*, traffic passing through. J_3 observes *Type 2* traffic, where an incoming and an outgoing packet share a common node G and the timestamp difference $|t_1 - t_2|$ is close to an expected value. J_4 observes *Type 3* traffic, where the observed incoming and outgoing packets do not share a node but the timestamp difference $|t_2 - t_3|$ is close to an expected value. The solid line shows the network paths and the dashed line shows the Tor circuit

3.2 Observable traffic

As a Tor user communicates with a website they both generate traffic data. The user sends data packets to the first node of the circuit, which forwards it to the next node in the circuit and so forth until the website receives the user's packets, similar for the website. Packet data sent between two circuit nodes are transferred over a network path and all jurisdictions in the path observe the packets' metadata information. We assume a jurisdiction learns the sender, receiver, and direction of the packet and has the timestamp for when it observed the packet. A packet is observed when it crosses the border of a jurisdiction. A packet can be incoming, entering the jurisdiction, outgoing, leaving the jurisdiction, or passing through a jurisdiction.

The jurisdictional adversaries want to reconstruct the Tor circuits to reveal the sender and user, breaking the *relationship anonymity* [30] of the Tor protocol. Using the observed packet data a jurisdiction can

combine incoming and outgoing packets using traffic analysis. When an onion router receives a packet it will either encrypt or decrypt it, to add or remove an onion layer. This cryptographic computation takes time and there will be a timestamp difference between the observed incoming and outgoing packet. If the timestamp difference is close to an expected value then the two observed incoming and outgoing packets are most likely part of the same circuit, which means they can be connected. The time it takes to send a packet over a network cable is negligible compared to the time it takes for a circuit node to do its cryptographic computations and we assume that sending packets over a cable takes no time at all. We classify the types of connections a jurisdiction can create from its observed packets into four categories, see Figure 1 for a visual description;

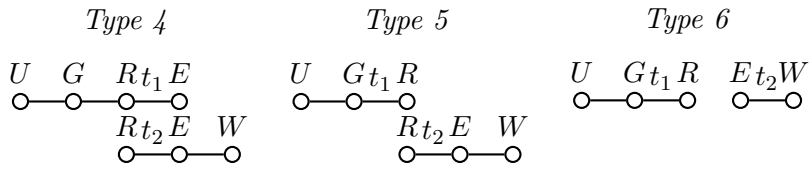
Type 0 A single packet passing through the jurisdiction, where the sender and receiver node of the packet is not located inside the jurisdiction's borders.

Type 1 A single packet ending in the jurisdiction, where either the sender or the receiver node of the packet is located inside the jurisdiction and is an endpoint of the circuit (a user or a website).

Type 2 One incoming and one outgoing packet share a common node inside the jurisdiction and the packets' timestamp difference is close to an expected value.

Type 3 One incoming and one outgoing packet that do not share a common node, but their timestamp difference is close to an expected value.

Packets that can be connected are combined and stored as *partial circuits*, each partial circuit contains a path of circuit nodes representing a partial Tor circuit and timestamps. We say a partial circuit has length n if its path consists of n circuit nodes. The timestamps are collected from the packet(s) it is created from, and all timestamps of any packet which would make the same partial circuit. (Many similar packets each with one timestamp makes one partial circuit with many timestamps.) The timestamps may be ordered into different sets to show the direction and flow of traffic over the partial circuit.



(a) Two partial circuits that share two nodes, and there are enough timestamps that are equal $t_1 = t_2$.

(b) Two partial circuits that share one node, and there are enough timestamps differences $|t_1 - t_2|$ that are close to an expected value.

(c) Two partial circuits that do not share any nodes, however, there are enough timestamps differences $|t_1 - t_2|$ that are close to an expected value.

Figure 2: Examples of partial circuits and how they can be connected. U denotes a user, G a guard node, R a relay node, E an exit node, W a website, and t_1, t_2 denotes timestamps for packets traveling between two circuit nodes.

3.3 Reconstructable traffic

It is very unlikely that a single jurisdiction is able to reveal the sender and receiver of any circuit. A coalition, however, can combine their analyzed partial circuits and potentially create complete Tor circuits, breaking the relationship anonymity. The colluding jurisdictions share their partial circuits with each other and try to combine them. We want to track the packets traveling from one jurisdiction to the next and look for partial circuits with paths that overlap, such that we can make a longer path by combining them. We classify the types of connections a coalition can create from its partial circuit, see Figure 2 for a visual description;

Type 4 Two partial circuits with paths that share two common nodes, where the two nodes are located at the end of the first and at the beginning of the second, and there are enough timestamp pairs, one from each partial circuit, that are identical.

Type 5 Two partial circuits with paths that share a common node, where that node is located at the end of the first and at the beginning of the second, and there are enough timestamp pairs,

one from each partial circuit, that have a timestamp difference that is close to an expected value.

Type 6 Two partial circuits with paths that do not share any common nodes, but there are enough timestamp pairs, one from each partial circuit, that have a timestamp difference that is close to an expected value.

Two partial circuits which combines into a *Type 4* connection should have identical timestamps. They share two nodes and the timestamps sent between these two nodes are should be present in both partial circuits.

Note that *Type 5* and *6* connections are similar to *Type 2* and *3* connections, the only difference is that partial circuits are being connected instead of single packets and in our reconstruction algorithm we reuse many of the methods for connecting these types of traffic.

4 Simulation algorithm

We describe our algorithm simulating the Tor network. The flow of information is sent in our overlay network, detailed in Section 3. In our simulator we assume that the adversaries are able to recognize Tor traffic and only generate Tor traffic, since onion routers usually only send Tor traffic to each other and all onion routers are known. Further, we assume that the adversaries have analyzed the distribution of timing patterns of Tor traffic. They will use this knowledge to statistically connect traffic entering and exiting their jurisdiction.

We initiate a network of jurisdictions, place onion routers, and define the user and website distributions. When a user communicates with a website, creates a new circuit, or destroys circuits we generate data. All traffic generated is sent between circuit nodes in the form of packets. Any data crossing the border of a jurisdiction is observed and stored, either as incoming or outgoing traffic. This data will be used in the reconstruction algorithm, detailed in Section 5.

We try to simulate the real world as best as we can, where information about the Tor network is gathered from the Tor data analysis website “Tor metrics” [36].

4.1 Initializing an overlay network

Each initialized jurisdiction represents a real world country. Two jurisdictions are connected by an edge if they share a border or if they connected to the same underwater internet cable [32].

Guard, Relay, and Exit nodes are placed in jurisdictions, where the location of each node is gathered from Tor metrics [36].

Users and websites are not placed in a jurisdiction at initialization, they are chosen and created during the simulation. The distribution of Tor users is gathered from Tor metrics [36] and a user can communicate with a website from any jurisdiction, hence, the websites are uniformly distributed.

4.2 Generate packets

The simulation runs for n iterations and each iteration generates data for a user that communicates with a website.

We pick either an existing ready user or create a new. If there is a ready user, we create a new circuit if its existing one has been up for more that 10 minutes. If there is no ready user we make a new user with a fresh circuit. Creating circuits generates traffic data.

When a user and website communicate they sends data over the circuit, where each node in the circuits sends packets to each other. A packet includes a timestamp, the sender and receiver node, the circuit ID of the Tor circuit, and packet length. It has the form

```
Timestamp Sender > Receiver (Circuit ID) Length.
```

Note that the circuit ID is only used to verify the output of the reconstruction algorithm.

4.3 Time and timestamps

A global `TIME` parameter is used to maintain the order of the user's activity, it is increased by a positive value between each iteration. The global parameter stays fixed during an iteration while the active user's time continues. As a packet is forwarded in the circuit the nodes perform cryptographic operations on it, although we do not do

the actual computations we add an *onion router delay* to the user's time. Similarly, we add a *sender delay* between packets sent from the user and website.

A lognormal distribution is used to sample these delays. Each onion router uses its own distribution to compute its delays, and it is sampled from a family of lognormal distributions. We do not know which distribution each onion router is using. We only know the family of distribution, which is chosen such that the circuit round-trip latency of the simulated network is close to the real Tor network's latency [36].

When the user has finished its activity in the current iteration it is getting ready for its next by waiting, as if reading the website it is communicating with. An activity delay is added to the user's time, which is uniform. Whenever the `TIME` parameter is larger than the user's time activity delay it can be chosen as a ready user.

4.4 Write observed traffic data

Each jurisdiction in the network path between two circuit nodes observes and stores the traffic. Each jurisdiction writes data to file as either incoming or outgoing traffic, this data will be used in the reconstruction algorithm.

4.5 Runtime

The runtime of the simulation algorithm is mainly dominated by writing and storing all data generated, the number of iterations n determines how much data is generated. For each iteration we generate traffic data for one Tor user, as it communicates with a website. Using the simulation parameters in Table 1, one iteration will on average generate 14,000 packets. However, more than one jurisdiction observes each packet and on average 60,000 packets are stored each iteration.

In wall-clock runtime for the simulation algorithm for: $n = 1000$ is roughly 5 minutes, $n = 10000$ is 4 hours, and $n = 100000$ is 11 hours.

5 Reconstruction algorithm

We describe our reconstruction algorithm, where a coalition of jurisdictions wants to reconstruct Tor circuits and reveal the sender and website.

We partially reconstruct a simulated Tor network, created using the algorithm described in Section 4, using the packets generated in the simulation. Each jurisdiction (from a chosen set of collaborators) process their observed data to make partial circuits, which will be connected further to create complete Tor circuits. The jurisdictions output is verified by comparing it with the real circuits created in the simulation. We assume all Tor circuits has length five, this is not essential for our algorithm.

5.1 Process observed packets

Each jurisdiction processes its observed packets to make partial circuits, using the classification discussed in Section 3. All packets are labeled either as incoming, entering the jurisdiction, or outgoing, leaving the jurisdiction. We iterate over all incoming packets and try to combine each one with an outgoing packet. We only look at the outgoing packets which are close to the incoming packet, with respect to time. We first look for trivial connections: incoming or outgoing packets which can be classified as a *Type 0* or *1* (packets passing through or packets ending inside the jurisdiction). If the incoming packet is not a trivial connection we look for an outgoing packet that shares a common node with the incoming packet, that is, we try to make a *Type 2* connection. If there are no outgoing packets with a common node we look for *Type 3* connections, where we want to find the outgoing packet which fits best with the incoming packet based on their timestamp difference. For *Type 2* and *3* we combine packets if their timestamp difference is close to some expected value, this value is the expected onion router delay and is derived from the family of lognormal distributions used in the simulation algorithm.

All connections made are stored as partial circuits, which contains the following information: a path of circuit nodes, four sets of timestamps, a probability score, and a list of circuit IDs.

The four sets of timestamps shows the flow of data traveling over the path. Two of the sets represents packets traveling in one direction, say from left to right, where one contains incoming timestamps on the left side and one contains outgoing timestamps on the right side. Similar for the remaining two sets where the direction is opposite of the first two (from right to left).

The probability score is used to evaluate how likely the circuit is part of a Tor circuit. The score is based on the time difference of the packets the partial circuit is made from. Partial circuits made from *Type 0* and *1* connections have a score of zero. The score is the output of the probability density function of a lognormal distribution with the time difference as input. The distribution is derived from the family of lognormal distributions. The closer the difference is to the expected onion router delay the higher the score is. For *Type 2* and *Type 3* connections we only connect an incoming packet with the outgoing packet which results in the highest score. A partial circuit's probability score is equal to the sum of each of its packet pair's score.

To verify our data we use the Tor circuit IDs, where each circuit ID is collected from the packets used to make the partial circuit. Each Tor circuit has one unique ID, but a partial circuit can have more than one ID stored. Note that we do not use the circuit IDs during the reconstruction process, they are only used to verify the output.

When a jurisdiction has analyzed all of its packets we discard all partial circuits that are most likely not part of a true Tor circuit, that is, if it has a low probability score. We set the discard limit based on trial and error, with a small discard limit we get a high false positive rate and with a large discard limit we get a low reconstruction rate.

5.2 Process partial circuits

Colluding jurisdictions share their partial circuits with each other to create complete Tor circuits. We start by finding partial circuits that share two nodes, to find *Type 4* connections. Then we look for partial circuits that share one node, to make *Type 5* connections. Following by looking for partial circuits that have enough timestamps pairs, one from each partial circuit, that have a timestamp difference close to

the expected value, that is, *Type 6* connections. If any new partial circuit was created while looking for these three types of connections we restart the partial circuit combination process, which is continued until there are no more circuits to combine.

We create a new partial circuit when we combine two, where we keep some of the data and discard some. We combine the two paths and any overlapping nodes are merged together. The new partial circuit only needs four sets of timestamps, where we take two from the first partial circuit, say the two leftmost, and two from the second, say the two rightmost. We calculate a new probability score for the circuit, where we use the timestamps that is going to be discarded to calculate the score, using the same method we use to scoring packets. All circuit IDs contained in the two partial circuits are included in the new.

The reconstruction algorithm terminates when there are no more partial circuits to combine. The output is all partial circuits of length three or longer, with a large enough probability score. Length five partial circuits are the potential complete Tor circuits.

5.3 Evaluate results

In the simulation we store all Tor circuits created, including their circuit IDs, and we use this to check the output of the reconstruction algorithm.

We compare the partial circuit's path with all simulated Tor circuits with a circuit ID that is equal to one of the partial circuit's stored IDs. The partial circuit is correct if the reconstructed path is equal to, or part of, one of the simulated paths. That is, a partial circuit is considered correct if its path is part of a simulated Tor circuit's path and it was created using packet data that was indeed sent over that Tor circuit.

We split the output into two groups: incorrect partial circuits, showing the false positive rate, and correct partial circuits, from which we get the reconstruction rate and the relationship revealing rate. The reconstruction rate shows how much of the simulated Tor circuits the algorithm reconstructed, and the relationship revealing rate shows how many user-website connections we found.

5.4 Runtime

The main contributing factor to the runtime of the reconstruction algorithm is the number of packets, the second is the number of timestamps. The algorithm can be split into two parts: the first analyzes packets to make partial circuits, the second analyzes and combines partial circuits.

Let $\mathcal{J}$ denote the set of colluding jurisdictions, cooperating in analyzing packets and partial circuits. Each jurisdiction $J \in \mathcal{J}$ analyzes its incoming and outgoing packets to combine them and stores the timestamps. Let p denote the number of incoming packets J has stored, we assume J also has p outgoing packets. We look through all p incoming packets and for each of them we compare it with some of the outgoing packets, we only connect packets that have timestamps close to each other and keep a short list of outgoing packets with timestamps close to the current incoming packet's timestamp. Iterating over the packets is at most $O(p \log p)$. For each packet we connect, we store its timestamp in a sorted list and inserting a timestamp is $O(p/k)$, where k is the number of partial circuits made by jurisdiction J. The runtime of the packet analysis algorithm, for each jurisdiction J, is $O((p^2 \log p)/k)$, where p is the number of packets (incoming or outgoing) J has stored and k is the number of partial circuits J has made. The number of colluding jurisdictions $|\mathcal{J}|$ is a small constant. As an optimization, we store all partial circuits a jurisdiction makes and we only need to run the packet analysis algorithm for a jurisdiction once. (If we want to change the set of collaborating jurisdictions we do not have to redo the packet analysis every time.)

All partial circuits made by the colluding jurisdictions are analyzed in the second part of the reconstruction algorithm. Let K denote the total number of partial circuits. When we combine partial circuits that share at least one common node we have a $O(K \log K)$ method for iterating through them, similar to the packet analysis method. However, when we combine partial circuits that do not share any common node we need a $O(K^2)$ method for iterating through them. We want to see if each partial circuit fits with all other, based on their combined path and timestamps. If two partial circuits' combined path is logical, looks like a Tor circuit, then we evaluate their

timestamps and give a probability score for how well the two partial circuits fit together. Let t denote the number of timestamps a partial circuit has. Evaluating the timestamps is $O(t \log t)$, using the method for packet analysis, and if the new probability score is high enough we create a new partial circuit and insert timestamps into sorted lists, which is $O(t)$. The runtime of the packet analysis algorithm is $O(K^2 t^2 \log t)$.

The wall-clock runtime of the reconstruction algorithm for: a simulated network with 1000 iterations and five jurisdictions is three hours, a simulated network with 10000 iterations and five jurisdictions is two and a half days.

5.5 Improvements

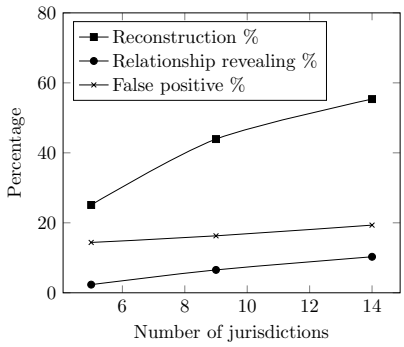
Our implementation is not perfect, and here we mention possible improvements to the reconstruction algorithm. The implementation is written in python, where an implementation written in a different language could be more efficient. Furthermore, each jurisdiction analyzes their own data and can be run in parallel.

6 Reconstruction results

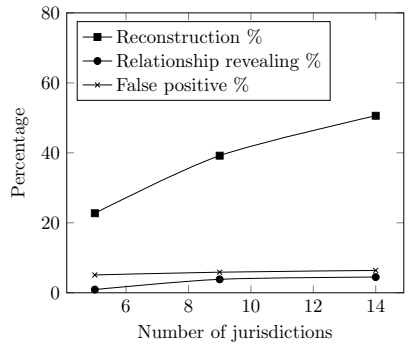
In the result we look at: the false positive rate, the reconstruction rate, and the relationship revealing rate. The false positive rate shows how many of the partial circuits are incorrect, the reconstruction and the relationship revealing rate only look at the partial circuits that are correct. The reconstruction rate shows how much of the simulated Tor circuits is reconstructed, partial circuits of length three or longer are used to find the reconstruction rate. The relationship revealing rate shows how many partial circuits reveal the user-website connection.

6.1 Results

We simulate a Tor network, with a specified number of iterations, and use the output of the simulation algorithm as input to the reconstruction algorithm. We run the simulation algorithm once and the reconstruction algorithm several times. Every time we reconstruct we



(a) Reconstructing a simulated network with a large number of iterations.



(b) Reconstructing a simulated network with a small number of iterations.

Figure 3: Comparing reconstruction results. The more iterations used in the simulation the more data is generated, more data means more connections can be made – both correct and false ones.

change the number of colluding jurisdictions. We include two reconstruction tests, in the first we look at how many simulation iterations affects the reconstruction results and in the second we look at how the coalition size affects the reconstruction results. The parameters used for the simulations are in Table 1, and the parameters used for the reconstructions are in Table 2.

In the first reconstruction test we compare reconstruction results from a simulation with a large number of iterations with reconstruction results from a simulation with a small number of iterations, see Figure 3. The average number of active Tor users in the simulation algorithm is close to the reported number of active users on Tor Metrics. The number of iterations specified for a simulation changes how many users have been active, but the average number of active users should still be the same for all simulations. We see that all three rates are higher in the reconstruction results of the larger simulation. The larger simulation generates more data and the reconstruction algorithm has more to process, this means the reconstruction algorithm

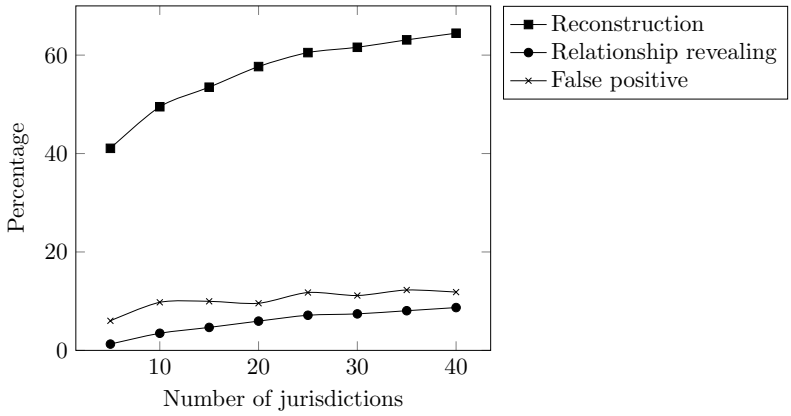


Figure 4: Reconstruction results for an increasing number of jurisdictions.

can make more connections and make more errors. By setting the cut-off bound for the probability score higher in the larger reconstruction we would get a lower false positive, reconstruction and relationship revealing rate, and get a result closer to the smaller reconstruction. Hence, having a larger number of iterations in the simulations only means longer computation time, and we will therefore only run the reconstruction algorithm on the smaller simulation for the second reconstruction test.

In the second reconstruction test we look at how the coalition size affects the reconstruction results, see Figure 4. The simulated network is always the same, we only change the number of jurisdictions cooperating for each run of the reconstruction algorithm. After the simulation algorithm is completed the jurisdictions are sorted based on the amount of data they have stored, from big to small. The jurisdiction coalition chosen for the n'th run of the reconstruction algorithm is the first $5n$ jurisdictions in the sorted list. This means that for each run of the reconstruction algorithm the coalition consists of the jurisdictions used in the previous run plus five new ones. For each new run of the reconstruction algorithm the five new jurisdictions have less and less data to contribute to the coalition, hence, all three rates flatten out and stabilize after 40 jurisdictions.

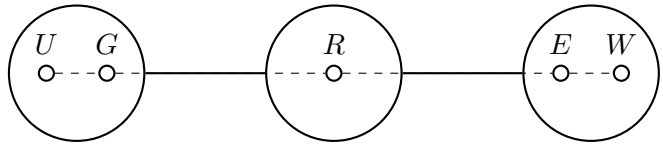


Figure 5: Path selection where the jurisdictional adversaries are unable to connect the user U with the website W, since they can only observe traffic sent to and from the guard node G, the relay node R, and the exit node E.

We can only speculate as to why the reconstruction rate peaks at 65 percent and the relationship rate peaks at 10 percent for the second reconstruction test. This is partly because of how the Tor network builds circuits and partly because of our implementation. If a Tor circuit's traffic doesn't cross the border of a jurisdiction, then no data is recorded and it can never be reconstructed. The jurisdictions used for the reconstruction simply do not observe enough data. Furthermore, we believe that each jurisdiction discards too much information when analyzing their own data. When we combine incoming and outgoing traffic we get a pile of leftovers, traffic that has not been used to reconstruct. A better algorithm could possibly reduce the amount of leftover traffic and find ways to use the leftovers.

7 Discussions

7.1 Path selection countermeasure

To be able to reconstruct a Tor circuit the jurisdictional adversaries need to observe traffic sent to and from the user and the website. If the traffic sent from the user to the guard node does not cross a jurisdictional border then no traffic can be observed and the circuit can never be fully reconstructed, similar for when the exit node and the website communicate. If the user specifies its Tor circuit such that the traffic that crosses the jurisdictional borders is sent between onion routers then the adversaries cannot see the user or the website and are never able to connect them.

The following Tor circuit selection prevents the jurisdictional adversaries breaking the relationship anonymity. A user U wants to connect to a website W. The user chooses the onion routers as follows: the guard node G should be situated in the same jurisdiction as the user U, the relay node R can be in any jurisdiction, and the exit node E should be located inside the same jurisdiction as the website W. See Figure 5.

Note that this path selections only avoids the jurisdictional adversaries, it is possible that other types of adversaries could break the relationship anonymity if the users use this path selections. For example, an adversary corrupting single onion routers can see traffic sent inside a jurisdiction if a circuit visits a corrupted node, and can possibly see the user or website of the circuit.

7.2 Passive global adversaries

We claim that the best attack the jurisdictional adversary can do is to passively observe Tor traffic, and a coalition of jurisdictions are therefore a passive global adversary versus the Tor network.

Tor uses a TLS connection between circuit nodes (except between the exit node and the website), which provides confidentiality and message integrity [8] and implies that Tor is IND–CCA [23]. Any active attack against messages sent between circuit nodes will be detected and prevented. The best attack the adversaries can do is to passively observe Tor traffic, and possibly stop traffic.

Each jurisdictional adversary indirectly monitor all onion routers inside its jurisdiction. If the jurisdictional adversaries cooperates in reconstructing Tor circuits they quickly become global, since they monitor a large portion of the onion router. In addition, a large set of jurisdictions has the power to reveal the relationship of a circuit if they choose to do so.

References

[1] Masoud Akhoondi, Curtis Yu, and Harsha V. Madhyastha. LASTor: A Low-latency AS-aware Tor Client. *IEEE/ACM Trans.*

Netw., 22(6):1742–1755, December 2014.

[2] Michael Backes, Aniket Kate, Praveen Manoharan, Sebastian Meiser, and Esfandiar Mohammadi. AnoA: A framework for analyzing anonymous communication protocols. In *Proceedings of the 2013 IEEE 26th Computer Security Foundations Symposium*, CSF '13, pages 163–178, Washington, DC, USA, 2013. IEEE Computer Society.

[3] Michael Backes, Aniket Kate, Sebastian Meiser, and Esfandiar Mohammadi. (Nothing else) MATor(s): Monitoring the Anonymity of Tor's Path Selection. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 513–524, New York, NY, USA, 2014. ACM.

[4] George Dean Bissias, Marc Liberatore, David Jensen, and Brian Neil Levine. Privacy vulnerabilities in encrypted http streams. In *Proceedings of the 5th International Conference on Privacy Enhancing Technologies*, PET'05, pages 1–11, Berlin, Heidelberg, 2006. Springer-Verlag.

[5] Avrim Blum, Dawn Song, and Shobha Venkataraman. Detection of interactive stepping stones: Algorithms and confidence bounds. In Erland Jonsson, Alfonso Valdes, and Magnus Almgren, editors, *Recent Advances in Intrusion Detection*, pages 258–277, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[6] Conseil d'Etat. Loi no. 2015-912 du 24 juillet 2015. Le livre VIII "Du renseignement", 2015.

[7] Thomas Demuth. A passive attack on the privacy of web users using standard log information. In Roger Dingledine and Paul Syverson, editors, *Proceedings of Privacy Enhancing Technologies workshop (PET 2002)*. Springer-Verlag, LNCS 2482, April 2002.

[8] T. Dierks and E. Rescorla. The transport layer security (TLS) protocol version 1.2. RFC 5246, RFC Editor, August 2008. `http://www.rfc-editor.org/rfc/rfc5246.txt`.

[9] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[10] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, SP '12, pages 332–346, Washington, DC, USA, 2012. IEEE Computer Society.

[11] Digitalt grenseforsvar. `https://forsvaret.no/etjenesten/dgf`. Accessed: 2017-10-04.

[12] Matthew Edman and Paul Syverson. As-awareness in Tor path selection. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 380–389, New York, NY, USA, 2009. ACM.

[13] Nathan S. Evans, Roger Dingledine, and Christian Grothoff. A Practical Congestion Attack on Tor Using Long Paths. In *Proceedings of the 18th Conference on USENIX Security Symposium*, SSYM'09, pages 33–50, Berkeley, CA, USA, 2009. USENIX Association.

[14] Nick Feamster and Roger Dingledine. Location diversity in anonymity networks. In *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*, WPES '04, pages 66–76, New York, NY, USA, 2004. ACM.

[15] Joan Feigenbaum, Aaron Johnson, and Paul Syverson. A model of onion routing with provable anonymity. In *Proceedings of the 11th International Conference on Financial Cryptography and 1st International Conference on Usable Security*, FC'07/USEC'07, pages 57–71, Berlin, Heidelberg, 2007. Springer-Verlag.

[16] Joan Feigenbaum, Aaron Johnson, and Paul Syverson. Probabilistic analysis of onion routing in a black-box model. *ACM Trans. Inf. Syst. Secur.*, 15(3):14:1–14:28, November 2012.

[17] Försvarsdepartementet. lag (2008:717), 2008.

[18] Forsvarsminister. Lov nr. 602 af 12-06-2013, 2013.

[19] Nethanel Gelernter and Amir Herzberg. On the limits of provable anonymity. In *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, WPES '13, pages 225–236, New York, NY, USA, 2013. ACM.

[20] Alejandro Hevia and Daniele Micciancio. An indistinguishability-based characterization of anonymous channels. In Nikita Borisov and Ian Goldberg, editors, *Privacy Enhancing Technologies: 8th International Symposium, PETS 2008 Leuven, Belgium, July 23-25, 2008 Proceedings*, pages 24–43, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[21] Andrew Hintz. Fingerprinting websites using traffic analysis. In *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies*, PET'02, pages 171–178, Berlin, Heidelberg, 2003. Springer-Verlag.

[22] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. Users get routed: Traffic correlation on Tor by realistic adversaries. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 337–348, New York, NY, USA, 2013. ACM.

[23] Jonathan Katz and Moti Yung. Unforgeable encryption and chosen ciphertext secure modes of operation. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, and Bruce Schneier, editors, *Fast Software Encryption: 7th International Workshop, FSE 2000 New York, NY, USA, April 10–12, 2000 Proceedings*, pages 284–299, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[24] Marc Liberatore and Brian Neil Levine. Inferring the source of encrypted http connections. In *Proceedings of the 13th ACM*

Conference on Computer and Communications Security, CCS '06, pages 255–263, New York, NY, USA, 2006. ACM.

[25] Steven J. Murdoch and George Danezis. Low-cost traffic analysis of Tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, SP '05, pages 183–195, Washington, DC, USA, 2005. IEEE Computer Society.

[26] Steven J. Murdoch and Piotr Zieliński. Sampled traffic analysis by internet-exchange-level adversaries. In *Proceedings of the 7th International Conference on Privacy Enhancing Technologies*, PET'07, pages 167–183, Berlin, Heidelberg, 2007. Springer-Verlag.

[27] Rishab Nithyanand, Oleksii Starov, Phillipa Gill, Adva Zair, and Michael Schapira. Measuring and Mitigating AS-level Adversaries Against Tor. In *Proceedings of the Network and Distributed Security Symposium - NDSS '16*. Internet Society, February 2016.

[28] Lasse Øverlier and Paul Syverson. Locating hidden servers. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, SP '06, pages 100–114, Washington, DC, USA, 2006. IEEE Computer Society.

[29] Parliament of the United Kingdom. Investigatory powers act 2016, 2016.

[30] Andreas Pfitzmann and Marit Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management, 2010.

[31] shadow-plugin-tor wiki. `https://github.com/shadow/shadow-plugin-tor/wiki`. Accessed: 2018-02-19.

[32] Submarine Cable Map. `https://github.com/telegeography/www.submarinecablemap.com`. Accessed: 2018-10-25.

[33] Qixiang Sun, Daniel R. Simon, Yi-Min Wang, Wilf Russell, Venkata N. Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, SP '02, Washington, DC, USA, 2002. IEEE Computer Society.

[34] Yixin Sun, Anne Edmundson, Laurent Vanbever, Oscar Li, Jennifer Rexford, Mung Chiang, and Prateek Mittal. RAPTOR: Routing attacks on privacy in Tor. In *Proceedings of the 24th USENIX Security Symposium*, August 2015.

[35] The Tor path simulator, torps. `https://github.com/torps/torps`. Accessed: 2018-06-07.

[36] Tor Metrics. `https://metrics.torproject.org/`. Accessed: 2018-10-23.

[37] Chris Wacek, Henry Tan, Kevin S. Bauer, and Micah Sherr. An empirical evaluation of relay selection in Tor. In *NDSS*, 2013.

[38] Xinyuan Wang, Douglas S. Reeves, and S. Felix Wu. Inter-packet delay based correlation for tracing encrypted connections through stepping stones. In *Proceedings of ESORICS 2002*, pages 244–263, October 2002.

[39] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson. Spot me if you can: Uncovering spoken phrases in encrypted voip conversations. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 35–49, May 2008.

A Simulation and reconstruction parameters

Table 1: Parameters and data of the simulations used in the results.

	Iterations	Packets sent	Data stored
Small simulation	1000	14,001,560	0.69 GB
Large simulation	10000	141,404,190	5.16 GB

Table 2: Parameters and data of the reconstruction algorithm.

	Jurisdictions	Probability score lower bound *Type 2*	*Type 3*
Figure 3	US, GB, AU, CA, NZ, DK, FR, NL, NO, BE, DE, IT, ES, SE	10	20
Figure 4	DE, US, FR, RU, GB, PL, NL, UA, JP, DK, CN, CA, NO, BG, AE, IT, SE, CH, TR, GR, FI, AT, RO, MD, CZ, ID, ES, PT, IN, IS, HU, BE, IE, BR, AU, SG, SC, TH, SK, LU, HR, HK, PK, MY, LV, DJ, ZA, KR, IR, EG, LT, VE, VN, MX, TW, PH, AR, CR, IL, EE, SA, PA, CL, SI, CO, BN, KZ, GE, BY, CY, AZ, MA, AO, OM, NZ, RS, CW, KG, CM, JO, KE, BD, ZM, KN, SN, QA, NA, TZ, BA, DO, UZ, AL, AM, MN, LK, GH, SD, MZ, PY, KW	10	20

Paper VI

Verifiable Random Secrets and Subliminal-Free Digital Signatures

Herman Galteland and Tjerand Silde

In submission

Verifiable Random Secrets and Subliminal-Free Digital Signatures

Herman Galteland*and Tjerand Silde

Department of Mathematical Sciences,
Norwegian University of Science and Technology, NTNU
{herman.galteland, tjerand.silde}@ntnu.no

Abstract

We propose a protocol for subliminal-free post-quantum digital signatures. This is the first construction to achieve subliminal-free digital signatures in the post-quantum setting. The core of our protocol is a way to generate subliminal-free verifiable secret randomness, called a verifiable random secret (VRS) scheme, which is of independent interest. The VRS ensures that the randomness used in the signature is honestly generated, and contains no extra embedded information.

A VRS is an interactive protocol between two parties where the goal is to create some randomness in a way such that at the end of the protocol run the randomness is known to the prover but not to the verifier. The verifier will learn a commitment to the randomness and a proof of correctness, so that he can verify that the randomness was generated honestly, and to allow him to verify that it is used properly in subsequent applications. We require the randomness to be unpredictable to both parties until the protocol is completed, and unpredictable to the verifier even after the protocol is over.

Subliminal digital signatures were introduced by Simmons (CRYPTO 1983) and he proposed an interactive subliminal-free

*This work is funded by Nasjonal sikkerhetsmyndighet (NSM), www.nsm.stat.no.

signature protocol for signatures based on the hardness of the discrete logarithm problem, but it was never proved secure. We propose a VRS similar to Simmons' solution, combine the VRS with Schnorr-signatures, and prove it secure.

Our main contribution is a post-quantum secure subliminal-free digital signature scheme combining a lattice-based VRS with lattice-based signatures. The VRS uses the commitment scheme from Baum et al. (SCN 2018) and their efficient zero-knowledge proofs of linear relations together with the proof of shuffle of known content by Baum et al. (IN SUBMISSION[‡]) which is built on top of the same commitment scheme. The VRS can be used to create verifiable secret randomness to be used in the lattice-based signature framework by Lyubashevsky (EUROCRYPT 2012).

Subliminal digital signatures can e.g. be a threat against two-factor authentication systems when the second device is malicious. Boneh et al. (IEEE S&P 2019) gave a solution to this problem for signatures based on the hardness of computing discrete logarithms over elliptic curves, and our protocol can be an alternative solution for lattice-based signatures.

Keywords: post-quantum cryptography, subliminal channels, digital signatures, verifiable random secrets, zero knowledge-proofs.

1 Introduction

A *subliminal channel* is a solution to Simmons's Prisoners' Problem [46], where two prisoners want to communicate covertly over an overt channel controlled by a Warden. The prisoners are allowed to send signed messages, to verify who sent the message, but the messages themselves have to be sent in the clear so that the Warden can read their content. To communicate covertly the prisoners can create a subliminal channel, where the subliminal messages are encoded into the signatures. Only the prisoners, which have some shared secret knowledge, can recover the subliminal messages, and all signatures look normal to everyone else.

The goal of the subliminal sender and receiver, the prisoners, is

[‡]Will be published on eprint and/or in conference proceedings spring 2020.

to communicate covertly, and the goal of the Warden is to prevent any subliminal channel. In this paper we will focus on achieving the Warden's goal of creating a subliminal-free signature scheme.

Constructing subliminal-free digital signature schemes for classical adversaries is solved [8, 9, 19, 52]. Constructing post-quantum secure subliminal-free digital signature schemes has been an open problem until now, and designing such a protocol is the main contribution in our paper.

1.1 Warden Model

The subliminal sender S and receiver R want to reliably communicate discretely over a communication channel controlled by the Warden W. Before a signature is generated S and W interact to jointly produce a random value, known to S but secret to W, which will be used to create a signature, and S creates a proof that the random value was honestly generated and that the signature was *indeed* created using that random value. The sender sends the message-signature-proof tuple to W, which verifies the signature and checks the proof. If, and only if, the proofs are valid then W sends the message-signature pair to R. The proofs are only sent to W and cannot be used to send subliminal messages.

The sender will be able to choose his own public and secret signing keys, however, S will not be able to change the keys during the signing process. Warden will abort any messages if the signature is invalid with respect to the verification key of S and close the channel. Also, if S aborts during the signing process, e.g. if the signature does not include the subliminal bits and he wants to re-try, Warden closes the channel.

We assume that S and R may have shared secret information before they start communicating: a secret key for a suitable symmetric cryptosystem, and the signing key. The secret key is used to encrypt the subliminal messages to make them indistinguishable from a random value, and the signing key can be used to recover subliminal messages. The sender is allowed to cheat during key generation to produce any desired key.

We are not interested in hiding information in the messages them-

selves, called steganography. We will assume that S is given a message to sign. We are only interested in the case where S tries to encode subliminal messages into the signature.

1.2 Subliminal-Free Digital Signatures

Our work builds upon the subliminal-free digital signature scheme with proof definition of Bohli et al. [8], where they constructed a subliminal-free variant of ECDSA. We give a similar definition of a subliminal-free digital signatures scheme and construct a post-quantum subliminal-free digital signatures scheme based on lattices.

We get a subliminal-free digital signature scheme if S is unable to choose any of the values used to generate a signature, specifically random values. If S can choose the randomness then he can decide the final outcome of the signature and insert a subliminal message. We introduce the verifiable random secrets (VRS) scheme and use it together with Schnorr-like digital signatures schemes to create a subliminal-free digital signature scheme. The VRS is used by S and W to jointly generate a verifiable random number, which will be used to produce a signatures. The sender also need to include a proof showing that the random number generated in the VRS was used to produce the signature.

A VRS is an interactive protocol between a prover and a verifier that generates verifiable random numbers, that are known only to the prover and unpredictable for everyone else, and with proofs to convince the verifier that the randomness was generated honestly. VRSs are inspired by the verifiable random functions (VRFs) of Micali et al. [36], the main difference between a VRF and a VRS is that the random number is secret in a VRS and public in a VRF.

Schnorr-like digital signature schemes follow the same structure as zero-knowledge proofs of knowledge of opening of a commitment. The prover sends a new commitment of a random value to the verifier, the verifier replies with a challenge, and the sender generates a response using the challenge and the secret opening. This protocol can be made non-interactive using the Fiat-Shamir heuristic [22], where the challenge is generated by a hash function. If the message is a part of the input to the hash function we get a digital signature scheme.

The lattice-based signature scheme of Lyubashevsky [33, 34] follows the same pattern as Schnorr's digital signature scheme: commitment, challenge, and response. The signer samples a random vector, computes the challenge using a hash function, and the response is generated using the random value, challenge, and secret signing key. Lyubashevsky's scheme uses rejection sampling to discard certain signatures, where a signature is sometimes rejected to make the signature distribution independent of the secret key.

We give two VRS schemes and two subliminal-free digital signature schemes: a Diffie-Hellman (DH) based VRS with Schnorr signatures and a lattice-based VRS with lattice signatures ala Lyubashevsky. Our main contribution is the lattice based subliminal-free digital signature scheme. The DH based subliminal-free digital signature scheme is similar to Simmons's scheme, where we show that it is secure and subliminal-free in the Warden Model.

1.3 Related Work

Simmons introduced the notion of subliminal channels as a solution to his prisoners' problem [46]. Two partners in crime are arrested and put into separate parts of a jail. The prisoners wish to communicate with each other, to plan their escape, and the Warden allows them to send messages if he can read the content of the messages sent, hoping to learn about any potential escape plan. The prisoners are allowed to sign their messages and can verify that they are sent from a prisoner and not from the Warden. This is an authentication without secrecy communication channel controlled by the Warden. The problem of the prisoners is to make a subliminal channel that stays undetected by the Warden, and the problem of the Warden is to prevent any subliminal channels. Simmons showed that subliminal channels exists in digital signature schemes [47–49] and since more have been found [3, 9, 23, 26, 32, 53].

Desmedt was the first to construct and locate subliminal-free digital signature schemes [15], which since has been continued [8, 9, 19, 45, 49, 50, 52]. Bohli et al. introduced the notion of a *subliminal-free with proof signature scheme*, where the sender of a signature sends a proof to the Warden, and only to the Warden, that proves the signa-

ture sent is subliminal-free [8]. In *divertible protocols* [6, 12, 13, 40] a third party can be inserted between the two communication principals, where the third party can remove or detect subliminal messages. A *cryptographic reverse firewalls* [14, 37] sits around a user's computer and modifies messages to: maintain the usability of the user's computer; preserve security of the protocol generating the message, and; hinder information leaking from the computer to the outside world.

NIST's post-quantum standardization project have asked for digital signature schemes [39], and all digital signature schemes accepted to the second round are not subliminal free [23]. The CRYSTAL-Dilithium [20] and qTesla [2] submissions, however, are similar to Schnorr signatures and can potentially become subliminal-free using our techniques.

The idea of verifiable random secrets is inspired by the notion of *verifiable random functions* (VRF) introduced by Micali, Rabin and Vadhan [36]. A VRF is a pseudo-random function that outputs a random number and a proof stating that the function, with a given input, produced the random number. The first VRF constructions [1, 17, 18] had limitations; they are either interactive, had a small input space, and/or did not achieve full adaptive security. In the recent years VRF constructions *"with all desired properties"* [10, 27–31, 43, 51] have been made, which do not have any of these limitations.

1.4 Our Contribution

We introduce the notion of verifiable random secrets scheme and show how it can be used, together with Schnorr-like digital signature schemes, to construct subliminal-free digital signature schemes. We give two constructions following this framework; one Diffie-Hellman based, similar to Simmons's subliminal-free DSA scheme, and one post-quantum scheme based on lattices. Our contribution in the Diffie-Hellman based scheme is to formalize Simmons's subliminal-free DSA scheme and prove it secure.

1.5 Organization

In Section 2 we give general background information and in Section 3 and 4 we give the background information needed for our Diffie-Hellman based schemes and lattice-based schemes, respectively. We introduce verifiable random secrets in Section 5 and give definitions related to subliminal channels and subliminal-free digital signature schemes in Section 6. Our constructions are in Section 7, and we comment about the size and efficiency of the schemes in Section 8

2 Preliminaries

2.1 Notation

Let S be a set and $\texttt{Alg}(\cdot)$ an algorithm. Then, by $s \leftarrow \texttt{Alg}(\cdot)$ we mean that s is assigned the output of $\texttt{Alg}(\cdot)$, by $s \xleftarrow{\$} S$ we mean that s is assigned an uniformly random element of S (unless a specific distribution is specified), and by $s \leftarrow s'$ we mean that s is assigned the value s'. Let λ be the security parameter, then $\epsilon(\lambda)$ is a negligible function in the security parameter.

2.2 Schnorr Groups

A Schnorr group is a large prime-order subgroup of $\mathbb{Z}_p^*$, the multiplicative group of integers modulo p for some prime p. Let q be a prime and r be a positive integer such that $p = qr + 1$. Further, for a h such that $1 < h < p$ and $h^r \not\equiv 1 \mod p$, then $g \equiv h^r \mod p$ is a generator for a subgroup of $\mathbb{Z}_p^*$ of order q.

2.3 The Polynomial Ring R_p and the Respective Norms

Let $p, r \in \mathbb{N}^+$ and $N = 2^r$. Then we define the rings $R = \mathbb{Z}[X]/\langle X^N + 1\rangle$ and $R_p = R/\langle p\rangle$, that is, R_p is the ring of polynomials modulo $X^N + 1$ with integer coefficients modulo p. We define the norms of elements

$$f(X) = \sum \alpha_i X^i \in R$$

to be the norms of the coefficient vector as a vector in $\mathbb{Z}^N$:

$$||f||_1 = \sum |\alpha_i| \quad ||f||_2 = \left(\sum \alpha_i^2\right)^{1/2} \quad ||f||_\infty = \max_{i\in\{1,\ldots,n\}} \{|\alpha_i|\}.$$

For an element $\bar{f} \in R_p$ we choose coefficients as the representatives in $\left[-\frac{p-1}{2}, \frac{p-1}{2}\right]$, and then compute the norms as if $\bar{f}$ is an element in R. For vectors $\boldsymbol{a} = (a_1, \ldots, a_k) \in R^k$ we define the 2-norm to be

$$\|\boldsymbol{a}\|_2 = \sqrt{\sum \|a_i\|_2^2},$$

and analogously for the ∞-norm. We omit the subscript in the case of the 2-norm.

2.4 Short elements in R_p

It can be seen from Corollary 1.2 in [35] that sufficiently short elements in R_p are invertible. In the following, we assume for simplicity that the parameters are set such that all non-zero elements of ∞-norm at most 2 are invertible in R_p. We furthermore define

$$\mathcal{C} = \{c \in R_p \mid \|c\|_\infty = 1, \|c\|_1 = \nu\},$$

which consists of all elements in R_p that have trinary coefficients and are non-zero in exactly ν positions. This means that for any two distinct $c, c' \in \mathcal{C}$, the difference $c - c'$ is invertible as well. For convenience, denote by

$$\bar{\mathcal{C}} = \left\{c - c' \mid c \neq c' \in \mathcal{C}\right\}$$

the set of such differences.

2.5 Discrete Gaussian Distribution

The continuous normal distribution over $\mathbb{R}^k$ centered at $\boldsymbol{v} \in \mathbb{R}^k$ with standard deviation σ is given by

$$\rho(\boldsymbol{x})_{\boldsymbol{v},\sigma}^N = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-||\boldsymbol{x} - \boldsymbol{v}||^2}{2\sigma^2}\right).$$

When sampling randomness for our lattice-based commitment scheme, we'll need samples from the *discrete Gaussian distribution.* This distribution is achieved by normalizing the continuous distribution over R^k by letting

$$\mathcal{N}^k_{\boldsymbol{v},\sigma}(\boldsymbol{x}) = \frac{\rho^{kN}_{\boldsymbol{v},\sigma}(\boldsymbol{x})}{\rho^{kN}_{\sigma}(R^k)},$$

where $\boldsymbol{x} \in R^k$ and $\rho^{kN}_{\sigma}(R^k) = \sum_{\boldsymbol{x}\in R^k} \rho^{kN}_{\sigma}(\boldsymbol{x})$. When $\sigma = 1$ or $\boldsymbol{v} = \mathbf{0}$, they are omitted.

The most efficient way to sample elements from a discrete Gaussian distribution is using rejection sampling. Rejection sampling is a technique used to sample from a distribution Φ_0, using samples from a similar distribution Φ_1. Let Φ_0 and Φ_1 be two distributions defined over the same domain S, whose probability mass functions are efficiently computable. Let M be a constant such that $\Phi_0(x) \leq M \cdot \Phi_1(x)$ for all $x \in S$. Then the distribution generated by the algorithm in Figure 1 will output samples distributed according to Φ_0 while only using samples from Φ_1 and a uniform distribution. The performance of the algorithm depends strongly on how similar Φ_0 and Φ_1 are, in particular, how small the scaling factor M is.

Rejection Sampling

Input: distributions Φ_0, Φ_1 and constant M s.t. $\Phi_0(x) \leq M \cdot \Phi_1(x) \forall x$

Output: x from the distribution Φ_0

1 : $x \xleftarrow{\$} \Phi_1$

2 : $u \xleftarrow{\$} [0, 1]$

3 : **if** $u \leq \frac{\Phi_0(x)}{M \cdot \Phi_1(x)}$ **then** goto 1

4 : **Return:** x

Figure 1: Rejection sampling algorithm.

There are other ways to sample elements from a discrete Gaussian distribution. Another method is called the Box-Muller transform [11]. This method is not as efficient, but on the other hand, it is deterministic when given samples from a uniform distribution. We will make

use of this in our protocol. The transform works as follows: given two independent samples u_1 and u_2 from a uniform distribution on the unit interval, let

$$z_0 = \sqrt{-2u_1}\cos(2\pi u_2), \text{ and}$$
$$z_1 = \sqrt{-2u_1}\sin(2\pi u_2).$$

Then z_0 and z_1 are independent random variables from the standard normal distribution $\mathcal{N}$ with mean $\mu = 0$ and standard deviation $\sigma = 1$. To get samples from a distribution $\mathcal{N}_\sigma$ for $\sigma \neq 1$ we just multiply z_0 and z_1 by σ. Lastly, we note that if we add z_0 and z_1, then $z_2 = z_0 + z_1$ is distributed accordingly to $\mathcal{N}_{2\sigma}$. However, these samples are continuous and not discrete; we solve this by rounding to the closest integer.

2.6 The k-SUM Problem

The *k-sum problem* (k-SUM) is a variant of the subset sum problem (SSP). SSP is a NP-complete decision problem where given a set of n integers $a_1, a_2, \ldots, a_n$ and a number S; is there a non-empty subset of $a_1, a_2, \ldots, a_n$ whose sum is S? This decision problem is as hard as its search-equivalent. The k-SUM problem is the problem of deciding if there is a subset of size k of $a_1, a_2, \ldots, a_n$ whose sum is S. It can easily be shown that SSP reduces to k-SUM, as a polynomial time k-SUM-solver easily could be used to solve SSP by trying $k = 1, 2, \ldots, n$ until it finds a solution. Further, the decision variant of k-SUM is as hard as the search variant of k-SUM, as one could find the subset of size k by removing elements one-by-one and check if we have a solution or not for the new set. Given a n and a k, the fastest algorithms for solving k-SUM runs in $\mathcal{O}(n^{k/2})$ [21].

2.7 Subliminal-Free with Proof Signature Scheme

We include the following definition of Bohli et al. [8] for comparison, where we give our definition of a subliminal-free signature scheme in Section 6.3 followed by a note on the differences between our definition and that of Bohli et al.

Definition 1 (Subliminal-Free with Proof Signature Scheme [8])**.** *A subliminal-free with proof signature scheme is a quintuple of algorithms* $(\mathcal{K}, \mathcal{K}_{SF}, \mathcal{S}, \mathcal{V}, \mathcal{C})$*, where*

- *The* key generation algorithm $\mathcal{K}$ *takes the security parameter* l *as input and returns a pair of verification and signing keys* (vk, sk)*.*
- *The* subliminal-free key generation algorithm $\mathcal{K}_{SF}$ *takes* vk *and* sk *as input and generates the information* ci *that the warden needs for checking the signature computation.*
- *The* signing algorithm $\mathcal{S}$ *takes a message* m *and the signing key* sk *as input and produces a valid signature* σ *for* m *under* vk *and a proof* t*.*
- *The* verification algorithm $\mathcal{V}$ *takes a message* m*, a signature* σ *and the public verification key* vk *as input and returns* 1 *if* σ *is a valid signature for* m *with respect to* vk*, and* 0 *otherwise.*
- *The* checking algorithm $\mathcal{C}$ *takes a message* m*, a signature* σ*, a verification key* vk*, the checking information* ci *and a proof* t *as input, and returns* 1 *if* $\mathcal{V}(m, \sigma, vk) = 1$ *and* (σ, t) *is a valid output of* $\mathcal{S}(m, sk)$*.*

Moreover, for any algorithm A *taking the security parameter* l *as input, the probability of giving as output values* $vk, sk, ci, m, \sigma_1, \sigma_2, t_1, t_2$ *such that* $(vk, sk), ci$ *are computationally indistinguishable from the output of* $\mathcal{K}$ *and* $\mathcal{K}_{SF}$*, respectively,* $\sigma_1 \neq \sigma_2$ *and* $\mathcal{C}(m, \sigma_1, vk, ci, t_1) = \mathcal{C}(m, \sigma_2, vk, ci, t_2) = 1$ *is negligible in the security parameter* l*.*

2.8 Diffie-Hellman Problems

Definition 2 (Discrete Logarithm Problem)**.** *Fix a cyclic group* G *of prime order* q *with generator* g*. The advantage of an algorithm* $\mathtt{A}$ *solving the* Discrete Logarithm (DL) *problem for* G *and* g *is*

$$\mathbf{Adv}_G^{\mathsf{DL}}(\mathtt{A}) = \Pr[\mathbf{Exp}_G^{\mathsf{DL}}(\mathtt{A}) = 1],$$

where the experiment $\mathbf{Exp}_G^{\mathsf{DL}}(\mathtt{A})$ *is given in Figure 2 (left).*

Definition 3 (Decisional Diffie-Hellman Problem)**.** *Fix a cyclic group G of prime order q with generator g. The advantage of an algorithm* $\mathtt{A}$ *solving the* Decisional Diffie-Hellman (DDH) *problem for G and g is*

$$\mathbf{Adv}_G^{\mathsf{DDH}}(\mathtt{A}) = \left|\mathbf{Pr}[\mathbf{Exp}_G^{\mathsf{DDH\text{-}1}}(\mathtt{A}) = 1] - \mathbf{Pr}[\mathbf{Exp}_G^{\mathsf{DDH\text{-}0}}(\mathtt{A}) = 1]\right|$$

where the experiment $\mathbf{Exp}_G^{\mathsf{DDH}\text{-}b}(\mathtt{A})$ *is given in Figure 2 (right).*

$\underline{\mathbf{Exp}_G^{\mathsf{DL}}(\mathtt{A}):}$
$X \xleftarrow{\$} G$
$x \leftarrow \mathtt{A}(g, X)$
if $g^x = X$
  return 1
else
  return 0

$\underline{\mathbf{Exp}_G^{\mathsf{DDH}\text{-}b}(\mathtt{A}):}$
$x, y, r \xleftarrow{\$} \mathbb{Z}_q$
$X \leftarrow g^x; Y \leftarrow g^y$
if $b = 0$
  $Z \leftarrow g^{xy}$
else
  $Z \leftarrow g^r$
$b' \leftarrow \mathtt{A}(g, X, Y, Z)$
return b'

Figure 2: The DL experiment $\mathbf{Exp}_G^{\mathsf{DL}}(\mathtt{A})$ (left). The DDH experiment $\mathbf{Exp}_G^{\mathsf{DDH}\text{-}b}(\mathtt{A})$ (right).

2.9 Commitment Schemes

Commitment schemes were first introduced by Blum [7], and have since become an essential component in many advanced cryptography protocols.

Definition 4 (Commitment Scheme)**.** *A commitment scheme consists of three algorithms: key generation* (KeyGen)*, commitment* (Com) *and opening* (Open)*, where*

- KeyGen*, on input the security parameter 1^λ, outputs public parameters* pp*,*
- Com*, on input a message m, outputs a commitment c and randomness r,*

- $\texttt{Open}$, *on input* m, c *and* r, *outputs either* 0 *or* 1,

and the public parameters $\texttt{pp}$ *are implicit input to* $\texttt{Com}$ *and* $\texttt{Open}$.

Definition 5 (Completeness)**.** *We say that the commitment scheme is* complete *if honestly generated commitment are accepted by the opening algorithm. Hence, we want that*

$$\Pr\left[\texttt{Open}(m,c,r)=1 \;:\; \begin{matrix}\texttt{pp}\leftarrow\texttt{KeyGen}(1^\lambda)\\(c,r)\leftarrow\texttt{Com}(m)\end{matrix}\right]=1.$$

Definition 6 (Hiding)**.** *We say that a commitment scheme is* hiding *if an adversary* $\texttt{A}$, *after giving two messages* m_1 *and* m_2 *to a commitment oracle* $\texttt{O}_{\texttt{com}}$ *and receiving the commitment* c *to either* m_1 *or* m_2 *(chosen at random), cannot distinguish which message* c *is a commitment to. Hence, we want that*

$$2\cdot|\Pr\left[b=b' \;:\; \begin{matrix}\texttt{pp}\leftarrow\texttt{KeyGen}(1^\lambda)\\(m_1,m_2)\leftarrow\texttt{A}(\texttt{pp})\\b\xleftarrow{\$}\{0,1\},c\leftarrow\texttt{O}_{\texttt{com}}(m_b)\\b'\leftarrow\texttt{A}(c)\end{matrix}\right]-\frac{1}{2}|\leq\epsilon(\lambda).$$

Definition 7 (Binding)**.** *We say that commitment scheme is* binding *if an adversary* $\texttt{A}$, *after creating a commitment* c *to a messages* m, *cannot find a valid opening of* c *to a different message* $\hat{m}$. *Hence, we want that*

$$\Pr\left[\begin{matrix}m\neq\hat{m}\\\texttt{Open}(m,c,r)=1\\\texttt{Open}(\hat{m},c,\hat{r})=1\end{matrix} \;:\; \begin{matrix}\texttt{pp}\leftarrow\texttt{KeyGen}(1^\lambda)\\m\leftarrow\texttt{A}(\texttt{pp})\\(c,r)\leftarrow\texttt{A}(m)\\(\hat{m},\hat{r})\leftarrow\texttt{A}(m,c,r)\end{matrix}\right]\leq\epsilon(\lambda).$$

Definition 8 (Unconditional and Computational Adversaries)**.** *We say that a commitment scheme is* unconditionally hiding *if the scheme is hiding against an unbounded adversary, and we say that it is* computationally hiding *if the scheme is hiding against a bounded probabilistic time adversary. We say that a commitment scheme is* unconditionally binding *if the scheme is binding against an unbounded adversary, and we say that it is* computationally binding *if the scheme is binding against a bounded probabilistic time adversary.*

2.10 Digital Signature Schemes

Definition 9 (Digital Signature Schemes)**.** *A digital signature scheme consists of three algorithms: key generation* ($\texttt{KeyGen}$), *signing* ($\texttt{Sign}$) *and verification* ($\texttt{Verify}$), *where*

- $\texttt{KeyGen}$, *on input the security parameter* 1^λ, *outputs public parameters* $\texttt{pp}$, *a signing key* $\texttt{sk}$, *and a verification key* $\texttt{vk}$,
- $\texttt{Sign}$, *on input a message* m *and* $\texttt{sk}$, *outputs a signature* σ,
- $\texttt{Verify}$, *on input* m, σ *and* $\texttt{vk}$, *outputs either* 0 *or* 1,

and the public parameters $\texttt{pp}$ *are implicit input to* $\texttt{Sign}$ *and* $\texttt{Verify}$.

We require the digital signature scheme to be *complete* (sometimes also referred to as *correct*), and to be secure against existential forgery under an adaptive chosen message attack, following the definitions from Goldwasser et al. [25].

Definition 10 (Completeness)**.** *We say that the digital signature scheme is* complete *if honestly generated signatures are accepted by the verification algorithm. Hence, we want that*

$$\Pr\left[\texttt{Verify}(m,\sigma,\texttt{vk})=1 \;:\; \begin{matrix}(\texttt{pp},\texttt{sk},\texttt{vk})\leftarrow\texttt{KeyGen}(1^\lambda)\\(\sigma)\leftarrow\texttt{Sign}(m,\texttt{sk})\end{matrix}\right]=1.$$

Definition 11 (Existential Forgeability)**.** *We say that the digital signature scheme is secure against* existential forgeability *if an adversary* $\texttt{A}$, *after given valid signatures* σ_i *of messages* m_i *of* $\texttt{A}$*'s choice from a signing oracle* $\texttt{O}_{\texttt{sign}}$, *cannot forge a signature on any new message under the same public-private key pair. Hence, we want that*

$$\Pr\left[\begin{matrix}\hat{m}\notin\{m_i\}\\ \texttt{Verify}(\hat{m},\hat{\sigma},\texttt{vk})=1\end{matrix} \;:\; \begin{matrix}(\texttt{pp},\texttt{vk})\leftarrow\texttt{KeyGen}(1^\lambda)\\(\hat{m},\hat{\sigma})\leftarrow\texttt{A}^{\texttt{O}_{\texttt{sign}}}(\texttt{pp},\texttt{vk},\{m_i\})\end{matrix}\right]\leq\epsilon(\lambda).$$

Here $\{m_i\}$ *is the set of messages signed by the signing oracle* $\texttt{O}_{\texttt{sign}}$.

2.11 Zero-Knowledge Proofs

These definitions are based on Goldwasser et al. [24]. Let $\texttt{L}$ be a language, and let $\texttt{R}$ be a relation on $\texttt{L}$. Then, x is an element in $\texttt{L}$, if there exists a witness w such that $(x,w)\in\texttt{R}$.

Definition 12 (Zero-Knowledge Proofs)**.** *An interactive zero-knowledge proof protocol* Π *consists of two parties: a prover* $\texttt{P}$ *and a verifier* $\texttt{V}$, *and a setup algorithm* ($\texttt{Setup}$), *where* $\texttt{Setup}$, *on input the security parameter* 1^λ, *outputs public setup parameters* $\texttt{sp}$. *The protocol consist of a transcript* $\texttt{T}$ *of the communication between* $\texttt{P}$ *and* $\texttt{V}$, *with respect to* $\texttt{sp}$, *and the conversation terminates with* $\texttt{V}$ *outputting either* 1 *or* 0. *Let* $\langle\texttt{P}(\texttt{sp},x,w),\texttt{V}(\texttt{sp},x)\rangle$ *denote the output of* $\texttt{V}$ *on input* x *after its interaction with* $\texttt{P}$, *who has a witness* w.

Definition 13 (Completeness)**.** *We say that a zero-knowledge proof protocol* Π *is* complete *if* $\mathtt{V}$ *outputs* 1 *when* $\mathtt{P}$ *knows a witness* w*. Hence, for any sampling algorithm* $\mathtt{P}_0$ *we want that*

$$\Pr\left[\langle \mathtt{P}(\mathtt{sp}, x, w), \mathtt{V}(\mathtt{sp}, x)\rangle = 1 \ : \ \begin{array}{c} \mathtt{sp} \leftarrow \mathtt{Setup}(1^\lambda) \\ (x, w) \leftarrow \mathtt{P}_0(\mathtt{sp}) \\ (x, w) \in \mathtt{R} \end{array}\right] = 1.$$

Definition 14 (Soundness)**.** *We say that a zero-knowledge proof protocol* Π *is* sound *if a cheating prover* $\mathtt{P}^*$ *that does not know a witness cannot convince* $\mathtt{V}$*. Hence, for any* x *not in the language* $\mathtt{L}$

$$\Pr\left[\langle \mathtt{P}^*(\mathtt{sp}, x, \cdot), \mathtt{V}(\mathtt{sp}, x)\rangle = 1 \ : \ \begin{array}{c} \mathtt{sp} \leftarrow \mathtt{Setup}(1^\lambda) \\ \forall x \notin \mathtt{L} \end{array}\right] \leq \frac{1}{2}.$$

Definition 15 (Honest-Verifier Zero-Knowledge)**.** *We say that a zero-knowledge proof protocol* Π *is* honest-verifier zero-knowledge *if a honest but curious verifier* $\mathtt{V}^*$ *that follows the protocol cannot learn anything else than the fact that* $x \in \mathtt{L}$*. Hence, we want, for real accepting transcript* $\mathtt{T}_{\langle \mathtt{P}(\mathtt{sp},x,w),\mathtt{V}(\mathtt{sp},x)\rangle}$ *between a prover* $\mathtt{P}$ *and a verifier* $\mathtt{V}$*, and a accepting transcript* $\mathtt{S}_{\langle \mathtt{P}(\mathtt{sp},x,\cdot),\mathtt{V}(\mathtt{sp},x)\rangle}$ *generated by simulator* $\mathtt{S}$ *that only knows* x*, that*

$$2 \cdot |\Pr\left[b = b' \ : \ \begin{array}{c} \mathtt{sp} \leftarrow \mathtt{Setup}(1^\lambda) \\ \mathtt{T}_1 = \mathtt{T}_{\langle \mathtt{P}(\mathtt{sp},x,w),\mathtt{V}(\mathtt{sp},x)\rangle} \leftarrow \Pi(\mathtt{sp}, x, w) \\ \mathtt{T}_2 = \mathtt{S}_{\langle \mathtt{P}(\mathtt{sp},x,\cdot),\mathtt{V}(\mathtt{sp},x)\rangle} \leftarrow \mathtt{S}(\mathtt{sp}, x) \\ b \xleftarrow{\$} \{0,1\}, \mathtt{T}' \leftarrow \mathtt{T}_b \\ b' \leftarrow \mathtt{V}^*(\mathtt{T}', \mathtt{sp}, x) \end{array}\right] - \frac{1}{2}| \leq \epsilon(\lambda).$$

2.12 Verifiable Random Functions

We give the definition of a verifiable random function based on the work by Micali et al. [36].

Definition 16 (Verifiable Random Functions)**.** *A verifiable random function scheme consists of three algorithms: key generation* (`KeyGen`)*, function evaluation* (`Eval`) *and verification* (`Verify`)*, where*

- `KeyGen`*, on input the security parameter* 1^λ*, outputs a public function* `f`*, an evaluation key* `sk`*, and a verification key* `vk`*,*
- `Eval`*, on input a element* x *and* `sk`*, outputs an evaluation* $y = \mathtt{f}(\mathtt{sk}, x)$ *and a proof* π*,*
- `Verify`*, on input* x*,* y*,* π *and* `vk`*, outputs either* 0 *or* 1*,*

and the public function $\mathtt{f}$ *is implicit input to* $\mathtt{Eval}$ *and* $\mathtt{Verify}$.

Definition 17 (Completeness)**.** *We say that a verifiable random function scheme is* complete *if the verification algorithm always accepts the result of a honest evaluation of the function. Hence, we want that*

$$\Pr\left[\mathtt{Verify}(x,y,\pi,\mathtt{vk})=1 \;:\; \begin{matrix}(\mathtt{f},\mathtt{sk},\mathtt{vk})\leftarrow\mathtt{KeyGen}(1^\lambda)\\(y,\pi)\leftarrow\mathtt{Eval}(x,\mathtt{sk})\end{matrix}\right]=1.$$

Definition 18 (Uniqueness)**.** *We say that a verifiable random function scheme is* uniquely provable *if an adversary* $\mathtt{A}$*, after creating an evaluation* y *of* x *together with a proof* π*, cannot find another valid evaluation* $\hat{y}$ *and proof* $\hat{\pi}$ *to* x*. Hence, we want that*

$$\Pr\left[\begin{matrix}y\neq\hat{y}\\ \mathtt{Verify}(x,y,\pi,\mathtt{vk})=1\\ \mathtt{Verify}(x,\hat{y},c,\hat{\pi},\mathtt{vk})=1\end{matrix} \;:\; \begin{matrix}(\mathtt{f},\mathtt{sk},\mathtt{vk})\leftarrow\mathtt{KeyGen}(1^\lambda)\\(x,y,\pi,\hat{y},\hat{\pi})\leftarrow\mathtt{A}(\mathtt{f},\mathtt{sk},\mathtt{vk})\end{matrix}\right]\leq\epsilon(\lambda).$$

Definition 19 (Pseudorandomness)**.** *We say that a verifiable random function scheme is* pseudorandom *if an adversary* $\mathtt{A}$*, after given valid evaluations* y_i *with proofs* π_i *of inputs* x_i *of* $\mathtt{A}$*'s choice from a evaluation oracle* $\mathtt{O}_{\mathtt{eval}}$*, for a known* $\mathtt{f}$*, cannot distinguish if a value* y *is a valid evaluation of a* x *of* $\mathtt{A}$*'s choice with respect to* $\mathtt{f}$*, or if* y *is a random string. Hence, we want that*

$$2\cdot\left|\Pr\left[\begin{matrix}x\notin\{x_i\}\\ b=b'\end{matrix} \;:\; \begin{matrix}(\mathtt{f},\mathtt{sk},\mathtt{vk})\leftarrow\mathtt{KeyGen}(1^\lambda)\\ \{(y_i,\pi_i)\}\leftarrow\mathtt{A}^{\mathtt{O}_{\mathtt{eval}}}(\mathtt{f},\mathtt{vk},\{x_i\})\\ x\leftarrow\mathtt{A}(\mathtt{f},\mathtt{vk},\{(x_i,y_i,\pi_i)\}),\\ (y',\pi)\leftarrow\mathtt{Eval}(x,\mathtt{sk}),\hat{y}\xleftarrow{\$}\{0,1\}^{\mathtt{len}(y')}\\ b\xleftarrow{\$}\{0,1\},y\leftarrow by'+(1-b)\hat{y}\\ b'\leftarrow\mathtt{A}(\mathtt{f},x,y)\end{matrix}\right]-\frac{1}{2}\right|\leq\epsilon(\lambda).$$

3 Pedersen Commitments and Schnorr Signatures

We present the two building blocks of our discrete logarithm vased subliminal-free digital signature scheme.

3.1 Pedersen Commitments

An example of a commitment scheme based on the discrete logarithm problem is the Pedersen commitment, introduced by Pedersen [41].

Definition 20 (Pedersen Commitments)**.** *Let $\langle g \rangle = G = \langle h \rangle$ be a Schnorr group of prime order q with generators g and h. The Pedersen commitment scheme consists of three algorithms: key generation* (`KeyGen`)*, commitment* (`Com`) *and opening* (`Open`)*, where*

- `KeyGen`*, on input the security parameter 1^λ, outputs public parameters g, h and G,*
- `Com`*, on input a message $m \in \mathbb{Z}_q$, outputs, for $r \xleftarrow{\$} \mathbb{Z}_q$, a commitment $c = g^m h^r$ and randomness r,*
- `Open`*, on input m, c and r, outputs 1 if $c \stackrel{?}{=} g^m h^r$, and 0 otherwise*

and the public parameters `pp` *are implicit input to* `Com` *and* `Open`*.*

This commitment scheme is computationally binding and unconditionally hiding, as long as solving the discrete logarithm of g to the base h is hard. However, if we always let $r = 0$, we get a commitment scheme that is unconditionally binding, but not hiding for arbitrary messages.

3.2 Schnorr Signatures

We will later combine Pedersen commitments with Schnorr signatures [44], which was proven secure in the random oracle model by Pointcheval and Stern [42].

Definition 21 (Schnorr Signatures)**.** *Let $G = \langle g \rangle$ be a Schnorr group of prime order q with generator g, and let $\mathtt{H} : G \times \{0,1\}^* \to \mathbb{Z}_q$ be a hash-function. The Schnorr digital signature scheme consists of three algorithms: key generation* (`KeyGen`)*, signing* (`Sign`) *and verification* (`Verify`)*, where*

- `KeyGen`*, on input the security parameter 1^λ, outputs public parameters g, q and G, a signing key $\mathtt{sk} \xleftarrow{\$} \mathbb{Z}_q$, and a verification key $\mathtt{vk} = g^{\mathtt{sk}}$,*
- `Sign`*, on input a message $m \in \{0,1\}^*$ and* `sk`*, outputs, for $r \xleftarrow{\$} \mathbb{Z}_q$, a signature $\sigma = (R, y) = (g^r, \mathtt{sk} \cdot \mathtt{H}(R, m) + r \mod q)$,*

- `Verify`, *on input* m, σ *and* `vk`, *outputs* 1 *if* $g^y \stackrel{?}{=} \mathtt{vk} \cdot R^{\mathtt{H}(R,m)}$, *and* 0 *otherwise,*

and the public parameters g, q *and* G *are implicit input to* `Sign` *and* `Verify`.

4 Lattice-Based Cryptography

Here we introduce the building blocks of our lattice-based verifiable random secret scheme and signature scheme.

4.1 Lattice-Based Commitments

We briefly present the lattice based commitment scheme by Baum et al. [4], and refer to the paper for more details. The commitment scheme come together with an efficient zero-knowledge proof of linear relations that will be useful later.

Definition 22 (Lattice-Based Commitments [4])**.** *Let* R_p *be the ring of polynomials modulo* X^N+1 *with integer coefficients modulo* p. *The lattice based commitment scheme consists of three algorithms: key generation* (`KeyGen`), *commitment* (`Com`) *and opening* (`Open`), *where*

- `KeyGen`, *on input the security parameter* 1^λ, *outputs a public matrix* $\boldsymbol{A}$ *such that*

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{a'} \end{bmatrix} = \begin{bmatrix} 1 & a_1 & a_2 \\ 0 & 1 & a_3 \end{bmatrix}, \text{where } a_1, a_2, a_3 \xleftarrow{\$} R_p,$$

- `Com`, *on input a message* $m \in R_p$, *samples an* $\boldsymbol{r} \xleftarrow{\$} R_p^3$ *where* $\|\boldsymbol{r}\|_\infty = 1$, *and computes*

$$\boldsymbol{c} = \mathtt{Com}(m; \boldsymbol{r}) = \boldsymbol{A} \cdot \boldsymbol{r} + \begin{bmatrix} 0 \\ m \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix},$$

and returns $\boldsymbol{c}$ *and* $d = (m; \boldsymbol{r}, 1)$,

- `Open`, *on input* $(m, \boldsymbol{r}, f)$ *with* $f \in \bar{\mathcal{C}}$, *verifies the opening by checking if*

$$f \cdot \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \stackrel{?}{=} \boldsymbol{A} \cdot \boldsymbol{r} + f \cdot \begin{bmatrix} 0 \\ m \end{bmatrix},$$

and that $\|r_i\| \leq 4\sigma\sqrt{N}$ *for* $\boldsymbol{r} = (r_0, r_1, r_2)$ *with* $\sigma = 11 \cdot \nu \cdot \sqrt{3N}$. *It outputs* 1 *if all these conditions holds, and* 0 *otherwise. The challenge space* $\bar{\mathcal{C}}$ *is defined as in Section 2.4*

4.2 Lattice-Based Zero-Knowledge Proofs

We present two zero-knowledge protocols later to be used in our lattice based verifiable random secret scheme.

Definition 23 (Zero-Knowledge Proof of Linear Relations)**.** *Define the following three commitments:*

$$[x_1] = \mathtt{Com}(x_1; \boldsymbol{r}) = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix},$$
$$[x_2] = \mathtt{Com}(x_2; \boldsymbol{r}') = \begin{bmatrix} c_1' \\ c_2' \end{bmatrix},$$
$$[x_3] = \mathtt{Com}(x_3; \boldsymbol{r}'') = \begin{bmatrix} c_1'' \\ c_2'' \end{bmatrix}.$$

Let $[x_1], [x_2]$ *and* $[x_3]$ *be such that* $x_3 = \alpha_1 x_1 + \alpha_2 x_2$ *for some* $\alpha_1, \alpha_2 \in R_p$. *Then* Π_{Sum} *in Figure 3 is a zero-knowledge proof of knowledge (ZKPoK) of this fact (it is an adapted version of the linearity proof in [4]), and Figure 4 is the verification algorithm for the proof. This protocol can easily be extended to prove the linear relation between more than three elements by increasing the dimension of the protocol to be the number of summands times the size of each commitment. Let* $\pi \leftarrow \Pi_{Sum}(([x_1], [x_2]), [x_3], (\alpha_1, \alpha_2))$ *denote the run of the* Π_{Sum}*-protocol to prove the relation* $x_3 = \alpha_1 x_1 + \alpha_2 x_2$ *producing a proof* $\pi = ((t, t', t''), \beta, (\boldsymbol{z}, \boldsymbol{z}', \boldsymbol{z}''))$. *If* $\alpha_1 = \alpha_2 = 1$, *then the scalars are omitted in the input. Let* $0 \vee 1 \leftarrow \Pi_{SumV}(([x_1], [x_2]), [x_3], (\alpha_1, \alpha_2), \pi)$ *denote the verification of this proof.*

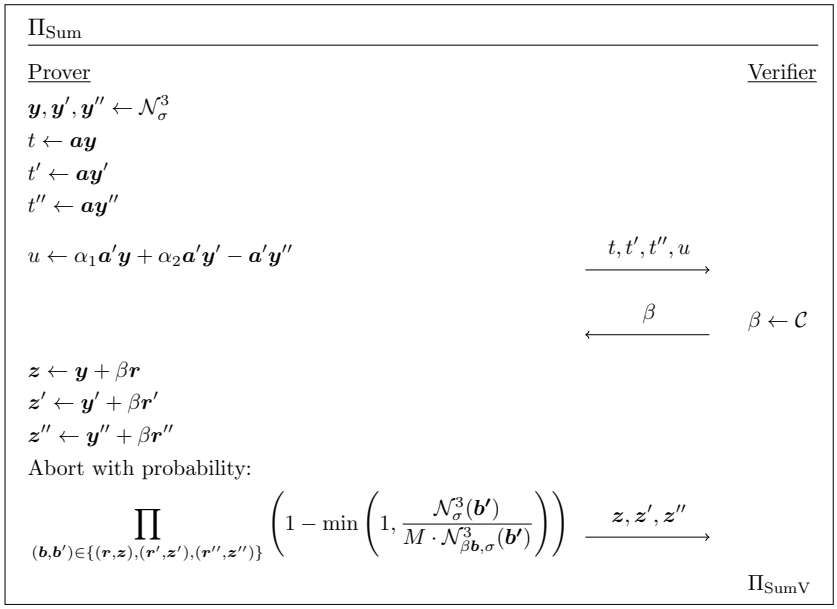


Figure 3: Protocol Π_{Sum} is a zero-knowledge protocol to prove the relation $x_3 = \alpha_1 x_1 + \alpha_2 x_2$, given the commitments $[x_1], [x_2], [x_3]$ and the scalars α_1, α_2.

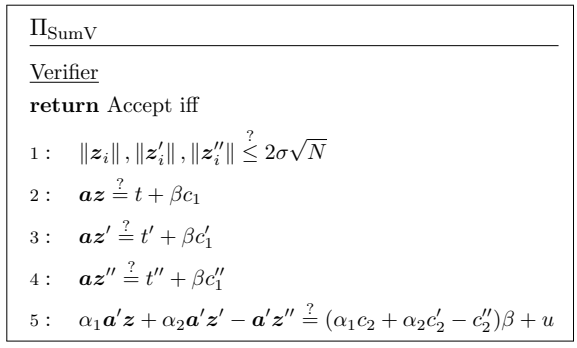


Figure 4: Verification step for the Π_{Sum} protocol.

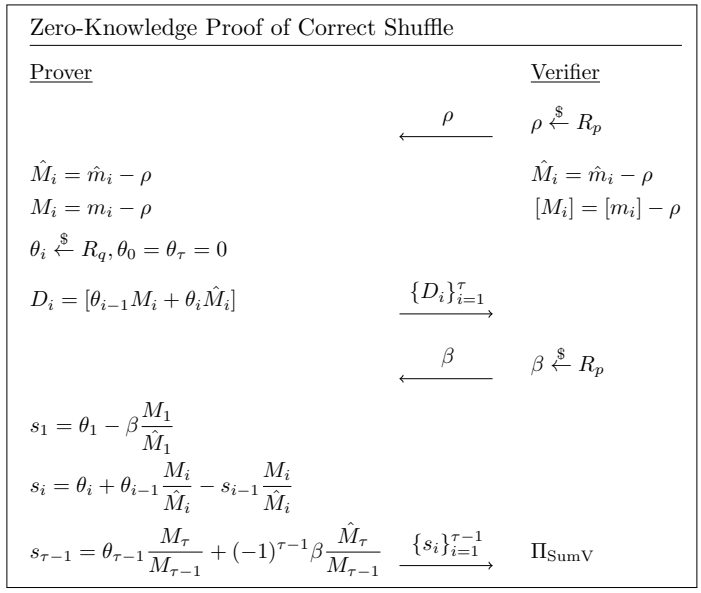


Figure 5: The public-coin zero-knowledge protocol of correct shuffle.

Definition 24 (Zero-Knowledge Proof of Correct Shuffle)**.** *In the unpublished work by Baum et al. [5] they give an efficient protocol* $\Pi_{Shuffle}$ *for a Neff-like shuffle of known values [38] for the lattice-based commitments by Baum et al. [4]. Given a list of elements* $(\hat{M}_1, \hat{M}_2, \ldots, \hat{M}_\tau)$ *from* R_p *and commitments* $(c_1, c_2, \ldots, c_\tau)$*, we can prove that the* c_i*'s are commitments to the* $\hat{M}_{\gamma(i)}$*'s, for some secret permutation* γ *of the indices. Let* $\pi \leftarrow \Pi_{Shuffle}(\{\hat{M}_i\}, \{c_i\}, \gamma)$ *denote the run of the shuffle-protocol, with proof* π*. Let* $0 \vee 1 \leftarrow \Pi_{ShuffleV}(\{\hat{M}_i\}, \{c_i\}, \gamma)$ *denote the verification of this proof.*

4.3 Lattice-Based Signatures

The most efficient lattice based signature schemes are based on the Schnorr-like signatures by Lyubashevsky [33, 34]. However, the zero-knowledge proof of opening given by Baum et al. [4] follows this exact structure. Let R_p be the message space, let H be a hash function $H : R_p \times R_p \to \mathcal{C}$ and let $\mathtt{vk} = \boldsymbol{c}$ be the public verification key,

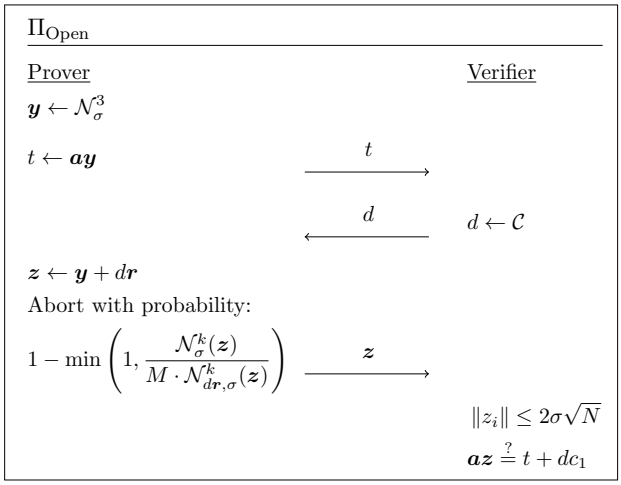


Figure 6: Zero-Knowledge Proof of Knowledge of Opening of $\boldsymbol{c} = \texttt{Com}(x; \boldsymbol{r})$.

where $\boldsymbol{c} = \texttt{Com}(0; \boldsymbol{r})$ is a commitment to 0 with randomness $\boldsymbol{r}$, and let the secret key $\texttt{sk} = \boldsymbol{r}$. Then the protocol Π_{Open} in Figure 6 can be turned into a signature scheme by applying the Fiat-Shamir transform where $d = H(t, m)$, for a message $m \in R_p$. Let $\sigma = (t, \boldsymbol{z}) \leftarrow \texttt{Sign}(m, \texttt{sk})$ denote the run of the signature algorithm with signature σ, and, further, let $0 \vee 1 \leftarrow \texttt{Verify}(\texttt{pk}, m, \sigma))$ denote the verification of this signature.

5 Verifiable Random Secrets

A VRF has a lot of useful properties. However, the constructions are based on the fact that you make the randomness public for anyone to verify. This makes it useless for e.g. digital signature schemes, which require the randomness to be secret to provide security. On the other hand, the verifier must be able to verify that the random value is generated in some certain unpredictable way, to avoid that the prover intentionally can choose randomness to his own advantage. One could imagine this being done in a zero-knowledge proof. The problem in this case is that we want to generate some random element

we want to use later, and hence, we cannot just prove that we have generated a random element, but also have to provide information that can be included in a subsequent application.

A verifiable random secret (VRS) scheme is an interactive protocol where the prover wants to produce a secret random value, and publish a commitment of the value together with a proof to convince the verifier that the secret is randomly generated. This can be done in the following way: first the prover commits to a random value and sends the commitment to the verifier. The verifier then returns a random challenge to the prover, which he in turn uses to generate a final commitment and a proof. The commitment contains a random value which was unpredictable for the prover until he got the challenge, and is secret to the verifier even after the protocol is completed. Anyone with access to the final commitment and the proof can check that the commitment was generated in a proper way, and hence, will be convinced that the content is random.

We want the VRS to have similar security properties as the VRF and the ZKP. Therefore, we adapt some of the security notions from VRFs and ZKPs, but change them slightly to fit the new setting.

Definition 25 (Verifiable Random Secrets)**.** *A Verifiable Random Secret-scheme (VRS) consists of a function* $r(\cdot,\cdot)$*, an interactive protocol seed* ($\Pi_{\texttt{Seed}}$) *and five algorithms: setup* (`Setup`)*, commit* (`Com`)*, challenge* (`Challenge`)*, generation* (`Generate`) *and check* (`Check`)*, where*

- `Setup`*, on input the security parameter* 1^λ*, outputs public parameters* `sp`*,*
- $\Pi_{\texttt{Seed}}$*, on input* `sp`*, outputs a random seed* s*,*
- `Com`*, on input a seed* s*, outputs a commitment* $\tilde{c}$ *of* s *and an opening* $\tilde{d}$*,*
- `Challenge`*, on no input, outputs a random challenge* t*,*
- `Generate`*, on input a commitment* $\tilde{c}$*, an opening* $\tilde{d}$ *and a challenge* t*, outputs a commitment* c*, an opening* d *of* c *(containing* $r = r(s,t)$*) and a proof* π*,*
- `Check`*, on input* $\tilde{c}$ *and* c*, challenge* t*, and proof* π*, outputs* 0 *or* 1*,*

and the public parameters $\mathtt{sp}$ *are implicit input to all algorithms following* $\mathtt{Setup}$.

Remark 1. *In the interactive protocol* $\Pi_{\mathtt{Seed}}$ *there may be one or more participants. Any participant of the protocol may be dishonest during the seed generation. The output seed may be given to any or all participants.*

The first thing we require from a VRS is that it is complete.

Definition 26 (Completeness)**.** *We say that the VRS is* complete *if a prover* $\mathtt{P}$ *always can convince a honest verifier* $\mathtt{V}$ *that a honestly generated proof is valid. Hence, we want that*

$$\Pr\left[\mathtt{Check}(\tilde{c},t,c,\pi)=1 \;:\; \begin{array}{c}\mathtt{sp}\leftarrow\mathtt{Setup}(1^\lambda)\\ s\leftarrow\Pi_{\mathtt{Seed}}(\mathtt{sp})\\ (\tilde{c},\tilde{d})\leftarrow\mathtt{Com}(s)\\ t\leftarrow\mathtt{Challenge}(\cdot)\\ (c,d,\pi)\leftarrow\mathtt{Generate}(\tilde{c},\tilde{d},t)\end{array}\right]=1.$$

The interactive VRS is visualized in Figure 7. Further, we use games to define the security of the scheme, and continue by defining *Binding*, *Prover bit-Unpredictability*, *Verifier Secrecy* and *Honest-Verifier Secrecy* for a VRS.

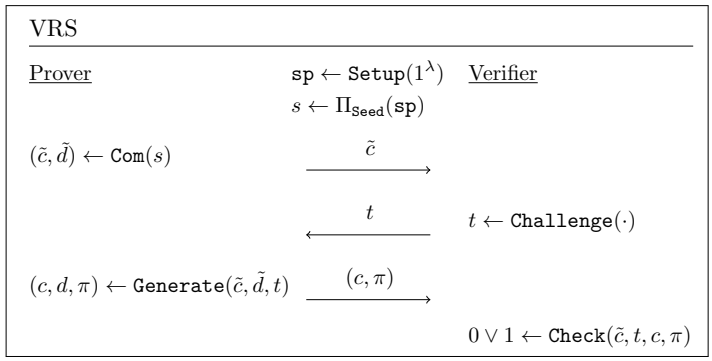


Figure 7: Our abstract verifiable random secret scheme.

Definition 27 (Binding)**.** *We say that the VRS is* binding*, if a cheating prover* $\mathtt{P}^*$ *cannot find a new opening* $\hat{d}\neq d$ *and a new commitment*

$\hat{c} \neq c$ accepted together with the proof π, where (c, d, π) was generated in a honest run of the protocol, depending on the commitment $\tilde{c}$ and the challenge t. We define the binding advantage $\mathbf{Adv}^{\mathbf{B}}$ of the cheating prover $\mathtt{P}^*$ to be:

$$\mathbf{Adv}^{\mathbf{B}}(\mathtt{P}^*) = \Pr\left[\begin{array}{c} c \neq \hat{c}, d \neq \hat{d} \\ \mathtt{Check}(\tilde{c}, t, c, \pi) = 1 \\ \mathtt{Check}(\tilde{c}, t, \hat{c}, \pi) = 1 \end{array} : \begin{array}{c} s \leftarrow \Pi_{\mathtt{Seed}}(\mathtt{sp}) \\ (\tilde{c}, \tilde{d}) \leftarrow \mathtt{P}^*(\cdot) \\ t \leftarrow \mathtt{Challenge}(\cdot) \\ (c, d, \pi) \leftarrow \mathtt{Generate}(\tilde{c}, \tilde{d}, t) \\ (\hat{c}, \hat{d}) \leftarrow \mathtt{P}^*(s, \tilde{c}, \tilde{d}, t, c, d, \pi) \end{array}\right].$$

Definition 28 (Prover bit-Unpredictability)**.** *We say that the VRS has* prover bit-unpredictability *if, given a balanced predicate function f of a cheating prover $\mathtt{P}^*$'s choice, the predicate $f(r)$ of the value r, where r is a function of the seed s and the challenge t, is unpredictable for $\mathtt{P}^*$. We let $\mathtt{P}^*$ choose a bit $\hat{b}$, and define the prover bit-unpredictability advantage $\mathbf{Adv}_{\mathbf{f}}^{\mathbf{PbU}}$ of a cheating prover $\mathtt{P}^*$, with respect to f, to be:*

$$\mathbf{Adv}_{\mathbf{f}}^{\mathbf{PbU}}(\mathtt{P}^*) = 2 \cdot |\Pr\left[\begin{array}{c} \mathtt{Check}(\tilde{c}, t, c, \pi) = 1 \wedge f(r) = \hat{b} \\ \vee \\ \mathtt{Check}(\tilde{c}, t, c, \pi) = 0 \wedge \tilde{b} = 1 \end{array} : \begin{array}{c} s \leftarrow \Pi_{\mathtt{Seed}}(\mathtt{sp}) \\ (\tilde{c}, \tilde{d}, f, \hat{b}) \leftarrow \mathtt{P}^*(\cdot) \\ t \leftarrow \mathtt{Challenge}(\cdot) \\ (c, d, r, \pi) \leftarrow \mathtt{P}^*(\tilde{c}, \tilde{d}, t, s) \\ \tilde{b} \xleftarrow{\$} \{0,1\} \end{array}\right] - \frac{1}{2}|.$$

Definition 29 ((Strong) Verifier Secrecy)**.** *We say that the VRS has* strong verifier secrecy*, if a cheating verifier $\mathtt{V}^*$ is unable to distinguish between a honestly generated value r and a random r sampled from the set $\mathtt{R} = \{r(s,t) : s \leftarrow \Pi_{\mathtt{Seed}}, t \leftarrow \mathtt{Challenge}\}$, where r is a function of the seed s and the challenge t. We define the (strong) verifier secrecy advantage $\mathbf{Adv}^{(\mathbf{s})\mathbf{VS}}$ of a cheating verifier $\mathtt{V}^*$ to be:*

$$\mathbf{Adv}^{(\mathbf{s})\mathbf{VS}}(\mathtt{V}^*) = 2 \cdot |\Pr\left[b = \hat{b} \ : \begin{array}{c} b \xleftarrow{\$} \{0,1\} \\ s \leftarrow \Pi_{\mathtt{Seed}}(\mathtt{sp}) \\ (\tilde{c}, \tilde{d}) \leftarrow \mathtt{Com}(s) \\ t \leftarrow \mathtt{V}^*(\tilde{c}) \\ (c, d, \pi) \leftarrow \mathtt{Generate}(\tilde{c}, \tilde{d}, t) \\ r_0 \leftarrow r(s,t), r_1 \xleftarrow{\$} \mathtt{R} \\ \hat{b} \leftarrow \mathtt{V}^*(\tilde{c}, t, c, \pi, r_b) \end{array}\right] - \frac{1}{2}|.$$

Definition 30 ((Strong) Honest-Verifier Secrecy)**.** *We say that the VRS has* strong honest-verifier secrecy*, if a honest but curious verifier $\mathtt{V}^*$ is unable to distinguish between a honestly generated value r and a random r sampled from the set $\mathtt{R} = \{r(s,t) : s \leftarrow \Pi_{\mathtt{Seed}}, t \leftarrow \mathtt{Challenge}\}$, where r is a function of the seed s and the challenge t.*

We define the (strong) honest-verifier secrecy advantage $\mathbf{Adv}^{\mathbf{(s)HVS}}$ *of a honest verifier* $\mathtt{V}^*$ *to be:*

$$\mathbf{Adv}^{\mathbf{(s)HVS}}(\mathtt{V}^*) = 2 \cdot |\Pr\left[b = \hat{b} \;:\; \begin{array}{c} b \xleftarrow{\$} \{0,1\} \\ s \leftarrow \Pi_{\mathtt{Seed}}(\mathtt{sp}) \\ (\tilde{c}, \tilde{d}) \leftarrow \mathtt{Com}(s) \\ t \leftarrow \mathtt{Challenge}(\cdot) \\ (c, d, \pi) \leftarrow \mathtt{Generate}(\tilde{c}, \tilde{d}, t) \\ r_0 \leftarrow r(s,t), r_1 \xleftarrow{\$} \mathtt{R} \\ \hat{b} \leftarrow \mathtt{V}^*(\tilde{c}, t, c, \pi, r_b) \end{array}\right] - \frac{1}{2}|.$$

Definition 31 ((Weak) Verifier Secrecy)**.** *We say that the VRS has* weak verifier secrecy *if, given a balanced predicate function* f *of a cheating verifier* $\mathtt{V}^*$*'s choice, the predicate* $f(r)$ *of the value* r*, where* r *is a function of the seed* s *and the challenge* t*, is unpredictable for* $\mathtt{V}^*$*. We let* $\mathtt{V}^*$ *choose a bit* $\hat{b}$*, and define the (weak) verifier secrecy advantage* $\mathbf{Adv}_{\mathbf{f}}^{\mathbf{(w)VS}}$ *of a cheating verifier* $\mathtt{V}^*$*, with respect to* f*, to be:*

$$\mathbf{Adv}_{\mathbf{f}}^{\mathbf{(w)VS}}(\mathtt{V}^*) = 2 \cdot |\Pr\left[\begin{array}{c} (\mathtt{Check}(\tilde{c}, t, c, \pi) = 1 \\ \wedge f(r) = \hat{b}) \\ \vee \\ (\mathtt{Check}(\tilde{c}, t, c, \pi) = 0 \\ \wedge \tilde{b} = 1) \end{array} \;:\; \begin{array}{c} s \leftarrow \Pi_{\mathtt{Seed}}(\mathtt{sp}) \\ (\tilde{c}, \tilde{d}) \leftarrow \mathtt{Com}(s) \\ (t, f) \leftarrow \mathtt{V}^*(\tilde{c}) \\ (c, d, \pi) \leftarrow \mathtt{Generate}(\tilde{c}, \tilde{d}, t) \\ \hat{b} \leftarrow \mathtt{V}^*(\tilde{c}, t, c, \pi) \\ \tilde{b} \xleftarrow{\$} \{0,1\} \end{array}\right] - \frac{1}{2}|.$$

Definition 32 ((Weak) Honest-Verifier Secrecy)**.** *We say that the VRS has* weak honest-verifier secrecy *if, given a balanced predicate function* f *of an honest verifier* $\mathtt{V}^*$*'s choice, the predicate* $f(r)$ *of the value* r*, where* r *is a function of the seed* s *and the challenge* t*, is unpredictable for a honest but curious verifier* $\mathtt{V}^*$*. We let* $\mathtt{V}^*$ *choose a bit* $\hat{b}$*, and define the (weak) honest-verifier secrecy advantage* $\mathbf{Adv}_{\mathbf{f}}^{\mathbf{(w)HVS}}$ *of a honest verifier* $\mathtt{V}^*$*, with respect to* f*, to be:*

$$\mathbf{Adv}_{\mathbf{f}}^{\mathbf{(w)HVS}}(\mathtt{V}^*) = 2 \cdot |\Pr\left[\begin{array}{c} \mathtt{Check}(\tilde{c}, t, c, \pi) = 1 \\ \wedge f(r) = \hat{b}) \end{array} \;:\; \begin{array}{c} s \leftarrow \Pi_{\mathtt{Seed}}(\mathtt{sp}) \\ (\tilde{c}, \tilde{d}) \leftarrow \mathtt{Com}(s) \\ t \leftarrow \mathtt{Challenge}(\cdot), f \leftarrow \mathtt{V}^*(\tilde{c}, t) \\ (c, d, \pi) \leftarrow \mathtt{Generate}(\tilde{c}, \tilde{d}, t) \\ \hat{b} \leftarrow \mathtt{V}^*(\tilde{c}, t, c, \pi) \end{array}\right] - \frac{1}{2}|.$$

Remark 2. *We note a cheating verifier* $\mathtt{V}^*$ *always can be turned into a honest verifier if* $\mathtt{V}^*$ *have to commit to his challenge before he receives the commitment of the seed. This would increase the communication complexity of the protocol, but at the same time make it easier to prove security.*

Remark 3. *For the honest-verifier secrecy we want the outcome to be unpredictable for a cheating verifier, and if he is unable to predict a bit of the random value then we have achieved what we want and weak honest-verifier secrecy. If the verifier is unable to distinguish between a randomly generated and a honestly generated random value, as in the strong honest-verifier secrecy, then he is unable to predict any bits of the honestly generated value. That is, strong honest-verifier secrecy implies weak honest-verifier secrecy.*

Definition 33 (ϵ-Security)**.** *A VRS is said to be ϵ-secure, for an ϵ negligible in the security parameter λ, if all of the advantages is less than ϵ. That is, we have ϵ-security if*

$$\mathbf{Adv}^{\mathbf{S}}(\mathtt{P}^*) \leq \epsilon(\lambda), \quad \mathbf{Adv}_{\mathbf{f}}^{\mathbf{PbU}}(\mathtt{P}^*) \leq \epsilon(\lambda), \quad \mathbf{Adv}_{\mathbf{f}}^{\mathbf{HVS}}(\mathtt{V}^*) \leq \epsilon(\lambda),$$

where $\mathbf{Adv}_{\mathbf{f}}^{\mathbf{HVS}}(\mathtt{V}^*)$ *could be* $\mathbf{Adv}_{\mathbf{f}}^{(\mathbf{w})\mathbf{HVS}}(\mathtt{V}^*)$ *or* $\mathbf{Adv}_{\mathbf{f}}^{(\mathbf{s})\mathbf{HVS}}(\mathtt{V}^*)$.

Remark 4. *We note that in this protocol, both the prover and the verifier are incentivised to be honest with regards to drawing properly random elements for the seed s and the challenge t, respectively. For the former, the prover wants to keep his secret value unpredictable for the verifier, so that he cannot guess the randomness used in the subsequent application. For the latter, the verifier wants to make it infeasible for the prover to control the outcome and infeasible to give a fake proof of the randomness used in the subsequent application.*

Remark 5. *We also note that the only contribution of the verifier (before verifying the proof) is to provide some randomness, and hence, the protocol is public coin. It follows that we can use the Fiat-Shamir heuristics to make the protocol non-interactive by letting $t = H(\mathtt{sp}, \tilde{c})$ (and s if it is public), for a hash-function H. In this scenario, the prover sends the transcript $(\tilde{c}, c, \pi)$ to the verifier, and store c, d himself. The verifier then run the* `Check` *algorithm as usual to make sure that everything is correct. However, not that in this case we lose Prover bit-Unpredictability, where $\mathtt{P}^*$ can try as many times that he wants before he sends the proof, and hence, he's able to control a few bits of r. This may or may not be important for the subsequent application.*

6 Subliminal-Free Digital Signatures

6.1 How to Achieve a Subliminal-Free Channel?

Let S be the sender, R be the receiver and W the warden. We consider S, R and W to all be probabilistic polynomial time algorithms. Let $\mathcal{C}$ be an information channel controlled by W, allowing S to send information to R. We want to define what it means for $\mathcal{C}$ to be a subliminal-free channel, and in particular, what it means for $\mathcal{C}$ to be a subliminal-free channel when $\mathcal{C}$ is a message authentication without secrecy channel. That is, a covert-free channel where messages from S are sent in the clear to R together with a signature. Then R can be sure that the message indeed is from S. Further, W want to be sure that there is no extra hidden information being sent. There are a lot of ways to encode information into a message-signature pair, where we are only interested preventing methods that encodes information into the signature, see the Warden Model in Section 1.1.

In our model S can hide a subliminal message in the signature, in which R later can extract using some pre-shared information. Signatures can be either deterministic or probabilistic. If the signature scheme has been made deterministic by derandomizing it, W could require S to prove in zero-knowledge that the randomness used in the signature is deterministically generated. If the signature scheme does not use any randomness, and is deterministic, S is not required to prove anything. In the case of probabilistic signatures, a subliminal signer could easily choose a subliminal message (or an encryption of a subliminal message) to be the randomness used in the signature, independent of the message being sent. Hence, if allowing probabilistic signature schemes, we must be able to restrict S's control over the choice of randomness used in the signatures. By using a VRS we are able to do this. Also in this case we require a proof stating that the signature is honestly generated.

Lastly, we require $\mathcal{C}$ to be a reliable channel. In some cases the honestly generated randomness used in the signature is equal to the subliminal message S wants to send. This could happen if the subliminal messages is very short, or divided into small pieces of sizes down to only 1 bit per signature. If S gets to decide if he want to

send the authenticated message or not, after given the message to send, or after interaction with W, or after signature-generation, then this can be used to create a subliminal channel. For every signature, S checks if his subliminal message is embedded or not; if it is, then S sends the message, otherwise he aborts and tries again. This was pointed out by Desmedt [16]. It follows that the warden cannot allow S to abort if he wants the channel to be subliminal-free. However, if S is using a probabilistic signature scheme and is allowed to generate the signatures in a verifiable but non-interactive manner, he will be able to abort without W noticing. This seems to be inherit in the construction, and we will thereby give two different definitions of subliminal-free digital signatures, where the relaxed notion will allow a small subliminal channel in the construction, given that S have to work exponentially hard to achieve this.

6.2 Subliminal and Subliminal-Free Digital Signatures

We continue by more informally define what we mean by subliminal and subliminal-free digital signature schemes.

Definition 34 (Subliminal Digital Signatures)**.** *Let m be a fixed message, $\mathcal{S}$ a signature scheme, $\hat{m}$ a subliminal message chosen by* $\mathtt{S}^*$ *and s some secret pre-shared information between* $\mathtt{S}^*$ *and* $\mathtt{R}^*$*. Then we say that $\mathcal{S}$ is a subliminal digital signature scheme if* $\mathtt{S}^*$ *can encode $\hat{m}$ into a signature σ using an algorithm* $\mathtt{Encode}_{\mathtt{S}^*}$*, where σ is a valid signature of m with respect to $\mathcal{S}$, that can be decoded by* $\mathtt{R}^*$ *using an algorithm* $\mathtt{Decode}_{\mathtt{R}^*}$ *but cannot be decoded nor detected by* W*. That is,*

- $\mathtt{Encode}_{\mathtt{S}^*}$ *on input the message m, signing key* sk*, and subliminal message $\hat{m}$, outputs a valid signature σ of the message m and a valid proof π,*
- $\mathtt{Decode}_{\mathtt{R}^*}$ *on input a message m, a signature σ, and secret pre-shared information s, outputs the subliminal message $\hat{m}$.*

Definition 35 (Subliminal-Free Digital Signatures)**.** *Let m be a fixed message, $\mathcal{S}$ a signature scheme, $\hat{m}$ a subliminal message chosen by* S *and s some secret pre-shared information between* S *and* R*. Further, let σ be a valid signature of m with respect to $\mathcal{S}$ and let π be a proof of*

correctness, both potentially generated by interaction between S *and* W*. Assume that* W *will close the channel if* S *deviates from the protocol. Then we say that* $\mathcal{S}$ *is a subliminal-free digital signature scheme if* S *cannot reliably both create a proof* π *accepted by* W *and at the same time encode a subliminal message* $\hat{m}$ *into* σ *that can be decoded by* R *but cannot be decoded nor detected by* W*.*

However, as noted earlier, if $\mathcal{S}$ is a probabilistic signature scheme, then S can undetectably abort the protocol after generating a signature on m and restart the signing algorithm. We therefore additionally include a relaxed notion of a subliminal-free digital signature scheme, where we allow a small subliminal channel.

Definition 36 (Subliminal l-Free Digital Signatures)**.** *Let* m *be a fixed message,* $\mathcal{S}$ *a signature scheme,* $\hat{m}$ *a subliminal message chosen by* S *and* s *some secret pre-shared information between* S *and* R*. Further, let* σ *be a valid signature of* m *with respect to* $\mathcal{S}$ *and let* π *be a proof of correctness, both potentially generated by interaction between* S *and* W*. Assume that* W *will close the channel if* S *deviates from the protocol. Then we say that* $\mathcal{S}$ *is a subliminal* l*-free digital signature scheme if* S *cannot reliably both create a proof* π *accepted by* W *and at the same time encode a subliminal message* $\hat{m}$ *of size greater or equal to* l *into* σ *that can be decoded by* R *but cannot be decoded nor detected by* W*; unless* S *does work exponentially in* l*.*

6.3 Subliminal-Free Digital Signature Scheme

Deterministic signatures seems to have an advantage over probabilistic signatures as they can guarantee a subliminal-free digital signature scheme in the non-interactive setting. However, they are usually harder to construct and even harder to prove being evaluated correctly. Probabilistic signatures schemes are used more in practice, and we want to extend our definition of subliminal-free digital signature schemes to allow interaction between S and W to ensure that we can achieve the strongest security guarantees also for these schemes.

Definition 37 (Subliminal-Free Digital Signature Scheme)**.** *A subliminal-free digital signature scheme consists of an interactive signing*

protocol ($\Pi_{\texttt{Sign}}$) *and five algorithms: key generation* (`KeyGen`), *setup* (`Setup`), *verification* (`Verify`), *and checking* (`Check`), *where*

- `KeyGen`, *on input the security parameter* 1^λ, *outputs public parameters* `pp`, *a signing key* `sk`, *and a verification key* `vk`,
- `Setup`, *on input the security parameter* 1^λ, *outputs public parameters* `sp`,
- $\Pi_{\texttt{Sign}}$, *on input a message* m *and* `sk`, *outputs a signature* σ *and a proof* π,
- `Verify`, *on input* m, σ *and* `vk`, *outputs either* 0 *or* 1,
- `Check`, *on input* m, σ, vk *and* π, *outputs either* 0 *or* 1,

and the public parameters `pp` *are implicit input to all algorithms following* `KeyGen` *and the public setup parameters* `sp` *are implicit input to* `Sign` *and* `Check`. *We require that* `Check` *returns* 1 *if and only if* `Verify` *returns* 1 *and* π *is valid.*

In our constructions, see Section 7, we use an interactive VRS as a part of the signature protocol to ensure that the randomness are honestly generated and give a proof of this statement. The checking algorithm is then the checking algorithm of the VRS combined with the verification algorithm of the signature scheme.

Remark 6. *Note that there are some small differences in how we define subliminal-free digital signatures compared to how it is defined by Bohli et al. [8]. They require it to be exponentially hard in the security parameter to find two different pairs of signatures and proofs that are valid for the same message. This is essentially the same as in our case when the signature scheme is deterministic, but we also allow for a probabilistic signature scheme. In addition, they have a* `SFKeyGen` *algorithm generating the checking information for warden. We have decided to include an algorithm* `Setup` *instead. The setup algorithms generates publicly available information used by the sender to give a proof of correct signing, and used by the warden to verify this proof. The proof itself should only be available to the warden, as it can contain subliminal information that can be decoded if shared with the receiver, but the setup parameters could be available to anyone.*

6.4 Subliminal-Free Digital Signatures with Pre-Processing

The subliminal-free digital signature scheme require `S` and `W` to interact every time `S` wants to sign a message. We can lower the communication complexity of our scheme by allowing some pre-processing between `S` and `W`. In this case we make the `Setup` algorithm into an interactive protocol $\Pi_{\texttt{Setup}}$, where `S` and `W` use the interactive VRS to generate lots of commitments to some randomness used in the signatures, together with proofs that the randomness is honestly generated. Then the commitment and the randomness is also an input to the signature scheme. As the randomness is independent of the message to be signed, this allows us to move all the extra communication from the signature procedure to the setup procedure, which can be executed before any messages need to be sent over the channel.

Note that in this case we get a stateful digital signature scheme, as the warden must ensure that the randomness is used in the correct order. Otherwise this would open a subliminal channel where the signer can chose the randomness from a set of values dependent on the subliminal message he wants to send.

Remark 7. *Note that in this case the setup parameters* `sp` *are still public, while the output of the VRS is not.*

6.5 Non-Interactive Subliminal-Free Digital Signatures

We can make a trade-off between communication and security by changing the interactive subliminal-free digital signature scheme into a non-interactive scheme by letting all the parts of the scheme be non-interactive algorithms. This would also enforce the VRS to be a non-interactive algorithm, as a part of the signing algorithm. We know from earlier that a non-interactive VRS is not bit-unpredictable. In this case we get a l-subliminal channel, as the sender can run the non-interactive VRS until he gets a l-bit subliminal message encoded into the randomness used in the signature, and hence, we have a l-subliminal signature scheme.

6.6 Security of Subliminal-Free Digital Signatures

We require that the subliminal-free digital signature scheme has the same security as a normal signature scheme, plus the additional requirement that a cheating prover cannot decide the randomness used in the signature, and that a cheating verifier can't use the proof of correctness to forge signatures. Hence, we want the subliminal-free digital signature scheme to be *complete*, *sound* and secure against *existential forgery*.

Definition 38 (Complete)**.** *We say that a subliminal-free digital signature scheme is* complete *if honestly generated pairs of signatures and proofs are accepted by both the verification and checking algorithms. Hence, we want that*

$$\Pr\left[\begin{array}{l}\mathtt{Verify}(m,\sigma,\mathtt{vk})=1\\ \mathtt{Check}(m,\sigma,\mathtt{vk},\pi)=1\end{array} : \begin{array}{c}(\mathtt{pp},\mathtt{sk},\mathtt{vk})\leftarrow\mathtt{KeyGen}(1^\lambda)\\ \mathtt{sp}\leftarrow\mathtt{Setup}(1^\lambda)\\ (\sigma,\pi)\leftarrow\mathtt{Sign}(m,\mathtt{sk})\end{array}\right]=1.$$

Definition 39 (Soundness)**.** *A subliminal-free signature scheme is* sound *if the sender is unable to establish a* subliminal bit channel *with the receiver. A subliminal bit channel is a channel where a subliminal signer* $\mathtt{S}^*$ *given a message* m *cannot encode a* subliminal bit $\hat{m}\in\{0,1\}$ *into a valid signature* σ *of* m*, that the subliminal receiver* $\mathtt{R}^*$ *can decode, when additionally producing a proof* π *of correctness accepted by the warden* $\mathtt{W}$ *with probability nonnegligible greater than one-half. That is, the subliminal-free signature scheme is sound if*

$$\Pr\left[\begin{array}{l}\mathtt{Decode}_{\mathtt{R}^*}(m,\sigma,s)=\hat{m}\\ \mathtt{Verify}(m,\sigma,\mathtt{vk})=1\\ \mathtt{Check}(m,\sigma,\mathtt{vk},\pi)=1\end{array} : \begin{array}{c}\hat{m}\xleftarrow{\$}\{0,1\}\\ (\mathtt{pp},\mathtt{sk},\mathtt{vk})\leftarrow\mathtt{KeyGen}(1^\lambda)\\ \mathtt{sp}\leftarrow\mathtt{Setup}(1^\lambda)\\ (\sigma,\pi)\leftarrow\mathtt{Encode}_{\mathtt{S}^*}(m,\mathtt{sk},\hat{m})\end{array}\right]<\frac{1}{2}+\epsilon(\lambda),$$

where the algorithms $\mathtt{Encode}_{\mathtt{S}^*}$ *and* $\mathtt{Decode}_{\mathtt{R}^*}$ *are as in Definition 34.*

A subliminal bit channel is the simplest subliminal channel, where the sender wants to send a signal, e.g., "escape tonight". With probability one-half the first bit of a signature is equal to the subliminal bit chosen by the sender, that is, by chance they are able to send a subliminal bit. However, if the received bit is the intended subliminal message with probability one-half then it cannot be trustworthy. The receiver cannot gamble and perform a predetermined, possibly daring, action based on a random bit. This cannot be a subliminal bit channel.

Definition 40 (Existential unforgeability)**.** *We say that a subliminal-free digital signature scheme is* existential unforgable *if an adversary* $\mathtt{A}$*, after given valid signatures* σ_i *and proofs* π_i *of messages* m_i *of* $\mathtt{A}$*'s choice from a signing oracle* $\mathtt{O}_{\mathtt{sign}}$*, cannot forge a signature on any new message under the same public-private key pair. We define the existential unforgability advantage* $\mathbf{Adv}^{\mathbf{EUF}}$ *of an adversary* $\mathtt{A}$ *to be:*

$$\mathbf{Adv}^{\mathbf{EUF}}(\mathtt{A}) = \Pr\left[\begin{array}{c} \hat{m} \notin \{m_i\} \\ \mathtt{Verify}(\hat{m}, \hat{\sigma}, \mathtt{vk}) = 1 \end{array} : \begin{array}{c} (\mathtt{pp}, \mathtt{vk}) \leftarrow \mathtt{KeyGen}(1^\lambda) \\ \mathtt{sp} \leftarrow \mathtt{Setup}(1^\lambda) \\ \{(\sigma_i, \pi_i)\} \leftarrow \mathtt{A}^{\mathtt{O}_{\mathtt{sign}}}(\mathtt{pp}, \mathtt{vk}, \{m_i\}) \\ (\hat{m}, \hat{\sigma}) \leftarrow \mathtt{A}(\{(m_i, \sigma_i, \pi_i)\}_i) \end{array}\right] \leq \epsilon(\lambda).$$

7 Our Schemes

In this section we present two subliminal-free digital signature schemes, one scheme where the security is based on the discrete logarithm problem, and one scheme where the security is based on the shortest vector problem in lattices. Both schemes use a VRS to generate a secret random value used as the randomness in the signature scheme.

7.1 Subliminal-Free Schnorr-Signatures

Our VRS definition and subliminal-free digital signature framework applied to the Schnorr signature scheme is similar to Simmons's subliminal-free digital signature scheme [45], where we include security proofs of our VRS and signature scheme to show that they are secure. The VRS scheme is detailed in Figure 8 and the subliminal-free digital signature scheme in Figure 9.

The Diffie-Hellman based VRS (DH-VRS) use deterministic Pedersen commitments [41] and all group operations are over the Schnorr group $G \subset \mathbb{Z}_p^*$ of prime order q with generator g, where p is prime and q divides $p-1$. Let $\mathtt{P}$ and $\mathtt{V}$ denote the prover and verifier, respectively. The idea of the DH-VRS is as follows.

1. $\mathtt{P}$ samples a uniformly random seed s and makes a commitment $\tilde{c}$ of this value.
2. $\mathtt{V}$ samples a uniformly random challenge t.
3. $\mathtt{P}$ makes a commitment of $s+t$. The proof π is the pair $(\tilde{c}, t)$, which the verifier has.

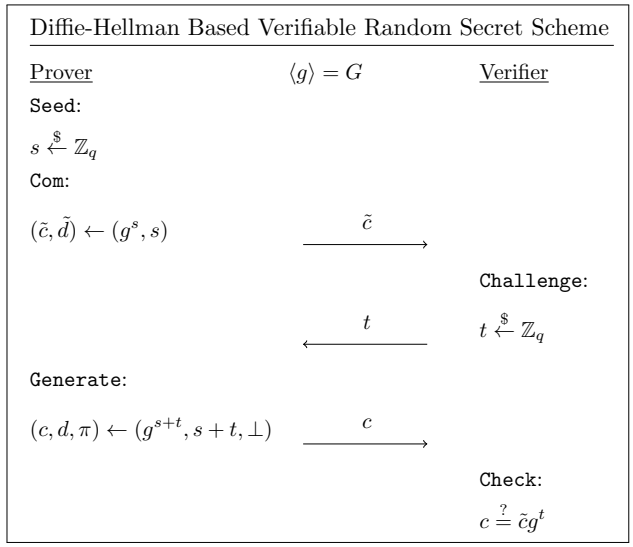


Figure 8: The Diffie-Hellman based verifiable random secret scheme, using deterministic Pedersen commitments.

4. Finally, V checks the output commitment and either accept or reject.

Below we show that the DH-VRS is complete and binding, and has prover bit-unpredictability and weak honest-verifier secrecy.

Lemma 1 (Completeness)**.** *The DH-VRS, detailed in Figure 8, is complete.*

Proof. The verify algorithm $\texttt{Verify}(\tilde{c}, t, c, \pi)$ outputs 1 if $c = \tilde{c}g^t$. In an honest run we have $c = g^{s+t} = g^s g^t = \tilde{c}g^t$. □

Lemma 2 (Binding)**.** *The DH-VRS, detailed in Figure 8, is binding and* $\mathbf{Adv}^{\mathbf{S}}$ *is negligible.*

Proof. Assume $\texttt{Verify}(\tilde{c}, t, c, \pi) = 1$ and $c = \tilde{c}g^t$, for some commitment $\tilde{c}$ and integer t. If $\texttt{P}^*$ want to find another commitment $\hat{c}$ satisfying $\hat{c} \neq c$ and $\texttt{Verify}(\tilde{c}, t, \hat{c}, \pi) = 1$ then $\hat{c} = g^{s+t}g^{kq}$, for an integer k. However, G is finite and $\hat{c}$ and $\hat{c} \equiv c$ in G. Similar for the opening d. The binding advantage of $\texttt{P}^*$ is zero. □

Lemma 3 (Prover bit-Unpredictability)**.** *The DH-VRS, detailed in Figure 8, has prover bit-unpredictability and* $\mathbf{Adv}_{\mathbf{f}}^{\mathbf{PbU}}$ *is negligible.*

Proof. The cheating prover wants to predict a bit $\hat{b}$ of $r = r(s,t) = s + t$. The `Check` algorithm forces c to be of the form $\tilde{c}g^t$, for some $\tilde{c} \in G$ and $t \in \mathbb{Z}_q$. Say $\tilde{c} = g^s$ for some $s \in \mathbb{Z}_q$ then $c = g^{s+t}$, the opening, $d = s+t$ and $r = s+t$. If this is not satisfied then $\mathtt{P}^*$ cannot win, the only value he can choose is $s \in \mathbb{Z}_q$. As long as t is in r the cheating prover will guess a bit in r with probability one-half, hence, to improve this we need to remove t from r. The prover can make a guess t' of the challenge t and set $s = s' - t'$, for an integer s'. If $t' = t$ then $\tilde{c}g^t = g^{s'-t+t} = g^{s'} = c$ and $r = s' - t + t = s'$, which $\mathtt{P}^*$ can predict. If $t' \neq t$ then the `Check` outputs 0 and the event will not occur. The probability of guessing t is q^{-1} and $\mathbf{Adv}_{\mathbf{f}}^{\mathbf{PbU}} = q^{-1}$, which can be made negligible for a large q. □

From Lemma 1 we know that the DH-VRS, detailed in Figure 8, is complete. We would also like to show that the DH-VRS is ϵ-Secure, however, this is not possible as it does not have weak honest-verifier secrecy. The reason is that the value g^r leaks a part of the DH-VRS randomness r, hence, the adversary $\mathtt{V}^*$ can use g^r to accurately predict certain bits of r.

Using the above DH-VRS we can construct verifiable random numbers. Together with the Schnorr digital signature scheme [44] we construct a Diffie-Hellman based subliminal-free digital signature scheme (DH-SFS), where the verifiable random number, generated in the DH-VRS, is used to produce a Schnorr signature. See Figure 9. Below we show that the DH-SFS is complete, sound and secure against existential forgeries.

Lemma 4 (Completeness)**.** *The DH-SFS scheme, described in Figure 9, is complete.*

Proof. In an honest run we have that $g^{\gamma}pk^{\beta} = g^{d-\beta \cdot sk}g^{\beta \cdot sk} = g^{s+t} = c$ and the `Verify` algorithm outputs 1, similarly $H(pk, g^{\gamma}pk^{\beta}, m) = H(pk, c, m) = \beta$ and the `Check` algorithm outputs 1. □

Lemma 5 (Soundness)**.** *The DH-SFS scheme, described in Figure 9, is sound.*

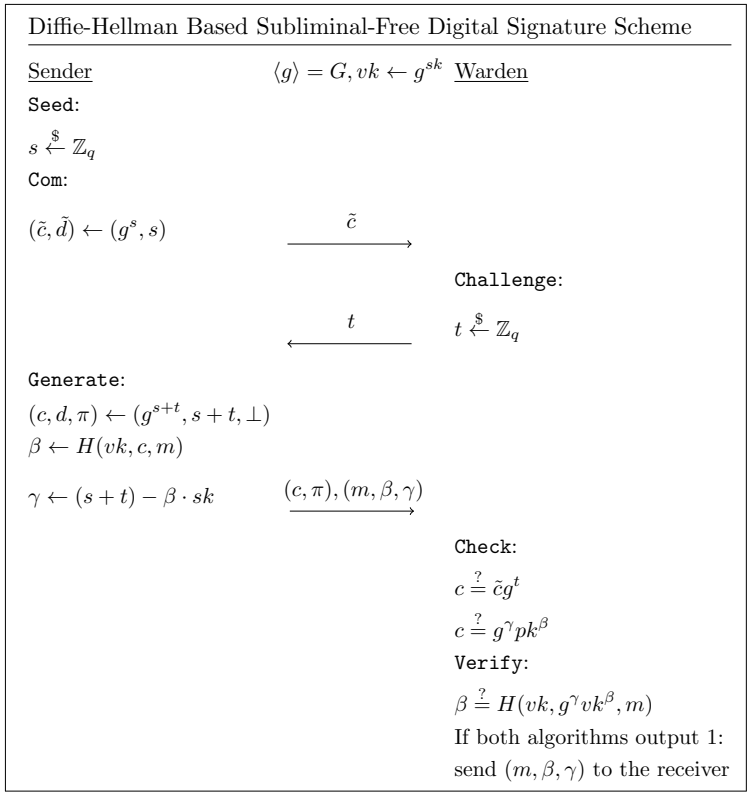


Figure 9: The Diffie-Hellman based subliminal-free digital signature scheme, using the DH-VRS, to generate a random value and proofs, and Schnorr digital signatures.

Proof. We claim that $\mathtt{P}^*$ is unable to encode a subliminal bit in the signature, that the receiver is able to decode, with probability greater than one-half. By Lemma 2 the sender cannot find a new commitment $\hat{c} \neq c$, which will be accepted by the warden, and by our Warden Model the sender cannot choose the message m and is not allowed to alter the secret and public key during the signing process. Hence, the value β is deterministic given $\tilde{c}$ and t. By Lemma 3 the sender is unable to predict the random value $s + t$ used to generate γ and the signature is deterministic. □

Lemma 6 (Existential Unforgability)**.** *The DH-SFS scheme, in Figure 9, is secure against existential forgability and* $\mathbf{Adv}^{\mathbf{EUF-CMA}}$ *is negligible.*

Proof. Let $\mathtt{A}$ be an adversary that can produce forgeries of the DH-SFS scheme in polynomial time. Then there exist an adversary $\mathtt{B}$ that solves the discrete logarithm problem (Definition 2) in polynomial time. Use $\mathtt{A}$ to produce two forgeries $(\hat{m}_1, \hat{\sigma}_1)$ and $(\hat{m}_2, \hat{\sigma}_2)$, on two different messages such that $\beta_1 \neq \beta_2$, using the same random values and public parameters. Then $sk = (\gamma_2 - \gamma_1)/(\beta_1 - \beta_2)$ and

$$\mathbf{Adv}^{\mathbf{EUF}}(\mathtt{A}) \leq \mathbf{Adv}^{\mathbf{DL}}(\mathtt{B}),$$

where the discrete logarithm advantage of $\mathtt{B}$ is negligible. □

7.2 Subliminal-Free Lattice-Based Signatures

Our lattice-based VRS scheme is built on top of the commitment scheme by Baum et al. [4] combined with the verifiable shuffle of known content by Baum et al. [5]. Let $\mathtt{P}$ denote the prover and let $\mathtt{V}$ denote the verifier. All the polynomials are from R_p.

The protocol works as follows: Warden draws 3τ Gaussian distributed polynomials s_i with standard deviation σ/κ and sends them to Sender. He then shuffles the polynomials, commits to them in the new order and sends the commitments to Warden. The Warden draws three subset T_j, for $i \leq j \leq 3$, of indices from 1 to 3τ at random and sends them to Sender. Note that these elements can be represented as uniform distributed numbers on the unit interval, converted

to Gaussians via the Box-Muller method as described earlier. The Sender computes the three sums of the underlying messages of the commitments with the respective indices, and proves that the sums are correct and that the initial shuffle was correct. Finally he computes the signature of the message, and sends the message-signature pair together with the commitments and proofs to the Warden. The Warden verifies that the commitments, the shuffle, and proofs are correct (the VRS part) and that the signature-message pair is correct. He also verifies that the randomness generated in the VRS is the randomness used in the signature. Let $\boldsymbol{y} = [y_1, y_2, y_3]$ be the underlying messages of the commitments c_1, c_2, c_3 generated by the VRS. They are all Gaussian distributed polynomials with standard deviation σ, as they are a sum of κ Gaussian distributed polynomials with standard deviation σ/κ. The Warden verifies the signature by checking that t' is the linear combination $y_1 + a_1 y_2 + a_2 y_3$, that is, $t = \boldsymbol{ay}$. If all the proofs are correct then the Warden forwards the message-signature pair to the Receiver.

The VRS scheme is visualized in Figure 10. In the end of the protocol, P ends up with a commitment to a random sum of randomly chosen polynomials. Intuitively, the prover should not be able to predict any of coordinates of the the final polynomial before he's given the set of indices, and the verifier should not be able guess any of the coordinates of the final polynomial when he only knows that it's a sum of a subset of the initial polynomials. We proceed by proving *Completeness*, *Binding*, *Prover bit-Unpredictability* and *Honest-Verifier Secrecy* for the VRS.

Lemma 7 (Completeness)**.** *The Lattice-VRS in Figure 10 is compete.*

Proof. Assuming that the prover is honest, he will commit to the given seed-values in a permuted order. He will also be able to generate an accepting zero-knowledge proof of correct shuffle, as the shuffle-protocol is complete. Further we assume that the prover commit to the sum of the correct values given the challenge set of indices, and then he will be able to generate an accepting zero-knowledge proof of correct sum, as the sum-protocol is complete. We conclude that the VRS is complete. □

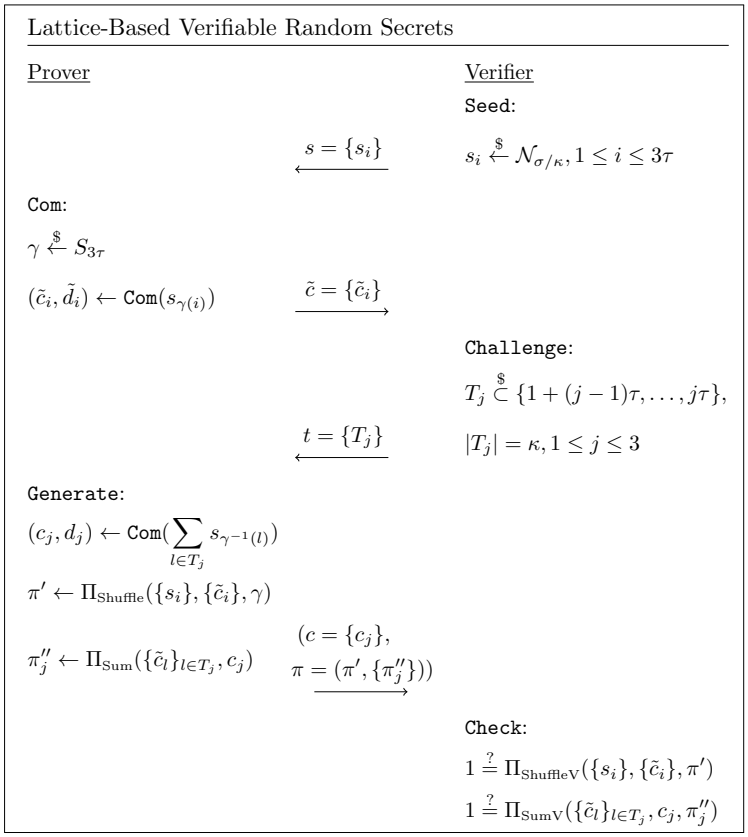


Figure 10: The lattice-based verifiable random secret scheme, using the commitment scheme by Baum et al. [4] and the verifiable shuffle of known content by Baum et al. [5].

Lemma 8 (Binding)**.** *The Lattice-VRS in Figure 10 is binding.*

Proof. Assuming that a cheating prover $\mathtt{P}^*$ has advantage ϵ of breaking the binding property of the VRS, then $\mathtt{P}^*$ also have a advantage ϵ of breaking the binding property of the commitment scheme. The binding property of the commitment scheme is based on the SIS-problem, and hence, an attacker $\mathtt{A}$ can then turn $\mathtt{P}^*$ into a SIS-solver with success probability ϵ. We conclude that the VRS is binding. □

Lemma 9 (Prover bit-Unpredictability)**.** *The Lattice-VRS in Figure 10 is prover bit-unpredictable.*

Proof. The cheating prover $\mathtt{P}^*$ wants to predict a bit $\hat{b}$ of $r_j = r(\{s_l\}_{l=1}^{\kappa}, \{\hat{c}_l\}_{l=1}^{\kappa}, T_j)$, for $0 \leq j \leq 3$, that is, a bit $\hat{b}$ of the values $r_j = \sum_{l \in T_j} s_{\gamma^{-1}(l)}$ for the permutation γ of the underlying messages of the set of commitments. All values s_i are drawn at random from a Gaussian distribution. $\mathtt{P}^*$ is allowed to guess after he know all s_i's, but before he has received the sets T_j of indices. As T_j are random sets of indices drawn uniformly at random, r_j is a random sum of κ random elements, and hence, r_j is random. The probability of guessing any bits if r_j correctly is $1/2$, and hence, the prover bit-unpredictability advantage $\mathbf{Adv}_{\mathbf{f}}^{\mathbf{PbU}}(\mathtt{P}^*)$ is negligible. We conclude that the VRS is prover bit-unpredictable. □

Lemma 10 ((Strong) Honest-Verifier Secrecy)**.** *The Lattice-VRS in Figure 10 has strong honest-verifier secrecy.*

Proof. Assume that a honest but curious verifier $\mathtt{V}^*$ is given the value r_b at the end of the protocol. His task is to decide if r_b is a sum of κ elements among the values $s_1, \ldots, s_\tau$ or not. This reduces to the k-SUM problem, for a subset of size κ out of τ elements. If $\mathtt{V}^*$ can break the strong honest-verifier secrecy of the VRS with a probability ϵ, then an attacker $\mathtt{A}$ can then turn $\mathtt{V}^*$ into a k-SUM solver with success probability ϵ. We conclude that the VRS has strong honest-verifier secrecy. □

Theorem 1. *The lattice-based VRS detailed in Figure 10 is complete and ϵ-Secure.*

Proof. This follows from combining Lemma 7, 8, 9 and 10. □

Lemma 11 (Completeness)**.** *The Lattice-SFS scheme in Figure 11 is complete.*

Proof. This follows directly from the fact that the VRS is complete and that the signature scheme (without the VRS) is complete. The VRS ensure that the randomness used in the signature is of the right form, and hence, the composition of the two schemes is then complete. We conclude that the signature scheme is complete. □

Lattice-Based Subliminal-Free Digital Signatures

Sender	$\mathtt{vk} = \mathtt{Com}(0, \boldsymbol{r})$	Warden
$\mathtt{sk} = \boldsymbol{r}$		Seed:
	$\xleftarrow{s = \{s_i\}}$	$s_i \xleftarrow{\$} \mathcal{N}_{\sigma/\kappa}, 1 \le i \le 3\tau$
Com:		
$\gamma \xleftarrow{\$} S_{3\tau}$		
$(\tilde{c}_i, \tilde{d}_i) \leftarrow \mathtt{Com}(s_{\gamma(i)})$	$\xrightarrow{\tilde{c} = \{\tilde{c}_i\}}$	
		Seed:
		$T_j \xleftarrow{\$} \subset \{1 + (j-1)\tau, \ldots, j\tau\},$
	$\xleftarrow{t = \{T_j\}}$	$\lvert T_j \rvert = \kappa, 1 \le j \le 3$
Generate :		
$(c_j, d_j) \leftarrow \mathtt{Com}(\sum_{l \in T_j} s_{\gamma^{-1}(l)})$		
$\pi' \leftarrow \Pi_{\mathrm{Shuffle}}(\{s_i\}, \{\tilde{c}_i\}, \gamma)$		
$\pi''_j \leftarrow \Pi_{\mathrm{Sum}}(\{\tilde{c}_l\}_{l \in T_j}, c_j)$		
$(t', \boldsymbol{z}) \leftarrow \mathtt{Sign}(m, \mathtt{sk})$		
$\pi''' \leftarrow \Pi_{\mathrm{Sum}}(\{c_j\}, t', (1, a_1, a_2))$	$\xrightarrow{(m, (t', \boldsymbol{z})), (\{c_j\}, (\pi', \{\pi''_j\}, \pi'''))}$	
		Check:
		$1 \stackrel{?}{=} \Pi_{\mathrm{ShuffleV}}(\{s_i\}, \{\tilde{c}_i\}, \pi')$
		$1 \stackrel{?}{=} \Pi_{\mathrm{SumV}}(\{\tilde{c}_l\}_{l \in T_j}, c_j, \pi''_j)$
		$1 \stackrel{?}{=} \Pi_{\mathrm{SumV}}(\{c_j\}, t', (1, a_1, a_2), \pi''')$
		Verify:
		$1 \stackrel{?}{=} \mathtt{Verify}(\mathtt{vk}, m, (t', \boldsymbol{z})))$
		If all algorithms output 1:
		Send $(m, (t', \boldsymbol{z}))$ to the receiver.

Figure 11: The lattice-based subliminal-free digital signature scheme, using the lattice-based VRS and a lattice signature scheme based on Lyubashevsky [33, 34].

Lemma 12 (Soundness)**.** *The Lattice-SFS scheme in Figure 11 is*

sound.

Proof. The VRS ensures that the randomness is honestly generated, and because of the prover bit-unpredictability of the VRS, a cheating prover $\mathtt{P}^*$ has a negligible probability of embedding any information into the randomness used in the signature. It follows that a cheating prover doesn't have a reliable subliminal channel. We conclude that the signature scheme is sound. □

Lemma 13 (Existential unforgability)**.** *The Lattice-SFS scheme in Figure 11 is secure against existential forgability.*

Proof. A honest but curious verifier $\mathtt{V}^*$ learns Ar for some public matrix A and randomness r, where r is the output of the VRS. By multiplying all $s_1, \ldots, s_\tau$ by A, this reduces to solving the k-SUM problem for $k = \kappa$ when the set is $\{As_i\}_{i=1}^{\tau}$ and $S = Ar$. This search problem is equivalent to the decision problem with the same input, which we know is hard. The VRS ensures that the randomness is honestly generated, and because of the strong honest verifier secrecy of the VRS, $\mathtt{V}^*$ has a negligible probability of learning any information about the secret signing key used to generate the signature. Furthermore, both zero-knowledge proofs in the VRS can be simulated, and hence, this additional information does not provide any information that $\mathtt{V}^*$ can use to win the game. □

8 Efficiency and Size

8.1 The Discrete Logarithm SFS

The efficiency of the discrete logarithm subliminal-free digital signature scheme is summarized in Figure 12, the detailed description is below.

In the DH-SFS scheme we do an interactive VRS exchange, generate a proof, and produce a normal Schnorr signature. The Sender computes two exponentiations, one which will be used to compute the usual Schnorr signature commitment, and sends two messages to the Warden. The Warden computes three exponentiations, two of which are to verify the signature, and sends one message to the Sender and

	Computation and interaction			Signature
	Sender	Warden	Receiver	size
Schnorr signature	1**E**+1**M**		2**E**	2**GE**
DH-SFS	2**E**+2**M**	3**E**+2**M**	2**E**	5**GE**

Figure 12: Efficiency of our subliminal-free digital signature schemes compared to their non-subliminal-free counterparts. **E** denotes an exponentiation **GE** a group element, and **M** a message sent.

one to the Receiver (if the proof is valid). The signature consists of two group elements of the Schnorr group G and three group elements of $\mathbb{Z}_q$. We ignore the hash function evaluations, and multiplication and addition of group elements. To produce a subliminal-free Schnorr signature we need four additional exponentiations, send three additional messages, and have three additional group elements (one from $\mathbb{Z}_q$ and two from G).

8.2 The Lattice-Based SFS

In the lattice-based SFS scheme we run the interactive VRS, generate proofs and produce a signature. The Sender computes $3\tau+3$ commitments, one shuffle proof, three proofs showing that each of the three commitments c_j is a sum of κ $\tilde{c}_i$'s, one proof showing the signature is honestly generated, and one signature. For a normal signature the Sender only sends one message, for a subliminal-free signature the Sender sends two and the Warden sends three.

A normal signature contains four elements from R_p, an subliminal-free signature requires $45\tau + 12\kappa + 40$ elements from R_p. To give an estimate of the size the signature we give a rough estimation of τ and κ. The security of the lattice-based scheme relies on the k-SUM problem and we want it to be hard to guess the subset of size κ from the super-set of size τ, to prevent brute force attacks we want τ choose κ to be larger than or equal to 2^{128} for 128 bit security. Furthermore, the best known algorithm for solving the k-SUM problem runs in $\mathcal{O}(\tau^{\kappa/2})$ [21], which means that we also need $\tau^{\kappa/2} \geq 2^{128}$. By setting $\tau = 256$ and $\kappa = 32$ we satisfies both of these equations, and a

subliminal-free signature contains 11944 elements from R_p. Using the parameters provided by Baum et al. [4] for the commitment scheme we have $p \approx 2^{32}$ and $N = 1024$. One element in R_p is then 4.1 KB, and hence, a normal signature is 16.4 KB and a subliminal-free signature is ≈ 49 MB. The difference is a factor of 3000.

However, we can make the protocol more efficient in the online phase of sending messages and signatures by doing a lot of work in a pre-processing phase. Assume that we want to send X messages. Then most of the work can be done in advance: creating the seeds, committing to the seeds and shuffle them, prove that the shuffle is done correct, choose indices, compute the large sums and prove that the sums are correct. This is almost all of the work in the whole protocol. This can be done X times in one go, and takes space $\approx 49X$ MB. Only the elements t' needs to be stored until the online phase, in the correct order, and hence, only $4.1X$ KB of data. The only thing to be done in the online phase is to generate the signature and send over $\boldsymbol{z}$ together with the message, which is only three ring elements, one less than a normal signature.

Acknowledgments

We thank Kristian Gjøsteen and Yao Jiang for fruitful discussions, aid and comments that greatly improved the manuscript.

References

[1] Michel Abdalla, Dario Catalano, and Dario Fiore. Verifiable Random Functions from Identity-Based Key Encapsulation. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, pages 554–571, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[2] Erdem Alkim, Paulo S. L. M. Barreto, Nina Bindel, Juliane Kramer, Patrick Longa, and Jefferson E. Ricardini. The Lattice-Based Digital Signature Scheme qTESLA. Cryptology ePrint

Archive, Report 2019/085, 2019. https://eprint.iacr.org/2019/085.

[3] Ross Anderson, Serge Vaudenay, Bart Preneel, and Kaisa Nyberg. The Newton channel. In Ross Anderson, editor, *Information Hiding*, pages 151–156, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[4] Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In Dario Catalano and Roberto De Prisco, editors, *Security and Cryptography for Networks*, pages 368–385, Cham, 2018. Springer International Publishing.

[5] Carsten Baum, Kristian Gjøsteen, Tjerand Silde, and Thor Tunge. Electronic voting using lattice-based commitments and verifiable encryption. Unpublished.

[6] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT'98*, pages 127–144, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[7] Manuel Blum. Coin flipping by telephone a protocol for solving impossible problems. *SIGACT News*, 15(1):23–27, January 1983.

[8] Jens-Matthias Bohli, María Isabel González Vasco, and Rainer Steinwandt. A Subliminal-Free Variant of ECDSA. In Jan L. Camenisch, Christian S. Collberg, Neil F. Johnson, and Phil Sallee, editors, *Information Hiding*, pages 375–387, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[9] Jens-Matthias Bohli and Rainer Steinwandt. On subliminal channels in deterministic signature schemes. In Choon-sik Park and Seongtaek Chee, editors, *Information Security and Cryptology – ICISC 2004*, pages 182–194, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[10] Dan Boneh, Hart William Montgomery, and Ananth Raghunathan. Algebraic Pseudorandom Functions with Improved Efficiency from the Augmented Cascade. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 131–140, New York, NY, USA, 2010. ACM.

[11] G. E. P. Box and Mervin E. Muller. A note on the generation of random normal deviates. *Ann. Math. Statist.*, 29(2):610–611, 06 1958.

[12] Mike Burmester and Yvo Desmedt. All Languages in NP Have Divertible Zero-Knowledge Proofs and Arguments Under Cryptographic Assumptions. In Ivan Bjerre Damgård, editor, *Advances in Cryptology — EUROCRYPT '90*, pages 1–10, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[13] Mike Burmester, Yvo G. Desmedt, Toshiya Itoh, Kouichi Sakurai, and Hiroki Shizuya. Divertible and subliminal-free zero-knowledge proofs for languages. *J. Cryptol.*, 12(3):197–223, June 1999.

[14] Rongmao Chen, Yi Mu, Guomin Yang, Willy Susilo, Fuchun Guo, and Mingwu Zhang. Cryptographic Reverse Firewall via Malleable Smooth Projective Hash Functions. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, pages 844–876, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[15] Yvo Desmedt. Subliminal-free authentication and signature. In D. Barstow, W. Brauer, P. Brinch Hansen, D. Gries, D. Luckham, C. Moler, A. Pnueli, G. Seegmüller, J. Stoer, N. Wirth, and Christoph G. Günther, editors, *Advances in Cryptology — EUROCRYPT '88*, pages 23–33, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[16] Yvo Desmedt. Simmons' protocol is not free of subliminal channels. *Proceedings 9th IEEE Computer Security Foundations Workshop*, pages 170–175, 1996.

[17] Yevgeniy Dodis. Efficient Construction of (Distributed) Verifiable Random Functions. In Yvo G. Desmedt, editor, *Public Key Cryptography — PKC 2003*, pages 1–17, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[18] Yevgeniy Dodis and Aleksandr Yampolskiy. A Verifiable Random Function with Short Proofs and Keys. In Serge Vaudenay, editor, *Public Key Cryptography - PKC 2005*, pages 416–431, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[19] Qingkuan Dong and Guozhen Xiao. A Subliminal-Free Variant of ECDSA Using Interactive Protocol. In *2010 International Conference on E-Product E-Service and E-Entertainment*, pages 1–3, Nov 2010.

[20] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTAL-Dilithium. `https://pq-crystals.org/dilithium/data/dilithium-specification-round2.pdf`. Submission to the NIST Post-Quantum Standardization Project, round 2.

[21] Jeff Erickson. Lower bounds for linear satisfiability problems. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '95, pages 388–395, Philadelphia, PA, USA, 1995. Society for Industrial and Applied Mathematics.

[22] Amos Fiat and Adi Shamir. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, pages 186–194, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

[23] Herman Galteland and Kristian Gjøsteen. Subliminal channels in post-quantum digital signature schemes. Cryptology ePrint Archive, Report 2019/574, 2019. `https://eprint.iacr.org/2019/574`.

[24] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth*

Annual ACM Symposium on Theory of Computing, STOC '85, page 291–304, New York, NY, USA, 1985. Association for Computing Machinery.

[25] Shafi. Goldwasser, Silvio. Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[26] Alexander Hartl, Robert Annessi, and Tanja Zseby. A Subliminal Channel in EdDSA: Information Leakage with High-Speed Signatures. In *Proceedings of the 2017 International Workshop on Managing Insider Security Threats*, MIST '17, pages 67–78, New York, NY, USA, 2017. ACM.

[27] Dennis Hofheinz and Tibor Jager. Verifiable Random Functions from Standard Assumptions. In Eyal Kushilevitz and Tal Malkin, editors, *Theory of Cryptography*, pages 336–362, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[28] Susan Hohenberger and Brent Waters. Constructing Verifiable Random Functions with Large Input Spaces. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 656–672, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[29] Tibor Jager. Verifiable Random Functions from Weaker Assumptions. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography*, pages 121–143, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[30] Shuichi Katsumata. On the Untapped Potential of Encoding Predicates by Arithmetic Circuits and Their Applications. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 95–125, Cham, 2017. Springer International Publishing.

[31] Lisa Kohl. Hunting and Gathering – Verifiable Random Functions from Standard Assumptions with Short Proofs. In Dongdai Lin and Kazue Sako, editors, *Public-Key Cryptography – PKC 2019*, pages 408–437, Cham, 2019. Springer International Publishing.

[32] Dai-Rui Lin, Chih-I Wang, Zhi-Kai Zhang, and D. J. Guan. A digital signature with multiple subliminal channels and its applications. *Comput. Math. Appl.*, 60(2):276–284, July 2010.

[33] Vadim Lyubashevsky. Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, pages 598–616, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[34] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 738–755, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[35] Vadim Lyubashevsky and Gregor Seiler. Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 204–224, Cham, 2018. Springer International Publishing.

[36] Silvio Micali, Salil Vadhan, and Michael Rabin. Verifiable random functions. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS '99, pages 120–, Washington, DC, USA, 1999. IEEE Computer Society.

[37] Ilya Mironov and Noah Stephens-Davidowitz. Cryptographic reverse firewalls. Cryptology ePrint Archive, Report 2014/758, 2014. `https://eprint.iacr.org/2014/758`.

[38] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, CCS '01, pages 116–125, New York, NY, USA, 2001. ACM.

[39] NIST Post-Quantum Cryptography, Round 1 Submissions. `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions`. Accessed: 2019-11-29.

[40] Tatsuaki Okamoto and Kazuo Ohta. Divertible zero knowledge interactive proofs and commutative random self-reducibility. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology — EUROCRYPT '89*, pages 134–149, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

[41] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 129–140, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[42] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, pages 387–398, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[43] Răzvan Roşie. Adaptive-Secure VRFs with Shorter Keys from Static Assumptions. In Jan Camenisch and Panos Papadimitratos, editors, *Cryptology and Network Security*, pages 440–459, Cham, 2018. Springer International Publishing.

[44] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO*, 1989.

[45] G. J. Simmons. An introduction to the mathematics of trust in security protocols. In *[1993] Proceedings Computer Security Foundations Workshop VI*, pages 121–127, June 1993.

[46] Gustavus J. Simmons. The prisoners' problem and the subliminal channel. *Advances in Cryptology: Proceedings of Crypto 83*, pages 51–67, 1984.

[47] Gustavus J. Simmons. The subliminal channel and digital signatures. In Thomas Beth, Norbert Cot, and Ingemar Ingemarsson, editors, *Advances in Cryptology*, pages 364–378, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.

[48] Gustavus J. Simmons. A secure subliminal channel (?). In Hugh C. Williams, editor, *Advances in Cryptology — CRYPTO*

'85 Proceedings, pages 33–41, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

[49] Gustavus J. Simmons. Subliminal Communication is Easy Using the DSA. In Tor Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, pages 218–232, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[50] Gustavus J. Simmons. Results concerning the bandwidth of subliminal channels. *IEEE Journal on Selected Areas in Communications*, 16(4):463–473, May 1998.

[51] Shota Yamada. Asymptotically Compact Adaptively Secure Lattice IBEs and Verifiable Random Functions via Generalized Partitioning Techniques. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 161–193, Cham, 2017. Springer International Publishing.

[52] Yinghui Zhang, Hui Li, Xiaoqing Li, and Hui Zhu. Provably Secure and Subliminal-Free Variant of Schnorr Signature. In Khabib Mustofa, Erich J. Neuhold, A. Min Tjoa, Edgar Weippl, and Ilsun You, editors, *Information and Communication Technology*, pages 383–391, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[53] Xianfeng Zhao and Ning Li. Reversible watermarking with subliminal channel. In Kaushal Solanki, Kenneth Sullivan, and Upamanyu Madhow, editors, *Information Hiding*, pages 118–131, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

ISBN 978-82-326-4580-0 (printed version)
ISBN 978-82-326-4581-7 (electronic version)
ISSN 1503-8181

Doctoral theses at NTNU, 2020:116

Herman Galteland

Malicious cryptography

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

NTNU
Norwegian University of
Science and Technology

Herman Galteland

Doctoral Thesis

Doctoral theses at NTNU, 2020:116

NTNU

NTNU
Norwegian University of
Science and Technology