Asgeir Steine

Privacy-Preserving Cryptographic Protocols

Thesis for the degree of Philosophiae Doctor

Trondheim, June 2012

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

NTNU – Trondheim
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Information Technology, Mathematics
and Electrical Engineering
Department of Mathematical Sciences

ISBN 978-82-471-3651-5 (printed ver.)
ISBN 978-82-471-3653-9 (electronic ver.)
ISSN 1503-8181

Doctoral theses at NTNU, 2012:179

Printed by NTNU-trykk

Acknowledgement

This thesis is the result of my four years of PhD-research with the Department of Mathematical Sciences and the Faculty of Information Technology, Mathematics and Electrical Engineering at NTNU.

During my time of PhD-studies some people have been present in my life that deserve a special thanks.

First and foremost I would like to thank my supervisor Kristian Gjøsteen and co-authors George Petrides and Øystein Thuen. You have been essential for the fulfillment of this thesis and invaluable as friends and moral support along the way. I would also like to thank Professor Ivan Damgård of the university of Århus and his wonderful group of PhD-students and post.docs. I learned a lot during my stay with you.

Lastly I want to thank my friends and family for always being there for me when I need you.

Thank you.

Introduction

The modern development of electronic communications has opened the door to new and fantastic possibilities, but also information security challenges. One particular concern is that the privacy of the individual user is failing. As more and more personal and sensitive information finds its way online, the power of data mining and automatic profiling grows. There are many examples where content providers gather sensitive information on their users. Mobile network providers store location data and communication logs. Banks store where and when their users make transactions. Automatic toll booth systems register when and where the cars pass them. It seems as if every new service adds another example to the list.

While laws often protect the users from abuse of their sensitive data, it seems difficult for the authorities to check whether or not the rules are being followed. Even when the service provider has honest intentions, dishonest employees may access and exploit the stored data. Recent attacks by hacker groups like Anonymous and LulzSec also illustrate the difficulty of securing sensitive data records from outside threats. We feel that it is important to indicate that privacy preserving alternatives to today's infrastructures can be developed and that these alternatives do not significantly hinder the efficiency of the services.

Privacy preserving protocols should be designed so that no service provider has access to sensitive information he does not strictly need to provide the service. By dividing the overall knowledge about users between multiple service providers the severity of security breaches can be limited. In two of the papers of this thesis we design alternative protocols for mobile network authentication and mobile payment systems. In our model for mobile networks the mobile service providers are not directly responsible for maintaining the network. Each service provider leases network access from network providers. We can thereby design our protocols so that the network provider only learns the location of users (not identity), while the service provider learns the identity (not location). Building on this mobile network protocol we consider a mobile payment system where the merchant does not learn the shopper's identity and the bank can not link user transactions to the merchants involved.

In the following sections we will briefly describe the tools and models that we use to analyse security, and the content of the five papers that constitute this thesis.

1 Protocol analysis

A cryptographic protocol can be thought of as a set of interactive algorithms for one or more communicating parties designed to securely realise some defined goal. The term securely in this context does not have a universal meaning, but must be explicitly defined. In fact defining the correct notion of security for a protocol is a significant part of its analysis. Consider the Needham-Schroeder authentication protocol [7] for example. Needham and Schroeder proved their authentication protocol secure in their model. Later however, Lowe discovered

[6] that the protocol was vulnerable to man in the middle attacks. This does not mean that there was something wrong with Needham and Schroeder's proof. Their model and security definition was simply to weak to capture these kinds of attacks. They assumed that users of the system would not participate in attacks.

The preceding example illustrates the importance of explicitly defining what kind of security our protocols can provide and the challenge of finding the correct level of security for the goals we want to achieve.

Game based security A common method to gain confidence in a cryptographic protocol's security is to show that any successful attacker can be used to solve one or more well-studied hard problems. This is usually done by so-called sequences-of-games as described in [8].

Initially one would define the desired security goal as a game played by the attacker. The game models some interaction between an attacker and the protocol in question. The attacker wins the game if he can compromise the protocol's security. Next one goes through a sequence of steps each time introducing a new game slightly different from the previous until one arrives at a final game where no compromise is possible. To complete the argument one bounds the attackers chances to distinguish each game from its predecessor.

One way to bound the attackers distinguishing chances for a given pair of games is to prove that a distinguishing attacker can be used to build a solver for one of the assumed hard problems (and that this solver uses essentially the same amount of resources as the attacker). By the above procedure one proves that any adversary has its advantage against the initial game bounded by the sum of its distinguishing advantages and thus that it is essentially as hard to win the security game as it is to solve the underlying hard problems.

Simulation based security Cryptographic protocols vary in complexity from protocols for achieving basic properties like authentication, integrity or confidentiality to sophisticated multi-party protocols like electronic elections or privacy preserving online auctions. Typically the more sophisticated protocols are constructed using the simpler protocols as subprotocols. It is therefore desirable to analyse protocols in a modular way by defining security notions that are preserved under composition of protocols.

A simple idea for achieving modular security is to define security as an ideal functionality modelling how the protocol ideally should behave if we had an incorruptible trusted third party to manage it. To prove that a protocol realises the security of this ideal functionality one proves the existence of an efficient simulator such that no (appropriately bounded) adversarial environment is able to distinguish if he is interacting with the protocol or with the ideal functionality composed with the simulator.

Since the adversarial environment is unspecified this way of defining security gives guarantees that are preserved under composition of protocols. Frameworks for this kind of security modelling are called simulation based and the two main

such frameworks are the universal composability framework of [2] and the reactive simulatability framework of [1]. Although the idea behind simulation based security is simple, many technical issues must be resolved to make such a model formal.

Formal methods Another approach for analysing complicated protocols is to assume "perfect" cryptographic primitives. By defining formal rules about the terms in the protocols and how they can be used to attack the system, one builds a logic for reasoning about protocol security. These kinds of formal models stem from the work of Dolev and Yao [5] and have lead to quite powerful tools (like Scyther and ProVerif) for automatic analysis of protocols. The analysis in such models are often simpler than in game based analysis. However since the proofs only apply to ideal crypto primitives, the path from a formal security proof to real world security is longer than for game based proofs.

Lately there has been some effort to come up with models that combine the best features from both the game based security paradigm and the formal models. A nice survey on these topics can be found in [4].

2 Our work

Our work consist of five papers all in the field of protocol analysis. We emphasise the privacy aspect of protocol security.

Paper I: Dynamic Group Signatures with Tokens A group signature scheme is a variant of cryptographic signatures allowing the signer to remain anonymous within a group. It allows any member of the group to sign messages on the group's behalf. Groups can be created by an authority called the group manager. Under special circumstances signatures can be opened by an authority called the opener to reveal the signers identity.

In this paper we develop a variant of group signatures that we call token based group signatures. With token based group signatures the group member issues tokens to each member of the group which allow them to sign messages within a time window. This simplifies the problem of dynamically updating the members of the group while maintaining anonymity. We define security notions for the token based group signatures and give a generic construction using non-interactive zero knowledge proofs.

Group signatures have a number of interesting properties that make them suitable for anonymous authentication schemes. By signing a challenge, a user can prove that he is indeed a member of the group without revealing his identity. The verifier also get a guarantee that the opener has the power to obtain the signer's identity from the signature. Group signatures can therefore be used to build a public-key alternative to the authentication protocol of paper IV.

Paper II: Formal Verification of Reductions in Cryptography In this paper we consider a way of formalising black-box reductions of indistinguishability games by modelling each game as an interactive state machine with an associated set of probability distributions. The idea is that proper standardisation of notation for security games is one step towards automatic verification of game based security proofs. We proceed by defining maps between such state machines and sufficient conditions for when such maps lead to reductions. As an example we apply our techniques to prove the standard security reduction (see [9]) of the ElGamal encryption scheme.

Paper III: A Novel Framework for Protocol Analysis Taking the universal composability framework of [2] as a starting point. We redo some of its definitions to avoid certain technicalities and better suit our needs as a simulation based security framework. Perhaps the most striking change we have made is to add a message queue for scheduling of messages. As in the original formulation we only allow one active party at a time, but where Canetti's version only allows the active party to send one single message we additionally allow the party to enqueue messages. The effect of this extension is that many protocols become easier to model, however it may introduce some technicalities in the security proofs as one has to show that the simulator can simulate the queue correctly.

In the universal composability framework it is a clear preference towards modelling single session protocols. For instance an encrypted channel would be modelled by a functionality that can send only one secure message. By composing multiple instances of the same functionality one is able to send multiple messages securely. As a result it becomes important that each machine is able to create new instances of its sub-protocols during protocol execution, which makes the framework quite technical. While this single session preference sometimes lead to simplified analysis, this approach also has its disadvantages. One issue is that the parties of a protocol instance has to agree in advance on a session identifier (name of the particular instance corresponding to that session).

In settings as in Paper IV, where one of the parties should remain anonymous, this issue becomes troublesome. We therefore prefer the multi-session approach and consider the simpler scenario where all machines are instantiated before protocol execution in a fixed communication graph, thereby avoiding these issues.

Paper IV: Towards Privacy Preserving Mobile Communications The mobile phone protocols in use today do not give a high priority to the user's privacy. The mobile service provider has easy access to each user's location, can see who the users call and listen in on their conversations. We sketch an alternative setup where the service provider is split into two parties. One party, the service provider, handles authentication and user identities. The other party, the network provider, handles the network but learns as little as possible about the user's identities.

We build our mobile phone protocol step by step using the adapted universal composability framework of Paper III. The first step in this work is to create a key establishment and authentication protocol and its corresponding functionality.

In this protocol the user first authenticates himself to the service provider to prove that he is a subscriber and should have access to the network. If the service provider is satisfied, he tells the network provider to proceed by establishing symmetric keys with the user.

On top of the key establishment functionality we build a secure channel between the network provider and users, using the shared keys. In the next step we use the secure channel to build an anonymous internet access functionality where the users can connect to a global network, while hiding behind pseudonyms managed by the network provider. Finally we sketch how the internet access functionality can be used to provide telephony over the global network, using public key encryption and pseudonyms.

Paper V: Weak Blind Signatures and Mobile Payment In this paper we introduce and analyse a mobile payment system in the adapted universal composability framework of Paper III. We use a simplified version of the internet access functionality defined in Paper IV. A mobile payment system allows users and merchants to perform transactions by means of the users mobile device. In addition to regular transaction security one important issue is the privacy of the users. The mobile device communicates with the merchant via some short range communication standard while simultaneously communicating with its bank over the anonymous internet connection.

The protocol uses the notion of blind signatures introduced initially in [3], but adapted to our framework. Blind signatures have been developed especially for the purpose of privacy preserving payment systems. By allowing users to request signatures from the bank without revealing the contents of the message to be signed, the user can have the bank sign transaction data and the merchant's identity without actually disclosing these values to the bank.

As opposed to previous adaptions of blind signatures to the universal composability framework, our definition is independent of the communication channels in use and so can be reused in different scenarios where other communication devices are assumed.

References

1. M. Backes, B. Pfitzmann, and M. Waidner. A general composition theorem for secure reactive systems. In *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 336–354. Springer, 2004.
2. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145. IEEE Computer Society, 2001.
3. D. Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203. Plenum Press, New York, 1982.

4. Véronique Cortier, Steve Kremer, and Bogdan Warinschi. A survey of symbolic methods in computational analysis of cryptographic systems. *J. Autom. Reasoning*, 46(3-4):225–259, 2011.
5. D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.
6. Gavin Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. In *TACAS*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer, 1996.
7. Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21:993–999, December 1978.
8. Victor Shoup. Sequences of games: A tool for taming complexity in security proofs, 2004.
9. Yiannis Tsiounis and Moti Yung. On the security of elgamal based encryption. In *Public Key Cryptography*, volume 1431 of *Lecture Notes in Computer Science*, pages 117–134. Springer, 1998.

Paper I

Dynamic Signatures with Tokens
Asgeir Steine and Øystein Thuen
Preprint

Dynamic Group Signatures with Tokens

Asgeir Steine and Øystein Thuen

NTNU, Trondheim, Norway.

Abstract. We propose a group signature scheme with efficient member revocation. In our scheme, special tokens are used in the creation of group signatures. These tokens can only be used by the group's members, and allows for anonymous group signatures. We present new security requirements for this setting and show the existence of such a scheme. Our proposed scheme uses general NIZK-proofs, and is thus not practical.

1 Introduction

Group signatures are digital signatures where users sign on behalf of a group. This allows the user to remain anonymous and still create valid signatures. A special authority known as the *opener* is allowed to reveal the identity that created a signature.

In most group signature schemes every eligible user is given its own secret key. This secret key along with the public information is enough to create signatures. One problem with this approach is to remove a user from the group, and revoke his ability to create valid signatures. Many attempts have been made to create efficient schemes where users can also easily and efficiently be revoked. In schemes where the group manager and the opener is the same entity, this is quite easy and several schemes exist. If we require that the group manager is unable to open signatures and reveal the signer, the problem of revocation becomes much harder.

We propose a new kind of group signature scheme. A user can sign messages only if he has a valid *token*. The token must be previously requested from a token issuer, and is valid for some predefined period. This approach very easily enables adding and revoking users of the system, but at the cost that users must occasionally acquire new tokens to create signatures. Such communication is not uncommon. In many networks, the users have continually contact with a base station, and receiving new tokens may not be a large extra expense.

We define new security requirements for this kind of scheme, based on the standard definitions of group signatures; *anonymity*, *untraceability* and *non-frameability*. We also show that it is possible to create a secure, fully dynamic group signature scheme using tokens. Our approach and definitions are based on the article [3], which defined dynamic group signatures without revocation.

The scheme we propose uses a general NIZK proof system. Due to these proofs, the scheme is not practical, but shows that such a system is achievable.

1.1 Related Work

We do not know of any work on group signatures using tokens as a way to implement revocation. There have however been proposed numerous group signature schemes.

Group signatures were first introduced by Chaum and van Heyst in 1991 [8]. In the original proposal the members of the group were static. Many efficient implementations of such a scheme was proposed, including [1], [4] and [6]. Some of these included adding and revoking users from the group. These schemes are all based around having a single group manager that both handles user keys and can open signatures. In this setting the users have to trust the group manager, as he will know the identity behind every signature.

In the 2004 paper [3], Bellare et al. proposed a unified framework to prove security for dynamic group signature schemes. Here they separate the Key Issuer from the Opener. This allows for stronger security definitions in which only the Opener can identify a signer, and the users remain anonymous even to the Key Issuer. Bellare et al. defines three security requirements for group signatures: *anonymity*, *traceability* and *non-frameability*. They show that various other security

definitions like unlinkability and coalition resistance are included in these three security goals. However they do not include revocation in their security definition.

Our work is strongly influenced by this paper.

After the paper [3] several group signature schemes using separate Key Issuer and Opener have been proposed. This includes [9], [13], [7] and [10]. These papers either do not include revocation or require expensive operations every time revocation occurs.

A different way to handle revocation is known as *verifier-local revocation*. Here only the verifiers have to do additional computations when revocation of users happens. Several papers use this approach [5], [2], [12], [11]. None of these papers however, separate the Key Issuer from the Opener. We do not know of any papers that use this approach and satisfies the strongest security requirements as defined by Bellare et al.

As of yet, we do not know of an efficient group signature scheme with secure revocation that satisfies the strongest security definitions. We propose a different approach that may be useful in certain applications.

2 Syntax and Notation

The group signature schemes we are considering consist of a tuple of algorithms **GS = (KeyGen, UserKeyGen, Revoke, TokenIssue, Sign, Verify, Open)**. The scheme is an interactive protocol between three types of parties, the *users*, the *token issuer* and the *opener*. The token issuer runs the algorithms *Join*, *Revoke* and *TokenIssue*. Since the token issuer decides who is eligible to receive tokens, he controls which users are allowed to create signatures. In other similar schemes, this entity is sometimes called the group manager. Each algorithm is described below.

KeyGen is the initial setup phase run by a trusted third party. The input is a security parameter k. **KeyGen** produces a triple of keys, (gpk, ok, tik). The public group key gpk is made available to everyone, the opening key ok is sent to the opener and the token issue key tik is sent to the token issuer.

UserKeyGen takes security parameter k as input and generates a key pair (rk, sk). We assume that **UserKeyGen** is trusted and every user can securely generate a key pair. We also assume that the request key rk is made public and identifies the user, while sk is securely sent to the user. This can all be implemented if the existence of a PKI is assumed.

Join A user wanting to join the group submits his identity. If the user is eligible to join, his request key is added to the list *CurrentUsers*.

Revoke has access to the user list *CurrentUsers*. On input a request key rk, the corresponding entry is removed from *CurrentUsers*.

TokenIssue takes as input the token issue key tik, a request key rk and a time t. The algorithm returns a token τ or $\perp$ if the request key is not in *CurrentUsers*.

Sign takes as input a users signing key sk, a token τ, a time t and a message m. It outputs a signature σ.

Verify takes as input the group public key, a message m, a signature σ and a time interval T. It returns 1 if σ is a valid signature on m for the given time interval and 0 otherwise.

Open takes as input the opening key ok, a message m and a valid signature σ. The algorithm tries to trace the user that produced the signature. If it is successful it returns its public key rk. Otherwise it returns 0.

These schemes work as follows. A user in the group contacts the Token Issuer with his id and a time stamp t. The time stamp is in the close future, say within one hour. The user receives a token which can be used to create signatures on any message, using the algorithm *Sign*. The signature is valid until the time specified and can be transmitted over the network. Note that even if a user is revoked, the user can use tokens until they expire.

The time period a token is valid can be adjusted to accommodate the needs of the scheme. Note that since a user identifies to the TokenIssuer, very short validity period could allow for traffic analysis when the user uses the token, and thus break the anonymity.

The Token Issuer can add and remove users using the algorithms *Join* and *Revoke*. Anyone can run Verify and choose their desired time interval. The algorithm *Open* can only be executed by the Opener, which has the opening key.

3 Security Goals

We define three experiments which define the security we are interested in. A group scheme is *traceable* if every valid signature can be opened to reveal the signer. It satisfies *non-frameability* if no subset of users, token issuer and opener together can create a signature that traces to an honest user. The last security property is *anonymity*, which ensures that only the opener can identify the signer that created a signature.

The adversaries attacking the different experiments defining security get access to a set of oracles. We describe these oracles.

AddUser takes no input and generates keys for a new user. This user is added to the group. Its public request key rk is returned.

CorruptUser takes a request key as input and returns the corresponding secret signing key.

Revoke takes a request key as input and removes this user from the group.

RequestToken takes a request key and a time stamp as input and returns a token if the request key is valid.

Sign takes a request key, a token, a message and a time stamp as input. It outputs a signature on the message which has been made by the corresponding signing key.

Open takes a message and a valid signature as input. It returns the request key of the creator of the signature.

We start by defining traceability. The adversary is given the group key and the opening key as well as access to the oracles *AddUser*, *CorruptUser*, *Revoke* and *RequestToken*. The adversary wins if he can produce a signature and a message that does not trace back to any of the users. We note that the adversary has access to *all* private keys of the system. We formally define traceability in the following experiment.

$Exp_A^{traceability}(k)$
1 $(gpk, ok, tik) \leftarrow KeyGen(k)$
2 $AllUsers \leftarrow \{\}$
3 $CurrentUsers \leftarrow \{\}$
4 $CorruptUsers \leftarrow \{\}$
5 $(m, \sigma, T) \leftarrow A(gpk, ok; AddUser, CorruptUser, Revoke, RequestToken)$
6 **if** $Verify(gpk, \sigma, m, T) = 1$ **and** $Open(ok, m, \sigma) \notin AllUsers$
7 **then return** 1
8 **return** 0

We define the advantage of an adversary A against the experiment $\mathrm{Exp}^{traceability}$ as

$$\mathrm{Adv}_A^{traceability}(k) = \Pr[Exp_A^{traceability}(k) = 1]$$

Non-frameability is defined by the experiment below. Here the adversary is given both the Opening Key and the Token Issue Key. He also has oracle access to *AddUser*, *CorruptUser*, *Revoke* and *Sign*. The adversary wins if he can produce a valid signature on a user he has not corrupted, without using the *Sign*-oracle.

$Exp_A^{non-frame}(k)$
1 $(gpk, ok, tik) \leftarrow KeyGen(k)$
2 $AllUsers \leftarrow \{\}$
3 $CurrentUsers \leftarrow \{\}$

4 $CorruptUsers \leftarrow \{\}$
5 $(m, \sigma, T) \leftarrow A(gpk, ok, tik; AddUser, CorruptUser, Revoke, Sign)$
6 **if** $Verify(gpk, m, \sigma, T) = 0$
7 **then return** 0
8 $rk \leftarrow Open(ok, m, \sigma)$
9 **if** the following hold, **then return** 1, **else return** 0
10 $rk \in AllUsers$
11 m was not queried to the $Sign$-oracle
12 $rk \notin CorruptUsers$

We define the advantage as

$$\mathrm{Adv}_A^{non-frameability}(k) = \Pr[Exp_A^{non-frame}(k) = 1],$$

and say that a scheme is non-frameable if the advantage is negligible for all polynomial time adversaries A.

For $b \in \{0, 1\}$ we use the two experiments given below to define anonymity. Here the adversary is given access to the Token Issue Key, but obviously not the opening key. The adversary has oracle access to $Challenge_b$, $Open$, $AddUser$, $CorruptUser$ and $Revoke$. The oracle $Challenge_b$ takes two request keys and one message and creates a signature using one of the keys, depending on the bit b. The adversary can access this many times, and at some point returns a guess of the value of this bit. The opening oracle will not open any of the signature that $Challenge$ returns. The adversary wins if he correctly guesses the value of b.

$Exp_A^{anonymity_b}(k)$
1 $(gpk, ok, tik) \leftarrow KeyGen(k)$
2 $CurrentUsers \leftarrow \{\}$
3 $CorruptUsers \leftarrow \{\}$
4 $b' \leftarrow A(gpk, tik; Challenge_b, Open, AddUser, CorruptUser, Revoke)$
5 **return** b'

We define the advantage of an adversary A against the experiment $\mathrm{Exp}^{anonymity}$ as

$$\mathrm{Adv}_A^{anonymity}(k) = |\Pr\left[Exp_A^{anonymity_1}(k) = 1\right] - \Pr\left[Exp_A^{anonymity_0}(k) = 1\right]|$$

and say that a scheme is anonymous if the advantage is negligible for all polynomial time adversaries.

We also need a game that describes that signatures should only be valid during some time interval. In this experiment we keep a list of users and the time interval they where eligible to receive tokens. The adversary wins if he can create a valid signature for a user in a time interval the user could not receive tokens.

The advantage is defined in the usual way.

$$\mathrm{Adv}_A^{time-non-malleability}(k) = \Pr[Exp_A^{time-non-mall}(k) = 1]$$

And we say that a scheme is time-non-malleable if the advantage is negligible for all polynomial time adversaries.

4 Building blocks

Our scheme uses three basic building blocks. We need an encryption scheme, a signature scheme and a non-interactive zero-knowledge (NIZK) proof system. We discuss the security properties we

$Exp_A^{time-non-mall}(k)$
1 $(gpk, ok, tik) \leftarrow KeyGen(k)$
2 $Users \leftarrow \{\}$
3 $CorruptUsers \leftarrow \{\}$
4 $(m, \sigma, t) \leftarrow A(gpk, ok; AddUser, CorruptUser, Revoke, TokenIssue)$
5 $rk \leftarrow Open(ok, m, \sigma)$
6 Let T be the time interval that rk could receive tokens
7 **if** $Verify(gpk, m, \sigma, t) = 1$**and** $t \notin T$**and** $rk \notin CorruptUsers$
8 **then return** 1
9 **else return** 0

require of these underlying schemes.

We denote an encryption scheme by three algorithms, the key generation algorithm K_e, the encryption Enc and decryption Dec. The security requirement we need is known as *indistinguishability under chosen ciphertext attack* (Ind-CCA) and is defined in the experiment below.

$Exp_A^{Ind-CCA_b}(k)$
1 $(pk, sk) \leftarrow K_e(k)$
2 $d \leftarrow A(pk; Ch_e(b, \cdot, \cdot), \mathrm{Dec}(sk, \cdot))$
3 **return** d

For $b \in \{0, 1\}$, we define the experiments $\mathrm{Exp}^{Ind-CCA_b}$. An adversary A is given a normal decryption oracle and a challenge oracle Ch_e. The challenge oracle takes two messages m_0 and m_1 as input and encrypts m_b according to the bit b. The decryption oracle will not decrypt any ciphertext which the challenge oracle has produced. We define the advantage of the adversaries against the two experiments in the normal way.

$$\mathrm{Adv}_A^{Ind-CCA}(k) = |\Pr[Exp_A^{Ind-CCA_1} = 1] - \Pr[Exp_A^{Ind-CCA_0} = 1]|$$

We say that the encryption scheme satisfies Ind-CCA if the advantage is negligible for all polynomial time adversaries.

The second building block is a signature scheme, denote as (K_s, Sig, Ver), where K_s is the key generation algorithm and Sig, Ver denotes the signature and verification algorithms. The security we need is *unforgeability under a chosen message attack* (unforge-CMA). This security is given in the experiment below.

$Exp_A^{unforge-CMA}(k)$
1 $(pk, sk) \leftarrow K_s(k)$
2 $(m, \sigma) \leftarrow A(pk; \mathrm{Sig}(sk, \cdot))$
3 **if** $Verify(m, \sigma) = 1$**and** A did not query $\mathrm{Sig}(sk, m)$
4 **then return** 1
5 **else return** 0

We define

$$\mathrm{Adv}_A^{unforge-CMA}(k) = \Pr[Exp_A^{unforge-CMA}(k) = 1]$$

and say that the scheme is unforgeable if the advantage is negligible for all polynomial time adversaries.

The last component we need is simulation-sound NIZK-proof for membership in a NP-language. An NP-relation ρ is a subset of $\{0, 1\}^* \times \{0, 1\}^*$ such that given (x, w), one can test for membership in ρ in polynomial time for all x. If $(x, w) \in \rho$ we say that x is a statement (or theorem) and w a corresponding witness (or proof). For a given ρ, we let L_ρ be the set of all valid statements.

The algorithms P (prove) and V (verify) has access to a common reference string. The proof systems we require has a simulator SIM. The simulator can simulate proofs that are indistinguishable from real proofs, for valid statements. The security requirements are defined below.

$Exp_A^{zk_0}(k)$
1 $(CRS, state) \leftarrow \text{SIM}(gen, k)$
2 $d \leftarrow A(CRS; Prove_0)$
3 **return** d

$Exp_A^{zk_1}(k)$
1 $(CRS, state) \xleftarrow{r} \{0,1\}^c$
2 $d \leftarrow A(CRS; Prove_1)$
3 **return** d

Let (P, V, SIM) be a proof system. To define zero-knowledge for (P, V, SIM) we define two experiments Exp^{zk_0} and Exp^{zk_1}. The oracles $Prove_0$ and $Prove_1$ takes a valid statement x and witness w as input. $Prove_0$ simulates a proof by using the proof stage of SIM, while $Prove_1$ uses the real prover P. An adversary wins if he has a non-negligible probability to distinguish these two experiments

$$\text{Adv}_A^{zero-knowledge}(k) = |\Pr[Exp_A^{zk_1}(k) = 1] - \Pr[Exp_A^{zk_0}(k) = 1]|$$

$Exp_A^{sim-soundness}(k)$
1 $(CRS, state) \leftarrow \text{SIM}(gen, k)$
2 $(x, \pi) \leftarrow A(CRS; \text{SIM}(prove, state, \cdot))$
3 **if** $x \notin L_\rho$
4 **and** π was not returned by $\text{SIM}(prove, state, \cdot)$
5 **and** $V(k, x, \pi, CRS) = 1$
6 **then return** 1
7 **else return** 0

Simulation soundness states that no adversary should be able to create valid proofs on false statements. It is defined in the experiment above. We define

$$\text{Adv}_A^{simulation-soundness}(k) = \Pr[Exp_A^{sim-soundness}(k) = 1],$$

and say that a scheme has simulation soundness if $\text{Adv}_A^{sim-soundness}(k)$ is negligible for all polynomial time adversaries A.

5 Our construction

We first explain the underlying ideas of the scheme. Each user of the system creates a pair (rk, sk) of signature keys. To create a group signature a user needs a token τ, which is a signature on his own key rk and a time stamp. The user then signs a message and token creating a signature σ_2. The user then encrypts the signature σ_2 together with the token and his own public key and proves using a NIZK-proof that everything is done properly. This binds everything together. The final group signature is the ciphertext and the NIZK-proof. The encryption is done using the Opener's public key, which enables him to reveal the identity of the signer.

We explain the scheme in detail. Let (K_e, Enc, Dec) be an IND-CCA secure encryption scheme. Let (K_s, Sig, Ver) be an Unforge-CMA secure signature scheme. Let (P, V, SIM) be a simulation sound NIZK proof system for an NP-relation ρ. We define the relation ρ as follows. Let $x = (pk_e, pk_s, t, m, c)$ be a statement and $w = (rk, \tau, \sigma_2, r)$ a corresponding witness. We define

$$(x, w) \in \rho \Leftrightarrow \begin{cases} \text{Ver}(pk_s, (rk, t), \tau) = 1 \\ \text{Ver}(rk, (\tau, m), \sigma_2) = 1 \\ \text{Enc}(pk_e, (\tau, \sigma_2, rk), r) = c \end{cases}$$

The scheme is described in detail below.

KeyGen(k)
1 $(pk_s, sk_s) \leftarrow \mathrm{K}_s(k)$
2 $(pk_e, sk_e) \leftarrow \mathrm{K}_e(k)$
3 $CRS \xleftarrow{r} \{0,1\}^c$
4 $gpk \leftarrow (k, CRS, pk_s, pk_e)$
5 $ok \leftarrow sk_e$
6 $tik \leftarrow sk_s$
7 **return** (gpk, ok, tik)

UserKeyGen(k)
1 $(rk, sk) \leftarrow K_S(k)$
2 **return** (rk, sk)

Join(rk)
1 **if** the user is not eligible to join the group
2 **then return** 0
3 Add rk to the lists *CurrentUsers* and *AllUsers*
4 **return** 1

TokenIssue(tik, rk, t)
1 **if** t is not a valid time **or** $rk \notin CurrentUsers$
2 **then return** 0
3 $\tau \leftarrow \mathrm{Sig}(tik, (rk, t))$
4 **return** τ

Sign(sk, τ, m, t)
1 $\sigma_2 \leftarrow \mathrm{Sig}(sk, (\tau, m))$
2 $r \xleftarrow{r} \{0,1\}^k$
3 $c \leftarrow \mathrm{Enc}(pk_e, (\tau, \sigma_2, rk), r)$
4 $x \leftarrow (pk_e, pk_s, t, m, c)$
5 $w \leftarrow (rk, \tau, \sigma_2, r)$
6 $\pi \leftarrow P_\rho(x, w)$
7 $\sigma \leftarrow (c, \pi)$
8 **return** σ

Verify(m, σ, t)
1 $(c, \pi) \leftarrow \sigma$
2 **if** t is not a valid time
3 **then return** 0
4 $x \leftarrow (t, m, c)$
5 **return** $V_\rho(x, \pi)$

Open(ok, m, σ, t)
1 $(c, \pi) \leftarrow \sigma$
2 $(\tau, \sigma_2, rk) \leftarrow \mathrm{Dec}(ok, c)$
3 **if** $\mathrm{Ver}(pk_s, (rk, t), \tau) = 1$**and** $\mathrm{Ver}(rk, (\tau, m), \sigma_2) = 1$
4 **then return** rk
5 **else** $return 0$

6 Security Proofs

In this section we prove that our proposed scheme satisfies the security requirements of a group signature scheme.

Theorem 1 *If* $(\mathrm{K}_e, \mathrm{Enc}, \mathrm{Dec})$ *is an IND-CCA secure encryption scheme,* (P, V, SIM) *is a NIZK proof system that satisfies soundness, simulation-soundness and zero-knowledge and* $(K_S, Sign, Verify)$ *is a signature scheme which is unforgeable against Chosen Message Attacks, then our scheme is anonymous, traceable and non-frameable.*

We first define a set of default oracles which will be used in the proofs. The oracles have access to all variables the underlying experiment creates.

Adduser()
1 $(rk, sk) \leftarrow \mathrm{K}_s(k)$
2 Add rk to *CurrentUsers* and *AllUsers*
3 $usk[rk] \leftarrow sk$
4 **return** rk

CorruptUser(rk)
1 **if** $rk \notin$ *CurrentUsers*
2 **then return** 0
3 Add rk to *CorruptUsers*
4 **return** $usk[rk]$

Revoke(rk)
1 **if** $rk \notin$ *CurrentUsers*
2 **then return**
3 **if** $rk \in$ *CorruptUsers*
4 **then** Remove rk from *CorruptUsers*
5
6 Remove rk from *CurrentUsers*

RequestToken(rk, t)
1 **if** t is not a valid time **or** $rk \notin$ *CurrentUsers*
2 **then return** 0
3 **return** $\mathrm{Sig}(tik, (rk, t))$

Sign(rk, τ, m, t)
1 **if** $rk \in$ *CurrentUsers*
2 **then return** $\mathrm{Sig}(usk[rk], \tau, m, t)$

Challenge$_b(rk_0, rk_1, m)$
1 **if** $rk_0 \notin$ *CurrentUsers***or** $rk_1 \notin$ *CurrentUsers*
2 **then return** 0
3 Let t be a valid time stamp
4 $\tau \leftarrow \mathrm{Sig}(tik, (rk_b, t))$
5 $\sigma \leftarrow Sign(usk[rk_b], \tau, m, t)$
6 Note that the adversary is not allowed to call the Open-oracle with (m, σ).
7 **return** σ

Open(m, σ)
1 **if** $Verify(m, \sigma) = 0$
2 **then return** 0
3 $rk \leftarrow Open(ok, m, \sigma)$

4 **return** rk

Lemma 2 *If* $(\mathrm{K}_e, \mathrm{Enc}, \mathrm{Dec})$ *is an IND-CCA secure encryption scheme and* (P, V, SIM) *is a NIZK proof system that satisfies simulation-soundness and zero-knowledge, then our scheme is anonymous.*

Proof. This proof follows the proof of Lemma 5.1 of [3].

To prove this lemma we create four adversaries: A_S attacking simulation soundness of (P, V, SIM), A_0 and A_1 attacking IND-CCA of $(K, \mathrm{Enc}, \mathrm{Dec})$ and A_{dist} attacking the zero knowledge of the proof system. All these algorithms use an adversary B attacking the anonymity of the group signature as a subroutine.

First the algorithm that will be used to attack the simulation soundness. We redefine *Challenge* and *Open*, while using the default oracles for the rest.

$A_S(CRS, SIM(prove, state,))$
1 $CurrentUsers \leftarrow \{\}$
2 $CorruptUsers \leftarrow \{\}$
3 $usk \leftarrow \{\}$
4 $y \leftarrow \perp$
5 $cList \leftarrow \{\}$
6 $(pk_s, tik) \leftarrow K_s(k)$
7 $(pk_e, ok) \leftarrow K_e(k)$
8 $gpk \leftarrow (k, CRS, pk_s, pk_e)$
9 $B(gpk, tik; Challenge, Open, Adduser, CorruptUser, Revoke)$
10 **return** y

Challenge(rk_0, rk_1, m)
1 **if** $rk_0 \notin CurrentUsers$**or** $rk_1 \notin CurrentUsers$
2 **then return** 0
3 $t \leftarrow 0$
4 $\tau \leftarrow \mathrm{Sig}(tik, (rk_0, t))$
5 $\sigma_2 \leftarrow \mathrm{Sig}(usk[rk_0], (\tau, m))$
6 $M \leftarrow 0^{|(\tau, \sigma_2, rk_0)|}$
7 $c \leftarrow \mathrm{Enc}(pk_e, M)$
8 Add c to $cList$
9 $x \leftarrow (pk_e, pk_s, t, m, c)$//Notice that $x \notin L_\rho$
10 $\pi \leftarrow \mathrm{SIM}(prove, state, x)$//oracle query
11 $\sigma \leftarrow (c, \pi)$
12 **return** (σ, t)

Open(m, σ, t)
1 $(c, \pi) \leftarrow \sigma$
2 $x \leftarrow (pk_e, pk_s, t, m, c)$
3 **if** $V_\rho(x, \pi) = 1$**and** $c \in cList$
4 **then** $y \leftarrow (x, \pi)$
5 $(\tau, \sigma_2, rk) \leftarrow \mathrm{Dec}(ok, c)$
6 **return** rk

Next is the two algorithms A_0 and A_1. They are symmetrical in design so we define both at the same time. Let $b \in \{0, 1\}$.

$A_b(pk; Ch_e, Dec)$
1 $(pk_s, tik) \leftarrow K_s(k)$

2 $CurrentUsers \leftarrow \{\}$
3 $CorruptUsers \leftarrow \{\}$
4 $usk \leftarrow \{\}$
5 $cList \leftarrow \{\}$
6 $d \leftarrow \perp$
7 $(CRS, state) \leftarrow SIM(gen, k)$
8 $gpk \leftarrow (k, CRS, pk_s, pk)$
9 $d' \leftarrow B(gpk, tik; Challenge_b, Open, AddUser, CorruptUser, Revoke)$
10 **if** $d \neq \perp$
11 **then return** d
12 **return** d'

Challenge$_b(rk_0, rk_1, m)$
1 **if** $rk_0 \notin CurrentUsers$**or** $rk_1 \notin CurrentUsers$
2 **then return** 0
3 $t \leftarrow 0$
4 $\tau \leftarrow \mathrm{Sig}(tik, (rk_b, t))$
5 $\sigma_2 \leftarrow \mathrm{Sig}(usk[rk_b], (\tau, m))$
6 $M_b \leftarrow (\tau, \sigma_2, rk_b)$
7 $M_{\bar{b}} \leftarrow 0^{|M_b|}$
8 $c \leftarrow Ch_e(M_b, M_{\bar{b}})$
9 $x \leftarrow (pk, pk_s, t, m, c)$
10 $\pi \leftarrow SIM(prove, state, x)$
11 Add c to $cList$
12 $\sigma \leftarrow (c, \pi)$
13 **return** (σ, t)

Open(m, σ, t)
1 We assume that (m, σ) was not returned from $Challenge_b$
2 $(c, \pi) \leftarrow \sigma$
3 $x \leftarrow (pk, pk_s, t, m, c)$
4 **if** $V_\rho(x, \pi) = 0$
5 **then return** 0
6 **if** $c \in cList$
7 **then** $d \leftarrow b$
8 **return** 0
9 $(\tau, \sigma_2, rk)Dec(c)$//oracle query
10 **return** rk

The last algorithm we need is a distinguisher for the zero knowledge experiment. This algorithm has the task of distinguishing between real proof made in accordance to a random reference string and simulated proofs made from a simulated reference string.

$A_{dist}(CRS; Prove)$
1 $(pk_s, tik) \leftarrow K_s(k)$
2 $(pk_e, ok) \leftarrow K_e(k)$
3 $gpk \leftarrow (k, CRS, pk_s, pk_e)$
4 $b \xleftarrow{r} \{0, 1\}$
5 $b' \leftarrow B(gpk, tik; Challenge_b, Open, AddUser, CorruptUser, Revoke)$
6 **if** $b' = b$
7 **then return** 1
8 **return** 0

Challenge$_b(rk_0, rk_1, m)$
1 **if** $rk_0 \notin CurrentUsers$**or** $rk_1 \notin CurrentUsers$

2 **then return** 0
3 $t \leftarrow 0$
4 $\tau \leftarrow Sig(tik, (rk_b, t))$
5 $\sigma_2 \leftarrow Sig(usk[rk_b], (\tau, m))$
6 $c \leftarrow Enc(pk_e, (\tau, \sigma_2, rk_b))$
7 $x \leftarrow (pk_e, pk_s, t, m, c)$
8 $\pi \leftarrow Prove(x)$//oracle query
9 $\sigma \leftarrow (c, \pi)$
10 **return** (σ, t)

Open(m, σ, t)
1 $(c, \pi) \leftarrow \sigma$
2 $(\tau, \sigma_2, rk) \leftarrow Dec(ok, c)$
3 **return** rk

We first look at the algorithm A_{dist} participating in the experiment $Exp^{zk_1}_{A_{dist}}(k)$. Notice that in this case, A_{dist} is given a real random string CRS and the *prove*-oracle uses the real prover P. Thus if B wins, B actually beats anonymity. The exact computation follows

$$\begin{aligned}
&\Pr[Exp^{zk_1}_{A_{dist}}(k) = 1] \qquad (1)\\
&= \Pr[B = 1 | b = 1] \cdot \Pr[b = 1] + \Pr[B = 0 | b = 0] \cdot \Pr[b = 0]\\
&= \frac{1}{2}\Pr[Exp^{anon_1}_b(k) = 1] + \frac{1}{2}\Pr[Exp^{anon_0}_b(k) = 0]\\
&= \frac{1}{2}\Pr[Exp^{anon_1}_b(k) = 1] + \frac{1}{2}(1 - \Pr[Exp^{anon_0}_b(k) = 1])\\
&= \frac{1}{2} + \frac{1}{2}\mathrm{Adv}^{anonymity}_B(k)
\end{aligned}$$

We now look at A_0 and A_1, the adversaries against Ind-CCA. The oracle $Challenge_b$ responds differently according to the underlying parameters. It either responds properly using the keys of user rk_0 or user rk_1 or c is the encryption of an all zero string. We call these different environments for 0, 1 and ϵ, and write $B(0)$, $B(1)$ and $B(\epsilon)$ respectively.

In some cases the Open oracle is unable to respond properly to B. This happens when (m, σ) is a valid signature pair that has not been returned from $Challenge_b$, but the ciphertext in σ has been returned from Ch_e. We call these events BBS_0, BBS_1 and BBS_ϵ according to how $Challenge_b$ behaves. The opposite events we call $NBBS_0$, $NBBS_1$ and $NBBS_\epsilon$.

We examine how the response of A_b corresponds to the sub algorithm B in the different settings. The events defined above helps our notation. The easiest case is when $b = 0$. Then A_0 only returns 1 if B returns 1 and $NBBS$ happens.

$$\begin{aligned}
\Pr[Exp^{Ind-CCA-0}_{A_0} = 1] &= \Pr[B(0) = 1 \wedge NBBS_0] \qquad (2)\\
&\leq \Pr[B(0) = 1]
\end{aligned}$$

$$\Pr[Exp^{Ind-CCA-1}_{A_0} = 1] = \Pr[B(\epsilon) = 1 \wedge NBBS_\epsilon] \qquad (3)$$

When $b = 1$ the algorithm A_1 may return 1 also in the case when BBS happens.

$$\Pr[Exp^{Ind-CCA-0}_{A_1} = 1] = \Pr[B(\epsilon) = 1 \wedge NBBS_\epsilon] + \Pr[BBS_\epsilon] \qquad (4)$$

$$\begin{aligned}
\Pr[Exp^{Ind-CCA-1}_{A_1} = 1] &= \Pr[B(1) = 1 \wedge NBBS_1] + \Pr[BBS_1] \qquad (5)\\
&\geq \Pr[B(1) = 1]
\end{aligned}$$

If we look at the algorithm A_S we notice that it behaves the same way A_b does when the all zero message is encrypted. A_S succeeds exactly in those cases A_b is unable to open signatures. In

these cases A_S creates a response y that breaks the simulation soundness. We get

$$\mathrm{Adv}_{A_S}^{simulation-soundness} = \Pr[A_S \text{ does not return } \perp] = \Pr[BBS_\epsilon] \tag{6}$$

We now combine this equation with 3 and 4 and we get

$$\mathrm{Adv}_{A_S}^{simulation-soundness} = \Pr[Exp_{A_1}^{Ind-CCA-0} = 1] - \Pr[Exp_{A_0}^{Ind-CCA-1} = 1] \tag{7}$$

The last case is when A_{dist} participates in $Exp_{A_{dist}}^{zk_0}(k)$. Here the prove oracle simulates the proofs using a non-random reference string. We can thus compare the probabilities with A_0 and A_1 in the Ind-CCA games. The computations follow, where we use the inequalities (2) and (5).

$$\begin{aligned}
\Pr[Exp_{A_{dist}}^{zk_0}(k) = 1] &= \Pr[B(0) = 0] \cdot \Pr[b = 0] + \Pr[B(1) = 1] \cdot \Pr[b = 1] \\
&= \frac{1}{2}(1 - \Pr[B(0) = 1] + \Pr[B(1) = 1]) \\
&\leq \frac{1}{2}(1 - \Pr[Exp_{A_0}^{Ind-CCA-0}(k) = 1] + \Pr[Exp_{A_1}^{Ind-CCA-1}(k) = 1]
\end{aligned}$$

We then combine this with equation 7.

$$2\Pr[Exp_{A_{dist}}^{zk-0}(k) = 1] \leq 1 + \mathrm{Adv}_{A_0}^{Ind-CCA}(k) + \mathrm{Adv}_{A_1}^{Ind-CCA}(k) + \mathrm{Adv}_{A_S}^{Simulation-Soundness}(k)$$

At last we combine this inequality with equation (1).

$$\mathrm{Adv}_{B}^{anonymity} \leq 2\mathrm{Adv}_{A_{dist}}^{zk} + \mathrm{Adv}_{A_0}^{Ind-CCA} + \mathrm{Adv}_{A_1}^{Ind-CCA} + \mathrm{Adv}_{A_S}^{Simulation-soundness}$$

Since this holds for any adversary B, we have proven the lemma.

Lemma 3 *If $(K_S, Sign, Verify)$ is a signature scheme which is unforgeable against Chosen Message Attacks and (P, V) is a proof system that satisfies the soundness property, then our scheme is traceable.*

Proof. Let B be any adversary against $Exp^{traceability}(k)$. We construct an adversary A against $Exp^{unforge-CMA}(k)$.

A$(pk; \mathrm{Sig}(sk, \cdot))$
1 $(pk_e, ok) \leftarrow \mathrm{K}_e(k)$
2 $CRS \xleftarrow{r} \{0,1\}^c$
3 $CurrentUsers \leftarrow \{\}$
4 $AllUsers \leftarrow \{\}$
5 $CorruptUsers \leftarrow \{\}$
6 $gpk \leftarrow (k, CRS, pk, pk_e)$
7 $(m, \sigma, t) \leftarrow B(gpk, ok; AddUser, CorruptUser, Revoke, TokenIssue$
8 $(c, \pi) \leftarrow \sigma$
9 $(\tau, \sigma_2, rk) \leftarrow \mathrm{Dec}(ok, c)$
10 **return** $((rk, t), \tau)$

We use the default oracles for AddUser, CorruptUser and Revoke, but change TokenIssue.

TokenIssue(rk, t)
1 **if** t is not a valid time **or** $rk \notin CurrentUsers$
2 **then return** 0
3 $\tau \leftarrow \mathrm{Sig}(sk, (rk, t))$ //oracle query
4 **return** τ

We now look at how the success probability of the algorithms A and B relate. Assume that B wins the traceability experiment and returns $(m, \sigma) = (m, (t, c, \pi))$. Let $x = (pk_e, pk_s, t, m, c)$,

following previous notation. It follows that $V_\rho(x,\pi)=1$. Since we are assuming that (V,P) is a sound proof system for the language L_ρ, only with a negligible probability is $x \notin L_\rho$. We can therefore assume that $x \in L_\rho$, i.e. that $\exists(rk,\tau,\sigma_2,r)$ such that

$$\begin{aligned}\mathrm{Ver}(pk_s,(rk,t),\tau)&=1\\ \mathrm{Ver}(rk,(\tau,m),\sigma_2)&=1\\ \mathrm{Enc}(pk_e,(\tau,\sigma_2,rk),r)&=c.\end{aligned}$$

Since $rk \notin AllUsers$ we can be sure that the signature τ made of the message (rk,t) was not made by the signature-oracle. It follows that if B wins the traceability experiment and we ignore the negligible probability that $x \notin L_\rho$, A succeeds in producing a forged signature.

Lemma 4 *If $(K_S, Sign, Verify)$ is a signature scheme which is unforgeable against Chosen Message Attacks and (P,V) is a proof system that satisfies the soundness property, then our scheme is non-frameable.*

Proof. Let B be any adversary against $Exp^{non-frame}(k)$. We construct an adversary A against $Exp^{unforge-CMA}(k)$.

A$(pk;\mathrm{Sig}(sk,\cdot))$
1 $(gpk,ok,tik) \leftarrow KeyGen(k)$
2 $CurrentUsers \leftarrow \{\}$
3 $CorruptUsers \leftarrow \{\}$
4 $usk \leftarrow \{\}$
5 $N \leftarrow MaxUsers(k)$
6 $v \xleftarrow{r} \{1,\ldots,N\}$
7 $counter \leftarrow 0$
8 $(m,\sigma,t) \leftarrow B(gpk,ok,tik; AddUser, CorruptUser, Revoke, Sign$
9 $(c,\pi) \leftarrow \sigma$
10 $(\tau,\sigma_2,rk) \leftarrow \mathrm{Dec}(ok,c)$
11 $x \leftarrow (pk_e,pk_s,m,c)$
12 **if** $rk = pk$**and** $V_\rho(x,\pi)=1$
13 **then return** $((\tau,m),\sigma_2)$
14 **return** 0

We need to modify the oracles $AddUser()$, $CorruptUser(rk)$ and $Sign(rk,\tau,m,t)$. We can use the default *Revoke*-oracle.

AddUser()
1 $counter \leftarrow counter+1$
2 **if** $counter = v$
3 **then** Add pk to $CurrentUsers$
4 $usk[pk] \leftarrow \perp$
5 **return** pk
6 **else** Use default $AddUser$-oracle

CorruptUser(rk)
1 **if** $rk = pk$
2 **then return** failure
3 **else** Use default $CorruptUser$-oracle

Sign(rk,τ,m,t)
1 **if** $rk = pk$**and** $rk \in CurrentUser$
2 **then** $\sigma_2 \leftarrow \mathrm{Sig}(sk,(\tau,m))$//oracle-query
3 $r \xleftarrow{r} \{0,1\}$
4 $c \leftarrow \mathrm{Enc}(pk_e,(\tau,\sigma_2,rk),r)$

```
5          x ← (pk_e, pk_s, t, m, c)
6          w ← (rk, τ, σ_2, r)
7          π ← P_ρ(x, w)
8          σ ← (c, π)
9          return σ
10     else Use default Sign-oracle
```

Assume that B wins non-frameability experiment and returns (m, σ). Using the notation from algorithm A we have that $rk \in \mathit{CurrentUsers}$, $rk \notin \mathit{CorruptUsers}$ and m was not queried to the *Sign*-oracle. We also notice that all the keys returned from *AddUser* are generated by K_s. It follows that B can not distinguish pk from the normal keys, without asking for the corresponding signing key. Thus the rk corresponding to the answer of B is equal to pk with a probability of at least $1/N$. A may still fail in this case, if x is not a valid statement even if $V_\rho(x, \pi) = 1$. However this only happens with a negligible probability under the soundness assumption. We get the following probabilities

$$\mathrm{Adv}_B^{non-frameability}(k) \leq 2^{-k} + N \cdot \mathrm{Adv}_A^{unforge-CMA}(k)$$

and the lemma is proven.

Lemma 5 *If $(K_S, Sign, Verify)$ is a signature scheme which is unforgeable against Chosen Message Attacks and (P, V) is a proof system that satisfies the soundness property, then our scheme satisfies time-non-malleability.*

Proof. Let B be any adversary against $Exp_B^{time-non-mall}(k)$. We construct an adversary A against $Exp_A^{unforge-CMA}(k)$.

```
A(pk; Sig(sk, ·))
 1  (pk_e, ok) ← K_e(k)
 2  CRS ←r {0,1}^c
 3  CurrentUsers ← {}
 4  AllUsers ← {}
 5  CorruptUsers ← {}
 6  gpk ← (k, CRSpk, pk_e)
 7  (m, σ, t) ← B(gpk, ok; AddUser, CorruptUser, Revoke, TokenIssue
 8  (c, π) ← σ
 9  (τ, σ_2, rk) ← Dec(ok, c)
10  return ((rk, t), τ)
```

We use the default oracles for *AddUser*, *CorruptUser* and *Revoke*, but change *TokenIssue*.

```
TokenIssue(rk, t)
1  if t is not a valid time  or  rk ∉ CurrentUsers
2     then return 0
3  τ ← Sig(sk, (rk, t)) //oracle query
4  return τ
```

We now look at how the success probability of the algorithms A and B relate. Assume that B wins the time-non-malleability experiment and returns $(m, \sigma, t) = (m, (c, \pi), t)$. Let $x = (pk_e, pk_s, t, m, c)$, following previous notation. It follows that $V_\rho(x, \pi) = 1$. Since we are assuming that (V, P) is a sound proof system for the language L_ρ, only with a negligible probability is $x \notin L_\rho$. We can therefore assume that $x \in L_\rho$, i.e. that $\exists (rk, \tau, \sigma_2, r)$ such that

$$\begin{aligned} \mathrm{Ver}(pk_s, (rk, t), \tau) &= 1 \\ \mathrm{Ver}(rk, (\tau, m), \sigma_2) &= 1 \\ \mathrm{Enc}(pk_e, (\tau, \sigma_2, rk), r) &= c. \end{aligned}$$

Since the TokenIssue-oracle did not issue the τ we now consider, we can be sure that the signature τ made of the message (rk, t) was not made by the signature-oracle. It follows that if B wins the experiment and we ignore the negligible probability that $x \notin L_\rho$, A succeeds in producing a forged signature.

References

1. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270. Springer, 2000.
2. G. Ateniese, D. Xiaodong Song, and G. Tsudik. Quasi-efficient revocation in group signatures. In *Financial Cryptography*, volume 2357 of *Lecture Notes in Computer Science*, pages 183–197. Springer, 2002.
3. M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 136–153. Springer, 2005.
4. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.
5. D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *ACM Conference on Computer and Communications Security*, pages 168–177. ACM, 2004.
6. X. Boyen and B. Waters. Compact group signatures without random oracles. *IACR Cryptology ePrint Archive*, 2005:381, 2005.
7. X. Boyen and B. Waters. Full-domain subgroup hiding and constant-size group signatures. In *Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2007.
8. D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer, 1991.
9. C. Delerablée and D. Pointcheval. Dynamic fully anonymous short group signatures. In *VIETCRYPT*, volume 4341 of *Lecture Notes in Computer Science*, pages 193–210. Springer, 2006.
10. J. Groth. Fully anonymous group signatures without random oracles. In *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 164–180. Springer, 2007.
11. B. Libert and D. Vergnaud. Group signatures with verifier-local revocation and backward unlinkability in the standard model. In *CANS*, volume 5888 of *Lecture Notes in Computer Science*, pages 498–517. Springer, 2009.
12. T. Nakanishi and N. Funabiki. A short verifier-local revocation group signature scheme with backward unlinkability. *IEICE Transactions*, 90-A(9):1793–1802, 2007.
13. T. Nakanishi, F. Kubooka, N. Hamada, and N. Funabiki. Group signature schemes with membership revocation for large groups. In *ACISP*, volume 3574 of *Lecture Notes in Computer Science*, pages 443–454. Springer, 2005.

Paper II

Formal Verification of Reductions in Cryptography
Kristian Gjøsteen, George Petrides, and Asgeir Steine
Published in NISK 2010

Formal Verification of Reductions in Cryptography

Kristian Gjøsteen, George Petrides, and Asgeir Steine

Department of Mathematical Sciences, NTNU

Abstract. We propose a framework for black-box security reductions suitable for computer verification. By describing protocols on the level of interactive state machines, we come up with verifiable conditions sufficient for black-box reductions. As an illustration, we rewrite the standard security proof for the ElGamal cryptosystem within our framework.

1 Introduction

As the design and analysis of cryptographic protocols develop, it is only natural that security proofs become longer and more complicated, and thus error-prone and harder to verify. As a countermeasure, we need to find new ways of presenting proofs that minimize their verification effort. In particular, the cryptographic community should aim for a "standard form" for security proofs that is (a) general so that any reduction proof can be written in that form, (b) easily verifiable so that even a computer program can verify it and (c) simple enough so that authors will actually use it.

A large class of security notions in cryptography are phrased in terms of indistinguishability games. The most fundamental examples are perhaps Ind-CPA and Ind-CCA, commonly used to define confidentiality properties for encryption schemes. Furthermore, the notion of security in the Universal Composability (UC) framework [3] is formulated in terms of the indistinguishability of the actual protocol and an ideal functionality composed with a simulator.

In this paper, we propose a framework for presenting black-box reductions of such indistinguishability games using interactive state machines.

Related Work. The sequences of games method (see for example [12]) provides a practical way of presenting security proofs, and is used by most authors in the cryptographic community today. Our approach uses the same idea but is different in that we represent the games as interactive state machines. In this way, we are able to formulate conditions for indistinguishability reductions.

Recent work in the field includes the development of CSLR, a logic for polynomial time computational indistinguishability by Zhang [15], and its use by Nowak and Zhang [7] in the implementation of security protocols and game indistinguishability proofs. Previously, Nowak [6] had developed a game indistinguishability toolbox for the proof assistant Coq. In contrast to Nowak and

Zhang, we adopt the concrete security notion of Bellare et al. [2] and parametrize the adversary's resources. A computer implementation of our framework is ongoing work.

Our state machines are similar to the Probabilistic I/O Automata (PIOAs) of Segala [11]. PIOAs are an alternative to Interactive Turing Machines (ITMs) for modelling interactive computations. Their extension to Time Bounded Task-PIOAs by Canetti et al. [4] is used in an effort to develop an analogue of the UC-framework.

Our notions of interactive state machines and systems generalize those used in [10] for studying signcryption in the UC-framework. In particular, we consider probabilistic machines rather than just deterministic ones.

Backes et al. [1] and Sprenger et al. [13] have also been working on automatic verification of cryptographic protocols.

Outline. The outline of the paper is as follows: In Section 2 we establish the standard definitions and notation needed for the rest of the paper. In Section 3 we define state machines and systems and develop the necessary theory for their usage in reductions. In Section 4 we demonstrate the theory by providing a reduction for the ElGamal cryptosystem. We end the paper with our conclusions in Section 5.

2 Preliminaries

Throughout the paper, $\mathbb{G}$ denotes a multiplicative cyclic group of order p with generator g. Moreover, we will be frequently using the notions of interactive, deterministic and probabilistic Turing machines as they appear in [8].

A triple $(g^a, g^b, g^c) \in \mathbb{G}^3$ is called a *DDH-triple* if $c = ab$. Informally, the Decisional Diffie Hellman (DDH) Assumption states that given a triple in $\mathbb{G}^3$, it is computationally hard to decide whether it is a DDH-triple or not.

In Section 4 we consider *ciphertext indistinguishability under chosen plaintext attack*, abbreviated to Ind-CPA security, as defined in [9] under the name of polynomial security.

For a set of symbols I, a finite *string* of symbols from I is a sequence of symbols $w_1 w_2 \cdots w_n$ with $w_i \in I$ for $1 \leq i \leq n$. We use the letter λ to denote the empty string. The set of finite strings of symbols from I is denoted by I^*. Given a string $v = v_1 v_2 \cdots v_n$ and $w \in I$ we will use $v||w$ to denote the appended string $v_1 v_2 \cdots v_n w$.

3 Interactive State Machines

3.1 State Machines

A *state machine* M is a tuple (S, I, f, W, s^i), where S is a set of *states*, I is a set of *inputs*, $f : S \times I \to S$ is a partial function called the *transition function*, $W \subseteq S$ is a set of *wait* states and $s^i \in W$ is the *initial* state, that satisfies the following conditions:

1. Each state $s \in S$ is a tuple of the form (i, d_s), where i is the index of the state, denoted by $\#s$, from some index set $\#S$, and the state data $d_s \in I^*$, the set of finite strings of symbols from I. For example, $s^i = (1, \lambda)$.
2. Each $s \in S$ is either saving or non-saving to mean that if $f(s', w) = s$ then $d_s = d_{s'}||w$ if s is saving and $d_s = d_{s'}$ if s is non-saving.
3. If we let $S_i = \{s \in S \mid \#s = i\}$ and for each $w \in I$ let $S_w = \{s \in S \mid (s, w) \in \text{Dom } f\} \subseteq S$ then,
   (a) $s \in W \Leftrightarrow S_{\#s} \subseteq W$.
   (b) $s \in S$ is saving $\Leftrightarrow$ all $s' \in S_{\#s}$ are saving.
   (c) for any $w \in I$ and $s, s' \in S_w$, $\#s = \#s' \Rightarrow \#f(s, w) = \#f(s', w)$.

By the last point, we can define a partial function $\tilde{f} : \#S \times I \to \#S$ by $\tilde{f}(i_1, w) = i_2$, whenever i_2 is the common index of $f(s, w)$ for all states $s \in S_{i_1} \cap S_w$. For our future convenience, we also let Alphabet $f = \{w \in I \mid S_w \neq \emptyset\}$ and for each $s \in S$ let $f_s = \{w \in I \mid \ (s, w) \in \text{Dom } f\}$.

A sequence of states $s_1, s_2, \ldots, s_{n+1}$ together with a sequence of inputs $w_1, w_2, \ldots, w_n$ such that $f(s_i, w_i) = s_{i+1}$ for $1 \leq i \leq n$, form a *transition sequence* of length n. We will call a transition sequence of length 1 simply a *transition*.

Definition 1. *A transition sequence is called* proper *if it starts and ends in wait states with only non-wait states in between.*

Definition 2. *A state machine M is called* n-proper *if:*

P1. Every $s \in S$ belongs in a finite transition sequence starting in s^i and ending in a wait state,
P2. Any proper transition sequence has length at most n.

Operation. The following informal description gives some insight to the intended operation of state machines.

When the machine is in a wait state $s \in W$, it stops and waits for *external* input (that is input given by an external source such as a user or another machine) in order to proceed, whereas when in a non-wait state $s' \notin W$, it proceeds to the next state using *internal* inputs (that is inputs that are not given nor can be observed by an external source unless output). The final input before reaching a wait state is given as *output*.

Diagram Representation. Following our definition, state machines can be represented by diagrams using the symbols of Figure 1 (μ-interior states are defined in Subsection 3.3). As an illustrative example, we consider the Coin Flipping State Machine M_{cf} of Figure 2. M_{cf} models the game where a coin is flipped and then the player is asked to guess whether it is heads or tails. The machine outputs 1 if the guess is correct and 0 if not. The states and transition function of M_{cf} are determined by the diagram.

In many cases, we can obtain a compact representation of state machines by grouping together all states with the same index and parametrizing the transitions by the state data. For example, by letting $S_1 = \{(1, \lambda)\}$, $S_2 =$

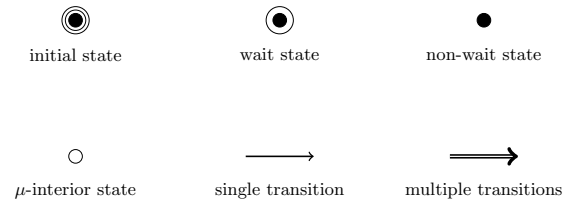

Fig. 1. The symbols used in the diagram representation of state machines.

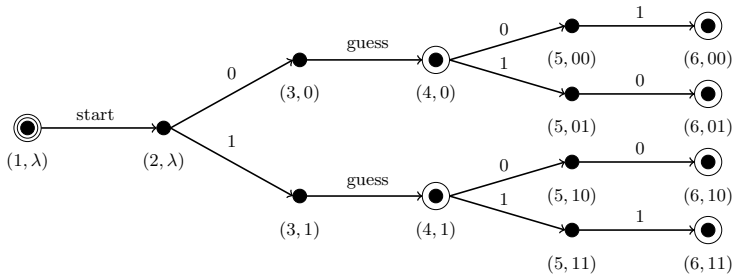

Fig. 2. The Coin Flipping State Machine M_{cf}.

$\{(2,\lambda)\}$, $S_3 = \{(3,b) \mid b \in \{0,1\}\}$, $S_4 = \{(4,b) \mid b \in \{0,1\}\}$, $S_5 = \{(5,bb^*) \mid b, b^* \in \{0,1\}\}$, and $S_6 = \{(6,bb^*) \mid b, b^* \in \{0,1\}\}$, we can represent the coin flipping machine as in Figure 3.

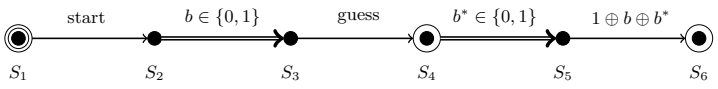

Fig. 3. Compact Representation of the Coin Flipping Machine M_{cf}.

To resolve non-determinism whenever there are multiple transitions out of a non-wait state, we associate a probability measure with each state machine, as seen in the next subsection.

3.2 Systems

A *probability measure* on a machine $M = (S, I, f, W, s^i)$ is a function $\delta_M : ((S \setminus W) \times I) \cap \operatorname{Dom} f \to [0,1]$ such that for any state $s \notin W$ with $f_s \neq \emptyset$, $\sum_{w \in f_s} \delta_M(s,w) = 1$. A pair $\Sigma_M = (M, \delta_M)$ forms a *system*. A system is *deterministic* if $\operatorname{Im} \delta_M = \{0,1\}$.

Intuitively, $\delta_M(s, w)$ gives the probability that the internal input w is used to proceed from state s to the next state.

Definition 3. *Consider a system (M, δ_M). Given $s \notin W$, we say we δ_M-sample w from f_s if for each $x \in f_s$, $\Pr[w = x] = \delta_M(s, x)$. We denote it by $w \leftarrow_{\delta_M} f_s$.*

Procedure 1 describes the behaviour of the system $\Sigma_M = (M, \delta_M)$ that when in a wait state is given input x. In words, if the next state is not a wait state, the system will δ_M-sample inputs and use them to transition until it enters a wait state.

Procedure 1: $\Sigma_M()$ (Σ_M is in any state $s \in W$)

input: $x \in I$
1. **if** $(s, x) \notin \text{Dom } f$ **then** return $\perp$
2. $s \leftarrow f(s, x)$
3. **while** $s \notin W$ **do**
   (a) $x \leftarrow_{\delta_M} f_s$
   (b) $s \leftarrow f(s, x)$
4. **return** x

The Coin Flipping Machine M_{cf} of Figure 3 can be made into a system by defining a probability measure $\delta_{M_{cf}}$. For example, if we want the coin to be fair, this can be done by defining

$$\delta_{M_{cf}}(s, w) = \begin{cases} \frac{1}{2} & \text{if } \#s = 2 \text{ and } w \in \{0, 1\} \\ 1 & \text{if } \#s = 3 \text{ and } w = guess \\ 1 & \text{if } s = (5, d_1, d_2) \text{ and } w = 1 + d_1 + d_2 \in \{0, 1\} \end{cases}.$$

3.3 Maps Between State Machines

Let $M = (S, I, f, W, s^i)$ and $N = (T, J, g, V, t^i)$ be machines and let $\mu : S \to T$ be a function.

Definition 4. *The* μ-Interior *of M is the set $S_\mu = \{s \in S \mid \mu(s) \in T \setminus V\}$.*

We can now classify inputs according to the type of the transitions that contain them as follows:
$I_i = \{w \in I \mid \exists\ s \in S_w$ such that $s, f(s, w) \in S_\mu\}$, the inputs in μ-interior to μ-interior transitions,
$I_o = \{w \in I \mid \exists\ s \in S_w$ such that $s, f(s, w) \notin S_\mu\}$, the inputs in non-$\mu$-interior to non-$\mu$-interior transitions,
$I_{oi} = \{w \in I \mid \exists\ s \in S_w$ such that $s \notin S_\mu$, and $f(s, w) \in S_\mu\}$, the inputs in non-μ-interior to μ-interior transitions, and
$I_{io} = \{w \in I \mid \exists\ s \in S_w$ such that $s \in S_\mu$, and $f(s, w) \notin S_\mu\}$, the inputs in μ-interior to non-μ-interior transitions.

Definition 5. *A transition sequence in M is μ-proper if its first and last states are its only states from $S \setminus S_\mu$.*

Definition 6. *A function $\mu : S \to T$ such that $\mu(s^i) = t^i$ and $\mu(W) \subseteq V$ is called:*

1. Alphabet-separating *if I_o, I_i, I_{io} and I_{oi} are pairwise disjoint.*
2. Sequence-preserving *if for any $s \in S \setminus S_\mu$ and $w \in I_o$ we have $(s, w) \in \mathrm{Dom}\, f \Rightarrow \mu(s) = \mu(f(s, w))$,*
3. *A* map between the machines M and N, *denoted by $\mu : M \to N$, if it is onto, alphabet separating and sequence preserving,* $\mathrm{Dom}\, g = \{(\mu(s), w) \mid (s, w) \in \mathrm{Dom}\, f$ *and $w \notin I_o\}$ and for all $(s, w) \in \mathrm{Dom}\, f$,*

$$g(\mu(s), w) = \begin{cases} \mu(f(s, w)) & \text{if } (s, w) \in \mathrm{Dom}\, f \text{ and } w \in I_{oi} \cup I_i \cup I_{io} \\ \text{undefined} & \text{otherwise} \end{cases}.$$

4. Normal *if for every $s \in S_\mu$, the index of the first state of all μ-proper transition sequences that contain a state $s' \in S_{\#s}$ is the same.*

By alphabet-separation, we can define $s|_\mu = (\#s, d_s|_{I_i})$, where for any $d \in I^*$, $d|_{I_i}$ denotes removing from d any symbols from I_i.

Let $\#S_\mu = \{\#s \mid s \in S_\mu\}$ and $\#(S \setminus S_\mu) = \{\#s \mid s \in S \setminus S_\mu\}$. By normality we can define a function $\xi : \#S_\mu \to \#(S \setminus S_\mu)$ by $\xi(i_1) = i_2$, where i_2 is the common index of the first states of all μ-proper transition sequences that contain a state $s \in S_{i_1}$.

Finally, when μ is a map between the machines M and N, we have by constructon that $\mathrm{Alphabet}\, g = \mathrm{Alphabet}\, f \setminus I_o$.

Definition 7. *A map $\mu : M \to N$ is called a* reduction map *if it is normal and*

R1. For any $w \in I_{io}$ and $s, s' \in S_w$, $\xi(\#s) = \xi(\#s') \Rightarrow \#f(s, w) = \#f(s', w)$,
R2. For any $s, s' \in S \setminus S_\mu$ we have $s|_\mu = s'|_\mu \Leftrightarrow f_s = f_{s'}$.

By Condition R1, the last states of all μ-proper transition sequences starting in states with the same index and containing $w \in I_{io}$ have the same index. Hence we can define a partial function $h : \#(S \setminus S_\mu) \times I_{io} \to \#(S \setminus S_\mu)$ by $h(i_1, w) = i_2$, whenever i_2 is the common index of the last states of all μ-proper transition sequences starting in states with index i_1 and containing w.

Definition 8. *Two systems (M, δ_M) and (N, δ_N) are called* μ-compatible *if there exists a reduction map $\mu : M \to N$ and*

C1. for all $(s, w) \in \mathrm{Dom}\, \delta_M$ for which $(\mu(s), w) \in \mathrm{Dom}\, \delta_N$ we have $\delta_M(s, w) = \delta_N(\mu(s), w)$,
C2. for any $w \in I_o \cup I_{oi}$ and $s, s' \in S_w \cap (S \setminus W)$ we have $s|_\mu = s'|_\mu \Rightarrow \delta_M(s, w) = \delta_M(s', w)$.

3.4 Reduced Systems

Let $\mu : M \to N$ be a reduction map. We define map ω with domain S by

$$\omega(s) = \begin{cases} ((\#s, 0), d_s|_\mu) & \text{if } s \notin S_\mu \\ ((\xi(\#s), 1), d_s|_\mu) & \text{if } s \in S_\mu \end{cases}.$$

Let $\bar{S} = \omega(S)$, $\bar{W} = \omega(W) \cup \omega(S_\mu)$ and $\bar{s}^i = \omega(s^i)$. We define the partial function $\bar{f} : \bar{S} \times I \to \bar{S}$ by

$$\bar{f}(\bar{s}, w) = \bar{f}(((i, b), d_{\bar{s}}), w)$$
$$= \begin{cases} ((\tilde{f}(i, w), 0), d^*_{\bar{s}}) & \text{if } w \in I_o \text{ and } \bar{s} \in \omega(S_w)\ (b = 0) \\ ((i, 1), d^*_{\bar{s}}) & \text{if } w \in I_{oi} \text{ and } \bar{s} \in \omega(S_w)\ (b = 0) \\ ((h(i, w), 0), d^*_{\bar{s}}) & \text{if } (i, w) \in \operatorname{Dom} h\ (w \in I_{io} \text{ and } b = 1) \\ \text{undefined} & \text{otherwise} \end{cases},$$

where $d^*_{\bar{s}} = \begin{cases} d_{\bar{s}} & \text{if } \bar{s} \text{ is not saving} \\ d_{\bar{s}}||_w & \text{if } \bar{s} \text{ is saving} \end{cases}$.

Note that by construction, Alphabet $\bar{f} = (\text{Alphabet } f) \setminus I_i$.

Proposition 1. *The tuple $R_\mu = (\bar{S}, I, \bar{f}, \bar{W}, \bar{s}^i)$ is a state machine.*

Proof. It is easy to see that $\bar{s}^i \in \bar{W} \subseteq \bar{S}$ and $\bar{f}$ is a partial function as, since both $\tilde{f}$ and h are partial functions, it has a unique image for each element in its domain. Moreover, each state in $\bar{S}$ is a tuple of the required form $(\#\bar{s}, d_{\bar{s}})$, where $\#\bar{s}$ is a tuple itself from $\#S \times \{0, 1\}$. We next show that $\bar{f}$ has co-domain $\bar{S}$.

In the first case, since $b = 0$ there exists $s_1 \in S \setminus S_\mu$ such that $\omega(s_1) = \bar{s}$, that is $\#s_1 = i$ and $d_{s_1}|_{I_i} = d_{\bar{s}}$. Since $\bar{s} \in \omega(S_w)$ there also exists $s_2 \in (S \setminus S_\mu)$ such that $f(s_1, w) = s_2$, implying that $\#s_2 = \tilde{f}(i, w)$ and $d_{s_2} = d^*_{s_1}$. Since $w \notin I_i$, $d^*_{s_1}|_{I_i} = d^*_{\bar{s}}$ and hence $\omega(s_2) = ((\tilde{f}(i, w), 0), d^*_{\bar{s}}) \in \bar{S}$.

In the second case, since $b = 0$ there exists $s_1 \in S \setminus S_\mu$ such that $\omega(s_1) = \bar{s}$, that is $\#s_1 = i$ and $d_{s_1}|_{I_i} = d_{\bar{s}}$. Since $\bar{s} \in \omega(S_w)$ there also exists $s_2 \in S_\mu$ such that $f(s_1, w) = s_2$ implying that $\xi(\#s_2) = i$ and $d_{s_2} = d^*_{s_1}$. As before, since $w \notin I_i$ we have $d^*_{s_1}|_{I_i} = d^*_{\bar{s}}$ and hence $\omega(s_2) = ((i, 1), d^*_{\bar{s}}) \in \bar{S}$.

Finally, in the third case, since $b = 1$ there exists $s_1 \in S_\mu$ such that $\omega(s_1) = \bar{s}$, that is $\xi(\#s_1) = i$ and $d_{s_1}|_{I_i} = d_{\bar{s}}$. Next, $(i, w) \in \operatorname{Dom} h$ implies that there exists $s_2 \in S \setminus S_\mu$ such that $f(s_1, w) = s_2$ with $\#s_2 = h(i, w)$ and $d_{s_2} = d^*_{s_1}$. Once more, since $w \notin I_i$ we have $d^*_{s_1}|_{I_i} = d^*_{\bar{s}}$ and hence $\omega(s_2) = ((h(i, w), 0), d^*_{\bar{s}}) \in \bar{S}$.

We call the state machine R_μ the *μ-reduced machine of M and N*.

Proposition 2. *Let (M, δ_M) and (N, δ_N) be μ-compatible systems. Then (R_μ, δ_{R_μ}) is a system where $\delta_{R_\mu} : (\bar{S} \setminus \bar{W} \times I) \cap \operatorname{Dom} \bar{f} \to [0, 1]$ is the following probability measure induced by δ_M:*

$$\delta_{R_\mu}(\bar{s}, w) = \delta_M(s, w), \text{ where } s \in S \text{ such that } \omega(s) = \bar{s}. \tag{1}$$

Proof. $\bar{s} = ((i, b), d_{\bar{s}}) \in \bar{S} \setminus \bar{W}$ implies that $b = 0$ and $(\bar{s}, w) \in \operatorname{Dom} \bar{f}$ further implies that $w \in I_o \cup I_{oi}$ and $\bar{s} \in \omega(S_w)$. Condition C2 guarantees that δ_{R_μ} has a unique image for each element in its domain, and consequently that it is a function.

By Condition R2, for any $s_1, s_2 \in S_i$ such that $\omega(s_1) = \omega(s_2) = \bar{s}$, we have $f_{s_1} = f_{s_2}$. We also have that $\bar{s} \in \omega(S_{w'})$ for all $w' \in f_{s_1}$ which implies $(\bar{s}, w') \in \operatorname{Dom} \bar{f}$ and hence $f_{s_1} = \bar{f}_{\bar{s}}$. Therefore, for every $s \in S$ such that $\omega(s) = \bar{s}$ we have $\sum_{w \in \bar{f}_{\bar{s}}} \delta_{R_\mu}(\bar{s}, w) = \sum_{w \in f_s} \delta_M(s, w) = 1$, as required for (R_μ, δ_{R_μ}) to be a system.

We call (R_μ, δ_{R_μ}) the μ-reduced system of (M, δ_M) and (N, δ_N).

3.5 Interleaving of Systems

Procedure 2 describes $(\Sigma_N + \Sigma_{R_\mu})$, the interleaving of systems Σ_N and Σ_{R_μ}. As we show in Proposition 3, this interleaving behaves in essentially the same way as Σ_M, a key result for the proof of our main result, Theorem 1.

Procedure 2: $(\Sigma_N + \Sigma_{R_\mu})()$ (Σ_N is in any state $t \in V$ and Σ_{R_μ} in any state $\bar{s} \in \bar{W}$)

---

input: $x \in I$

1. $z \leftarrow \Sigma_{R_\mu}(x)$
2. **if** $z \neq \perp$ **then**
   (a) $y \leftarrow \Sigma_N(x)$
   (b) **if** $y \neq \perp$ **then** $z \leftarrow \Sigma_{R_\mu}(y)$
   (c) **while** $z \in I_{oi}$ **do**
      i. $y \leftarrow \Sigma_N(z)$
      ii. $z \leftarrow \Sigma_{R_\mu}(y)$
3. **return** z

Proposition 3. *Let Σ_M and Σ_N be μ-compatible systems in states $s \in W$ and $\mu(s)$ respectively. Let Σ_{R_μ} be their μ-reduced system in state $\omega(s)$. Then, for any sequence of inputs $x, y_1, \ldots, y_n$,*

1. $\Pr[y_1, \ldots, y_n$ *is sampled when* $\Sigma_M(x)$ *is run*$] = \Pr[y_1, \ldots, y_n$ *is sampled when* $(\Sigma_N + \Sigma_{R_\mu})(x)$ *is run*$]$,
2. *if* $(\Sigma_N + \Sigma_{R_\mu})(x) = \Sigma_M(x)$ *and* Σ_M *is in state* s' *then* Σ_N *and* Σ_{R_μ} *are in states* $\mu(s')$ *and* $\omega(s')$ *respectively.*

The proof of Proposition 3 is a case by case comparison of $(\Sigma_N + \Sigma_{R_\mu})$ and Σ_M.

3.6 Simulation of Systems

Let $\Sigma_M = (M, \delta_M)$ be a system, with $M = (S, I, f, W, s^i)$. We say that a triple $X_{\Sigma_M} = (X_w, X_s, X_t)$ of Turing machines *TM-simulates* Σ_M if X_s is a probabilistic machine and X_w and X_t are deterministic machines such that X_w recognises wait states, X_t for given s and w models the transition function $f(s, w)$ and X_s assigns to each state s a random input w with probability $\delta_M(s, w)$.

Definition 9. *A system Σ_M is* simulatable *if there exists a triple X_{Σ_M} that TM-simulates it.*

Let (X_s, X_t, X_w) TM-simulate Σ_M and let Z be any interactive Turing machine (ITM). Let $I_{X_{\Sigma_M}}$ be the ITM that runs internal copies of X_s, X_t and X_w and during an interaction with Z, which it starts with $s = s^i$, follows Procedure 3 whenever Z outputs message x and gives control to $I_{X_{\Sigma_M}}$.

Procedure 3: The behaviour of $I_{X_{\Sigma_M}}$ after Z outputs message x during an interaction with $I_{X_{\Sigma_M}}$

1. **if** $x \in \{0,1\}$ **then** halt
2. **if** $X_t(s,x) = \perp$ **then** output $\perp$ and return control to Z
3. $s \leftarrow X_t(s,x)$
4. **while** $X_w(s) \neq 1$ **do**
   (a) $x \leftarrow X_s(s)$
   (b) $s \leftarrow X_t(s,x)$
5. **output** x and return control to Z

We call the interaction between $I_{X_{\Sigma_M}}$ and Z an *execution* of Z and Σ_M. Procedure 3 is a Turing machine realization of Procedure 1.

Definition 10. *A system Σ_M is* T_*-responsive *if there exists a triple X_{Σ_M} that TM-simulates Σ_M such that for any $s \in W$ and $w \in I$, the time used by $I_{X_{\Sigma_M}}$ to run Procedure 3 from state s is bounded by T_*.*

3.7 Indistinguishability of Systems

We define $E_{\Sigma_M,Z}$ as the event "during an execution with Σ_M, Z outputs 1".

Definition 11. *Two simulatable systems $\Sigma_{M,0}$ and $\Sigma_{M,1}$ are* (T_0, k, ϵ)-indistinguishable *if for any ITM Z bounded by running time T_0 and at most k outputs in total, it holds that*

$$Adv(Z, \Sigma_{M,0}, \Sigma_{M,1}) = |\Pr[E_{\Sigma_{M,0},Z}] - \Pr[E_{\Sigma_{M,1},Z}]| \leq \epsilon \ . \tag{2}$$

Furthermore, if Inequality (2) does not hold for a given Z, we will call it a (T_0, k, ϵ)-distinguisher.

Theorem 1. *Let M be an n-proper state machine and $\Sigma_{M,i} = (M, \delta_{M,i})$ and $\Sigma_{N,i} = (N, \delta_{N,i})$ be μ-compatible simulatable systems for $i \in \{0,1\}$. If*

1. *the probability measures $\delta_{R_\mu,i}$ induced on R_μ by $\delta_{M,i}$ as in Equation (1) are equal,*
2. *$\Sigma_{R_\mu} = (R_\mu, \delta_{R_\mu})$ is T_*-responsive, where $\delta_{R_\mu} = \delta_{R_\mu,i}$,*

then for any (T_0, k, ϵ)-distinguisher Z for $\Sigma_{M,0}$ and $\Sigma_{M,1}$, there exist a $(T_0 + nk(T_* + c), nk, \epsilon)$-distinguisher for $\Sigma_{N,0}$ and $\Sigma_{N,1}$, where c is the time required to determine the type of a given input $w \in I_o \cup I_{oi} \cup I_{io}$.

Proof. Since $\Sigma_{N,i}$ and Σ_{R_μ} are all simulatable for $i \in \{0,1\}$, there exist ITMs $C_{N,i+R_\mu}$ that model the behaviour of $(\Sigma_{N,i} + \Sigma_{R_\mu})$. Let Z be a (T_0, k, ϵ)-distinguisher for $\Sigma_{M,0}$ and $\Sigma_{M,1}$. By Proposition 3, an interaction of Z with $C_{N,i+R_\mu}$ as in Procedure 4 is identical to an execution of Z and $\Sigma_{M,i}$ for $i \in \{0,1\}$, and therefore the probability that Z outputs 1 is the same in both cases. Therefore, if we let Σ_N be one of $\Sigma_{N,i}$, then Z interacting with C_{N+R_μ} as in Procedure 4 gives a distinguisher for $(N, \delta_{N,i})$.

Procedure 4: Interaction of Z and C_{N+R_μ}

1. Z outputs x
2. **while** $x \notin \{0,1\}$ **do**
   (a) $u \leftarrow C_{N+R_\mu}(x)$
   (b) send u to Z and obtain new output x
3. **return** x

By definition of Z, $C_{N+R_\mu}(x)$ is invoked at most k times, and each time, $\Sigma_{R_\mu}(x)$ and $\Sigma_N(x)$ are run internally at most n times. Each run of $\Sigma_{R_\mu}(x)$ is at simulation time cost T_* and after each run, a type check of the output is performed at time cost c. The total runtime of Z is bounded by T_0. Summing up, the new distinguisher is a $(T_0 + nk(T_* + c), nk, \epsilon)$-distinguisher as required.

4 Application to the ElGamal Cryptosystem

In this section we use the state machine theory we have developed to prove that the ElGamal cryptosystem described in [5] is Ind-CPA secure, a result already established in [14]. The idea is to construct systems that model the underlying hard problem of the ElGamal cryptosystem, namely the DDH assumption, and the Ind-CPA game for ElGamal, and then apply Theorem 1 to obtain the required indistinguishability result.

The two systems $\Sigma_{N_{DDH},i}$ for $i \in \{0,1\}$ formed by pairing the machine N_{DDH} of Figure 5 with each of the probability measures $\delta_{N_{DDH},i}$ of Table 3, model the DDH assumption, which translates to saying that $\Sigma_{N_{DDH},0}$ and $\Sigma_{N_{DDH},1}$ are $(T_0, 2, \epsilon)$-indistinguishable for some large value T_0 and small value ϵ.

Next, consider the four systems $\Sigma_{M_{ElG},i_1 i_2}$ for $i_1, i_2 \in \{0,1\}$ obtained by pairing the machine M_{ElG} of Figure 4, which is 4-proper as by Definition 1, with each of the probability measures $\delta_{M_{ElG},i_1 i_2}$ of Table 1. For $i_1 = 0$, the two systems $\Sigma_{M_{ElG},0 i_2}$ model the Ind-CPA game for ElGamal. Moreover, it is a fact that $\Sigma_{M_{ElG},10}$ and $\Sigma_{M_{ElG},11}$ are $(T_0', k', 0)$-indistinguishable for any T_0' and any k' as they model random encryption of messages.

We are now in a position to define a map μ between machines M_{ElG} and N_{DDH}, as shown in Figure 6. The alphabet subsets are $I_o = \mathbb{G}$, $I_i = \mathbb{Z}_p$, $I_{io} = \mathbb{G}$ and $I_{oi} = \{\text{start, bl}\}$. Note that in the way machine M_{ElG} is described, μ will

not be alphabet separating since $I_o = I_{io}$. This can be overcome by encoding the group elements in two different ways. The seemingly redundant transition arrows with input *bl* are used to separate inputs into the desired input subsets.

It is straightforward to check that μ satisfies all of the conditions for a reduction map and hence we can construct the μ-reduced machine R_μ, given in Figure 7. Furthermore, by comparing Table 1 with Table 3, it is straightforward to verify that the systems Σ_{M_{ElG},i_1i_2} and Σ_{N_{DDH},i_1} are μ-compatible for $i_1, i_2 \in \{0,1\}$. Let $\Sigma_{R_\mu,i_1i_2} = (R_\mu, \delta_{R_\mu,i_1i_2})$ be the corresponding four μ-reduced systems, where δ_{R_μ,i_1i_2} is defined as in Equation (1) and given in Table 3. Since these are independent of i_1, we deduce that the two μ-reduced systems Σ_{R_μ,i_10} are the same, and so are Σ_{R_μ,i_11}.

Note that all the systems we have defined are simulatable. Also, since the most computationally expensive transition in Σ_{R_μ,i_1i_2} is the multiplication of two elements in $\mathbb{G}$, these systems are T_*-responsive for some small constant T_*.

Now, suppose that there exists a $(T_0, k, 2\epsilon)$-distinguisher Z for the systems $\Sigma_{M_{ElG},00}$ and $\Sigma_{M_{ElG},01}$. By the triangle inequality and the fact that $Adv(Z, \Sigma_{M_{ElG},10}, \Sigma_{M_{ElG},11}) = 0$, we have $2\epsilon \leq Adv(Z, \Sigma_{M_{ElG},00}, \Sigma_{M_{ElG},01}) \leq Adv(Z, \Sigma_{M_{ElG},00}, \Sigma_{M_{ElG},10}) + Adv(Z, \Sigma_{M_{ElG},01}, \Sigma_{M_{ElG},11})$. Therefore, we must have $Adv(Z, \Sigma_{M_{ElG},0i_2}, \Sigma_{M_{ElG},1i_2}) \geq \epsilon$ for at least one $i_2 \in \{0,1\}$. Thus, Z gives rise to a (T_0, k, ϵ)-distinguisher for at least one of the pairs $(\Sigma_{M_{ElG},0i_2}, \Sigma_{M_{ElG},1i_2})$.

Since all the conditions of Theorem 1 are satisfied, we deduce that any (T_0, k, ϵ)-distinguisher Z_{i_2} for the systems $\Sigma_{M_{ElG},0i_2}$ and $\Sigma_{M_{ElG},1i_2}$, for $i_2 \in \{0,1\}$, gives rise to a $(T_0 + 4k(T_* + c), 4k, \epsilon)$-distinguisher Z'_{i_2} for the systems $\Sigma_{N_{DDH},0}$ and $\Sigma_{N_{DDH},1}$.

We have shown that any $(T_0, k, 2\epsilon)$-distinguisher for the systems $\Sigma_{M_{ElG},00}$ and $\Sigma_{M_{ElG},01}$ modelling the Ind-CPA security of the ElGamal cryptosystem gives rise to a $(T_0 + 4k(T_* + c), 4k, \epsilon)$-distinguisher for the systems $\Sigma_{N_{DDH},0}$ and $\Sigma_{N_{DDH},1}$ modelling the DDH assumption.

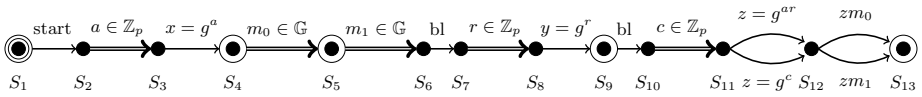

Fig. 4. The state machine M_{ElG}. Its state sets are given in Table 1.

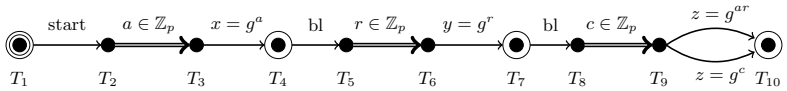

Fig. 5. The state machine N_{DDH}. Its state sets are given in Table 2.

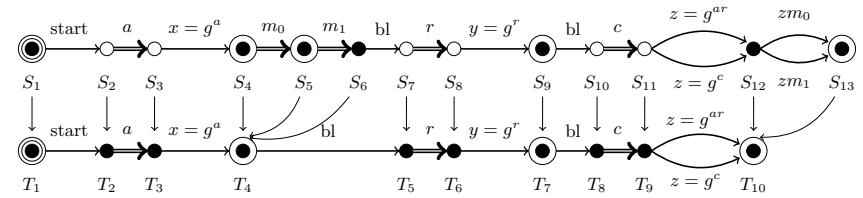

Fig. 6. The reduction map μ between M_{ElG} and N_{DDH}.

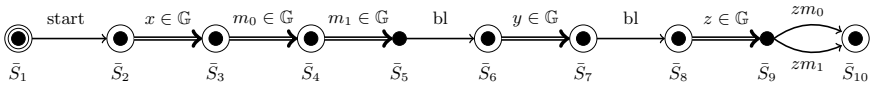

Fig. 7. The reduced state machine R_μ. Its state sets are given in Table 2.

M_{ElG}	$\delta_{M_{ElG},i_1i_2}((j, d_1 \dots d_n), w)$
$S_1 = \{(1, \lambda)\}$	
$S_2 = \{(2, \lambda)\}$	
$S_3 = \{(3, a) \mid a \in \mathbb{Z}_p\}$	
$S_4 = \{(4, a) \mid a \in \mathbb{Z}_p\}$	$\frac{1}{p}$ if $j \in \{2, 7, 10\}$ and $w \in \mathbb{Z}_p$
$S_5 = \{(5, am_0) \mid a \in \mathbb{Z}_p,\ m_0 \in \mathbb{G}\}$	1 if $j = 6$ and $w = \text{bl}$
$S_6 = \{(6, am_0m_1) \mid a \in \mathbb{Z}_p,\ m_0, m_1 \in \mathbb{G}\}$	1 if $j \in \{3, 8\}$ and $w = g^{d_n}$
$S_7 = \{(7, am_0m_1) \mid a \in \mathbb{Z}_p,\ m_0, m_1 \in \mathbb{G}\}$	1 if $j = 11,\ w = g^{d_1 \cdot d_3}$ and $i_1 = 0$
$S_8 = \{(8, am_0m_1r) \mid a, r \in \mathbb{Z}_p,\ m_0, m_1 \in \mathbb{G}\}$	1 if $j = 11,\ w = g^{d_n}$ and $i_1 = 1$
$S_9 = \{(9, am_0m_1r) \mid a, r \in \mathbb{Z}_p,\ m_0, m_1 \in \mathbb{G}\}$	1 if $j = 12,\ w = d_2 \cdot d_n$ and $i_2 = 0$
$S_{10} = \{(10, am_0m_1r) \mid a, r, \in \mathbb{Z}_p,\ m_0, m_1 \in \mathbb{G}\}$	1 if $j = 12,\ w = d_3 \cdot d_n$ and $i_2 = 1$
$S_{11} = \{(11, am_0m_1rc) \mid a, r, c \in \mathbb{Z}_p,\ m_0, m_1 \in \mathbb{G}\}$	
$S_{12} = \{(12, am_0m_1rcz) \mid a, r, c \in \mathbb{Z}_p,\ m_0, m_1, z \in \mathbb{G}\}$	
$S_{13} = \{(13, am_0m_1rcz) \mid a, r, c \in \mathbb{Z}_p,\ m_0, m_1, z \in \mathbb{G}\}$	

Table 1. The sets of states of machine M_{ElG} and the probability measures δ_{M_{ElG},i_1i_2}

N_{DDH}	R_μ
$T_1 = \{(1, \lambda)\}$	$\bar{S}_1 = \{((1,0), \lambda)\}$
$T_2 = \{(2, \lambda)\}$	$\bar{S}_2 = \{((1,1), \lambda)\}$
$T_3 = \{(3, a) \mid a \in \mathbb{Z}_p\}$	$\bar{S}_3 = \{((4,0), \lambda)\}$
$T_4 = \{(4, a) \mid a \in \mathbb{Z}_p\}$	$\bar{S}_4 = \{((5,0), m_0) \mid m_0 \in \mathbb{G}\}$
$T_5 = \{(5, a) \mid a \in \mathbb{Z}_p\}$	$\bar{S}_5 = \{((6,0), m_0 m_1) \mid m_0, m_1 \in \mathbb{G}\}$
$T_6 = \{(6, ar) \mid a, r \in \mathbb{Z}_p\}$	$\bar{S}_6 = \{((6,1), m_0 m_1) \mid m_0, m_1 \in \mathbb{G}\}$
$T_7 = \{(7, ar) \mid a, r \in \mathbb{Z}_p\}$	$\bar{S}_7 = \{((9,0), m_0 m_1) \mid m_0, m_1 \in \mathbb{G}\}$
$T_8 = \{(8, ar) \mid a, r, \in \mathbb{Z}_p\}$	$\bar{S}_8 = \{((9,1), m_0 m_1) \mid m_0, m_1 \in \mathbb{G}\}$
$T_9 = \{(9, arc) \mid a, r, c \in \mathbb{Z}_p\}$	$\bar{S}_9 = \{((12,0), m_0 m_1 z) \mid m_0, m_1, z \in \mathbb{G}\}$
$T_{10} = \{(10, arc) \mid a, r, c \in \mathbb{Z}_p\}$	$\bar{S}_{10} = \{((13,0), m_0 m_1 z) \mid m_0, m_1, z \in \mathbb{G}\}$

Table 2. The sets of states of machines N_{DDH} and R_μ

$\delta_{N_{DDH},i}((j, d_1 \dots d_n), w)$	$\delta_{R_\mu, i_1 i_2}((j, d_1 \dots d_n), w)$
$\frac{1}{p}$ if $j \in \{2,5,8\}$ and $w \in \mathbb{Z}_p$ 1 if $j \in \{3,6\}$ and $w = g^{d_n}$ 1 if $j = 9,\ w = g^{d_1 \cdot d_3}$ and $i = 0$ 1 if $j = 9,\ w = g^{d_n}$ and $i = 1$	1 if $j = 5$ and $w = \text{bl}$ 1 if $j = 9,\ w = d_1 \cdot d_n$ and $i_2 = 0$ 1 if $j = 9,\ w = d_2 \cdot d_n$ and $i_2 = 1$

Table 3. The probability measures $\delta_{N_{DDH},i}$ and $\delta_{R_\mu, i_1 i_2}$

5 Conclusions

In this paper we have developed a new framework for providing black-box reductions for cryptographic protocols. By representing security games and assumptions by state machines, providing a reduction is equivalent to providing a map having certain properties between the state machines. The potential of these properties for automated verification could allow for error-free reductions.

Plans for future work include application of the technique to more complex protocols and investigating the possibility for the extension to non-black-box reductions.

References

1. M. Backes and C. Jacobi: Cryptographically Sound and Machine-Assisted Verification of Security Protocols, Proc. STACS 2003, LNCS 2607 (2003), 675–686, Springer.
2. M. Bellare, J. Kilian, P. Rogaway: The Security of Cipher Block Chaining, Proc. CRYPTO 1994, LNCS 839 (1994), 341–358, Springer.
3. R. Canetti: Universally Composable Security: A New Paradigm for Cryptographic Protocols, Report 2000/067, IACR ePrint Archive (2005).
4. R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, O. Pereira and R. Segala: Analyzing Security Protocols Using Time-Bounded Task-PIOAs, Discrete Event Dyn. Syst., Vol. 18, (2008), 111–159, Springer.

5. T. Elgamal: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, IEEE Trans. Inf. Theory, Vol. 31, (1985), 469–472.
6. D. Nowak: On Formal Verification of Arithmetic-Based Cryptographic Primitives, Proc. ICISC 2008, LNCS 5461 (2009), 368–382, Springer.
7. D. Nowak and Y. Zhang: A Calculus for Game-based Security Proofs, Report 2010/230, IACR ePrint Archive (2010).
8. O. Goldreich: Foundations of Cryptography, Cambridge University press (2001).
9. S. Goldwasser and S. Micali: Probabilistic Encryption, Journal of Computer and System Sciences, Vol. 28 (1984), 270–299.
10. L. Kråkmo: Privacy Preserving Protocols and Security Proof Techniques, PhD-Thesis, NTNU (2009).
11. R. Segala: Modeling and Verification of Randomized Distributed Real-time Systems. PhD-Thesis, MIT (1995).
12. V. Shoup: Sequences of Games: A Tool for Taming Complexity in Security Proofs, Report 2004/332, IACR ePrint Archive (2004).
13. C. Sprenger and D. Basin: A Monad-Based Modeling and Verification Toolbox with Application to Security Protocols, Proc. TPHOLs 2007, LNCS 4732 (2007), 302–318, Springer.
14. Y. Tsiounis and M. Yung: On the Security of ElGamal Based Encryption, Proc. PKC 1998, LNCS 1431 (1998), 117–134, Springer.
15. Y. Zhang: The Computational SLR: A Logic for Reasoning about Computational Indistinuishabilty, Proc. TLCA 2009, LNCS 5608 (2009), 401–415, Springer.

Paper III

A Novel Framework for Protocol Analysis
Kristian Gjøsteen, George Petrides and Asgeir Steine
Published in JISIS vol. 1, issue 2/3 2011
(Extended abstract published in ProvSec 2011)

A Novel Framework for Protocol Analysis*

Kristian Gjøsteen, George Petrides, and Asgeir Steine

NTNU, Trondheim, Norway

Abstract. We describe a novel reformulation of Canetti's Universal Composability (UC) framework for the analysis of cryptographic protocols. Our framework is different mainly in that it is (a) based on systems of interactive Turing machines with a fixed communication graph and (b) augmented with a global message queue that allows the sending of multiple messages per activation. The first feature significantly simplifies the proofs of some framework results, such as the UC theorem, while the second can lead to more natural descriptions of protocols and ideal functionalities. We illustrate how the theory can be used with several examples.

Keywords: Protocol Security, Universal Composability.

1 Introduction

Protocol analysis is arguing whether a given protocol has desirable functional and security properties or not. Canetti [2] introduced the Universal Composability framework, which has since become the most common model for analysis of cryptographic protocols (although there are other more or less equivalent models [7]).

In Canetti's framework, analysis begins by defining a so-called ideal functionality that encapsulates some desired functionality and security properties in the form of a trusted third party. The idea is to compare a cryptographic protocol with this ideal functionality. If the protocol is in a specific sense indistinguishable from the functionality, we say that the protocol realises the functionality. If the functionality has the desired functional and security properties, then so will the protocol.

However, Canetti's definitions have more powerful properties. The most interesting property is that of composability, where any subprotocol can be replaced by an ideal functionality that is realised by the subprotocol. This is interesting from a practical point of view, since ideal functionalities are typically much easier to work with than cryptographic protocols. Replacing subprotocols by ideal functionalities can therefore simplify analysis.

We note that ideal functionalities are also used to provide ideal models for the facilities underlying protocols, such as communication networks, common reference strings, random oracles, out-of-band key agreement, etc.

* An extended abstract of this paper has been published in LNCS 6890, Springer, pp. 340–347, 2011.

One interesting feature of Canetti's approach is a strong preference for studying the single-session case, since this significantly simplifies analysis. The multi-session analysis follows from the single-session case by composing multiple instances. For many protocols, this approach is quite simply not feasible (see [4] for one example). In other cases, multi-session analysis may result in tighter concrete results (the key agreement functionality in Sect. 5.3 is one example).

Some drawbacks of Canetti's framework are more apparent in the setting of anonymous communications. One such drawback is that all protocol machines must agree on a unique session identifier before the execution. This is awkward because the anonymity requirements explicitly forbid most obvious approaches for agreeing on session identifiers.

Another drawback is the way the activation of interactive Turing machines (ITMs) is handled. When an ITM wants to send a message to another ITM, it has to write the message to the appropriate communication tape and stop its execution. The message recipient is activated, but the sender cannot control when itself will be reactivated.

Our contribution. Based on our ongoing work with practical protocols (among others with anonymous communications and electronic elections), we have developed a reformulation of Canetti's framework in terms of systems of ITMs where communication is regulated by a fixed communication graph.

In this paper we formally describe this reformulation, for which we also define the notions of emulation and realisation, and prove the usual composition theorem from [2]. Additionally, we provide several examples that demonstrate various framework features.

There are two main differences between our formulation and that of Canetti:

- Canetti uses a very dynamic setting, where instances of ITMs come into existence as needed. Instead, we consider a fixed system of ITMs with communication regulated by a fixed communication graph.
- We augment Canetti's system of activation with a message queue, where ITMs can submit multiple messages into the queue for later delivery. Moreover, ITMs can send messages to themselves and, therefore, also self-reactivate once all previously enqueued messages have been processed.

Since we use a fixed system of ITMs and a fixed communication graph, we shall usually let our protocol machines handle multiple sessions. As we have already argued, this is already necessary for some protocols, and may have other advantages as well.

The main advantage of adding a message queue is to get simpler and more natural protocol descriptions. In the field of anonymous communications, this also gives us quite natural solutions to certain problems that seem difficult to solve in Canetti's framework.

A final advantage of our formulation is that it significantly simplifies proofs of framework results. For instance, the proof of our composition theorem is essentially contained in the drawing given in Fig. 2.

We follow the concrete security approach of [1], which we believe is essential for practical applications. Canetti uses local resource bounds on the parts of the system that imply global resource bounds for the entire system. Instead, we consider systems that "usually terminate", without exceeding global resource bounds. In our experience, this simplifies many arguments significantly.

Overview. Section 2 provides the basic definitions for systems and partial systems of interactive Turing machines. Once we have a notion of executions of systems, in Sect. 3 we introduce the notion of indistinguishable systems and partial systems, and prove some basic results about indistinguishability. Section 4 defines the notion of emulation and proves our composition theorem. Finally, our framework is illustrated via examples in Sect. 5, before our conclusions appear in Sect. 6.

1.1 Definitions

An *unordered pair* is a set with one or two elements.

Our graphs will be undirected multigraphs with loops. A *graph* is a tuple (V, E, λ), where V is a set of vertices, E is a set of edges and λ is a function that assigns an unordered pair of vertices to every edge.

An *interactive Turing machine* (ITM) is a Turing machine with one read-and-write working tape and five other tapes: random, input, incoming communication, output, and outgoing communication. The first three are read-only, the latter two are write-only. The ITM may enter a special *wait state*, in which processing temporarily stops, but may be resumed later from the same state. We refer the reader to [5] for more information on ITMs.

2 Systems of Interactive Turing Machines

In this section we introduce the notion of a system, essentially an assignment of ITMs to vertices in an undirected multigraph where the edges in the graph decide which ITMs can communicate with each other. An execution of such a system begins by activating the ITM instance attached to the initial vertex. Each active instance may add any number of messages to the message queue. When done, the active instance may choose to deliver a message immediately to another instance, thereby activating it, or deliver the first message in the queue, activating its recipient.

We also define partial systems, essentially systems with "loose edges" attached to vertices in the system. Partial systems can be composed by composing the communication graphs and replacing identical "loose edges" with real edges.

2.1 Definitions

A *system* $\Sigma = (V, E, \lambda, v_0, M)$ consists of the following:

- A *communication graph* (V, E, λ).

– A distinguished *initial vertex* $v_0 \in V$.
– A function M that assigns an ITM to each $v \in V$. We denote this ITM by M_v.

The communication graph can have parallel edges, and we shall assume that every vertex has a loop.

A *partial system* $\Pi = (W, F, \lambda', N, X, \kappa)$ consists of a graph (W, F, λ'), a function N that assigns an ITM to each $v \in W$, a set X of *external* edges (disjoint from the *internal* edges F) and a function $\kappa : X \to W$ that assigns a vertex to every external edge.

A partial system is *initial* if a vertex is designated as initial. Two partial systems are *interchangeable* if they have identical external edges, and they are either both initial or both not initial.

Two partial systems $\Pi_1 = (W_1, F_1, \lambda'_1, N_1, X_1, \kappa_1)$ and $\Pi_2 = (W_2, F_2, \lambda'_2, N_2, X_2, \kappa_2)$ are *composable* if they are not both initial and have disjoint vertex and edge sets. Their *composition*, denoted by $\Pi_1 \odot \Pi_2$, is the partial system $\Pi_3 = (W_3, F_3, \lambda'_3, N_3, X_3, \kappa_3)$, where:

– $W_3 = W_1 \cup W_2$,
– $F_3 = F_1 \cup F_2 \cup (X_1 \cap X_2)$,
– $\lambda'_3(e) = \begin{cases} \lambda'_1(e) & \text{if } e \in F_1, \\ \lambda'_2(e) & \text{if } e \in F_2, \\ \{\kappa_1(e), \kappa_2(e)\} & \text{if } e \in X_1 \cap X_2, \end{cases}$
– $N_{3,v} = \begin{cases} N_{1,v} & \text{if } v \in W_1, \\ N_{2,v} & \text{if } v \in W_2, \end{cases}$
– $X_3 = (X_1 \cup X_2) \setminus (X_1 \cap X_2)$, and
– $\kappa_3(e) = \begin{cases} \kappa_1(e) & \text{if } e \in X_1, \\ \kappa_2(e) & \text{if } e \in X_2. \end{cases}$

If either Π_1 or Π_2 is initial, then so is $\Pi_1 \odot \Pi_2$. If $\Pi_1 \odot \Pi_2$ is initial and has no external edges then it is a system and we say that Π_1 *completes* Π_2 (and vice versa).

Remark 1. Given two interchangeable partial systems, a third partial system may be composable with only one of them due to conflicts with vertices or internal edges in the other. For the same reason, several other natural and desirable operations, such as the composition of a partial system with itself, are forbidden by the above constructions.

The simple solution to this technical problem is to rename edges. We shall assume that such renaming can be done without cost. Also, trivial renaming will typically go unmentioned.

2.2 Execution Model

An *execution* of a system $\Sigma = (V, E, \lambda, v_0, M)$ works with the following state:

– For each vertex $v \in V$, one instance of the ITM M_v.
– A message queue Q containing tuples $V \times E \times \{0,1\}^*$.

When the execution starts, any input to the system is written to the input tape of the M_{v_0} instance, which is then activated. Whenever an instance of an ITM M_v enters its wait state, the execution proceeds as follows:

– *Queuing a message.* If M_v has written a tuple (queue, e, m) to its communication tape and $\lambda(e) = \{v, v'\}$ for some $v' \in V$ then add (v, e, m) to Q and reactivate M_v.
– *Handing over.* If M_v has written $(\mathsf{hand{-}over}, e, m)$ to its communication tape and $\lambda(e) = \{v, v'\}$ then write (e, m) to the incoming communication tape of $M_{v'}$ and activate $M_{v'}$.
– *Delivering a message.* If M_v did not write to its communication tape and (v', e, m) is the first entry in Q with $\lambda(e) = \{v', v''\}$ then remove the entry from the queue, write (e, m) to the incoming communication tape of $M_{v''}$ and activate $M_{v''}$.
– *Default activation.* If M_v did not write to its communication tape, $v \neq v_0$ and Q is empty then activate M_{v_0}.
– *Termination.* If $v = v_0$ and M_v has halted after writing a (possibly empty) string x to its output tape then copy x to the system's output tape and stop.

Remark 2. Note that queuing messages via loop edges allows for some degree of controlled self-reactivation of ITMs.

Resource Bounds There is no requirement that an execution must terminate. However, we shall mainly be interested in systems that with high probability terminate, in which case it is interesting to consider the time required before termination. We define the total execution time of a system to be the sum of the execution time of all the ITMs plus the cost of managing activation and the message queue.

We say that a system is (t, δ)-*bounded* if it terminates after time at most t, except with probability at most δ.

Apart from execution time, there are other interesting resource measures in protocol analysis, such as the number of sessions run or the number of corrupted parties. In particular, the size of the ITM description is relevant in preventing trading execution time for description size.

In general, for some tuple of resource bounds $R = (r_1, \ldots, r_n)$, we shall say that a system is (R, δ)-*bounded* or $(r_1, \ldots, r_n, \delta)$-*bounded* if the probability that the system terminates without exceeding any of the resource bounds in R is at least $1 - \delta$.

2.3 Protocols

The notion of a system of ITMs described above is very general. Our next aim is to show how protocols fit into this general system. For this, we shall define two notions, those of *protocol machine* and *ideal functionality*.

Protocol machines and *ideal functionalities* are ITMs that expect to be attached to vertices in a communication graph with a loop and specific number of incident non-loop edges. An ideal functionality will distinguish a special *attacker edge* and consider the remaining edges as *input/output* (*i/o*) *edges*. A protocol machine will distinguish up to three classes of edges: one or more *i/o edges*, zero or more *subprotocol edges* and zero or one *corruption edge*.

Remark 3. The interpretation of this is as follows. A protocol machine is attached to a vertex in a communication graph and receives instructions over its i/o edges directing its operation. If the protocol produces output, this is sent out via the i/o edges.

An attacker may be able to corrupt the protocol machine in various ways. In this case, the attacker will send special messages to the protocol machine over its corruption edge, and the protocol machine will respond according to its programming.

The protocol machine may delegate work to subprotocols. If so, the protocol machine controls the subprotocol by sending messages over its subprotocol edges and possibly receiving subprotocol output via the same subprotocol edges.

An ideal functionality either models an ideal subprotocol or some idealisation of some feature a protocol relies on, such as a physical communications network. The i/o edges play the same role as for protocol machines.

Protocol machines and ideal functionalities are organised in a layered hierarchical fashion with protocol machines above their subprotocol machines and ideal functionalities at the bottom. This is made formal by the notion of *protocol system*, which is a non-initial partial system satisfying:

1. All the ITMs are either protocol machines or ideal functionalities.
2. Every corruption and attacker edge is external.
3. Any non-external edge is considered an i/o edge by one of its incident ITMs and a subprotocol edge by the other.
4. If every edge in the graph is considered to be directed from the ITM that considers it as a subprotocol edge to the ITM that considers it as an i/o edge, then the communication graph has no cycles except for trivial cycles made up of loop edges.

If no ITM considers an external edge to be a subprotocol edge, we say that the protocol system is *closed*. Two protocol systems are *composable* if they are composable as partial systems and the composition is again a protocol system.

Remark 4. It is possible to have a closed protocol system consisting of a single ideal functionality. As we will see later (Sect. 5), this is an important class of protocol systems.

Remark 5. Traditionally, the equivalent of a closed protocol system is assumed to interact with a partial system composed of an environment and an attacker. In our formulation, a closed protocol will be composed with an initial partial system (the environment) so that it forms a system. Informally, we may consider part of the environment to be the attacker.

Remark 6. It will be useful to assume that every protocol machine is given an "identity", and every ideal functionality associates an "identity" with each i/o edge. These identities must be consistent, in the sense that subprotocol and parent protocol machines have the same identities, and the identity the ideal functionality associates to an i/o edge corresponds to the identity of any protocol machine attached via that edge.

3 Indistinguishability

The notion of indistinguishability is central in cryptography, and we shall define two notions of indistinguishability for systems and partial systems.

A distinguisher is an algorithm that provides input to an unknown system and then tries to say something about the system based on its output. Formally, a *distinguisher* $D = (D_1, D_2)$ is a pair of algorithms such that D_1 outputs a string and a state, while D_2 takes a string and a state as input and outputs 0 or 1. An execution of D with a system Σ proceeds as follows:

1. D_1 is run and outputs x and a state.
2. The system Σ gets x as input to an execution and outputs y.
3. D_2 is given y and the state as input and outputs 0 or 1.

Remark 7. Since systems are not guaranteed to terminate, the same holds for distinguisher executions. However, if the system execution terminates, the distinguisher execution will also terminate.

We say that a distinguisher interacting with a system Σ is (R, δ)-bounded with respect Σ if the above execution does not exceed the resource bounds in R except with probability δ. Note that we consider the time used to be not just the time used for the system execution, but also the time used by the distinguisher algorithms.

3.1 Definition of Indistinguishability of Systems

Let Σ_1 and Σ_2 be two systems, D a distinguisher and R_1 and R_2 resource bounds. Let E_i be the event that the execution of D with Σ_i outputs 1, and let G_i be the event that the execution of D with Σ_i terminates without exceeding the resource bound R_i. Note that D interacting with Σ_i is (R_i, δ_i)-bounded if $\Pr[\neg G_i] \leq \delta_i$.

The *distinguishing advantage* of D with respect to the resource bounds R_1 and R_2 is defined to be

$$\mathrm{Adv}(D, \Sigma_1, \Sigma_2; R_1, R_2) = |\Pr[E_1 \wedge G_1] - \Pr[E_2 \wedge G_2]| \ .$$

A system Σ_1 is $(R_1, \delta_1, R_2, \delta_2, \epsilon)$-*indistinguishable* from system Σ_2 if for any D the following holds:

1. If (D, Σ_1) is (R_1, δ_1)-bounded then (D, Σ_2) is (R_2, δ_2)-bounded ($\Pr[\neg G_1] \leq \delta_1 \Rightarrow \Pr[\neg G_2] \leq \delta_2$).

2. $\mathrm{Adv}(D, \Sigma_1, \Sigma_2; R_1, R_2) \leq \epsilon$.

Remark 8. Note that this notion of indistinguishability need not be symmetric.

We have now defined a notion of bounded indistinguishability. Next, we define a similar notion of unbounded indistinguishability. The *unbounded distinguishing advantage* of D is defined to be

$$\mathrm{Adv}(D, \Sigma_1, \Sigma_2) = |\Pr[E_1] - \Pr[E_2]| \ .$$

We say that Σ_1 and Σ_2 are ϵ-*indistinguishable* if any distinguisher D has unbounded distinguishing advantage at most ϵ.

Remark 9. If two systems are ϵ-indistinguishable, it is easy to show that for any input distribution, the probability that the executions terminate may differ by at most ϵ.

These notions of indistinguishability for systems carry over in a natural way to partial systems. Let Π_1 and Π_2 be interchangeable partial systems. We say that Π_1 is $(R_1, \delta_1, R_2, \delta_2, \epsilon)$-*indistinguishable* (respectively ϵ-*indistinguishable*) from Π_2 if for any partial system Π_3 that completes Π_1, we have that $\Pi_1 \odot \Pi_3$ is $(R_1, \delta_1, R_2, \delta_2, \epsilon)$-indistinguishable (respectively ϵ-indistinguishable) from $\Pi_2 \odot \Pi_3$ as systems.

3.2 Basic Results

Transitivity We have a limited form of transitivity for indistinguishability. This follows from the fact that if two systems are distinguishable then one or both must be distinguishable from any third system.

Theorem 1. *Suppose Π_1, Π_2 and Π_3 are interchangeable partial systems such that Π_1 is $(R_1, \delta_1, R_2, \delta_2, \epsilon)$-indistinguishable from Π_2 and Π_2 is $(R_2, \delta_2, R_3, \delta_3, \epsilon')$-indistinguishable from Π_3. Then, Π_1 is $(R_1, \delta_1, R_3, \delta_3, \epsilon + \epsilon')$-indistinguishable from Π_3.*

Proof. Let Π_4 be any partial system that completes Π_1 and D be a distinguisher. We define E_i to be the event that the execution of D with $\Pi_i \odot \Pi_4$ outputs 1 and G_i the event that this execution terminates before exceeding the bound R_i, $1 \leq i \leq 3$.

The first condition of $(R_1, \delta_1, R_3, \delta_3, \epsilon + \epsilon')$-indistinguishability is given by the first condition of the indistinguishability assumptions:

$$\Pr[\neg G_1] \leq \delta_1 \Rightarrow \Pr[\neg G_2] \leq \delta_2 \Rightarrow \Pr[\neg G_3] \leq \delta_3.$$

The second condition can be verified by the following computation using the second condition of the indistinguishability assumptions:

$$\begin{aligned} &|\Pr[E_1 \wedge G_1] - \Pr[E_3 \wedge G_3]| \leq \\ &|\Pr[E_1 \wedge G_1] - \Pr[E_2 \wedge G_2]| + |\Pr[E_2 \wedge G_2] - \Pr[E_3 \wedge G_3]| \leq \epsilon + \epsilon' \ , \end{aligned}$$

which concludes the proof.

Composition It is clear that if we have two indistinguishable partial systems and compose them with the same partial system, the resulting (partial) systems are indistinguishable.

Theorem 2. *Suppose Π_1 and Π_2 are interchangeable partial systems such that Π_1 is $(R_1, \delta_1, R_2, \delta_2, \epsilon)$-indistinguishable from Π_2. Then, for any partial system Π_3 composable with Π_1 we have $\Pi_1 \odot \Pi_3$ is $(R_1, \delta_1, R_2, \delta_2, \epsilon)$-indistinguishable from $\Pi_2 \odot \Pi_3$.*

Proof. This follows immediately from the definition of indistinguishability since either Π_3 completes Π_i or for any partial system Π_4 that completes $\Pi_i \odot \Pi_3$, the partial system $\Pi_3 \odot \Pi_4$ completes Π_i.

Adding Resource Bounds Suppose we have two ϵ-indistinguishable partial systems. What happens if we impose resource bounds on them ? It is clear that it depends on the resource bounds since a bounded distinguisher is considered successful if, with high probability, it keeps the bound in the execution with one of the partial systems but not with the other. Theorem 3 gives us a way to convert unbounded indistinguishability to bounded indistinguishability.

Theorem 3. *Let Π_1 and Π_2 be 0-indistinguishable partial systems, and let (R_1, δ_1) and (R_2, δ_2) be resource bounds. Suppose that for any distinguisher D and partial system Π_3 that completes Π_1, if $(D, \Pi_1 \odot \Pi_3)$ is (R_1, δ_1)-bounded then $(D, \Pi_2 \odot \Pi_3)$ is (R_2, δ_2)-bounded. Then, Π_1 is $(R_1, \delta_1, R_2, \delta_2, \max\{\delta_1, \delta_2\})$-indistinguishable from Π_2.*

Proof. The first condition of indistinguishability is given by the assumption, hence we only need to show that for any distinguisher D and partial system Π_3 that completes Π_1 we have $\mathrm{Adv}(D, \Pi_1 \odot \Pi_3, \Pi_2 \odot \Pi_3; R_1, R_2) \leq \max\{\delta_1, \delta_2\}$.

For a distinguisher D and partial system Π_3 as above, let E_i be the event that an execution of D with $\Pi_i \odot \Pi_3$ outputs 1 and G_i the event that this execution terminates without exceeding the bound R_i. Then, the distinguishing advantage is $|\Pr[E_1 \wedge G_1] - \Pr[E_2 \wedge G_2]|$. First note that, trivially,

$$\Pr[E_i \wedge G_i] \leq \Pr[E_i] \ . \tag{1}$$

Also, since

$$\Pr[E_i] = \Pr[E_i \wedge G_i] + \Pr[E_i \wedge \neg G_i] \leq \Pr[E_i \wedge G_i] + \Pr[\neg G_i] \ ,$$

it follows that

$$\Pr[E_i \wedge G_i] \geq \Pr[E_i] - \Pr[\neg G_i] \ . \tag{2}$$

We now need to show that the advantage is bounded by $\max\{\delta_1, \delta_2\}$. There are two cases to consider:

Case 1: If $\Pr[E_1 \wedge G_1] - \Pr[E_2 \wedge G_2] > 0$ then by inequalities (1) and (2), and the definition of 0-indistinguishability we have

$$\begin{aligned}\Pr[E_1 \wedge G_1] - \Pr[E_2 \wedge G_2] &\leq \Pr[E_1] - (\Pr[E_2] - \Pr[\neg G_2]) \\ &= (\Pr[E_1] - \Pr[E_2]) + \Pr[\neg G_2] \\ &\leq \delta_2.\end{aligned}$$

Case 2: Similarly, if $\Pr[E_1 \wedge G_1] - \Pr[E_2 \wedge G_2] < 0$ we have

$$\begin{aligned}\Pr[E_2 \wedge G_2] - \Pr[E_1 \wedge G_1] &\leq \Pr[E_2] - (\Pr[E_1] - \Pr[\neg G_1]) \\ &= (\Pr[E_2] - \Pr[E_1]) + \Pr[\neg G_1] \\ &\leq \delta_1.\end{aligned}$$

In either case the distinguishing advantage is bounded by $\max\{\delta_1, \delta_2\}$ as required.

The same arguments can be extended to ϵ-indistinguishable partial systems which with added resource bound gives $(R_1, \delta_1, R_2, \delta_2, \epsilon + \max\{\delta_1, \delta_2\})$-indistinguishability.

Subsystem Simulator Any partial system can be simulated by a single "simulator" vertex. The "simulator" simply runs the partial system as described in the execution model. However, the simulating machine will most usually need some extra time for accounting. It is possible to prove the following theorem.

Theorem 4. *For any partial system Π there is an interchangeable single-vertex partial system Π_S that is 0-indistinguishable from Π.*

3.3 Techniques

How do we prove that two partial systems are indistinguishable? One approach is to build on the existing work for indistinguishable probability spaces. Essentially, two probability spaces are indistinguishable if it is hard to say from which space an element was sampled. Such indistinguishable spaces can be used to build indistinguishable partial systems.

Another approach for constructing indistinguishable partial systems is to identify an "exceptional" event, prove that as long as that event does not occur the partial systems are indistinguishable, and finally bound the probability of that event occurring. This bound will also bound the distinguishing advantage.

Indistinguishable Probability Spaces Let Z_1 and Z_2 be two probability spaces and assume that sampling from Z_1 and Z_2 requires identical time. A t-distinguisher for Z_1 and Z_2 is an algorithm that on input z sampled either from Z_1 or Z_2 requires time at most t (including sampling time) before outputting either 0 or 1.

We say that Z_1 and Z_2 are (t, ϵ)-*indistinguishable* if for any t-distinguisher it holds that

$$|\Pr[E_1] - \Pr[E_2]| \leq \epsilon \ ,$$

where E_i is the event that the t-distinguisher outputs 1 on input sampled from Z_i.

Define two partial systems Π_1 and Π_2 having one vertex and one external edge as follows: The first time it receives (sample) via the external edge, the machine attached to the corresponding vertex samples z from Z_i and sends $(\mathsf{sampled}, z)$ over its external edge, and nothing else happens.

Theorem 5. *Let Z_1 and Z_2 be (t, ϵ)-indistinguishable probability spaces, Π_1 and Π_2 be as above, and (R, δ) be any resource bound that has t as its time bound. Then Π_1 is $(R, \delta, R, \delta + \epsilon, \epsilon)$-indistinguishable from Π_2.*

The proof proceeds in two steps. First, Lemma 1 proves the resource bound. Then Lemma 2 bounds the distinguishing advantage.

Algorithm A with input z:	Algorithm B with input z:
1. Let D interact with the system $\Pi_1 \odot \Pi_3$ with the exception that the message Π_1 sends over its external edge is substituted by $(\mathsf{sampled}, z)$.	1. Let D interact with the system $\Pi_1 \odot \Pi_3$ with the exception that the message Π_1 sends over its external edge is substituted by $(\mathsf{sampled}, z)$.
2. If D stops without reaching the resource bound R, output 0.	2. If D outputs b without reaching the resource bound R, output b.
3. If D reaches the resource bound R, stop the interaction and output 1.	3. If D reaches the resource bound R, stop the interaction and output 0.

Fig. 1. Algorithms for the proofs of Lemma 1 and Lemma 2.

Lemma 1. *Let Z_1, Z_2, Π_1 and Π_2 be as above, Π_3 be any partial system that completes Π_1 and Π_2, and D be a distinguisher such that D interacting with $\Pi_1 \odot \Pi_3$ is (R, δ)-bounded. Then D interacting with $\Pi_2 \odot \Pi_3$ is $(R, \delta+\epsilon)$-bounded.*

Proof. Let δ_i be the probability that D interacting with $\Pi_i \odot \Pi_3$ exceeds the resource bound R. Note that $\delta_1 \leq \delta$.

The algorithm A given in Fig. 1 is a t-distinguisher for Z_1 and Z_2. We see that if its input has been sampled from Z_i, it simulates the system $\Pi_i \odot \Pi_3$ perfectly until cut-off, and the probability that it outputs 1 is δ_i.

Therefore, A is a $(t, |\delta_1 - \delta_2|)$-distinguisher for Z_1 and Z_2. This means that $|\delta_1 - \delta_2| \leq \epsilon$, by assumption, which implies that $\delta_2 \leq \delta_1 + \epsilon \leq \delta + \epsilon$.

Hence, the probability that D interacting with $\Pi_2 \odot \Pi_3$ exceeds the resource bound R is at most $\delta + \epsilon$.

Lemma 2. *Let Π_3 be a partial system that completes Π_1 and D a distinguisher such that D interacting with $\Pi_1 \odot \Pi_3$ is (R, δ)-bounded. Then the distinguishing advantage of D is at most ϵ.*

Proof. The algorithm B given in Fig. 1 is a t-distinguisher for Z_1 and Z_2. For a distinguisher D we define the events E_i that D interacting with $\Pi_i \odot \Pi_3$ outputs 1 and G_i that the interaction is time bounded by R. We see that if B's input has been sampled from Z_i, it simulates the system $\Pi_i \odot \Pi_3$ perfectly until cut-off, and the probability that it outputs 1 equals $\Pr[E_i \wedge G_i]$.

The distinguishing advantage of B is therefore equal to $|\Pr[E_1 \wedge G_1] - \Pr[E_2 \wedge G_2]|$, which, by assumption, must be at most ϵ.

This completes the proof of Theorem 5.

Exceptional Events The technique of *exceptional events* [8] is central in modern cryptography. The idea is to isolate the conditions under which two systems can deviate, and then somehow bound the probability of this exceptional event occurring. This bound then becomes a bound on how often the systems can deviate, and therefore be distinguished.

Theorem 6. *Let Π_1 and Π_2 be interchangeable partial systems, and let F_1, F_2 be ("exceptional") events for the respective partial systems. For some resource bound R, any partial system Π_3 that completes Π_1 and any distinguisher D such that $(D, \Pi_1 \odot \Pi_3)$ is (R, δ)-bounded, we have that Π_1 is $(R, \delta, R, \delta + \epsilon, \epsilon)$-indistinguishable from Π_2 if the following hold:*

1. *The probability of F_1 and F_2 occurring is ϵ.*
2. *If F_1 and F_2 do not occur then the probabilities that*
   (i) the resource bound R is exceeded, and
   (ii) the distinguisher outputs 1 without exceeding the resource bound R,
   are each identical for the two systems.

Proof. For a distinguisher D and a partial system Π_3 that completes Π_1 and Π_2 we define the events E_i that D outputs 1 when interacting with $\Pi_i \odot \Pi_3$ and G_i that the interaction does not exceed the resource bound R. The conditions for the theorem are then:

1. $Pr[F_1] = Pr[F_2] = \epsilon$,
2. $Pr[\neg G_1 | \neg F_1] = Pr[\neg G_2 | \neg F_2]$,
3. $Pr[E_1 \wedge G_1 | \neg F_1] = Pr[E_2 \wedge G_2 | \neg F_2]$.

We first assume that $Pr[\neg G_1] \leq \delta$ and prove that it implies $Pr[\neg G_2] \leq \epsilon + \delta$. Using Conditions 1 and 2 above, we have that

$$\begin{aligned} Pr[\neg G_2] &= Pr[\neg G_2 \wedge F_2] + Pr[\neg G_2 \wedge \neg F_2] \\ &\leq Pr[F_2] + Pr[\neg G_2 | \neg F_2] Pr[\neg F_2] \\ &\leq Pr[F_2] + Pr[\neg G_1 \wedge \neg F_1] \\ &\leq Pr[F_2] + Pr[\neg G_1] \\ &\leq \epsilon + \delta \ . \end{aligned}$$

Next we need to show that $|Pr[E_1 \wedge G_1] - Pr[E_2 \wedge G_2]| \leq \epsilon$. Firstly, notice that

$$Pr[E_i \wedge G_i] = Pr[E_i \wedge G_i \wedge F_i] + Pr[E_i \wedge G_i \wedge \neg F_i] \ .$$

Secondly, by Conditions 1 and 3 above, we have that

$$\begin{aligned} Pr[E_1 \wedge G_1 \wedge \neg F_1] - Pr[E_2 \wedge G_2 \wedge \neg F_2] &= Pr[E_1 \wedge G_1 | \neg F_1] Pr[\neg F_1] - Pr[E_2 \wedge G_2 | \neg F_2] Pr[\neg F_2] \\ &= (1-\epsilon)(Pr[E_1 \wedge G_1 | \neg F_1] - Pr[E_2 \wedge G_2 | \neg F_2]) \\ &= 0 \ . \end{aligned}$$

It follows that

$$\begin{aligned} |Pr[E_1 \wedge G_1] - Pr[E_2 \wedge G_2]| &= |Pr[E_1 \wedge G_1 \wedge F_1] - Pr[E_2 \wedge G_2 \wedge F_2] + 0| \\ &\leq \max\{Pr[F_1], Pr[F_2]\} \\ &= \epsilon \ , \end{aligned}$$

which concludes the proof.

4 Composition Theorem

One very useful notion is that of composability. A protocol is said to be secure if it is in some sense indistinguishable from an appropriate ideal functionality. When this secure protocol is used as a subprotocol, the following theorem proves that analysis of the parent protocol can be done using the ideal functionality abstraction, possibly a major simplification.

Let Π_1 and Π_2 be protocol systems with the same external i/o edges. We say that Π_1 $(R_1, \delta_1, R_2, \delta_2, \epsilon)$-*emulates* Π_2 if there exists a partial system $\mathcal{S}$ such that Π_1 and $\Pi_2 \odot \mathcal{S}$ are interchangeable, and Π_1 is $(R_1, \delta_1, R_2, \delta_2, \epsilon)$-indistinguishable from $\Pi_2 \odot \mathcal{S}$. In such case, we call $\mathcal{S}$ a *simulator*.

If the partial system Π_2 consists of a single ideal functionality $\mathcal{F}$, we say that Π_1 $(R_1, \delta_1, R_2, \delta_2, \epsilon)$-*realises* Π_2 and, informally, that it is $\mathcal{F}$-*secure*.

Theorem 7 (Composition). *Let Π_1, Π_2 and Π_4 be closed protocol systems and let Π_3 be a protocol system composable with Π_1 and Π_2. If Π_1 $(R_1, \delta_1, R_2, \delta_2, \epsilon)$-emulates Π_2 and $\Pi_2 \odot \Pi_3$ $(R_2, \delta_2, R_3, \delta_3, \epsilon')$-emulates Π_4 then $\Pi_1 \odot \Pi_3$ $(R_1, \delta_1, R_3, \delta_3, \epsilon + \epsilon')$-emulates Π_4.*

Proof. Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be simulators such that the partial system Π_1 is $(R_1, \delta_1, R_2, \delta_2, \epsilon)$-indistinguishable from $\Pi_2 \odot \mathcal{S}_1$ and $\Pi_2 \odot \Pi_3$ is $(R_2, \delta_2, R_3, \delta_3, \epsilon')$-indistinguishable from $\Pi_4 \odot \mathcal{S}_2$.

Firstly, by Theorem 2, the system $\Pi_1 \odot \Pi_3$ is $(R_1, \delta_1, R_2, \delta_2, \epsilon)$-indistinguishable from $\mathcal{S}_1 \odot \Pi_2 \odot \Pi_3$, and $\mathcal{S}_1 \odot \Pi_2 \odot \Pi_3$ is $(R_2, \delta_2, R_3, \delta_3, \epsilon')$-indistinguishable from $\mathcal{S}_1 \odot \mathcal{S}_2 \odot \Pi_4$.

Secondly, by Theorem 1, $\Pi_1 \odot \Pi_3$ is $(R_1, \delta_1, R_3, \delta_3, \epsilon + \epsilon')$-indistinguishable from $\mathcal{S}_1 \odot \mathcal{S}_2 \odot \Pi_4$.

Finally, by letting $\mathcal{S}_3 = \mathcal{S}_1 \odot \mathcal{S}_2$, it follows that $\Pi_1 \odot \Pi_3$ is $(R_1, \delta_1, R_3, \delta_3, \epsilon + \epsilon')$-indistinguishable from $\mathcal{S}_3 \odot \Pi_4$, as required.

Note that the technicalities encountered in the corresponding proof of [2] are avoided due to our use of a fixed communication graph and global resource bounds. A visual version of the proof is given in Fig. 2.

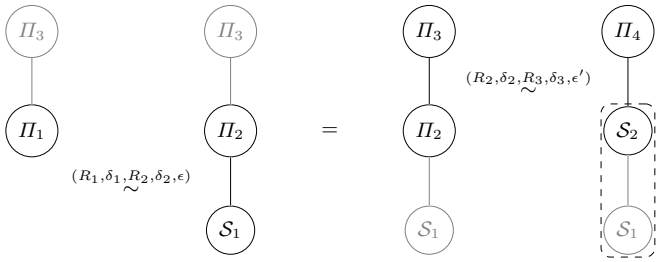

Fig. 2. Proof of Theorem 7. The bold parts of the diagram contain the indistinguishability assumptions. Theorem 2 says that adding the faded parts does not affect indistinguishability. The result follows from transitivity and taking the subsystem enclosed by the dashed lines as the required simulator.

5 Examples

We shall provide three examples of protocol analysis in our framework. The first is the now standard proof that secure digital signature schemes realise a natural signature functionality. The second example shows how a protocol can be used to simulate multiple, independent communication networks, even if only one physical network is available. Similar techniques are very useful in protocol design. The final example shows how we can prove that Diffie-Hellman is a secure key agreement protocol against passive attackers.

5.1 Signatures

A *signature scheme* Υ consists of a probabilistic key generation algorithm $\mathcal{K}$ that outputs descriptions of two algorithms, a signing algorithm s and a deterministic verification algorithm v. The signing algorithm takes a message from some message space as input, and outputs a *signature*. The verification algorithm takes a message and a signature as input, and outputs 0 or 1. We require that for any signature σ output by the signature algorithm for message m, $v(m, \sigma) = 1$.

We say that the signature scheme Υ is (t, n, ϵ)-secure if for any pair (s, v) generated by $\mathcal{K}$ and any algorithm $\mathcal{A}$ that (i) is given v as input, (ii) is allowed to get signatures on at most n messages and (iii) requires at most time t before producing some output (m, σ), the probability that $v(m, \sigma) = 1$ for a σ that was not given to $\mathcal{A}$ as a signature on m is at most ϵ.

We get a "protocol" from the signature scheme Υ as described in Fig. 3. It realises the functionality given in Fig. 4 under static corruptions.

On input (keygen):
1. Stop if s has been recorded.
2. $(s, v) \leftarrow \mathcal{K}$.
3. Record s. Output (key, v).

On input (sign, m):
1. $\sigma \leftarrow s(m)$.
2. Output $(\mathsf{signature}, \sigma)$.

On input $(\mathsf{verify}, m, \sigma, v')$:
1. Output $(\mathsf{verify}, v'(m, \sigma))$.

Fig. 3. Signature protocol from signature scheme Υ.

On (keygen) from player P:
1. Stop if (P) has been recorded.
2. Record (P).
3. Hand over (keygen, P) to $\mathcal{A}$.
4. Wait for $(\mathsf{keygen}, P, s, v)$ from $\mathcal{A}$.
5. Record (P, s, v).
6. Send (key, v) to player P.

On (sign, m) from player P:
1. Stop if (P, s, v) is not recorded.
2. Compute $\sigma \leftarrow s(m)$.
3. Record (m, σ, v).
4. Send $(\mathsf{signature}, \sigma)$ to P.

On $(\mathsf{verify}, m, \sigma, v)$ from player P:
1. If (m, σ, v) is recorded, send $(\mathsf{verify}, 1)$ to P and stop.
2. If (P', s, v) is recorded for some s and honest P', send $(\mathsf{verify}, 0)$ to P and stop.
3. If $v(m, \sigma) = 1$, record (m, σ, v).
4. Send $(\mathsf{verify}, v(m, \sigma))$ to P.

Fig. 4. Signature functionality. The adversary-supplied s and v are descriptions of algorithms, where s is probabilistic and v deterministic. We require that for any allowed m, if $\sigma \leftarrow s(m)$ then $v(m, \sigma) = 1$.

The proof proceeds as follows. First, we apply Theorem 4 to gather all honest protocol machines into a single ITM.

Next, we reorganise the functionality and add bookkeeping work (that is, recording public keys and generated signatures, and looking up keys and message/signature pairs), as well as outsource key generation to a new ITM, namely the simulator given in Fig. 5, so that the new system is identical to the ideal functionality composed with the simulator except that all valid signatures are accepted.

On (keygen, P) from the functionality:
1. $(s, v) \leftarrow \mathcal{K}$.
2. Output $(\mathsf{keygen}, P, s, v)$.

Fig. 5. Simulator for the proof of Theorem 8.

It is a tedious exercise to verify that the resulting system is 0-indistinguishable from the first system. We need extra resources for bookkeeping and sending messages to the simulator, and the amount of work depends on the exact implementation of the bookkeeping, as well as the number N of players, the number n of

requests for signing and verification, and the total length l of those requests. Let us fix some resource bound R_1 and δ and increase the resource bound R_1 to R_2 by adding sufficient resources for the bookkeeping work. Now R_2 depends on R_1, N, n and l. Increasing the resource bound to (R_2, δ) and applying Theorem 3 costs us a distinguishing advantage of δ.

Next, we begin rejecting as invalid any technically valid signatures that have not been recorded as honestly generated. Unless a forgery is generated within the time bound, the system is indistinguishable.

Assume that the time bound implied by the resource bound R_2 is t. To bound the probability of the exceptional event, namely the generation of a forgery within time bound t, we construct an attacker against the signature scheme as follows: It chooses one honest user at random from the N honest users. For that user, it uses the verification key it gets as an attacker against the signature scheme. Instead of signing messages, it queries its signing oracle. When the forgery is detected, it is a forgery for the chosen user with probability $1/N$, which means that if our time bound is t and the signature is (t, n, ϵ)-secure, then the probability of the exceptional event is upper-bounded by $N\epsilon$. Application of Theorem 6 concludes the proof sketch of the following theorem:

Theorem 8. *Let Υ be a (t, n, ϵ)-secure signature scheme. Let R_1 and R_2 be the resource bounds discussed above. Then the protocol in Fig. 3 $(R_1, \delta, R_2, \delta + N\epsilon, 2\delta + N\epsilon)$-realises the functionality in Fig. 4.*

5.2 Radio Link

Sometimes, it is easier to construct complex protocols if we can assume multiple independent communication networks, one for each subprotocol. In the real world, we usually make do with one network, and instead attach some prefix to messages identifying which subprotocol it belongs to.

As an example, we shall describe a functionality for modelling a certain form of radio communication between a set of users and a collection of network operators. Each network consists of many base stations connected by a secure network. Radio communication always happens between users and base stations. Users may communicate with multiple base stations at any one time. Which base stations they communicate with may change over time.

Each network base station is assigned a *position*. Once a user has entered the vicinity, he will receive messages broadcast by the network and he will be able to send messages to the network. The adversary may choose to listen to a certain position, in which case we model the adversary as unrealistically powerful and let the radio link around this base station degenerate into a normal adversary-controlled network. The functionality is described in Fig. 6.

Now we shall describe a multiplexing protocol that essentially pretends to provide n independent radio networks while only using one network. The protocol machine has n input edges, and communicates with one radio link functionality. Every user and network player runs one copy of the protocol machine. The protocol is described in Fig. 7.

On (enter, pos) from U:
1. Record (U, pos).

On (leave, pos) from U:
1. Erase any record (U, pos).

On (listen, pos) from $\mathcal{A}$:
1. Record (listen, pos), then hand over $(\mathsf{listen{-}ok})$ to $\mathcal{A}$.

On $(\mathsf{send}, pos, sid, m, N)$ from U:
1. Stop if (U, pos) is not recorded.
2. If (listen, pos) is recorded, hand over $(\mathsf{send}, pos, sid, m, N)$ to $\mathcal{A}$.
3. Else send $(\mathsf{recv}, pos, sid, m)$ to N.

On $(\mathsf{send}, pos, sid, m)$ from N:
1. If (listen, pos) is recorded, hand over $(\mathsf{send}, pos, sid, m, N)$ to $\mathcal{A}$.
2. Else, for any (U, pos) recorded, send $(\mathsf{recv}, pos, sid, m, N)$ to U.

On $(\mathsf{send}, pos, sid, m, N)$ from $\mathcal{A}$:
1. Stop if (listen, pos) is not recorded.
2. Send $(\mathsf{recv}, pos, sid, m)$ to N.

On $(\mathsf{send}, pos, sid, m)$ from $\mathcal{A}$:
1. Stop if (listen, pos) is not recorded.
2. For any (U, pos) recorded, send $(\mathsf{recv}, pos, sid, m)$ to U.

Fig. 6. The radio link functionality. User players are denoted by U, network players by N.

Note that the command (Leave, pos) has to be treated with a bit of care. Any message enqueued from $\mathcal{F}_{RL}$ to ϕ_n before (Leave, pos) is received by ϕ_n has to be treated as if (Leave, pos) was never received. This makes the protocol a bit non-intuitive.

If the protocol ϕ_n is properly composed with $\mathcal{F}_{RL}$, it realises n copies of $\mathcal{F}_{RL}$ composed in parallel. Verifying that ϕ_n composed with $\mathcal{F}_{RL}$ and n copies of $\mathcal{F}_{RL}$ composed with the simulator given in Fig. 8 are 0-indistinguishable is as usual a tedious affair.

All that is left is to realise that resource bounds may change, which calls for an application of Theorem 3. We fix a resource bound R_1 and increase R_1 to R_2 by adding the potential extra cost of running $\mathcal{F}_{RL1} \odot \ldots \odot \mathcal{F}_{RLn}$ instead of $\phi_n \odot \mathcal{F}_{RL}$ and arrive at the following theorem.

Theorem 9. *The protocol* $\phi_n \odot \mathcal{F}_{RL}$ $(R_1, \delta, R_2, \delta, \delta)$*-realises* $\mathcal{F}_{RL1} \odot \ldots \odot \mathcal{F}_{RLn}$.

5.3 Key Agreement

In this section we illustrate how passive security of the Diffie-Hellman key agreement can be proven within our framework. The section also serves as an example where multi-session analysis gives a tighter security reduction than basic application of the UC-theorem. For this section we shall assume that $\mathbb{G} = \langle g \rangle$ is a cyclic group of prime order p for which the distribution of random triples of the form (g^a, g^b, g^{ab}) and the distribution of completely random triples from $\mathbb{G}$ are (t, ϵ)-indistinguishable. For simplicity we assume that the protocol players communicate through an authenticated eavesdropping channel modeled as the functionality $\mathcal{F}_N$.

Figure 11 gives a protocol description of the standard Diffie-Hellman key agreement (Π_{KA}) and Fig. 10 describes the functionality that captures its security properties ($\mathcal{F}_{KA}$).

Part 1: User

On (enter, pos) from input i:
1. Record $(pos, i, 0)$ and hand over (enter, pos) to $\mathcal{F}_{RL}$.

On (leave, pos) from input i:
1. Stop if $(pos, i, 0)$ is not recorded.
2. Change the record to $(pos, i, 1)$ and send (leave, pos, i) to self.
3. If no record $(pos, j, 0)$ is left then hand over (leave, pos) to $\mathcal{F}_{RL}$.

On (leave, pos, i) from self:
1. Remove the record $(pos, i, 1)$.

On $(\mathsf{send}, pos, sid, m, N)$ from input i:
1. Stop if $(pos, i, 0)$ is not recorded.
2. Hand over $(\mathsf{send}, pos, i||sid, m, N)$ to $\mathcal{F}_{RL}$.

On $(\mathsf{recv}, pos, i||sid, m, N)$ from $\mathcal{F}_{RL}$:
1. Stop if (pos, i, b) is not recorded.
2. Hand over $(\mathsf{recv}, pos, sid, m, N)$ on input i.

Part 2: Network

On $(\mathsf{send}, pos, sid, m)$ from input i:
1. Hand over $(\mathsf{send}, pos, i||sid, m)$ to $\mathcal{F}_{RL}$.

On $(\mathsf{recv}, pos, i||sid, m)$ from $\mathcal{F}_{RL}$:
1. Hand over $(\mathsf{recv}, pos, sid, m)$ on input i.

Fig. 7. The multiplexing protocol ϕ_n. Each instance has n input/output edges, and one connection to an $\mathcal{F}_{RL}$ functionality. Its behaviour depends on whether it is acting for a user player or for a network player.

On (listen, pos) from $\mathcal{A}$:
1. For i from 1 to n, do: hand over (listen, pos) to $\mathcal{F}_{RLi}$, then wait for $(\mathsf{listen{-}ok})$ from $\mathcal{F}_{RLi}$.

On $(\mathsf{listen{-}ok})$ from $\mathcal{F}_{RLn}$:
1. Hand over $(\mathsf{Listen{-}ok})$ to $\mathcal{A}$.

On $(\mathsf{send}, pos, sid||i, m, N)$ from $\mathcal{A}$:
1. Hand over $(\mathsf{send}, pos, sid, m, N)$ to $\mathcal{F}_{RLi}$.

On $(\mathsf{send}, pos, sid||i, m)$ from $\mathcal{A}$:
1. Hand over $(\mathsf{send}, pos, sid, m)$ to $\mathcal{F}_{RLi}$.

On $(\mathsf{send}, pos, sid, m)$ from $\mathcal{F}_{RLi}$:
1. Hand over $(\mathsf{send}, pos, sid||i, m)$ to $\mathcal{A}$.

On $(\mathsf{send}, pos, sid, m, N)$ from $\mathcal{F}_{RLi}$:
1. Hand over $(\mathsf{send}, pos, sid||i, m, N)$ to $\mathcal{A}$.

Fig. 8. The simulator from the proof of Theorem 9.

We prove that the protocol Π_{KA} realises $\mathcal{F}_{KA}$ by a sequence of steps where we gradually alter the protocol system Π_{KA}. As the first step we gather all protocol parties into one simulating machine M_1 using Theorem 4. Next we facilitate the use of random self reducibility [6] by altering M_1 to a machine M_2 and introducing a new machine M_{DDH}. At the beginning of the execution, M_2 queries M_{DDH} to receive a random DDH-triple (g_1, g_2, g_k). After this, every h_i from an honest session is generated by raising g_i to a random power and multiplying with a random power of g, $i = 0, 1$:

$$h_i \leftarrow g_i^{r_i} g^{s_i} \ .$$

The r_i and s_i are recorded so that the key can be generated as

$$k \leftarrow g_k^{r_1 r_2} g_1^{r_1 s_2} g_2^{r_2 s_1} g^{s_1 s_2}.$$

Apart from this M_2 behaves exactly as M_1.

By a bit of calculation, it can be verified that the new system is 0-indistinguishable from the previous one as the triples (h_1, h_2, k) are random DDH-triples. Adding resource bounds by Theorem 3 we get that Π_{KA} is $(R_1, \delta, R_2, \delta, \delta)$-indistinguishable from $M_2 \odot M_{DDH}$, where R_1 is a bound for interactions with Π_{KA} and R_2 is obtained from R_1 by adding the resources needed to generate DDH-triples less efficiently, and simulate the protocol in one vertex.

We can now replace the machine M_{DDH} by a machine M_{RAND} that generates a random triple instead of a DDH-triple. By Theorem 5 the previous partial system is $(R_2, \delta, R_2, \delta + \epsilon, \epsilon)$-indistinguishable from the new one if the time component of R_2 is t or less. At this point each key k (with honest players) is a random group element completely independent of the corresponding h_1 and h_2. It follows that this partial system is 0-indistinguishable from a partial system $\mathcal{F}_{KA} \odot \mathcal{S}_{KA}$ where $\mathcal{F}_{KA}$ handles key generation and a machine $\mathcal{S}_{KA}$ handles the simulation of the messages with h_1 and h_2 and every corrupted key agreement session. Adding the potential extra resources needed for running $\mathcal{F}_{KA} \odot \mathcal{S}_{KA}$ instead of $M_2 \odot M_{RAND}$ and applying Theorem 3 one final time we get that $M_2 \odot M_{RAND}$ is $(R_2, \delta + \epsilon, R_3, \delta + \epsilon, \delta + \epsilon)$-indistinguishable from $\mathcal{F}_{KA} \odot \mathcal{S}_{KA}$. By the transitivity of Theorem 1 we get the following theorem.

Theorem 10. *The partial system Π_{KA} $(R_1, \delta, R_3, \delta + \epsilon, 2\delta + 2\epsilon)$-realises $\mathcal{F}_{KA}$ under the assumptions above.*

On (m, V) from a party U:
1. Send (m, U) to V and hand over $(\mathsf{m}, \mathsf{U}, \mathsf{V})$ to $\mathcal{A}$.

Fig. 9. The authenticated insecure network functionality $\mathcal{F}_N$.

On $(\mathsf{Establish}, V)$ from a party U:
1. Generate a new random identifier id and a random key $k \in \mathbb{G}$.
2. Record (U, V, id, k) and (V, U, id, k).
3. Send $(\mathsf{Establish}, U, V, id)$ to $\mathcal{A}$.

On $(\mathsf{Output}, U, \tilde{k}, id)$ from $\mathcal{A}$:
1. Stop if (U, V, id, k) is not recorded for any value of V and k.
2. If V is corrupted then send $(\mathsf{Key}, \tilde{k}, V)$ to U.
3. Else send (Key, k, V) to U.
4. In either case, remove the record (U, V, id, k).

Fig. 10. The key agreement functionality $\mathcal{F}_{KA}$.

On input $(\mathsf{Establish}, V)$:
1. Generate a random $x \in \mathbb{Z}_p$ and set $h_1 \leftarrow g^x$.
2. Record (V, x, h_1).
3. Send $((\mathsf{Establish}, h_1), V)$ to $\mathcal{F}_N$.

On $((\mathsf{Establish}, h_1), V)$ from $\mathcal{F}_N$:
1. Generate a random $y \in \mathbb{Z}_p$ and set $h_2 \leftarrow g^y$.
2. Send $((\mathsf{Respond}, h_1, h_2), V)$ to $\mathcal{F}_N$.
3. Output (Key, h_1^y, V).

On $((\mathsf{Respond}, h_1, h_2), V)$ from $\mathcal{F}_N$
1. Stop if (V, x, h_1) is not recorded.
2. Output (Key, h_2^x, V).
3. Remove record (V, x, h_1).

Fig. 11. The key agreement protocol Π_{KA} for party U.

6 Concluding Remarks

We have defined a novel framework for doing protocol analysis. The framework is based on the ideas embodied in Canetti's Universal Composability [2] and Pfitzmann–Waidner's reactive simulatability [7], but the exact formalisation is novel.

The new framework is based on our ongoing work of analysing practical protocols (for anonymous communication and electronic voting to name a few). It is our opinion that our framework is highly suitable for analysing practical protocols.

We have provided three simple examples of theorems proven in our framework. The first example is the now standard proof that a natural digital signature functionality can be realised using secure digital signatures. The second example is less standard, but shows how our framework naturally encapsulates the ideas concerning joint state composability [3]. The last example shows how passive security of the Diffie-Hellman key agreement can be proven in our framework.

References

1. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *Proc. FOCS 1997*, pp. 394–403, 1997.

2. R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. Report 2000/067, IACR ePrint Archive, 2005.
3. R. Canetti and T. Rabin. Universal Composition with Joint State. In *CRYPTO*, volume 2729 of *LNCS*, pp. 265–281, 2003.
4. K. Gjøsteen and L. Kråkmo. Universally Composable Signcryption. In *EuroPKI*, volume 4582 of *LNCS*, pp. 346–353, 2007.
5. O. Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.
6. M. Naor and O. Reingold. Number-Theoretic Constructions of Efficient Pseudo-Random Functions. *J. ACM*, 51(2) pp. 231–262, 2004.
7. B. Pfitzmann and M. Waidner. A Model for Asynchronous Reactive Systems and its Application to Secure Message Transmission. In *IEEE Symposium on Security and Privacy*, pp. 184–200, 2001.
8. V. Shoup. Sequences of Games: A Tool for Taming Complexity in Security Proofs. Report 2004/332, IACR ePrint Archive, 2004.

Paper IV

Towards Privacy Preserving Mobile Communications
Kristian Gjøsteen, George Petrides, and Asgeir Steine
Preprint

Towards Privacy Preserving Mobile Communications*

Kristian Gjøsteen, George Petrides, and Asgeir Steine

NTNU, Trondheim, Norway.

Abstract. In today's mobile communications, the privacy concerns of mobile phone users are not dealt with. For example, mobile network operators need to know the location of each device at any given time in order to provide seamless services. In this paper we develop a universally composable anonymous internet access protocol and sketch how this can be the basis for privacy preserving mobile communications. As a side contribution, to simplify protocol analysis we describe, ideal functionalities for all the related notions.

1 Introduction

In today's mobile communications, it is a functional requirement that users of mobile phones give frequent updates of their location to the *mobile network operator* (MNO) they are connected to in order to be continuously able to receive calls and enjoy seamless communication. This updating is done by initially sending the *International Mobile Subscriber Identity* (IMSI) that is embedded on their smart card and uniquely identifies them, and subsequently a *Temporary Mobile Subscriber Identity* (TMSI) generated by the MNO, to the MNO [1].

Despite the fact that IMSIs are transmitted as rarely as possible and TMSIs are changed frequently by the MNO in order to prevent user tracing by eavesdroppers on the radio link, some privacy concerns are raised by the MNO's ability to learn each user's location at any given time.

Our contribution. In this article we show that the above concerns can be addressed in a setting where MNOs own and maintain the network infrastructure but do not have direct relationship with customers. Instead, distinct middle agents, the *service providers* (SPs), deal with user subscriptions for network access. Our preference for this over the current setting will be discussed later on.

Assuming such a setting, we demonstrate that it is possible for users to (i) anonymously establish encrypted communication channels with MNOs with indirect authentication via their subscribed SP every time they change their location, and consequently to (ii) anonymously access the internet using ephemeral pseudonyms provided by MNOs. In this way, MNOs are prevented from following specific users as they move around network locations, while at the same time important features like user and network authentication and radio link encryption are maintained.

* Funded by the Norwegian Research Council's VERDIKT programme project 183195.

Although anonymous internet access does not directly imply privacy preserving seamless communications, it can be used as a foundation. We illustrate this through a protocol that allows users to make and receive seamless calls over the internet that are, unlike today, anonymous and private with respect to MNOs, SPs and *telephony providers* (TPs). Similar constructions can be used for different kinds of mobile services.

A sequence of ideal functionalities captures the desired properties we mentioned. More precisely, anonymous and secure communication channel is captured by $\mathcal{F}_{\mathsf{AKE}}$ and $\mathcal{F}_{\mathsf{SARL}}$ of Sect. 3 and anonymous internet access by $\mathcal{F}_{\mathsf{AIA}}$ of Sect. 4 . These functionalities are contributions of this work on their own, and the corresponding protocols that we present in each section UC-realise them (see Sect. 2 for an informal definition).

Using such a modular approach helps to better understand and more easily analyse the various protocols. Furthermore, each subsequent protocol is built using the functionality realised by the preceding protocol, instead of the protocol itself, again to simplify analysis. All protocols and functionalities are described in the model of [7], summarised in Sect. 2, as the traditional sequential scheduling model of the universal composability framework [3] would lead to a rather unnatural flow of communication.

At a first glance, the solutions we have proposed might seem not to comply with the laws of several countries, such as the European Union's Data Retention Directive [6], which require that records of information related to the activity of users, like network entry point location for instance, are kept for some specified amount of time. However, as will be seen in Sect. 3, anonymity and location hiding are made possible by replacing TMSIs by temporary user generated session identifiers (SIDs) shared between MNOs and SPs. Therefore, if MNOs keep records of the locations of SIDs, and SPs records of users' identities associated with SIDs, a judicially invoked MNO-SP collaboration can disclose all the required information about specific users. Similarly, MNOs, SPs and TPs can together reveal all information regarding internet telephony since pseudonyms are associated with SIDs. Moreover, dividing the stored information into parts that on their own leak almost none of the private data of users, might have the effect of making such controversial laws be better received by the public. Discussing this further, including the exact details, is beyond the scope of this work.

Another issue that needs to be addressed is that of billing, even though precise details are not in the scope of the present article. Possible solutions that do not violate any of the anonymity requirements include fixed term contracts between users and SPs and SPs and MNOs, or having MNOs charge SPs according to the number of their (anonymous) subscribers that connect to the network, without revealing any location information. TPs can also charge a fixed fee, or, since their sole function is that of an online directory and we do not require that they should be distinct from SPs, offer free services.

Finally, we point out that the kind of setting we consider is not unrealistic as it is similar to what is realised today with *virtual mobile network operators* in the

role of SPs. This is an indication that, even though it is not a common setting, given appropriate incentives, like those stated above, it can become widespread.

Related work. Anonymity has been considered in many fields of cryptography. Modern anonymising networks, which are based on the mix networks of [5] and onion routers (most notably the TOR Project [10]), enhance user privacy by mixing up the network traffic of different users to make tracing and traffic analysis more difficult. Although they are interesting tools for obtaining unlinkability, in the mobile devices case they can only be used once the device has established connection to an MNO, a connection we require to be anonymous.

It is theoretically possible to fulfill the privacy requirements we have set even without the distinction of MNOs and SPs, by using anonymous authentication schemes (for example [8, 9]) or the more general anonymous credential schemes of [2]. Modern mobile devices often have adequate computing power to handle asymmetric cryptography. Nevertheless, we do not consider it to be a practical solution for user authentication purposes since, in our case, it opens up for denial of service attacks against SPs (see Sect. 3) by anonymous attackers. Preferably, we would like to reserve a player's ability to mount denial of service attacks until after that player has been authenticated and could thus be identified.

Other work on anonymity in mobile communications concerns a user's privacy with respect to various content service providers and eavesdroppers, whereas the MNO is assumed to know both its identity and location (e.g. [11, 12]).

Outline. Section 2 contains all preliminary results, notation and definitions needed for the remaining sections. In Sects. 3 and 4 we describe the functionalities that respectively achieve secure and anonymous network connection and anonymous internet access, together with the protocols that UC-realise them. In section 5 we sketch how these functionalities can be used as a basis for anonymous telephony services. We end the paper with our conclusions in Sect. 6.

2 Preliminaries

2.1 The Universal Composability Framework

In [3], Canneti proposed *universally composable (UC) security* that allows preservation of protocol security when several protocols are composed into larger ones. Briefly, the UC model can be described as follows:

A protocol π is executed in an environment $\mathcal{Z}$ which provides the input to each honest party P participating in π through a unique communication channel, which we denote by $\tilde{P}$. When an honest party P completes its part of π, it sends its output to $\mathcal{Z}$, also through $\tilde{P}$. An adversary $\mathcal{A}$ attacking π interacts with $\mathcal{Z}$ and entirely controls the behaviour of all corrupted parties participating in π.

An *ideal functionality* $\mathcal{F}$ is an ideal process for carrying out a given cryptographic task. If a protocol π is replaced by an ideal functionality $\mathcal{F}$ which interacts with $\mathcal{Z}$ through $\tilde{P}$ for each honest player P in π, and there exists (an

efficient) simulator $\mathcal{S}$ such that no $\mathcal{Z}$ can distinguish the effect of an adversary $\mathcal{A}$ attacking π from that of $\mathcal{S}$ attacking $\mathcal{F}$, we say that π *UC-realises* $\mathcal{F}$.

Finally, the universal composition theorem asserts that if we replace a subprotocol π_s of a protocol π by the functionality $\mathcal{F}_s$ UC-realised by π_s, the security of π is not affected - it only becomes easier to analyse. In general, a protocol that contains calls to an ideal functionality $\mathcal{F}$ is called an $\mathcal{F}$-hybrid protocol.

Remark 1. As proofs of indistinguishability are usually lengthy and full of technicalities and no tools are currently available to make their verification easier, due to page restrictions we will only argue that a successful simulator can exist. There are three key points to such a simulator, which we label for future reference:

($\mathcal{S}_1$) It can simulate the cryptographic primitives from the information provided.
($\mathcal{S}_2$) It must be able to keep account of the different sessions so that it knows
  (i) what kind of information leakage to $\mathcal{A}$ to simulate and
  (ii) how messages from $\mathcal{A}$ would translate to ideal world influence by $\mathcal{A}$.
($\mathcal{S}_3$) It must allow $\mathcal{A}$ to have the same influence in $\mathcal{F}$ as the one it has in π.

Helpful as it may be, the UC-framework has some limitations when it comes to specific applications. In particular, the inability of participants in UC protocols to send more than one message per activation (as messages are immediately delivered, senders become inactive and recipients are thereby activated) would distort the natural flow of communication in multi-party protocols such as ours. For example, after a party outputs, $\mathcal{Z}$ has to activate the next outputting party by sending it an otherwise redundant message. The problem worsens when such protocols are used as subprotocols. To solve this, we choose to use the scheduling model described in a reformulation of Canetti's work [7], summarised below:

The active party can send several messages to other parties by enqueuing them in a global message queue so that they get delivered once all earlier messages have been delivered. The active party can either deactivate itself, thus initiating delivery of messages from the queue, or *hand over* a single message to another party to mean that the message is immediately delivered, just as in Canetti's work. Recipients of delivered messages become the protocol's active party. If all parties are inactive and the queue is empty (as in the beginning and end of a protocol execution), the environment $\mathcal{Z}$ is considered the active party.

Remark 2. A side effect of using a messages queue as in [7] is that some extra work is needed to prove indistinguishability, namely to show that proper scheduling order is preserved. To avoid littering the functionalities presented in Sects. 3 and 4 with scheduling order related messages, we assume that each time the functionality receives a message m through $\tilde{P}$, before processing it, it creates a scheduling identifier *scid* and hands over a dummy message (e.g. $(\mathsf{Scheduling}, scid)$) to $\mathcal{A}$. When it receives the same message as a reply the message is processed. Similarly when it is about to send a message to $\tilde{P}$.

2.2 Notation, Definitions and Corruption Model

In our protocols, U, N, S and T denote user, MNO, SP and TP players, and *pos* denotes a location in the network. To be able to simulate symmetric encryption we need to restrict to static corruption of protocol players. However, we allow network locations to be adaptively corrupted by the adversary. Note that a corrupted user or MNO implies that the corresponding position is corrupted.

By Sign_k, Ver_k, Enc_k, Dec_k and MAC_k we denote signing, signature verification, encryption, decryption and MAC algorithms keyed with k, by vk_P the signature verification key of player P and by H a hash function. For simplicity, we model the hash function as a random oracle instead of using randomness extraction. Moreover, $a|b$ denotes the concatenation of strings a and b.

We let $\mathbb{G}$ denote a multiplicative cyclic group of prime order p with generator g and let $\mathbb{Z}_p$ denote the ring of integers modulo p. A triple $(g^a, g^b, g^c) \in \mathbb{G}^3$ is called a *DDH-triple* if $c \equiv ab \mod p$. The Decisional Diffie–Hellman (DDH) assumption holds in $\mathbb{G}$ if, stated informally, given a triple in $\mathbb{G}^3$ it is computationally hard to decide whether it is a *DDH*-triple or not.

2.3 Radio Link, Secure Channel and Internet Access

Users and MNOs communicate over a radio link which we model by the ideal functionality $\mathcal{F}_{\mathsf{RL}}$ of Fig. 1. Users enter and leave network positions (corresponding to today's base stations) and once they have entered, they receive all messages sent to that position. In order to identify messages intended for them, they need to prepend every message they send to an MNO with a session identifier *sid* (like the IMSI in the present-day setting), with which the MNO will also prepend the reply.

An eavesdropper may choose to listen to the radio link at certain positions. In this case we make the adversary unrealistically powerful by giving him full control over the radio link (i.e. the ability to intercept and inject messages) at any position where he listens.

On (Listen, pos) from $\mathcal{A}$:
– Record *pos* as corrupted and hand over (Listening, pos) to $\mathcal{A}$.
On (Enter, pos) from $\tilde{U}$:
– Record U as present at *pos*.
On (Leave, pos) from $\tilde{U}$:
– Remove the record of U's presence at *pos*.
On (Send, sid, m, pos, N) from $\tilde{U}$:
– If U is present at *pos* then if *pos* is honest, send (Recv, sid, m, pos) to $\tilde{N}$, else hand over (Send, sid, m, pos, N) to $\mathcal{A}$.
On (Send, sid, m, pos) from $\tilde{N}$:
– If *pos* is corrupted hand over (Send,sid,m,N, pos) to $\mathcal{A}$, otherwise send (Recv, sid, m, pos) to $\tilde{U}$, for each honest user U present at *pos*.
On (Send,sid,m,pos,N) or (Send,sid,m,pos) from $\mathcal{A}$:
– If *pos* is corrupted send (Recv,sid,m,pos) to $\tilde{N}$ or to $\tilde{U}$ for each honest user U present at *pos*.

Fig. 1. The radio link functionality $\mathcal{F}_{\mathsf{RL}}$.

We also assume the existence of a secure communication channel between all MNOs and SPs, which we model by the ideal functionality $\mathcal{F}_{\mathsf{SEC}}$ of Fig. 2.

On $(\mathsf{Send}, \mathsf{sid}, \mathsf{m}, \mathsf{P_s}, \mathsf{P}_r)$ from $\mathcal{A}$: – If P_r is honest and P_s is corrupted then send $(\mathsf{Recv}, \mathsf{sid}, \mathsf{m}, \mathsf{P_s})$ to $\tilde{P}_r$.	**On** $(\mathsf{Send}, \mathsf{sid}, \mathsf{m}, \mathsf{P}_r)$ from $\tilde{P}_s$: – If P_r is honest then send $(\mathsf{Recv}, \mathsf{sid}, \mathsf{m}, \mathsf{P_s})$ to $\tilde{P}_r$, else hand over $(\mathsf{Send}, \mathsf{sid}, \mathsf{m}, \mathsf{P_s}, \mathsf{P_r})$ to $\mathcal{A}$.

Fig. 2. The secure channel functionality $\mathcal{F}_{\mathsf{SEC}}$. P_s and P_r denote MNO or SP players.

Finally, we model access to the internet for the MNOs, where the adversary is assumed to be in full control, by the ideal functionality $\mathcal{F}_{\mathsf{IA}}$ of Fig. 3.

On $(\mathsf{Send}, \mathsf{P_r}, \mathsf{m})$ from $\tilde{P}_s$: – Hand over $(\mathsf{Send}, \mathsf{P_s}, \mathsf{P_r}, \mathsf{m})$ to $\mathcal{A}$.	**On** $(\mathsf{Send}, \mathsf{P_s}, \mathsf{P_r}, \mathsf{m})$ from $\mathcal{A}$: – If P_r is honest then send $(\mathsf{Recv}, \mathsf{P_s}, \mathsf{m})$ to $\tilde{P}_r$.

Fig. 3. The internet access functionality $\mathcal{F}_{\mathsf{IA}}$. P_s and P_r denote non-user players.

3 Secure and Anonymous Network Connection

In this section we describe how a user can achieve secure and anonymous connection to an MNO by first anonymously establishing session keys with it and then using them to encrypt the radio link communication.

3.1 Anonymous Key Establishment

The kind of key establishment suitable for the privacy issues we are trying to address is authenticated with one-sided anonymity, in the sense that only one of the parties (the user) is anonymous with respect to the other (the MNO). To achieve such anonymous authentication, a third party (the SP) needs to vouch that the user is a subscriber without disclosing the user's identity and, as mentioned before, without learning the user's location.

In Fig. 4 we summarise one such $\mathcal{F}_{\mathsf{RL}}/\mathcal{F}_{\mathsf{SEC}}$–hybrid protocol, π_{AKE}. Its main point is that instead of having a unique IMSI and a subscriber key (both shared with the MNO) embedded on their smart cards, users have a subscriber key shared with their subscribed SP, and an identity token created by the SP. As a result, instead of identifying themselves directly to the MNO using the IMSI, they can do so to the SP using the token which is indecipherable to the MNO. The SP can then confirm to the MNO that users are subscribers, without disclosing their identity. Moreover, users and MNOs can establish authenticated keys for encrypting radio link communication and a temporary session identifier using essentially a Diffie–Hellman key exchange. The full details are provided in Fig. 5.

Adversarial goals: The aim of a corrupted user is to get authenticated without subscription. That of an MNO to learn the user's identity and that of an SP to learn the user's location. The only sensible collusion is an MNO–SP one.

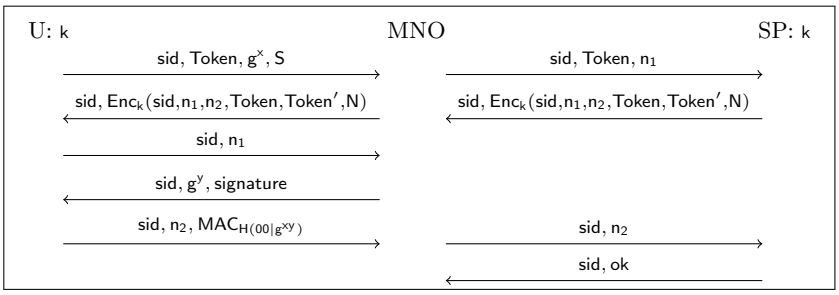

Fig. 4. Summary of the anonymous key establishment protocol π_{AKE}. Communication between users and MNOs is via $\mathcal{F}_{\mathsf{RL}}$ and between MNOs and SPs via $\mathcal{F}_{\mathsf{SEC}}$.

Authentication: First note that the user will never run two instances of the protocol in parallel. If the ciphertext is decipherable, it knows it came from its SP, and if it contains the identifier sid it is not a replayed message. If it contains the original token sent, it means no one is trying to link this session to a previous one. When it successfully verifies the MNO's signature, it is convinced about the MNO's identity and hence the origin of the Diffie–Hellman partial key.

When the MNO receives its nonce n_1 from the user, it concludes that the user is a subscriber of SP since it was able to decrypt the ciphertext. On receiving the MAC from the user, the MNO is convinced that the user accepted the signature and agrees on all previous messages. At this point there are two possibilities for the origin of the Diffie–Hellman partial key: either an honest subscriber of the SP or a corrupted SP eavesdropping the radio link at this position, masquerading as a user. The latter case, though possible, is improbable as it lacks motivation.

Finally, when the SP receives back its nonce n_2, it is convinced of the user's identity and acknowledges this with the final ok message to the MNO.

Anonymity and location privacy: The SP constructs tokens as independent encryptions of the user's identity and since no other part of the protocol contains information about the user's identity, users remain anonymous to anyone else.

The SP, unless listening from beforehand at the particular position and sees the token, learns nothing about the user's location.

Linkability issues: To avoid linkability, fresh keys should be established using distinct tokens every time a user enters a new location. For this reason, tokens are independent encryptions of a user's identity and fresh tokens are sent encrypted during a protocol run. However, in case the protocol fails before a user receives a new token, reuse of the old token can make linking the new session with previous ones possible both for the MNO and any eavesdroppers. Nevertheless, we prefer using tokens for the user's identification to the SP rather than public key cryptography which could prevent linkability. The reasons are that (i) in such case, linkability leads to tracing an anonymous user as opposed to tracing

User:

Stored: S, Token, k, g, $\{(\mathsf{N}, \mathsf{vk_N})\}$.
On $(\mathsf{Enter}, \mathsf{pos}, \mathsf{N})$ from $\mathcal{Z}$:
- If nothing is recorded then generate random exponent $\mathsf{x} \in \mathbb{Z}_p$ and session identifier sid, record $(\mathsf{pos},\mathsf{N},\mathsf{Token},\mathsf{sid},\mathsf{x})$ and send $(\mathsf{Enter},\mathsf{pos})$ and $(\mathsf{Send}, \mathsf{sid}, (\mathsf{Token}, \mathsf{g}^{\mathsf{x}}, \mathsf{S}), \mathsf{pos}, \mathsf{N})$ to $\mathcal{F}_{\mathsf{RL}}$.

On $(\mathsf{Leave}, \mathsf{pos})$ from $\mathcal{Z}$:
- Send $(\mathsf{Leave}, \mathsf{pos})$ to $\mathcal{F}_{\mathsf{RL}}$.
- If a record with pos exists remove it and output $(\mathsf{Est.\ failed\ unlinkable})$ if Token was updated or $(\mathsf{Est.\ failed\ linkable})$ otherwise.

On $(\mathsf{Recv}, \mathsf{sid}, \mathsf{m}, \mathsf{pos})$ from $\mathcal{F}_{\mathsf{RL}}$:
- If $\mathsf{Dec_k}(\mathsf{m})=(\mathsf{sid}, \mathsf{n}_1, \mathsf{n}_2, \mathsf{T}, \mathsf{T}_1, \mathsf{N})$ and $(\mathsf{pos}, \mathsf{N}, \mathsf{T}, \mathsf{sid}, \mathsf{x})$ is recorded, replace it by $(\mathsf{pos},\mathsf{N},\mathsf{T},\mathsf{sid},\mathsf{n}_1,\mathsf{n}_2,\mathsf{x},\mathsf{m})$, update $\mathsf{Token}\leftarrow\mathsf{T}_1$ and send $(\mathsf{Send}, \mathsf{sid}, (\mathsf{n}_1), \mathsf{pos}, \mathsf{N})$ to $\mathcal{F}_{\mathsf{RL}}$.
- Else if $\mathsf{m}=(\mathsf{h}, \sigma)$, $\mathsf{h} \in \mathbb{G}$, $(\mathsf{pos},\mathsf{N},\mathsf{T},\mathsf{sid},\mathsf{n}_1,\mathsf{n}_2,\mathsf{x},\mathsf{c})$ is recorded and $\mathsf{Ver}_{\mathsf{vk_N}}((\mathsf{sid},\mathsf{n}_1,\mathsf{g}^{\mathsf{x}},\mathsf{h},\mathsf{T},\mathsf{c},\mathsf{pos},\mathsf{S}), \sigma)=\mathsf{True}$ set $\mu\leftarrow\mathsf{MAC}_{\mathsf{H}(00|\mathsf{h}^{\mathsf{x}})}(\mathsf{sid},\mathsf{n}_1,\mathsf{n}_2,\mathsf{g}^{\mathsf{x}},\mathsf{h},\mathsf{T},\mathsf{c}, \sigma, \mathsf{pos}, \mathsf{S}, \mathsf{N})$, send $(\mathsf{Send}, \mathsf{sid}, (\mu, \mathsf{n}_2), \mathsf{pos}, \mathsf{N})$ to $\mathcal{F}_{\mathsf{RL}}$, output $(\mathsf{Est.},\mathsf{H}(01|\mathsf{h}^{\mathsf{x}}),\mathsf{H}(10|\mathsf{h}^{\mathsf{x}}),\mathsf{H}(11|\mathsf{h}^{\mathsf{x}}), \mathsf{pos}, \mathsf{N})$ and remove the record.
- Else do as second point of $(\mathsf{Leave}, \mathsf{pos})$ above.

MNO:

Stored: sk, g.
On $(\mathsf{Recv}, \mathsf{sid}, \mathsf{m}, \mathsf{pos})$ from $\mathcal{F}_{\mathsf{RL}}$:
- If $\mathsf{m}=(\mathsf{T}, \mathsf{h}, \mathsf{S})$ and sid is not on any record generate random nonce n_1, record $(\mathsf{sid},\mathsf{T},\mathsf{h},\mathsf{n}_1, \mathsf{pos},\mathsf{S})$ and send $(\mathsf{Send},\mathsf{sid},(\mathsf{T},\mathsf{n}_1,\mathsf{S})$ to $\mathcal{F}_{\mathsf{SEC}}$.
- Else if $\mathsf{m}=(\mathsf{n}_1)$ and $(\mathsf{sid}, \mathsf{T}, \mathsf{h}, \mathsf{n}_1, \mathsf{pos}, \mathsf{S}, \mathsf{c})$ is recorded, generate random exponent $\mathsf{y} \in \mathbb{Z}_p$ and $\sigma=\mathsf{Sign}_{\mathsf{sk}}(\mathsf{sid}, \mathsf{n}_1, \mathsf{h}, \mathsf{g}^{\mathsf{y}}, \mathsf{T}, \mathsf{c}, \mathsf{pos}, \mathsf{S})$, send $(\mathsf{Send}, \mathsf{sid}, (\mathsf{g}^{\mathsf{y}}, \sigma), \mathsf{pos})$ to $\mathcal{F}_{\mathsf{RL}}$ and replace the record by $(\mathsf{sid}, \mathsf{T}, \mathsf{h}, \mathsf{y}, \mathsf{n}_1, \mathsf{pos}, \mathsf{S}, \mathsf{c}, \sigma)$.
- Else if $\mathsf{m}=(\mu,\mathsf{n}_2)$, $(\mathsf{sid},\mathsf{T},\mathsf{h},\mathsf{y},\mathsf{n}_1,\mathsf{pos},\mathsf{S},\mathsf{c},\sigma)$ is recorded and $\mu=\mathsf{MAC}_{\mathsf{H}(00|\mathsf{h}^{\mathsf{y}})}(\mathsf{sid},\mathsf{n}_1,\mathsf{n}_2,\mathsf{h},\mathsf{g}^{\mathsf{y}}, \mathsf{T},\mathsf{c},\sigma,\mathsf{pos},\mathsf{S},\mathsf{N})$, send $(\mathsf{Send}, \mathsf{sid}, (\mathsf{n}_2), \mathsf{S})$ to $\mathcal{F}_{\mathsf{SEC}}$ and replace the record by $(\mathsf{sid}, \mathsf{h}, \mathsf{y}, \mathsf{pos}, \mathsf{S})$.

On $(\mathsf{Recv}, \mathsf{sid}, \mathsf{m}, \mathsf{S})$ from $\mathcal{F}_{\mathsf{SEC}}$:
- If $\mathsf{m}=(\mathsf{c})$ and $(\mathsf{sid},\mathsf{T},\mathsf{h},\mathsf{n}_1,\mathsf{pos},\mathsf{S})$ is recorded replace it by $(\mathsf{sid},\mathsf{T},\mathsf{h},\mathsf{n}_1,\mathsf{pos},\mathsf{S},\mathsf{c})$ and send $(\mathsf{Send},\mathsf{sid},\mathsf{m},\mathsf{pos})$ to $\mathcal{F}_{\mathsf{RL}}$.
- Else if $\mathsf{m}=(\mathsf{ok})$ and $(\mathsf{sid}, \mathsf{h}, \mathsf{y}, \mathsf{pos}, \mathsf{S})$ is recorded then remove it and output $(\mathsf{Est.}, \mathsf{H}(01|\mathsf{h}^{\mathsf{y}}), \mathsf{H}(10|\mathsf{h}^{\mathsf{y}}), \mathsf{H}(11|\mathsf{h}^{\mathsf{y}}), \mathsf{pos}, \mathsf{S})$.

SP:

Stored: $\mathsf{k_T}$, $\{(\mathsf{U}, \mathsf{k_U})\}$.
On $(\mathsf{Recv}, \mathsf{sid}, \mathsf{m}, \mathsf{N})$ from $\mathcal{F}_{\mathsf{SEC}}$:
- If $\mathsf{m}=(\mathsf{T}, \mathsf{n}_1)$, sid is not on any record and $\mathsf{Dec}_{\mathsf{k_T}}(\mathsf{T})=\mathsf{U}$, generate random nonce n_2 and token $\mathsf{T}_1=\mathsf{Enc}_{\mathsf{k_T}}(\mathsf{U})$, record $(\mathsf{sid}, \mathsf{n}_2, \mathsf{U}, \mathsf{N})$ and send $(\mathsf{Send},\mathsf{sid},(\mathsf{Enc}_{\mathsf{k_U}}(\mathsf{sid},\mathsf{n}_1,\mathsf{n}_2,\mathsf{T},\mathsf{T}_1,\mathsf{N})),\mathsf{N})$ to $\mathcal{F}_{\mathsf{SEC}}$.
- Else if $\mathsf{m}=(\mathsf{n}_2)$ and $(\mathsf{sid}, \mathsf{n}_2, \mathsf{U}, \mathsf{N})$ is recorded then remove it, send $(\mathsf{Send}, \mathsf{sid}, (\mathsf{ok}), \mathsf{N})$ to $\mathcal{F}_{\mathsf{SEC}}$ and output $(\mathsf{Est.}, \mathsf{U}, \mathsf{N})$.

Fig. 5. The anonymous key establishment protocol π_{AKE}.

identified users that is possible today (by forcing transmission of IMSIs instead of TMSIs[1]) and (ii) token verification uses symmetric encryption which allows for fast and inexpensive ciphertext validity checks by the SP, whereas if using public key methods this could lead to troublesome denial of service attacks. For example, since it is much easier to create a fake ciphertext than to validate one, a malicious user can anonymously send large amounts of fake ciphertexts to the SP, thus forcing it to exhaust its resources in trying to check their validity.

π_{AKE} UC-realises the ideal functionality $\mathcal{F}_{\mathsf{AKE}}$ we describe in Fig. 6 under the DDH assumption. This is the first functionality to model an anonymous key establishment between a user, subscribed to an SP, and an MNO, and is somewhat complicated, at least compared to the usual key establishment functionalities. The main reason is that we are attempting to realise it under realistic assumptions on available infrastructure and attacker power while minimising computational costs and communication overhead. This means that what can be achieved is less than perfect, and the functionality reflects that, in that describing the imperfections is quite complex.

[1] Such an attack is possible since an MNO can request IMSI transmission from the user before it authenticates itself [1].

On $(\mathsf{Enter}, \mathsf{pos}, \mathsf{N})$ from $\tilde{U}$:
– If no record $(\mathsf{id}, \mathsf{pos}, \mathsf{U}, \mathsf{N}, \mathsf{S})$ exists, where S is U's SP, then create one (if no record $(\mathsf{U}, \mathsf{id})$ exists, generate random identifier id) and hand over $(\mathsf{Enter}, \mathsf{leak}(\mathsf{id}, \mathsf{pos}, \mathsf{U}, \mathsf{N}, \mathsf{S}))$ to $\mathcal{A}$, where $\mathsf{leak}(\mathsf{id}, \mathsf{pos}, \mathsf{U}, \mathsf{N}, \mathsf{S})$ contains U and N if S is corrupted, pos, N and S if *pos* is corrupted, and id.
On $(\mathsf{Leave}, \mathsf{pos})$ from $\tilde{U}$:
– If $(\mathsf{id}, \mathsf{pos}, \mathsf{U}, \mathsf{N}, \mathsf{S})$ is recorded then hand over $(\mathsf{Leave}, \mathsf{id})$ to $\mathcal{A}$.
On $(\mathsf{Deny\ linkable}, \mathsf{id})$ or $(\mathsf{Deny\ unlinkable}, \mathsf{id})$ from $\mathcal{A}$:
– If $(\mathsf{id}, \mathsf{pos}, \mathsf{U}, \mathsf{N}, \mathsf{S})$ is recorded then remove it and send $(\mathsf{Est.\ failed\ linkable})$ or $(\mathsf{Est.\ failed\ unlinkable})$ to $\tilde{U}$. Additionally, if S is honest record $(\mathsf{U}, \mathsf{id})$ or remove any such record.
On $(\mathsf{Listen}, \mathsf{pos})$ from $\mathcal{A}$:
– Record *pos* as corrupted, collect $\mathsf{leak}(\mathsf{id}, \mathsf{pos}, \mathsf{U}, \mathsf{N}, \mathsf{S})$ from all records containing the appropriate input data, and hand over the collection to $\mathcal{A}$.
On $(\mathsf{Est.\ User}, \mathsf{k}_1, \mathsf{k}_2, \mathsf{k}_3, \mathsf{id})$ from $\mathcal{A}$:
– If $(\mathsf{id}, \mathsf{pos}, \mathsf{U}, \mathsf{N}, \mathsf{S})$ is recorded and N is honest then generate random keys $\mathsf{k}'_1, \mathsf{k}'_2$ and k'_3, replace the record by $(\mathsf{id}, \mathsf{pos}, \mathsf{U}, \mathsf{N}, \mathsf{S}, \mathsf{k}'_1, \mathsf{k}'_2, \mathsf{k}'_3)$ and send $(\mathsf{Est.}, \mathsf{k}'_1, \mathsf{k}'_2, \mathsf{k}'_3, \mathsf{pos}, \mathsf{N})$ to $\tilde{U}$.
– Else if $(\mathsf{id}, \mathsf{pos}, \mathsf{U}, \mathsf{N}, \mathsf{S})$ is recorded and N is corrupted, send $(\mathsf{Est.}, \mathsf{k}_1, \mathsf{k}_2, \mathsf{k}_3, \mathsf{pos}, \mathsf{N})$ to $\tilde{U}$ and either replace the record by $(\mathsf{id}, \mathsf{U}, \mathsf{N}, \mathsf{S})$ if S is honest or remove it otherwise.
On $(\mathsf{Est.\ SP}, \mathsf{id})$ from $\mathcal{A}$:
– If $(\mathsf{id}, \mathsf{U}, \mathsf{N}, \mathsf{S})$ or $(\mathsf{id}, \mathsf{pos}, \mathsf{U}, \mathsf{N}, \mathsf{S}, \mathsf{k}_1, \mathsf{k}_2, \mathsf{k}_3)$ is recorded then send $(\mathsf{Est.}, \mathsf{U}, \mathsf{N})$ to $\tilde{S}$ and either remove the record or replace it by $(\mathsf{id}, \mathsf{pos}, \mathsf{N}, \mathsf{S}, \mathsf{k}_1, \mathsf{k}_2, \mathsf{k}_3)$.
On $(\mathsf{Est.\ SP}, \mathsf{id}, \mathsf{U}, \mathsf{N}, \mathsf{S})$ from $\mathcal{A}$:
– If S is honest SP of corrupted U, send $(\mathsf{Est.}, \mathsf{U}, \mathsf{N})$ to $\tilde{S}$. In addition, if N is honest record $(\mathsf{id}, \mathsf{S}, \mathsf{N})$.
On $(\mathsf{Est.\ MNO}, \mathsf{k}_1, \mathsf{k}_2, \mathsf{k}_3, \mathsf{id})$ from $\mathcal{A}$:
– If $(\mathsf{id}, \mathsf{pos}, \mathsf{N}, \mathsf{S}, \mathsf{k}'_1, \mathsf{k}'_2, \mathsf{k}'_3)$ is recorded then remove it and send $(\mathsf{Est.}, \mathsf{k}'_1, \mathsf{k}'_2, \mathsf{k}'_3, \mathsf{pos}, \mathsf{S})$ to $\tilde{N}$.
On $(\mathsf{Est.\ MNO}, \mathsf{k}_1, \mathsf{k}_2, \mathsf{k}_3, \mathsf{id}, \mathsf{pos})$ or $(\mathsf{Est.\ MNO}, \mathsf{k}_1, \mathsf{k}_2, \mathsf{k}_3, \mathsf{pos}, \mathsf{S}, \mathsf{N})$ from $\mathcal{A}$:
– If $(\mathsf{id}, \mathsf{S}, \mathsf{N})$ is recorded or *pos* and S are corrupted then send $(\mathsf{Est.}, \mathsf{k}_1, \mathsf{k}_2, \mathsf{k}_3, \mathsf{pos}, \mathsf{S})$ to $\tilde{N}$ and in the first case remove the record.

Fig. 6. The anonymous key establishment functionality $\mathcal{F}_{\mathsf{AKE}}$.

Remark 3. One can always decide to modify π_{AKE} to use the public key cryptography alternative we mentioned. Doing so will simplify $\mathcal{F}_{\mathsf{AKE}}$ and will not affect any of our other functionalities or protocols, apart from removing the linkability possibilities inherited from the current version of $\mathcal{F}_{\mathsf{AKE}}$.

On the successful completion of a session, the user and the MNO will share three secrets and the SP will know that the user has established connection to the MNO, but not from where. A session is initiated when the user instructs the functionality that he wants to establish a key with an MNO at a certain position. The adversary $\mathcal{A}$ is notified about the attempt and controls when and how the key establishment completes. He can either allow it to (partially) complete, or he can interfere. The amount of information revealed about the attempt depends on which parties are corrupt, whether the adversary listens (or begins to listen) at the position, and how the user's previous key establishment attempt ended.

Simulation: As mentioned in Remark 1, we will simply argue that a successful simulator $\mathcal{S}_{\mathsf{AKE}}$ satisfying the three points $(\mathcal{S}_1)$ to $(\mathcal{S}_3)$ can exist.

It is fairly straight forward to satisfy $(\mathcal{S}_1)$: Using standard definitions for UC-secure signatures and symmetric encryption [3, 4], $\mathcal{S}_{\mathsf{AKE}}$ need not worry about forged signatures or maliciously created ciphertexts. Encrypted messages can be replaced by random ciphertexts of the same length and a secure MAC algorithm can be modeled as a pseudorandom function. By the DDH-assumption on $\mathbb{G}$,

each DDH-triple can be replaced by a random triple. Finally we disregard the negligible event that independent nonces coincide.

For ($\mathcal{S}_2$) there are many technicalities to work out: By going through the leakage of $\mathcal{F}_{\mathsf{RL}}$ and $\mathcal{F}_{\mathsf{AKE}}$ in the various corruption scenarios one can check that after the cryptographic changes have been made, the leakage from $\mathcal{F}_{\mathsf{AKE}}$ (that is the output of the leak function) only depends on the information that would also leak from $\mathcal{F}_{\mathsf{RL}}$. In other words, by replacing unknown values by fake ones and running the real world protocol internally, $\mathcal{S}_{\mathsf{AKE}}$ can simulate the leakage from $\mathcal{F}_{\mathsf{RL}}$ in the real world. When the adversary starts to listen to a new position, $\mathcal{F}_{\mathsf{AKE}}$ leaks a list with additional information needed to "upgrade" the sessions to the new corruption scenario. By linking the identifiers id to the current simulated sessions, $\mathcal{S}_{\mathsf{AKE}}$ can replace any fake value by the newfound information.

Finally, for ($\mathcal{S}_3$) we must verify that $\mathcal{F}_{\mathsf{AKE}}$ allows sufficient influence: Whenever one of the parties of the key-agreement (the user or MNO) is corrupted, $\mathcal{S}_{\mathsf{AKE}}$ is given the power to choose the keys. This is a standard simplification for key-agreements to ensure that any malicious attempt to influence the distribution of the keys in the real world can be reproduced by the simulator in the ideal world. If both participants are honest then the MAC and signature ensure that any successful output will have correctly distributed keys. Apart from the above, the only kind of influence $\mathcal{A}$ can make in the real world is to make sessions fail and to affect the order in which players output. In the ideal world, $\mathcal{S}_{\mathsf{AKE}}$ can make any session fail by sending $(\mathsf{Deny\ linkable}, id)$ or $(\mathsf{Deny\ unlinkable}, id)$ to $\mathcal{F}_{\mathsf{AKE}}$. If the user is honest, the session will remain traceable until the simulated user receives a new token from its SP. It is now just a case by case verification that every order of output that is possible in the real world is also possible in the ideal world, including the improbable case we mentioned of a corrupted SP fooling the MNO into outputting without any user participation.

3.2 Secure and Anonymous Radio Link

A user and an MNO that establish three secrets using $\mathcal{F}_{\mathsf{AKE}}$ can use two of them to encrypt their communication over the radio link and one as a temporary session identifier using the $\mathcal{F}_{\mathsf{AKE}}/\mathcal{F}_{\mathsf{RL}}$–hybrid protocol π_{SARL} of Fig. 7.

Informally, most of the security properties of π_{SARL} follow from those of $\mathcal{F}_{\mathsf{AKE}}$ and of symmetric encryption. If the secret keys are authenticated and uncorrelated, then so are the communication channels. The use of different keys for upstream and downstream communication is a standard tool. We assume that ciphertexts are authenticated so that the adversary cannot create valid ciphertexts, only replay observed ones.

π_{SARL} realises the the ideal functionality $\mathcal{F}_{\mathsf{SARL}}$ of Fig. 8, which captures the desired security properties of authenticated and encrypted channels. $\mathcal{F}_{\mathsf{SARL}}$ is an extension, in some sense, of $\mathcal{F}_{\mathsf{AKE}}$ and must therefore contain its imperfections.

Simulation: The existence of a successful simulator $\mathcal{S}_{\mathsf{SARL}}$ is more straight forward to argue for than $\mathcal{S}_{\mathsf{AKE}}$. Firstly, for ($\mathcal{S}_1$) we do the standard cryptographic

User:

On $(\mathsf{Enter}, \mathsf{pos}, \mathsf{N})$ from $\mathcal{Z}$:
– Send $(\mathsf{Enter}, \mathsf{pos}, \mathsf{N})$ to $\mathcal{F}_{\mathsf{AKE}}$.
On $(\mathsf{Leave}, \mathsf{pos})$ from $\mathcal{Z}$:
– If a record with *pos* exists then remove it and send $(\mathsf{Leave}, \mathsf{pos})$ to $\mathcal{F}_{\mathsf{RL}}$, else send it to $\mathcal{F}_{\mathsf{AKE}}$.
On m from $\mathcal{F}_{\mathsf{AKE}}$:
– If $\mathsf{m} = (\mathsf{Est.}, \mathsf{sid}, \mathsf{k_e}, \mathsf{k_d}, \mathsf{pos}, \mathsf{N})$ then record $(\mathsf{sid}, \mathsf{k_e}, \mathsf{k_d}, \mathsf{pos}, \mathsf{N})$ and send $(\mathsf{Enter}, \mathsf{pos})$ to $\mathcal{F}_{\mathsf{RL}}$, else output m.
On $(\mathsf{Send}, \mathsf{sid}, \mathsf{m})$ from $\mathcal{Z}$:
– If $(\mathsf{sid}, \mathsf{k_e}, \mathsf{k_d}, \mathsf{pos}, \mathsf{N})$ is recorded then send $(\mathsf{Send}, \mathsf{sid}, (\mathsf{Enc}_{\mathsf{k_e}}(\mathsf{m})), \mathsf{N})$ to $\mathcal{F}_{\mathsf{RL}}$.
On $(\mathsf{Recv}, \mathsf{sid}, \mathsf{c})$ from $\mathcal{F}_{\mathsf{RL}}$:
– If $(\mathsf{sid}, \mathsf{k_e}, \mathsf{k_d}, \mathsf{pos}, \mathsf{N})$ is recorded then output $(\mathsf{Received}, \mathsf{sid}, \mathsf{Dec}_{\mathsf{k_d}}(\mathsf{c}))$ and, if it is the first message received, also $(\mathsf{Est.}, \mathsf{sid}, \mathsf{pos}, \mathsf{N})$.

MNO:

On $(\mathsf{Est.}, \mathsf{sid}, \mathsf{k_d}, \mathsf{k_e}, \mathsf{pos}, \mathsf{S})$ from $\mathcal{F}_{\mathsf{AKE}}$:
– Record $(\mathsf{sid}, \mathsf{k_e}, \mathsf{k_d}, \mathsf{pos}, \mathsf{S})$, output $(\mathsf{Est.}, \mathsf{sid}, \mathsf{pos}, \mathsf{S})$ and send $(\mathsf{Send}, \mathsf{sid}, (\mathsf{Enc}_{\mathsf{k_e}}(\mathsf{ok}), \mathsf{pos})$ to $\mathcal{F}_{\mathsf{RL}}$.
On $(\mathsf{Send}, \mathsf{sid}, \mathsf{m})$ from $\mathcal{Z}$:
– If $(\mathsf{sid}, \mathsf{k_e}, \mathsf{k_d}, \mathsf{pos}, \mathsf{S})$ is recorded then send $(\mathsf{Send}, \mathsf{sid}, (\mathsf{Enc}_{\mathsf{k_e}}(\mathsf{m}), \mathsf{pos})$ to $\mathcal{F}_{\mathsf{RL}}$.
On $(\mathsf{Recv}, \mathsf{sid}, \mathsf{c}, \mathsf{pos})$ from $\mathcal{F}_{\mathsf{RL}}$:
– If $(\mathsf{sid}, \mathsf{k_e}, \mathsf{k_d}, \mathsf{pos}, \mathsf{S})$ is recorded then output $(\mathsf{Received}, \mathsf{sid}, \mathsf{Dec}_{\mathsf{k_d}}(\mathsf{c}))$.

Fig. 7. The secure and anonymous radio link protocol π_{SARL}.

On $(\mathsf{Enter}, \mathsf{pos}, \mathsf{N})$ from $\tilde{U}$ and $(\mathsf{Listen}, \mathsf{pos})$ and $(\mathsf{Deny\ linkable/unlinkable}, \mathsf{id})$ from $\mathcal{A}$:
– Do as $\mathcal{F}_{\mathsf{AKE}}$.
On $(\mathsf{Leave}, \mathsf{pos})$ from $\tilde{U}$:
– Replace U by $\perp$ in any record $(\mathsf{mid}, \mathsf{sid}, \mathsf{m}, \mathsf{U}, \mathsf{pos})$ and $(\mathsf{sid}, \mathsf{pos}, \mathsf{U}, \mathsf{X})$, $\mathsf{X} \in \{\mathcal{A}, \mathsf{N}\}$, and do as $\mathcal{F}_{\mathsf{AKE}}$.
On $(\mathsf{Est.\ MNO}, \mathsf{sid}', \mathsf{id})$ from $\mathcal{A}$:
– If $(\mathsf{id}, \mathsf{pos}, \mathsf{U}, \mathsf{N}, \mathsf{S})$ is recorded and N is honest then generate random secure session identifier sid, replace the record by $(\mathsf{id}, \mathsf{pos}, \mathsf{U}, \mathsf{N}, \mathsf{sid})$ and send $(\mathsf{Est.}, \mathsf{sid}, \mathsf{pos}, \mathsf{S})$ to $\tilde{N}$.
On $(\mathsf{Est.\ MNO}, \mathsf{sid}, \mathsf{pos}, \mathsf{S}, \mathsf{N}, \mathsf{U})$ $\underline{\text{or}}$ $(\mathsf{Est.\ MNO}, \mathsf{sid}, \mathsf{pos}, \mathsf{S}, \mathsf{N})$ from $\mathcal{A}$:
– If U is corrupted and S is U's SP $\underline{\text{or}}$ *pos* and S are corrupted then send $(\mathsf{Est.}, \mathsf{sid}, \mathsf{pos}, \mathsf{S})$ to $\tilde{N}$ and record $(\mathsf{sid}, \mathsf{pos}, \mathcal{A}, \mathsf{N})$.
On $(\mathsf{Est.\ User}, \mathsf{sid}', \mathsf{id})$ from $\mathcal{A}$:
– If $(\mathsf{id}, \mathsf{pos}, \mathsf{U}, \mathsf{N}, \mathsf{S})$ with corrupted N $\underline{\text{or}}$ $(\mathsf{id}, \mathsf{pos}, \mathsf{U}, \mathsf{N}, \mathsf{sid})$ is recorded then replace the record by $(\mathsf{sid}', \mathsf{pos}, \mathsf{U}, \mathcal{A})$ $\underline{\text{or}}$ $(\mathsf{sid}, \mathsf{pos}, \mathsf{U}, \mathsf{N})$ and send $(\mathsf{Est.}, \mathsf{sid}', \mathsf{pos}, \mathsf{N})$ $\underline{\text{or}}$ $(\mathsf{Est.}, \mathsf{sid}, \mathsf{pos}, \mathsf{N})$ to $\tilde{U}$.
On $(\mathsf{Send}, \mathsf{sid}, \mathsf{m})$ from $\tilde{U}$ $\underline{\text{or}}$ $\tilde{N}$:
– If $(\mathsf{sid}, \mathsf{pos}, \mathsf{U}, \mathsf{N})$ is recorded and *pos* is honest then send $(\mathsf{Received}, \mathsf{sid}, \mathsf{m})$ to $\tilde{N}$ $\underline{\text{or}}$ $\tilde{U}$.
– Else if $(\mathsf{sid}, \mathsf{pos}, \mathsf{X}, \mathsf{N})$ is recorded, *pos* is corrupted and $\mathsf{X} = \mathsf{U}$ $\underline{\text{or}}$ $\mathsf{X} \in \{\mathsf{U}, \perp\}$, generate random message identifier mid, record $(\mathsf{mid}, \mathsf{sid}, \mathsf{m}, \mathsf{N}, \mathsf{pos})$ $\underline{\text{or}}$ $(\mathsf{mid}, \mathsf{sid}, \mathsf{m}, \mathsf{X}, \mathsf{pos})$ and hand over $(\mathsf{mid}, \mathsf{sid}, |\mathsf{m}|, \mathsf{pos}, \mathsf{N})$ $\underline{\text{or}}$ $(\mathsf{mid}, \mathsf{sid}, |\mathsf{m}|, \mathsf{N}, \mathsf{pos})$ to $\mathcal{A}$, where $|\mathsf{m}|$ is the length of m.
– Else if $(\mathsf{sid}, \mathsf{pos}, \mathsf{U}, \mathcal{A})$ $\underline{\text{or}}$ $(\mathsf{sid}, \mathsf{pos}, \mathcal{A}, \mathsf{N})$ is recorded, hand over $(\mathsf{sid}, \mathsf{m}, \mathsf{pos}, \mathsf{N})$ $\underline{\text{or}}$ $(\mathsf{sid}, \mathsf{m}, \mathsf{N}, \mathsf{pos})$ to $\mathcal{A}$.
On $(\mathsf{Deliver}, \mathsf{mid})$ from $\mathcal{A}$:
– If $(\mathsf{mid}, \mathsf{sid}, \mathsf{m}, \mathsf{X}, \mathsf{pos})$ is recorded and $\mathsf{X} \in \{\mathsf{U}, \mathsf{N}\}$ then send $(\mathsf{Received}, \mathsf{sid}, \mathsf{m})$ to $\tilde{X}$.
On $(\mathsf{Send}, \mathsf{sid}, \mathsf{m}, \mathsf{pos})$ $\underline{\text{or}}$ $(\mathsf{Send}, \mathsf{sid}, \mathsf{m}, \mathsf{pos}, \mathsf{N})$ from $\mathcal{A}$:
– If $(\mathsf{sid}, \mathsf{pos}, \mathsf{U}, \mathcal{A})$ $\underline{\text{or}}$ $(\mathsf{sid}, \mathsf{pos}, \mathcal{A}, \mathsf{N})$ is recorded then send $(\mathsf{Received}, \mathsf{sid}, \mathsf{m})$ to $\tilde{U}$ $\underline{\text{or}}$ to $\tilde{N}$.

Fig. 8. The secure radio link functionality $\mathcal{F}_{\mathsf{SARL}}$.

changes to the π_{SARL} protocol. UC-secure ciphertexts are replaced by random encryptions and ciphertexts created and injected by $\mathcal{A}$ are considered invalid.

Once this is done, one can check that the leakage from $\mathcal{F}_{\mathsf{RL}}$ and $\mathcal{F}_{\mathsf{AKE}}$ in the protocol only depends on values that would also leak from $\mathcal{F}_{\mathsf{SARL}}$. Like $\mathcal{S}_{\mathsf{AKE}}$, $\mathcal{S}_{\mathsf{SARL}}$ can replace unknown values with fake ones and run the π_{SARL} protocol internally. This takes care of $(\mathcal{S}_2)$.

Lastly, we consider $(\mathcal{S}_3)$. The adversary can influence the establishment of secure sessions in π_{SARL} by influencing $\mathcal{F}_{\mathsf{AKE}}$. $\mathcal{S}_{\mathsf{SARL}}$ is therefore given the same power to influence $\mathcal{F}_{\mathsf{SARL}}$. Apart from this, the only way the adversary can influence the output of the user or the MNO is by delaying, deleting or replaying messages from honest players or sending messages from corrupted players (all over $\mathcal{F}_{\mathsf{RL}}$), something $\mathcal{S}_{\mathsf{SARL}}$ can also do with $\mathcal{F}_{\mathsf{SARL}}$.

4 Anonymous Internet Access

A user having a secure and anonymous connection to an MNO can anonymously access the internet (modelled by $\mathcal{F}_{\mathsf{IA}}$ of Fig. 3) using ephemeral pseudonyms securely obtained from the MNO via the $\mathcal{F}_{\mathsf{SARL}}/\mathcal{F}_{\mathsf{IA}}$-hybrid protocol π_{AIA} (Fig. 9).

User:

On $(\mathsf{Enter}, \mathsf{pos}, \mathsf{N})$ from $\mathcal{Z}$:
– Send $(\mathsf{Enter}, \mathsf{pos}, \mathsf{N})$ to $\mathcal{F}_{\mathsf{SARL}}$.
On m=(Est. failed linkable/unlinkable) from $\mathcal{F}_{\mathsf{SARL}}$:
– Output m.
On $(\mathsf{Est.}, \mathsf{sid}, \mathsf{pos}, \mathsf{N})$ from $\mathcal{F}_{\mathsf{SARL}}$:
– Record $(\mathsf{sid}, \mathsf{pos})$ and output $(\mathsf{Est.}, \mathsf{sid}, \mathsf{pos}, \mathsf{N})$.
On $(\mathsf{Leave}, \mathsf{pos})$ from $\mathcal{Z}$:
– Remove any record containing pos and send $(\mathsf{Leave}, \mathsf{pos})$ to $\mathcal{F}_{\mathsf{SARL}}$.
On $(\mathsf{Req.\ Pseud.}, \mathsf{l}, \mathsf{pos})$ from $\mathcal{Z}$:
– If $(\mathsf{sid}, \mathsf{pos})$ is recorded then generate random nonce n, record $(\mathsf{sid}, \mathsf{pos}, \mathsf{n}, \mathsf{l})$ and send $(\mathsf{Send}, \mathsf{sid}, (\mathsf{Req.\ Pseud.}, \mathsf{n}, \mathsf{l}))$ to $\mathcal{F}_{\mathsf{SARL}}$.
On $(\mathsf{Received}, \mathsf{sid}, \mathsf{m})$ from $\mathcal{F}_{\mathsf{SARL}}$:
– If m=$(\mathsf{Pseud.}, \mathsf{n}, \mathsf{ps}_1, \ldots, \mathsf{ps}_\mathsf{l})$ and $(\mathsf{sid}, \mathsf{pos}, \mathsf{n}, \mathsf{l})$ is recorded then remove the record, record each $(\mathsf{ps}_\mathsf{i}, \mathsf{sid}, \mathsf{pos})$ and output $(\mathsf{Pseud.}, \mathsf{pos}, \mathsf{ps}_1, \ldots, \mathsf{ps}_\mathsf{l})$.
– Else if m=$(\mathsf{Recv}, \mathsf{P_s}, \mathsf{ps}, \mathsf{m}')$ and $(\mathsf{ps}, \mathsf{sid}, \mathsf{pos})$ is recorded then output $(\mathsf{Received}, \mathsf{P_s}, \mathsf{ps}, \mathsf{m}')$.
On $(\mathsf{Send}, \mathsf{ps}, \mathsf{P_r}, \mathsf{m})$ from $\mathcal{Z}$:
– If $(\mathsf{ps}, \mathsf{sid}, \mathsf{pos})$ is recorded then send $(\mathsf{Send}, \mathsf{sid}, (\mathsf{Send}, \mathsf{P_r}, (\mathsf{ps}, \mathsf{m})))$ to $\mathcal{F}_{\mathsf{SARL}}$.

MNO:

On $(\mathsf{Est.}, \mathsf{sid}, \mathsf{pos}, \mathsf{S})$ from $\mathcal{F}_{\mathsf{SARL}}$:
– Output $(\mathsf{Est.}, \mathsf{sid}, \mathsf{pos}, \mathsf{S})$.
On $(\mathsf{Received}, \mathsf{sid}, \mathsf{m})$ from $\mathcal{F}_{\mathsf{SARL}}$:
– If m=$(\mathsf{Req.\ Pseud.}, \mathsf{n}, \mathsf{l})$, generate l random pseudonyms $\mathsf{ps}_1, \ldots, \mathsf{ps}_\mathsf{l}$, record each $(\mathsf{ps}_\mathsf{i}, \mathsf{sid})$ and send $(\mathsf{Send}, \mathsf{sid}, (\mathsf{Pseud.}, \mathsf{n}, \mathsf{ps}_1, \ldots, \mathsf{ps}_\mathsf{l}))$ to $\mathcal{F}_{\mathsf{SARL}}$.
– Else if m=$(\mathsf{Send}, \mathsf{P_r}, (\mathsf{ps}, \mathsf{m}'))$ and $(\mathsf{ps}, \mathsf{sid})$ is recorded then send m to $\mathcal{F}_{\mathsf{IA}}$.
On $(\mathsf{Recv}, \mathsf{P_s}, (\mathsf{ps}, \mathsf{m}))$ from $\mathcal{F}_{\mathsf{IA}}$:
– If $(\mathsf{ps}, \mathsf{sid})$ is recorded then send $(\mathsf{Send}, \mathsf{sid}, (\mathsf{Recv}, \mathsf{P_s}, \mathsf{ps}, \mathsf{m}'))$ to $\mathcal{F}_{\mathsf{SARL}}$.

Non-user player P:

On $(\mathsf{Send}, \mathsf{P_r}, \mathsf{m})$ from $\mathcal{Z}$:
– Send $(\mathsf{Send}, \mathsf{P_r}, \mathsf{m})$ to $\mathcal{F}_{\mathsf{IA}}$.
On $(\mathsf{Recv}, \mathsf{P_s}, \mathsf{m})$ from $\mathcal{F}_{\mathsf{IA}}$:
– Output $(\mathsf{Received}, \mathsf{P_s}, \mathsf{m})$.

Fig. 9. The anonymous internet access protocol π_{AIA}. P_s and P_r denote non-user player.

The ideal functionality $\mathcal{F}_{\mathsf{AIA}}$ we describe in Fig. 10 is UC-realised by π_{AIA} and captures the security properties we want from such a scheme. This functionality is an extension of $\mathcal{F}_{\mathsf{SARL}}$ and incorporates the internet access functionality $\mathcal{F}_{\mathsf{IA}}$.

On (Enter, pos, N) from $\tilde{U}$, (Send, sid, m) from $\tilde{U}$ and $\tilde{N}$, and (Deny linkable/unlinkable, id), (Deliver,mid), (Est. User, sid, id) and (Est. MNO, sid, X) from $\mathcal{A}$:
– Do as $\mathcal{F}_{\mathsf{SARL}}$.
On (Leave, pos) from $\tilde{U}$:
– Remove any record (ps, sid, pos, U, N), and do as $\mathcal{F}_{\mathsf{SARL}}$.
On (Req. Pseud., l, pos) from $\tilde{U}$:
– If (sid, pos, U, N) is recorded, generate random pseudonym request identifier prid, record (prid, l, sid, pos, U, N) and hand over (Req. Pseud., leak(prid, l, sid, pos, U, N)) to $\mathcal{A}$, where leak(prid, l, sid, pos, U, N) contains l, sid, pos and N if *pos* is corrupted, and prid.
On (Pseud., prid, $\mathsf{ps}'_1, \ldots, \mathsf{ps}'_{l'}$) from $\mathcal{A}$:
– If (prid, l, sid, pos, U, N) is recorded and either N is honest or $\mathsf{l}' = \mathsf{l}$, generate l random N-pseudonyms $\mathsf{ps}_1, \ldots, \mathsf{ps}_\mathsf{l}$ or set $\mathsf{ps}_\mathsf{i} \leftarrow \mathsf{ps}'_\mathsf{i}$ for $1 \leq \mathsf{i} \leq l$. Then record each (ps_i, sid, pos, U, N), remove the record (prid, l, sid, pos, U, N) and send (Pseud., pos, $\mathsf{ps}_1, \ldots, \mathsf{ps}_\mathsf{l}$) to $\tilde{U}$.
On (Send, ps, $\mathsf{P_r}$, m) from $\tilde{U}$:
– If (ps, sid, pos, U, N) is recorded then hand over (Send, $\mathsf{P_r}$, (ps, m), leak(ps, sid, pos, U, N)) to $\mathcal{A}$, where leak(ps, sid, pos, U, N) contains sid and pos if *pos* is corrupted, and N.
On (Send, $\mathsf{P_r}$, m) from $\tilde{P}_s$:
– If P_s is a non-user player then hand over (Send, $\mathsf{P_s}$, $\mathsf{P_r}$, m) to $\mathcal{A}$.
On (Send, $\mathsf{P_s}$, $\mathsf{P_r}$, m) from $\mathcal{A}$:
– If P_r is a non-user player then send (Received, $\mathsf{P_s}$, m) to $\tilde{P}_r$. Otherwise, if ($\mathsf{P_r}$, sid, pos, U, N) is recorded then send (Received, $\mathsf{P_s}$, $\mathsf{P_r}$, m) to $\tilde{U}$.
On (Listen, pos) from $\mathcal{A}$:
– Collect leak(prid, l, sid, pos, N, U) from all records and do as $\mathcal{F}_{\mathsf{SARL}}$.

Fig. 10. The anonymous internet access functionality $\mathcal{F}_{\mathsf{AIA}}$.

Simulation: First notice that the part about establishing a secure session in $\mathcal{F}_{\mathsf{AIA}}$ is essentially identical to that of $\mathcal{F}_{\mathsf{SARL}}$. We therefore focus on the sending of messages and handling of pseudonyms. The simulator can distinguish pseudonym request messages from sent messages using the leakage from $\mathcal{F}_{\mathsf{AIA}}$.

As usual we can disregard the negligible event that independent nonces coincide. This means that the adversary can never replay pseudonyms from an honest MNO to an honest user. Thus the only influence the adversary can have in pseudonym requests is to delay or delete them, which the simulator can also do via $\mathcal{F}_{\mathsf{AIA}}$. For messages sent over the internet, the simulator has full control and can inject or delete messages freely. Also, all relevant data leaks through $\mathcal{F}_{\mathsf{AIA}}$, except from the user's identity and the sender's position, unless corrupted. The view of the adversary in the π_{AIA} protocol is also independent of these values. Finally, when a new position is corrupted, the simulator learns additional information and it needs to update the leakage about pending pseudonym requests and sent messages.

5 Seamless Internet Telephony Services

Another of today's mobile communication related privacy issues is that when a mobile phone user wants make a call, it has to inform its MNO of the callee's identity in order for a connection to be made. In addition, the contents of a telephone conversation are available to the MNO of both the caller and the callee.

In this section we sketch how seamless, private and anonymous telephony services can be achieved from the $\mathcal{F}_{\mathsf{AIA}}$ functionality by means of public key cryptography and a new protocol player called the *telephony provider* (TP). The idea is that every time a user U enters a new position it registers a pseudonym with its subscribed TP through the registration protocol of Fig. 11. Different users can now call U by requesting U's pseudonym via TP. Once U accepts a call, further communication with the caller will be independent of the TP by means of direct exchange of pseudonyms and symmetric encryption keys. By renewing pseudonyms on the go, U can stay connected as it moves around the network or attempts to protect itself from traffic analysis.

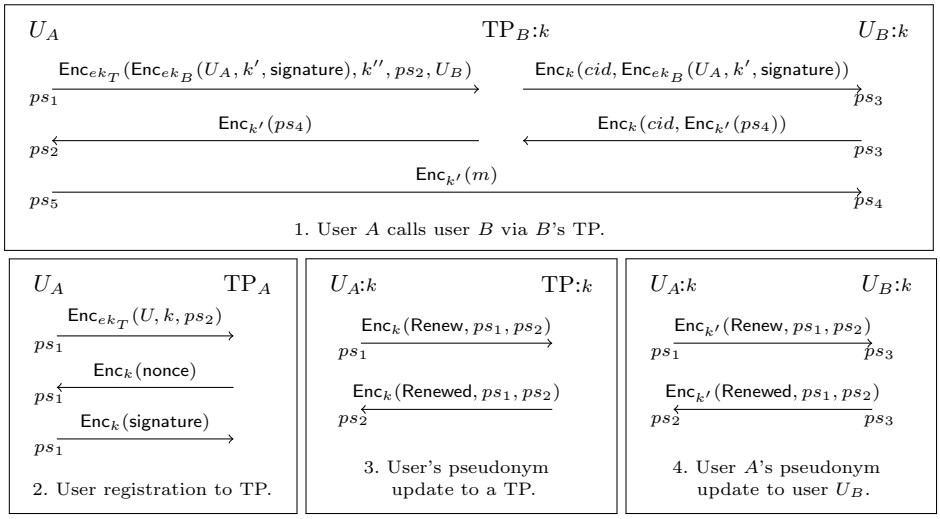

Fig. 11. The calling, pseudonym registration and pseudonym update protocols. All communication is through $\mathcal{F}_{\mathsf{AIA}}$.

In the first communication between a user and a TP, public key encryption is used which would allow for denial of service attack. Our protocol could be extended by letting the TPs share symmetric keys with MNOs so that the MNOs can authenticate registered pseudonyms. In that case denial of service attacks can be traced back to the owner of the pseudonym launching the attack.

Authentication: A caller authenticates to the callee via the signature. The callee is authenticated by using the key sent encrypted by the caller, as he is the sole owner of his decryption key. Similarly, a user authenticates to a TP during registration using a signature that includes the fresh nonce sent by the TP. The TP is authenticated by using the key sent by the user, as he is the sole owner of

his decryption key. Use of the shared symmetric key authenticates any further communication between any two parties.

6 Conclusions

In this paper we have proposed a sequence of universally composable protocols and ideal functionalities for mobile devices that in turn achieve secure and anonymous network connection and anonymous internet access. We have also sketched how the anonymous internet access functionality can be used as a basis for internet telephony services. This is a privacy-preserving alternative to the current state of affairs where users have to inform mobile network operators both of their location and identity as they move around the network, and where network operators have full overview of a user's telephone conversations.

References

1. 3GPP TS 33.102: Security Architecture, Ver. 11.1.0 (2011). Available at http://www.3gpp.org/ftp/Specs/html-info/33102.htm
2. Camenisch, J., Lysyanskaya, A.: An Efficient System for Non-transferable Anonymous Credentials With Optional Anonymity Revocation. In: Pfitzmann, B. (ed.) Eurocrypt 2001. LNCS, vol. 2045, pp. 93–118, Springer (2001)
3. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols, Report 2000/067, IACR ePrint Archive (2005)
4. Canetti, R.: Universally Composable Signature, Certification, and Authentication. In: CSFW 2004, pp. 219–233. IEEE press, (2004)
5. Chaum, D: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Commun. ACM, Vol. 24, Iss. 2, pp. 84–88. ACM press (1981)
6. The European Parliament and Council: Directive 2006/24/EC. L 105, pp. 54–63 (2006)
7. Gjøsteen, K., Petrides, G., Steine, A.: A Novel Framework for Protocol Analysis. In: Boyen, X., Chen, X. (eds.) ProvSec 2011. LNCS, vol. 6890, pp. 340–347, Springer (2011)
8. Lindell, Y.: Anonymous Authentication. Journal of Privacy and Confidentiality, Vol. 2, Iss. 2, pp. 35–63 (2010)
9. Nguyen, L., Safavi-Naini, R.: Dynamic k-times Anonymous Authentication. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 318–333, Springer (2005)
10. The Tor project. Available at https://www.torproject.org/index.html.en
11. Wachsmann, C., Chen, L., Dietrich, K., Löhr, H., Sadeghi, A.-R., Winter, J.: Lightweight Anonymous Authentication with TLS and DAA for Embedded Mobile Devices. In: Burmester, M., Tsudik, J., Magliveras, S., Ilić, I. (eds.) ISC 2010. LNCS, vol. 6531, pp. 84–98, Springer (2011)
12. Xu, J., Zhu, W.-T., Feng, D.-G.: An Efficient Mutual Authentication and Key Agreement Protocol Preserving User Anonymity in Mobile Networks. Computer Communications, Vol. 34, pp. 319–325, Elsevier (2011)

Paper V

Weak Blind Signatures and Mobile Payment
Asgeir Steine
Preprint

Weak Blind Signatures and Mobile Payment *

Asgeir Steine

NTNU, Trondheim, Norway.

Abstract. In this paper we introduce a protocol for an online mobile payment service using blind signatures. Security is defined and analysed in a variant of the universal composability framework under standard cryptographic assumptions. We assume fairly technical communication channels that help us to address the privacy issues related to such a service. Previous definitions of universally composable blind signature functionalities have always been dependent on the channels in use. As a result the functionalities would have to be redefined every time they are used together with a new channel. As a side contribution we define our functionality independently of the channels. This definition is more in line with the way functionalities for standard crypto schemes such as encryption and signature schemes are modelled.

1 Introduction

During the last twenty years or so electronic payment services have become more and more common. In many countries the traditional cash system with bills and coins is now obsolete. In the credit/debit cards systems of today's electronic payment services each merchant has a persistent connection with the bank. Another emerging technology for electronic payments is based on near field communications, allowing a payment device to communicate with the merchants terminal via short range radio and with the bank via the internet or mobile network. One example of this type is the Google Wallet system.

One problem with traditional credit card systems is that the bank learns to much about its users. The bank has full overview of where and when a transaction takes place, and the size of the transaction. One can argue that this practice is not desirable from a privacy perspective.

We build an alternative near field communications based payment scheme using the privacy preserving mobile networks of [9].

1.1 Related Work

To our knowledge the first cryptographic paper to address the issue of privacy for electronic payment was [3] from 1982. The notion of blind signatures was also introduced in that paper as a tool for achieving secure payments with enhanced user privacy. Following in its path many papers introduced examples of blind

* Funded by the Norwegian Research Council's VERDIKT programme project 183195.

signature schemes as well as improvements and adjustments to the approach of [3]. One notable adaption allowing users and merchants to conduct transactions without a persistent connection to the bank can be found in [4]. The merchant can then later contact the bank to cash in on its successful transactions. Another well known payment system can be found in [1].

The universal composability framework for analysing protocols was introduced in [2]. The paper [5] and one of the papers in [8] are devoted to modelling blind signatures within that framework. Security games for analysing blind signatures was originally defined in [7].

1.2 Our Contribution

We build a payment protocol from blind signatures in the standard way by having the bank blindly sign challenges from the merchants. The main novelty is the communication devices we use in our model. By modelling communication functionalities for mobile phones as in [9] our mobile payment functionality gives a detailed overview of security properties that can be achieved in different corruption scenarios. The protocol is analysed in the modified universal composability framework of [6].

As a side contribution we define and use a weak notion of blind signature schemes and prove its security properties in the same universal composability framework. The notion of blind signatures in this paper differs from the signatures of [5] and [8] as it is made independent of communication networks used between the signer and the player that blinds/unblinds the signature.

1.3 Assumptions

The universal composability framework comes in many flavors depending on the setup assumptions for the system and corruption capabilities of the adversary. We only consider static corruptions. That means that the adversary gets full control of the corrupted players, however he can only corrupt players before the actual run of the protocol starts. In the mobile payment protocol where we model physical networks by $\mathcal{F}_{\mathsf{AIA}}$ and $\mathcal{F}_{\mathsf{NFC}}$ we allow the adversary to corrupt positions in the networks adaptively during execution, as specified by the functionalities.

We assume honestly generated preshared public keys for our blind signature protocol. In practise it means that every protocol player has access to the key generation functionality of Fig. 1. In the mobile payment protocol of Fig. 11 we give the players access to a standard key registration functionality for public key encryption and traditional signatures. In that protocol the keys for the blind signature scheme have been abstracted away to the functionality $\mathcal{F}_{\mathsf{BS}}$.

Note that when the communication functionality $\mathcal{F}_{\mathsf{AIA}}$ was realised in [9] preshared symmetric keys between the each user and its service provider, and signature key pairs for each network provider was also assumed.

On (Key, Q) from a Player P:
- If (Key pair, pk, sk, Q) is not stored, generate $(pk, sk) \leftarrow$ Gen and store (Key pair, pk, sk, Q).
- In either case hand over (Pubic key, pk, Q) to P.

On (Keys) from a player P:
- If (Key pair, pk, sk, P) is not stored, generate $(pk, sk) \leftarrow$ Gen and store (Key pair, pk, sk, P).
- In either case and hand over (Key pair, pk, sk) to P.

On (Keys, Q) from $\mathcal{A}$:
- If (Key pair, pk, sk, Q) is not stored, generate $(pk, sk) \leftarrow$ Gen and store (Key pair, pk, sk, Q).
- In either case hand over (Public key, pk, Q) to $\mathcal{A}$ if Q is honest and (Key pair, pk, sk, Q) if he is corrupted.

Fig. 1. Key generation functionality $\mathcal{F}_{\mathsf{KG}}$ for a public-key crypto system with key generation algorithm Gen.

2 Weak Blind Signatures

A blind signature scheme consists of the following.
- A probabilistic key generation algorithm Gen that produces key pairs (pk, sk).
- Three probabilistic algorithms Request, Issue and Unblind such that: 1) Request on input (pk, m) outputs (ρ, s) where ρ is a request token and s is some state information. 2) Issue on input (sk, ρ) outputs a blinded signature $\tilde{\sigma}$ or $\perp$. 3) Unblind on input $(s, \tilde{\sigma})$ outputs a signature σ or $\perp$.
- An algorithm Verify that on some input (pk, m, σ) outputs either 0 or 1.

It is required that for any message m and any key pair $(pk, sk) \leftarrow$ Gen the signature σ obtained by the steps $(\rho, s) \leftarrow \mathsf{Request}(pk, m)$, $\tilde{\sigma} \leftarrow \mathsf{Issue}(sk, \rho)$, $\sigma \leftarrow \mathsf{Unblind}(s, \tilde{\sigma})$, satisfies $\mathsf{Verify}(pk, m, \sigma) = 1$ (completeness).

2.1 Game Based Blind Signatures

Game based security definitions for blind signatures are usually made via a blindness game and an unforgeability game. Each game describes how input to an attacker is generated and how the outcome of the game is decided from the attackers output. The security games we propose are quite close to the games described in [7]. We will consider a weak kind of blindness which as described in Def. 1 essentially says that an adversary given an honestly generated signature/verification key-pair and acting as the signer of two sessions cannot recognise whether or not the request tokens have been swapped between the sessions. Note that it could also be sensible to consider a stronger notion where the adversary is allowed to generate the keys maliciously and keep the private key secret.

The unforgeability requirement is described in Def. 2. Again for our purposes it is enough to consider a weak kind of unforgeability. The definition essentially

says that the adversary given an honestly generated verification key and access to an issuing oracle, cannot obtain $n+1$ valid signatures on distinct messages while querying the oracle only n times. This definition of unforgeability is weak in the sense that the adversary might obtain many signatures on the same message, and that we do not require that messages can be extracted from their request tokens as some approaches towards UC-secure blind signatures do.

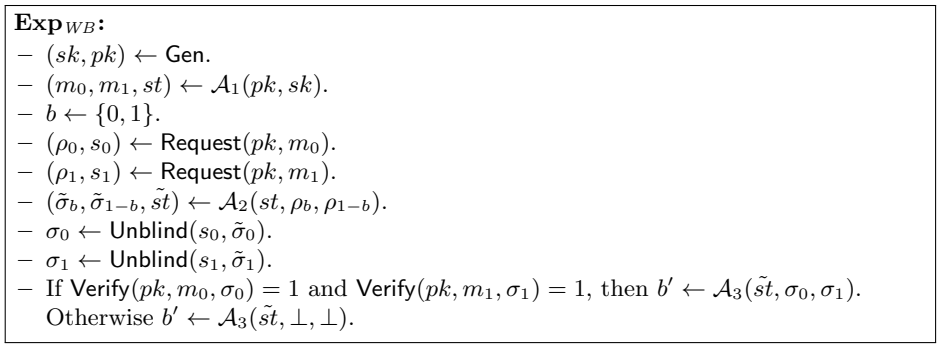

Fig. 2. The weak blindness experiment.

Definition 1 (Weak Blindness). *We define E_i to be the event that $b \oplus b' = i$ in the experiment $\boldsymbol{Exp}_{WB}$ of Fig. 2. The advantage of an attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ is defined to be*

$$\boldsymbol{Adv}_{\mathcal{A}}^{WB} = |Pr[E_0] - Pr[E_1]|.$$

If for any adversary $\mathcal{A}$ bounded by time t we have $\boldsymbol{Adv}_{\mathcal{A}}^{WB} \leq \epsilon$ we say that the blind signature scheme is (t, ϵ)-secure with respect to weak blindness.

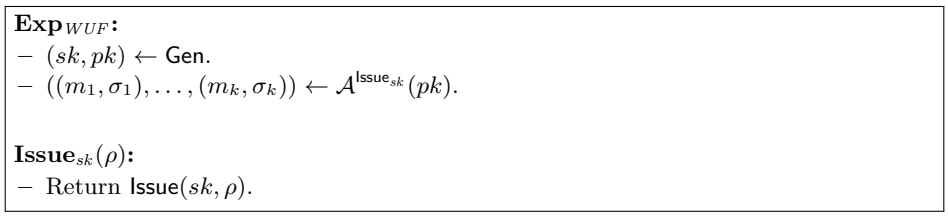

Fig. 3. The weak unforgeability experiment

Definition 2 (Weak unforgeability). *For an attacker $\mathcal{A}$ of the experiment $\boldsymbol{Exp}_{WUF}$ of Fig. 3 the event E is that for some $k \in \mathbb{Z}^+$ the attacker outputs*

valid signatures on $k+1$ distinct messages after at most k queries to Issue_{sk}. *The success probability of the attacker is defined as*

$$\boldsymbol{Succ}_{\mathcal{A}}^{WUF} = Pr[E].$$

If for any adversary $\mathcal{A}$ bounded by time t and n queries we have $\boldsymbol{Succ}_{\mathcal{A}}^{WUF} \leq \epsilon$, we say that the blind signature scheme is (t, n, ϵ)-secure with respect to weak unforgeability.

2.2 Request Integrity

To realise universally composable blind signatures independent of channel we need an extra security condition. The request integrity requirement described in Def. 3 ensures that it is infeasible for an attacker to alter an observed request token to obtain a blinded signature that unblinds with the same state information as the original request token without ever making an issue request for that particular request token. This requirement is non-standard in the theory of blind signatures, however as stated in Thm. 1 any blind signature scheme can be made into a blind signature scheme that satisfies request integrity requirements by using an IND-CCA secure public key encryption scheme to encrypt all request tokens.

Definition 3. *Define the success probability $\boldsymbol{Succ}_{\mathcal{A}}^{RI}$ of an attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ to be the probability that $b = 1$ after the interaction in the experiment $\boldsymbol{Exp}_{RI}$ of Fig.4.*

We say that the blind signature scheme is (t, n, ϵ)-secure with respect to request integrity if for any attacker bounded by time t and n queries to Issue_{sk} *we have $\boldsymbol{Succ}_{\mathcal{A}}^{RI} \leq \epsilon$.*

Definition 4. *Let E_i be the event that $b \oplus b' = i$ and c is not stored in* Dec_{dk} *in the experiment $\boldsymbol{Exp}_{CCA}$ of Fig. 5. We define the advantage of an attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against the experiment to be*

$$\boldsymbol{Adv}_{\mathcal{A}}^{CCA} = |Pr[E_0] - Pr[E_1]|.$$

If for any adversary $\mathcal{A}$ bounded by time t and n queries to Dec_{dk} *we have $\boldsymbol{Adv}_{\mathcal{A}}^{CCA} \leq \epsilon$ we say that the blind signature scheme is (t, n, ϵ)-secure with respect to IND-CCA.*

Theorem 1. *Let* $(\mathsf{Gen}_{\mathsf{Sign}}, \mathsf{Request}, \mathsf{Issue}, \mathsf{Unblind})$ *be a blind signature scheme and* $(\mathsf{Gen}_{\mathsf{Enc}}, \mathsf{Enc}, \mathsf{Dec})$ *a public key encryption scheme that is (t, n, ϵ)-secure with respect to IND-CCA. Then there exists a λ and a small constant t_0 such that* $(\mathsf{Gen}', \mathsf{Request}', \mathsf{Issue}', \mathsf{Unblind}')$ *of Fig. 6 is $(t - t_0 n, n, 2\epsilon)$-secure with respect to request integrity.*

$\mathbf{Exp}_{RI}$:
- $(pk, sk) \leftarrow \mathsf{Gen}$.
- $(m, st) \leftarrow A_1^{\mathsf{Issue}_{sk}}(pk)$.
- $(\rho, s) \leftarrow \mathsf{Request}(pk, m)$.
- $\tilde{\sigma} \leftarrow A_2^{\mathsf{Issue}_{sk}}(\rho, st)$.
- $\sigma \leftarrow \mathsf{Unblind}(s, \tilde{\sigma})$.
- If ρ is not stored in Issue_{sk}, $b \leftarrow \mathsf{Verify}(pk, m, \sigma)$, otherwise $b \leftarrow 0$.

$\mathbf{Issue}_{sk}(\rho')$:
- $\tilde{\sigma} \leftarrow \mathsf{Issue}(sk, \rho')$
- Store ρ'.
- Return $\tilde{\sigma}$.

Fig. 4. The request integrity experiment

$\mathbf{Exp}_{CCA}$:
- $(ek, dk) \leftarrow \mathsf{Gen}$.
- $(m_0, m_1, st) \leftarrow A_1^{\mathsf{Dec}_{dk}}(ek)$.
- $b \leftarrow \{0, 1\}$.
- $c \leftarrow \mathsf{Enc}(ek, m_b)$.
- $b' \leftarrow A_2^{\mathsf{Dec}_{dk}}(c, st)$.

$\mathbf{Dec}_{dk}(c')$:
- $m' \leftarrow \mathsf{Dec}(dk, m')$
- Store c'.
- Return m'.

Fig. 5. The IND-CCA experiment

Gen':
- $(vk, sk) \leftarrow \mathsf{Gen}_{\mathsf{Sign}}$.
- $(ek, dk) \leftarrow \mathsf{Gen}_{\mathsf{Enc}}$.
- $vk' \leftarrow (vk, ek)$.
- $sk' \leftarrow (sk, dk)$.
- Return (vk', sk').

Issue'$((sk, dk), \rho')$:
- $(\rho, \tilde{k}) \leftarrow \mathsf{Dec}(dk, \rho')$.
- $\tilde{\sigma} \leftarrow \mathsf{Issue}(sk, \rho)$.
- $\tilde{\sigma}' \leftarrow (\tilde{\sigma}, \tilde{k})$.
- Return $\tilde{\sigma}'$.

Request'$((vk, ek), m)$:
- $(\rho, s) \leftarrow \mathsf{Request}(vk, m)$.
- $k \leftarrow \{0, 1\}^{\lambda}$.
- $\rho' \leftarrow \mathsf{Enc}(ek, (\rho, k))$.
- $s' \leftarrow (s, k)$.
- Return (ρ', s').

Unblind'$((s, k), (\tilde{\sigma}, \tilde{k}))$:
- If $k = \tilde{k}$, return $\mathsf{Unblind}(s, \tilde{\sigma})$.
- Otherwise return $\perp$.

Fig. 6. Request integrity construction for a security parameter λ.

Proof. We can use any λ that satisfies $2^{-\lambda} \leq \epsilon$. Let us assume that $\mathcal{A}_{RI}$ is an attacker of the request integrity experiment. We build an attacker $\mathcal{A}_{CCA}$ against the IND-CCA experiment as follows. Upon receiving ek from the IND-CCA experiment $\mathcal{A}_{CCA}$ generates blind signature keys (vk, sk) with $\mathsf{Gen}_{\mathsf{Sign}}$ and sends $vk' = (vk, ek)$ to $\mathcal{A}_{RI}$. Issue requests from $\mathcal{A}_{RI}$ are handled as Issue' in Fig. 6 with the exception that the decryption oracle of the IND-CCA experiment is used instead of dk. When $\mathcal{A}_{RI}$ provides it's target message m, $\mathcal{A}_{CCA}$ generates a request token $(\rho, s) \leftarrow \mathsf{Request}(vk, m)$ and two (different) random λ-bit strings (k_0, k_1) for m and outputs $((\rho, k_0), (\rho, k_1))$ as a message pair to the IND-CCA experiment. Next $\mathcal{A}_{CCA}$ receives an encryption ρ' of one of the messages, this is the target request token for $\mathcal{A}_{RI}$. When $\mathcal{A}_{RI}$ finally outputs a blinded signature $\tilde{\sigma}' = (\tilde{\sigma}, \tilde{k})$, $\mathcal{A}_{CCA}$ outputs 0 if $\tilde{k} = k_0$, 1 if $\tilde{k} = k_1$ and guesses a random bit otherwise.

Let us now analyse the advantage of $\mathcal{A}_{CCA}$. Suppose $\mathcal{A}_{RI}$ has a success probabilty greater than 2ϵ. that means that $Pr[\tilde{k} = k_b] > 2\epsilon$. On the other hand since the view of $\mathcal{A}_{RI}$ is information theoretically independent of k_{1-b} we must have $Pr[\tilde{k} = k_{1-b}] \leq 2^{-\lambda} \leq \epsilon$. Considering the following identities we obtain the desired result.

$$Pr[b \oplus b' = 0] = Pr[\tilde{k} = k_b] + \frac{1}{2} \cdot Pr[\tilde{k} \notin \{k_0, k_1\}].$$

$$Pr[b \oplus b' = 1] = Pr[\tilde{k} = k_{1-b}] + \frac{1}{2} \cdot Pr[\tilde{k} \notin \{k_0, k_1\}].$$

$$\mathbf{Adv}^{CCA} \geq Pr[b \oplus b' = 0] - Pr[b \oplus b' = 1] = Pr[\tilde{k} = k_b] - Pr[\tilde{k} = k_{1-b}] > \epsilon.$$

In other words, any attacker with success probability greater than 2ϵ against the request integrity experiment gives an attacker with advantage greater than ϵ against the IND-CCA experiment. The overhead running time of $\mathcal{A}_{CCA}$ can always be bounded by $t_0 n$ for some small constant t_0 since it corresponds to the sending of one decryption query for each Issue query from $\mathcal{A}_{RI}$. □

2.3 Universally Composable Blind Signatures

The following theorem shows that the games defined in the previous section guarantee the security notion defined by the functionality of Fig. 8. The reductions for Thm. 2 are generic and might in practice be tighter for concrete solutions.

To analyse the security with concrete security parameters we consider two resource bounds R and R' that both bound the number of requested tokens for each player to the same constant n and the number of players to k. R bounds the total time usage to t and R' to some $t' > t$. We will assume that t' is sufficiently large so that it covers the extra time consumption related to accounting and extra computation for unblind queries in $\mathcal{F}_{\mathsf{MP}}$.

Theorem 2. *Let R and R' be as above, and $BS = (\mathsf{Gen}, \mathsf{Request}, \mathsf{Issue}, \mathsf{Unblind}, \mathsf{Verify})$ a blind signature scheme that is (t', ϵ)-secure w.r.t. weak blindness, and (t', n, ϵ)-secure w.r.t. weak unforgeability and request integrity. Then for any fixed message*

On $(\mathsf{Request}, m, Q)$ from $\mathcal{Z}$:
- Send (Key, Q) to $\mathcal{F}_{\mathsf{KG}}$ and obtain $(\mathsf{Public\ key}, pk)$.
- Generate $(\rho, s) \leftarrow \mathsf{Request}(pk, m)$, store $(\mathsf{RequestState}, pk, \rho, s, m)$ and output $(\mathsf{Request}, \rho)$.

On (Issue, ρ) from $\mathcal{Z}$:
- Send (Keys) to $\mathcal{F}_{\mathsf{KG}}$ and obtain $(\mathsf{Key\ pair}, pk, sk)$.
- Generate $\tilde{\sigma} \leftarrow \mathsf{Issue}(sk, \rho)$ and output $(\mathsf{Issue}, \tilde{\sigma})$.

On $(\mathsf{Unblind}, \rho, \tilde{\sigma})$ from $\mathcal{Z}$:
- Send (Key, Q) to $\mathcal{F}_{\mathsf{KG}}$ and obtain $(\mathsf{Public\ key}, pk)$.
- If $(\mathsf{RequestState}, pk, \rho, s, m)$ is recorded, then $\sigma \leftarrow \mathsf{Unblind}(s, \tilde{\sigma})$ and delete the request state, otherwise output (Fail) and stop.
- If $\mathsf{Verify}(pk, m, \sigma) = 1$ output $(\mathsf{Signature}, \sigma)$ otherwise output (Fail).

On $(\mathsf{Verify}, m, \sigma, Q)$ from $\mathcal{Z}$:
- Send (Key, Q) to $\mathcal{F}_{\mathsf{KG}}$ and obtain $(\mathsf{Public\ key}, pk)$.
- Run $\mathsf{Verify}(pk, m, \sigma)$ to obtain an outcome bit b and output (Verify, b).

Fig. 7. The blind signature $\mathcal{F}_{\mathsf{KG}}$-hybrid protocol for an honest player P.

m^* the protocol for blind signatures $(R, \delta, R', \delta+(2n+1)\epsilon k, \delta+(2n+1)\epsilon k)$-realises the weak blind signature functionality.

Proof. To prove the theorem we proceed by a sequence of games starting with the interaction between an environment $\mathcal{Z}$ and the protocol of Fig. 7 and ending with the interaction between $\mathcal{Z}$ and the functionality of Fig. 8 composed with a simulator that specifies the blind signature scheme.

Game 0: This is the interaction between $\mathcal{Z}$ and the protocol of Fig.7.

Game 1: In this game we gather up the protocol in one machine M_1 that 1) identically simulates the execution of Game 0, 2) keeps records and does the checks of the weak blind signature functionality of Fig. 8, and 3) on $(\mathsf{Unblind}, \rho, \tilde{\sigma})$ leaves a sufficient time gap so that the unblind procedure can later be swapped by the procedure of the functionality without additional time consumption.

It is clear that the final output by $\mathcal{Z}$ is identically distributed to the previous game which means that the two games are $(R, \delta, R', \delta, \delta)$-indistinguishable.

Game 2: In this game the machine M_1 is replaced by M_2 that instead of unblinding every $(\mathsf{Unblind}, \rho, \tilde{\sigma})$ query according to protocol, it (like the blind signature functionality) outputs (Fail) if the issuing player Q is honest and (Issue, ρ) has not previously been received by Q in M_2. By Lemma 1 the request integrity property makes this game $(R', \delta, R', \delta + n\epsilon k, n\epsilon k)$-indistingushable from the previous game.

Game 3: In this game the machine M_2 is replaced by M_3 that instead of requesting, issuing and unblinding as the protocol of Fig. 7 follows the functionality of Fig. 8. By Lemma 2 the weak blindness property makes this game

Setup Phase:

On any input m not from $X \neq \mathcal{A}$:

– Store $(\mathsf{Premature\ input}, m, X)$ and hand over (Setup) to $\mathcal{A}$.

On input a blind singature scheme $(\mathsf{Gen}, \mathsf{Request}, \mathsf{Issue}, \mathsf{Unblind})$ from $\mathcal{A}$:

– Move to execution phase and process every entry $(\mathsf{Premature\ input}, m, X)$ as if m was received from X in that phase.

Execution phase:

On $(\mathsf{Request}, m, Q)$ or $(\mathsf{Verify}, m, \sigma, Q)$ from P, (Issue, ρ) from Q or (Keys, Q) from $\mathcal{A}$:

– If $(\mathsf{Key\ pair}, pk, sk, Q)$ is not stored, generate $(pk, sk) \leftarrow \mathsf{Gen}$ and store $(\mathsf{Key\ pair}, pk, sk, Q)$. In either case follow the instructions below.

On $(\mathsf{Request}, m, Q)$ from P:

– Generate a request token $(\rho, s) \leftarrow \mathsf{Request}(pk, m^*)$, record $(\mathsf{Request}, pk, \rho, s, m, P, Q)$ and $(\mathsf{Good\ request}, \rho, Q)$ and send $(\mathsf{Request}, \rho)$ to P.

On (Issue, ρ) from P:

– Generate a blinded signature $\tilde{\sigma} \leftarrow \mathsf{Issue}(sk, \rho)$.

– If $(\mathsf{Good\ request}, \rho, Q)$ is recorded, store $(\mathsf{Good\ issue}, \rho, Q)$. Otherwise store $(\mathsf{Bad\ issue}, \rho, Q)$. In either case send $(\mathsf{Issue}, \tilde{\sigma})$ to P.

On $(\mathsf{Unblind}, \rho, \tilde{\sigma})$ from P:

– If $(\mathsf{Request}, pk, \rho, s, m, P, Q)$ is recorded then generate $\sigma^* \leftarrow \mathsf{Unblind}(s, \tilde{\sigma})$ and delete $(\mathsf{Request}, pk, \rho, s, m, P, Q)$. Otherwise send (Fail) to P and stop.

– If $\mathsf{Verify}(pk, m^*, \sigma^*) = 1$ and either $(\mathsf{Good\ issue}, \rho, Q)$ is recorded or Q is corrupt, construct new values $(\hat{\rho}, \hat{s}) \leftarrow \mathsf{Request}(pk, m)$, $\hat{\sigma} \leftarrow \mathsf{Issue}(sk, \hat{\rho})$ and $\sigma \leftarrow \mathsf{Unblind}(\hat{s}, \hat{\sigma})$, store $(\mathsf{Signature}, m, Q)$ and send $(\mathsf{Signature}, \sigma)$ to P. Otherwise send (Fail) to P.

On $(\mathsf{Verify}, m, \sigma, Q)$ from P:

– If $(\mathsf{Signature}, m, Q)$ is stored or Q is corrupted, run $b \leftarrow \mathsf{Verify}(pk, m, \sigma)$ and send (Verify, b) to P.

– Otherwise if $\mathsf{Verify}(pk, m, \sigma) = 1$ and there is a message $(\mathsf{BadIssue}, \rho, Q)$ recorded for some ρ, remove one such record entry, store $(\mathsf{Signature}, m, Q)$ and send $(\mathsf{Verify}, 1)$ to P.

– Otherwise send $(\mathsf{Verify}, 0)$ to P.

On (Keys, Q) from $\mathcal{A}$:

– Hand over $(\mathsf{Public\ key}, pk, Q)$ to $\mathcal{A}$ if Q is honest and $(\mathsf{Key\ pair}, pk, sk, Q)$ if he is corrupted.

Fig. 8. The blind signature functionality $\mathcal{F}_{\mathsf{BS}}$ for a honest player P and a fixed message m^*.

$(R', \delta + n\epsilon k, R', \delta + 2n\epsilon k, n\epsilon k)$-indistinguishable from the previous game.

Game 4: In this game the machine M_3 is replaced by M_4 that verifies signatures as the functionality of Fig. 8. By Lemma 3 the weak unforgeability property makes this game $(R', \delta + 2n\epsilon k, R', \delta + (2n+1)\epsilon k, \epsilon k)$-indistinguishable from the previous game.

Using the transitivity of indistinguishability we get that Game 0 is $(R, \delta, R', \delta + (2n+1)\epsilon k, \delta + (2n+1)\epsilon k)$-indistinguishable from Game 4. □

Lemma 1. *Under the conditions of Thm. 2 the machines M_1 and M_2 are $(R', \delta, R', \delta + n\epsilon k, n\epsilon k)$ indistinguishable, where k is the bound on the number of players and n the bound on the number of requests for each player.*

Proof. We start by defining an exceptional event for the execution of M_1 and M_2. Let E_i be the event that during an execution with M_i the following occurs for some (not necessarily distinct) honest players P and Q:

- P has received $(\mathsf{Request}, m, Q)$ and produced ρ with state information s.
- P has received $(\mathsf{Unblind}, \rho, \tilde{\sigma})$ without Q first receiving (Issue, ρ).
- $\sigma \leftarrow \mathsf{Unblind}(\tilde{\sigma}, s)$ is a valid signature on m.

As long as the events E_i do not occur the two games proceed identically.

Next we must bound the probability of the events E_i occuring. Suppose some adversary $\mathcal{A}$ can make the events E_i occur within time t with probability greater than $n\epsilon k$, where k is the bound on the number of players and n is the bound on the number of requests for each player. We can then create an attacker $\mathcal{A}_{RI}$ on the request integrity experiment as follows. Upon receiving pk from $\mathbf{Exp}_{RI}$, $\mathcal{A}_{RI}$ samples random numbers $i \leftarrow \{1, \ldots, k\}$ and $j \leftarrow \{1, \ldots, n\}$. $\mathcal{A}_{RI}$ proceeds by executing the interaction between M_2 and $\mathcal{A}$ with the exception that for the i'th player, pk and Issue_{sk} are used instead of the registered key pair. If some player receives a j'th request $(\mathsf{Request}, m, Q)$ where Q is the i'th player, then $\mathcal{A}_{RI}$ outputs m to the experiment and obtains a request token ρ. $\mathcal{A}_{RI}$ then uses this request token instead of generating a new request token and answers the i'th request query. If E_i happens for that particular request token $\mathcal{A}_{RI}$ outputs the $\tilde{\sigma}$ received by Q and wins the request integrity experiment. The probability of this happening is greater than ϵ making a contradiction to the assumption of Thm.2. □

Lemma 2. *Under the conditions of Thm. 2 the machines M_2 and M_3 are $(R', \delta, R', \delta + \epsilon nk, \epsilon nk)$-indistinguishable, where k is the bound on the number of registered keys pairs and n is the bound on the number of request tokens for each key pair.*

Proof. First of all notice that M_2 and M_3 can only differ when the request token have been generated by a honest player. We make a sequence of machines $M'_0, \cdots M'_{kn}$ as follows. Initially we define M'_0 to be M_2. For $i \in [0, k-1]$ and $j \in [1, n]$, let M'_{in+j} be the machine that acts like M_3 on the sessions corresponding

to the i first blind signature key pairs and on the j first honestly generated request tokens corresponding to the $(i+1)$'th key pair and acts like M_2 for all the other request tokens. It is clear that $M'_0 = M_2$ and $M'_{kn} = M_3$.

If every pair (M_{r-1}, M_r) are $(R', \delta + (r-1)\epsilon, R', \delta + r\epsilon, \epsilon)$-indistinguishable then we are done by transitivity of indistinguishability. Let us suppose that there is an $r = in + j$ such that M_{r-1} and M_r are distinguishable. For a distinguisher D we define the events E_r that D interacting with M'_r outputs 1 and G_r that the run is bounded by the resource bound R'. There are two ways that M_{r-1} and M_r can fail to be $(R', \delta + (r-1)\epsilon, R', \delta + r\epsilon, \epsilon)$-indistinguishable:

Case 1: There exist a D such that $|Pr[E_{r-1} \wedge G_{r-1}] - Pr[E_r \wedge G_r]| > \epsilon$.

Case 2: There exist a D such that $Pr[\neg G_{r-1}] \leq \delta + (r-1)\epsilon$ but $Pr[\neg G_r] > \delta + r\epsilon$.

In both cases this can be used to create a $(R', \delta + (r-1)\epsilon, R', \delta + r\epsilon, \epsilon')$-distinguisher between M'_{r-1} and M'_r with $\epsilon' > \epsilon$. We now define an attacker $A = (A_1, A_2, A_3)$ for the weak blindness game. A_1 receives a blind signature key pair (pk, sk) from the game, then interacts with D by running $M'_r = M'_{in+j}$ with the only exception that the $(i+1)$'th key pair is the one obtained from the game. When D asks to generate the j'th request token under the $(i+1)$'th key pair with message m, A_1 outputs (m, m^*, st) where st is the current state of the simulation of M'_{in+j}.

When A_2 receives (st, ρ_b, ρ_{1-b}) he continues to simulate M'_{in+j} with the exception that instead of creating a new j'th request token he uses the request token ρ_b and if the unblind query $(\mathsf{Unblind}, \rho_b, \tilde{\sigma})$ is made he generates $\tilde{\sigma}' \leftarrow \mathsf{Issue}(sk, \rho_{1-b})$ and outputs $(\tilde{\sigma}, \tilde{\sigma}', \tilde{st})$ where $\tilde{st}$ is the new simulation state. If A_3 receives $(\tilde{st}, \perp, \perp)$ he knows that $\tilde{\sigma}$ was not a valid blinded signature and doesn't need to come up with any corresponding unblinded signature to proceed with the simulation. If A_3 receives $(\tilde{st}, \sigma_0, \sigma_1)$ he continues the simulation using the σ_0 value for the ongoing unblind query. Finally when the simulation stops and D outputs a bit b', A_3 forwards this bit to the game.

Notice that if b is 1 in the game then A perfectly simulates M'_r and if b is 0 it simulates M'_{r-1}.

We thus have an attacker with advantage in guessing b within time t' greater than ϵ. By modifying our attacker to never exceed t' we get a (t', ϵ')-distinguisher for the weak blindness game. Which contradicts the hypothesis of Thm. 2.

□

Lemma 3. *Under the conditions of Thm. 2 the machines M_2 and M_3 are $(R', \delta, R', \delta + \epsilon k, \epsilon k)$-indistinguishable, where k is the bound on the number of registered key pairs in R.*

Proof. The machines M_2 and M_3 act identically unless the exceptional event that at some point during the execution for some (honest) key pair the adversary has created valid blind signatures for more messages than he has queried for issues. We can make a weak unforgeability attacker $\mathcal{A}_{WUF}$ by choosing a random

number i uniformly from $\{1, \ldots, k\}$ and running M_2 with the exception that the i'th key pair is replaced by the public key and oracle of the weak unforgeability experiment. If the probability of the exceptional event is larger than ϵk, then $\mathcal{A}_{WUF}$ has success probability larger than ϵ. By assumption the probability of the exceptional event occurring for an adversary bounded by time t is therefore bounded by ϵk. By the theorem for exceptional events in [6] the lemma follows. □

2.4 Partially Blind Signatures

The notion of blind signatures defined above allows us to generate "coins" of a fixed value. By creating multiple key pairs corresponding to various values one can make payments more efficient, however multiple transactions for each payment can still be considered suboptimal. An extension of blind signatures called partially blind signatures allows the signer to see a public part of the message to be signed while still being blind to the private part. This enables the user to specify the value of the coins to be signed and thus streamlines the payment systems.

For the security of partially blind signatures the blindness requirement only makes sense for signature requests with the same public part. Integrity of the public part is important to ensure that the value of a "coin" is fixed when the coin is issued.

3 Anonymous Mobile Payment

In this section we present our proposal for a mobile payment scheme and its corresponding ideal functionality. We sketch how the protocol can be simulated in a static corruption model assuming honestly generated preshared keys. Before we describe the protocol formally we must introduce the communication devices that the players use.

3.1 Near Field Communication

We model the communication between merchants and users by the functionality of Fig. 9. We will assume that physical restrictions prevent the adversary from active attacks on the near field communications, but the adversary can eavesdrop on the channel. This is modeled by a (Hack, M) message from the adversary.

3.2 Anonymous Internet Access

The functionality $\mathcal{F}_{AIA}$ of Fig. 10 is a fairly technical functionality modelling a pseudonymous connection between users and their banks. It is a modification of the anonymous internet access functionality of [9] made to suit our application.

The functionality models a global adversarially controlled network (internet), and a more local mobile network. The users can send "enter" and "leave" messages to the functionality to symbolise movement between base stations in the

On (Enter, M) from P:
- If there is already an entry with M recorded, send $(\mathsf{Merchant\ unavailable}, M)$ to P and stop.
- If P is an honest user U or the adversary $\mathcal{A}$ record (P, M) and send $(\mathsf{Entered}, M)$ to P.
- If (Hack, M) is recorded hand over $(\mathsf{Entered}, M)$ to $\mathcal{A}$.

On (Leave, M) from P:
- If there is a record (P, M), delete it and send (Left) to M, otherwise stop.
- If (Hack, M) is recorded hand over (Left, M) to $\mathcal{A}$.

On (Send, M, m) from P:
- If (P, M) is recorded send (m) to M.
- If (Hack, M) is recorded then hand over (Send, m, M) to $\mathcal{A}$.

On (Send, m) from M:
- If (P, M) is recorded send (m) to P.
- If (Hack, M) is recorded then hand over (Send, m, M) to $\mathcal{A}$.

On (Hack, M) from $\mathcal{A}$:
- Record (Hack, M).

Fig. 9. The near field communication functionality $\mathcal{F}_{\mathsf{NFC}}$.

local network. The "Deny" messages model denial of service attacks performed on the network to block the initial communication between a user and the base station. The way that $\mathcal{F}_{\mathsf{AIA}}$ was constructed in [9] allows the adversary to trace a user as he moves through the network by continuously denying service to that user. The user is however able to distinguish the traceable type of failure from the untraceable type.

Once a user has successfully entered a position he receives a pseudonym from his network provider. Hidden behind this pseudonym the user can send and receive messages on the global network.

Two new player entities are introduced in the functionality, the mobile service provider Sp and the network provider Np. These players do not play an essential role in the payment protocol of Fig. 11, but by keeping them in the model we also keep track of lower layer attacks. For example: If a session involves a corrupt Sp, then the users identity U leaks to the adversary. If Np is corrupt the pseudonyms are chosen by the adversary and the mobile position leaks.

3.3 Merchant - Bank Communication

We assume that each merchant and each bank can communicate through an insecure unauthenticated channel. There is no reason to assume a stronger channel in this setting as even with full power to replace, delete and inject messages the adversary cannot do any harm except denial of service.

3.4 Mobile Payment Protocol

Using the blind signature functionality $\mathcal{F}_{\mathsf{BS}}$ and the communication devices above we build a mobile payment protocol Π_{MP} as described in Fig. 11. For brevity we will write "send (m, ps) to B" instead of "send $(\mathsf{Send}, ps, B, m)$ to $\mathcal{F}_{\mathsf{AIA}}$" and "send m to ps" instead of "send (Send, ps, m) to $\mathcal{F}_{\mathsf{AIA}}$". Similarly for

On (Enter, pos, Np) from U:
- Stop if a record (id, pos, U, Np, Sp) exists. Else, if no record (U, id) exists, generate random identifier id, in either case record (id, pos, U, Np, Sp), where Sp is U's Sp, and handover (Enter, $id, leak$) to $\mathcal{A}$. $leak$ contains pos, Sp and Np if pos is corrupted and U if Sp is corrupted.

On (Leave, pos) from U:
- Remove any records (ps, sid, pos, U, Np), (U, ps, sid, pos), $(prid, l, sid, pos, U, Np)$ and (sid, pos, U, Np).
- Stop if (id, pos, U, Np, Sp) is not recorded, otherwise hand over (Leave, id) to $\mathcal{A}$.

On (Deny linkable, id) or (Deny unlinkable, id) from $\mathcal{A}$:
- Stop if (id, pos, U, Np, Sp) is not recorded, otherwise remove it and send (Est. failed linkable) or (Est. failed unlinkable) to U. Additionally, if Sp is honest then record (U, id) or remove any such record.

On (Entered, ps, id) from $\mathcal{A}$:
- Stop if (id, pos, U, Np, Sp) is not recorded, otherwise:
  - If U and Np are honest then generate random pseudonym ps', replace the record by (ps', pos, U, Np) and send (Entered, ps', pos, Np) to U.
  - Else, replace record by (ps, pos, U, Np) and send (Entered, ps, pos, Np) to U.

On (Send, ps, B, m) from U:
- If (ps, pos, U, Np) is not recorded, stop. Otherwise, if pos is corrupted, hand over (Send, $B, (ps, m), pos, Np$) to $\mathcal{A}$. If pos is uncorrupted, hand over (Send, $B, (ps, m), Np$) to $\mathcal{A}$.

On (Send, ps, m) from B:
- Hand over (Send, B, ps, m) to $\mathcal{A}$.

On (Deliver, ps, B, m) from $\mathcal{A}$:
- If (ps, pos, U, Np) is recorded send (m, ps) to B.

On (Deliver, B, ps, m) from $\mathcal{A}$:
- If (ps, pos, U, Np) is recorded then send (m, B, ps) to U.

On (Listen, pos) from $\mathcal{A}$:
- Collect
  - (id, Np, Sp) from every entry (id, pos, U, Np, Sp),
  - (ps, Np) from every entry (ps, pos, U, Np),

  in a list L and hand over (Listen, pos, L) to $\mathcal{A}$.

Fig. 10. The anonymous internet access functionality $\mathcal{F}_{\mathsf{AIA}}$.

(Recv., m, ps), (Recv., m, B, ps) and communication through $\mathcal{F}_{\mathsf{NFC}}$. We will be operating with signature keys (vk_U, sk_U) for any user U and encryption keys (ek_B, dk_B) for any bank B.

User Protocol:

On (Enter, pos, Np) from $\mathcal{Z}$:
- Send (Enter, pos, Np) to $\mathcal{F}_{\mathsf{AIA}}$.

On (Leave, pos) from $\mathcal{A}$:
- If there is an entry (Request, $tr, M, B, ps, pos, \rho, k$) output (Transaction lost, tr) and remove the entry.
- Send (Leave, pos) to $\mathcal{F}_{\mathsf{AIA}}$.

On $m \in \{$(Est. failed linkable), (Est. failed unlinkable)$\}$ from $\mathcal{F}_{\mathsf{AIA}}$:
- Send m to $\mathcal{Z}$.

On (Entered, ps, pos, Np) from $\mathcal{F}_{\mathsf{AIA}}$:
- Store (Pseudonym, pos, ps) and output (Entered, pos, Np).

On (Pay, tr, M, B, pos) from $\mathcal{Z}$:
- If (Pseudonym, pos, ps) is stored and no entries (Pay, tr', M', B', ps', pos') or (Request, $tr', M', B', ps', pos', \rho', k'$) are stored, then replace the entry (Pseudonym, pos, ps) by (Pay, tr, M, B, ps, pos) and send (Enter, M) to $\mathcal{F}_{\mathsf{NFC}}$.

On (Entered, M) from $\mathcal{F}_{\mathsf{NFC}}$:
- If (Pay, tr, M, B, ps, pos) is stored, send (Pay, tr, B) to M.

On (Merchant unavailable, M) from $\mathcal{F}_{\mathsf{NFC}}$:
- If (Pay, tr, M, B, ps, pos) is stored, remove it and output (Merchant unavailable, M).

On (Challenge, c, tr, B) from a merchant M:
- If there is an entry (Pay, tr, M, B, ps, pos) in record continue, otherwise stop.
- hand over (Request, $(c, M), B$) to $\mathcal{F}_{\mathsf{BS}}$ to obtain ρ. Generate a signature $\sigma_U \leftarrow \mathsf{Sign}(sk_U, (B, k, \rho))$ and encryption $e \leftarrow \mathsf{Enc}(ek_B, (U, k, \rho, \sigma_U))$ and send (Req, e, ps) to B. Replace the record entry by (Request, $tr, M, B, ps, pos, \rho, k$).

On (Leave, M) from $\mathcal{Z}$:
- If (Pay, tr, M, B, ps, pos) is stored, remove it. If (Request, $tr, M, B, ps, pos, \rho, k$) is stored, output (Transaction lost, tr) and remove it. In either case send (Leave, M) to $\mathcal{F}_{\mathsf{NFC}}$.

On (BlSig, e, ps) from B:
- Stop if there is no entry (Request, $tr, M, B, ps, pos, \rho, k$).
- If $\mathsf{Dec}(k, e) = \mathsf{Reject}$, remove the entry (Request, $tr, M, B, ps, pos, \rho, k$) and output (Payment rejected, tr).
- If $\mathsf{Dec}(k, e) = \tilde{\sigma}$ then hand over (Unblind, $\rho, \tilde{\sigma}$) to $\mathcal{F}_{\mathsf{BS}}$ to obtain σ.
- If $\mathsf{Verify}(B, (c, M), \sigma) = 1$ then send (Sign, tr, σ, B) to M, remove the record entry (Request, $tr, M, B, ps, pos, \rho, k$) and output (Payment complete, tr).

Fig. 11. Mobile payment protocol part I

3.5 Mobile Payment Functionality

The mobile payment functionality $\mathcal{F}_{\mathsf{MP}}$ of Fig. 13 captures the security properies of the protocol of the payment protocol Π_{MP}. Unfortunately to be able to realise

Merchant Protocol:

On (Pay, tr, B) from $\mathcal{F}_{\mathsf{NFC}}$:
- Store the message $(\mathsf{Entered}, tr, B)$ and output it.

On (Left) from $\mathcal{F}_{\mathsf{NFC}}$:
- If there is an entry $(\mathsf{Challenge}, c, tr, B)$ remove it and send $(\mathsf{Transaction\ lost}, tr, B)$ to $\mathcal{Z}$.
- If there is an entry $(\mathsf{Entered}, tr, B)$ stored, remove it.

On (Charge) from $\mathcal{Z}$:
- If $(\mathsf{Entered}, tr, B)$ is stored, generate a random challenge c, replace the record by $(\mathsf{Challenge}, c, tr, B)$ and send it through $\mathcal{F}_{\mathsf{NFC}}$.

On $(\mathsf{Sign}, tr, \sigma, B)$ from $\mathcal{F}_{\mathsf{NFC}}$:
- If there is an entry $(\mathsf{Challenge}, c, tr)$ and $\mathsf{Verify}(B, (c, M), \sigma) = 1$ then replace $(\mathsf{Challenge}, c, tr)$ by $(\mathsf{Payment}, c, B, \sigma)$ and output $(\mathsf{Payment\ Complete}, B, tr)$.

On (Claim) from $\mathcal{Z}$:
- For every entry $(\mathsf{Payment}, c, B, \sigma)$ stored send $(\mathsf{Claim}, c, \sigma)$ to B and remove the entry.

Bank Protocol:

On (Req, e) from ps:
- If $\mathsf{Dec}(dk_B, e)$ is of the form (U, k, ρ, σ_U), $(\mathsf{Requested}, \rho)$ is not stored and $\mathsf{Ver}(vk_U, (B, k, \rho)) = 1$ then record $(\mathsf{Pending}, \rho, k, ps)$ and $(\mathsf{Requested}, \rho)$, and output $(\mathsf{Request}, U)$.

On (Accept, U) or (Reject, U) from $\mathcal{Z}$:
- If $(\mathsf{Pending}, \rho, k, ps)$ is recorded, continue, otherwise stop.
- If accepted hand over (Issue, ρ) to $\mathcal{F}_{\mathsf{BS}}$ to obtain a blinded signature $\tilde{\sigma}$, make encryption $e \leftarrow \mathsf{Enc}(k, \tilde{\sigma})$ and send (BlSig, e) to ps.
- If rejected, make encryption $e \leftarrow \mathsf{Enc}(k, \mathsf{Reject})$ and send (BlSig, e) to ps.

On $(\mathsf{Claim}, c, \sigma)$ from M:
- If $\mathsf{Verify}(B, (M, c), \sigma) = 1$ and $(\mathsf{Claimed}, c)$ is not stored, then store $(\mathsf{Claimed}, c)$ and output $(\mathsf{Valid\ claim}, M)$.

Fig. 11. Mobile payment protocol part II

it from the functionalities $\mathcal{F}_{\mathsf{AIA}}$ and $\mathcal{F}_{\mathsf{NFC}}$ we must keep some of the quirks of those functionalities, which makes $\mathcal{F}_{\mathsf{MP}}$ quite large and difficult to read. Each transaction is maintained by a session entry $(\mathsf{Session}, trid, U, M, B, tr, ps, pos)$ where $trid$ is a string that identifies the particular session. The progression of the transaction is to a degree scheduled by the adversary and the functionality keeps track by means of state entries for the three involved players U,M and B. To accurately schedule when near field areas and mobile positions are entered or left, we additionally store the entries $(\mathsf{Leaving}, pos, trid)$, $(\mathsf{Leaving}, M, trid)$, $(\mathsf{Left}, trid)$ and $(\mathsf{Entered}, M, trid)$.

Leakage

To simplify the description of the functionality $\mathcal{F}_{\mathsf{MP}}$ we parametrise it by a leakage variable that specifies the leakage from the functionality in the different corruption scenarios. Each column in the table of Fig. 12 tells us what informa-

Leak. \ Corr.	pos	Nf	U	B	M	Np	Sp	All honest
Bank	x	x	x	x	x	x	x	x
Merchant		x	x		x			
Mobile network provider	x	x	x	x	x	x	x	x
Mobile position	x		x			x		
Mobile service provider			x			x	x	
Transaction data		x	x		x			
User identity			x	x			x	

Fig. 12. Table for leakage of $\mathcal{F}_{\mathsf{MP}}$. If for a session the top entry of a collumn is corrupted, then for each entry with an x in that column, the value of the leftmost entry in the same row is contained in $leak$.

tion leaks if the top entry is corrupted. For instance in a session where the bank B and the near field Nf are corrupted the user identity U would leak since B is corrupt likewise for the banks identity B and the network provider identity. Since Nf is corrupted the bank identity, transaction data, merchant identity and mobile network provider leaks. So in this example $leak = (U, B, Np, tr, M)$.

Scheduling and Delayed Output

The scheduling queue of [6] makes it somewhat difficult to ensure that the order of messages is the same for the protocol and a simulated execution with the ideal functionality. To keep the functionality general but still ensure that scheduling can be made correct for any appropriate protocol we must give the ideal attacker a lot of power to influence the scheduling of events in the execution.

All messages from the environment Z to $\mathcal{F}_{\mathsf{MP}}$ are on the form $(tag, content)$ where tag is a short string that describes the type of message (for example Enter

or Leave). For any such message the functionality $\mathcal{F}_{\mathsf{MP}}$ creates a scheduling identity *scid*, stores the message and hands over $(tag, scid)$ to the adversary $\mathcal{A}$. Upon receiving *scid* from $\mathcal{A}$ the functionality processes the message. This allows the simulator to determine how much delay there should be on input to $\mathcal{F}_{\mathsf{MP}}$ based on the type of message it receives. For presentational purposes we omit describing this in the functionality of Fig. 13.

Blank Values and Corrupt Switch

In the cases where the user is corrupt the adversary can split sessions into two indepentent interactions. One interaction between the merchant and the user, and one interaction between the user and the bank. The way we have modelled blind signatures makes it impossible to link a corrupt interaction with a merchant to a particular corrupt interaction with a bank in the protocol. We must therefore allow the adversary to combine a successful bank interaction of one corrupt session with a successful merchant interaction of another session in $\mathcal{F}_{\mathsf{MP}}$. The way we do this is by the corrupt switch messages.

Another issue is that when receiving values from the adversary in corrupt enter or corrupt state messages it is unreasonable to require that the adversary reveals all the session data. For example since we assume that users are anonymous when interacting with the merchant, it is unreasonable that the adversary must reveal the identity of a corrupt user in such an interaction. We therefore allow the adversary to input also empty values '$\perp$' on its corrupt pay messages. Of course if the adversary wants to interact with a particular honest merchant M (or a particular bank B) in a session he must send a corrupt pay message using M (or B) and not an empty value '$\perp$'.

4 Properties

Some desirable properties are apparent from the definition of $\mathcal{F}_{\mathsf{MP}}$.

- **Bank security:** The number of valid claims done by merchants to a honest bank can never exceed the number of accepted requests issued by that bank.
- **Merchant security:** Successful payments can only be used to create valid claims for the merchant involved. Additionally unless the communication between the bank and the merchant is broken, any successful transaction leads to a valid claim.
- **User security:** After a successful interaction with a bank, only the user himself (and the bank) can use this to complete a transaction. Moreover unless the communication between the bank and the user or between the user and the merchant is broken, any request from the user accepted by the bank leads to a successful payment.
- **Privacy preservation:** Privacy properties are inherited from the $\mathcal{F}_{\mathsf{AIA}}$ functionality. The user identity does not leak unless Sp or B is corrupted. A user can be traced through the mobile network by means of a denial of service

On $m = (\mathsf{Enter}, pos, Np)$ from U or $m \in \{(\mathsf{Deny\ linkable}, id), (\mathsf{Deny\ unlinkable}, id), (\mathsf{Entered}, ps, id)\}$ from $\mathcal{A}$:

– Do as $\mathcal{F}_{\mathsf{AIA}}$.

On $(\mathsf{Entered}, ps, id)$ from $\mathcal{A}$:

– Stop if (id, pos, U, Np, Sp) is not recorded, otherwise:
  - Send $(\mathsf{Entered}, pos, Np)$ to U.
  - If U and Np are honest then generate random pseudonym ps' and replace the record by (ps', pos, U, Np).
  - Else, replace the record by (ps, pos, U, Np).

On (Leave, pos) from U:

– If there is an entry $(\mathsf{Session}, trid, U, M, B, tr, ps, pos)$ stored, remove any user state with $trid$ and store $(\mathsf{Leaving}, trid, pos)$. If the user state was $(\mathsf{User\ state}, \mathsf{Request}, trid)$, send $(\mathsf{Transaction\ lost}, tr)$ to U.

– Hand over $(\mathsf{Leaving\ position}, trid)$ to $\mathcal{A}$.

On (Leave, M) from U:

– If there is no session entry $(\mathsf{Session}, trid, U, M, B, tr, ps, pos)$, stop.

– Remove any user state with $trid$ and store $(\mathsf{Leaving}, trid, M)$. If the user state was $(\mathsf{User\ state}, \mathsf{Request}, trid)$ send $(\mathsf{Transaction\ lost}, tr)$ to U.

– Hand over $(\mathsf{Leaving\ Merchant}, trid)$ to $\mathcal{A}$.

On $(\mathsf{Left}, trid)$ from $\mathcal{A}$:

– Continue if there is a session $(\mathsf{Session}, trid, U, M, B, tr, ps, pos)$ and either U is corrupt or there is an entry $(\mathsf{Leaving}, trid, X)$, where $X = pos$ or $X = M$. Otherwise stop.

– Remove $(\mathsf{Leaving}, trid, X)$ if it exists.

– If $X = pos$, do as $\mathcal{F}_{\mathsf{AIA}}$ on (Leave, pos).

– If $X = M$, remove any entry $(\mathsf{Entered}, M, trid)$, store $(\mathsf{Left}, trid)$ and return control to $\mathcal{A}$.

On $(\mathsf{Merchant\ exit}, trid)$ from $\mathcal{A}$:

– If $(\mathsf{Left}, trid)$ is stored for a session $(\mathsf{Session}, trid, U, M, B, tr, ps, pos)$, remove $(\mathsf{Left}, trid)$ and any merchant state with $trid$.

– If the merchant state was $(\mathsf{Merchant\ state}, \mathsf{Challenge}, trid)$, send $(\mathsf{Transaction\ lost}, tr)$ to M and return control to $\mathcal{A}$.

On $(\mathsf{Pay}, tr, M, B, pos)$ from U:

– If there is no user state corresponding to a session with U, but there is an entry (ps, pos, U, Np) stored, remove it and continue, otherwise stop.

– Generate a transaction identifier $trid$ and store the session $(\mathsf{Session}, trid, U, M, B, tr, ps, pos)$. Store $(\mathsf{User\ state}, \mathsf{entering}, trid)$, generate leakage according to the table of Fig. 12 and hand over $(\mathsf{User\ entering}, trid, leak)$ to $\mathcal{A}$.

On $(\mathsf{Entered}, trid)$ from $\mathcal{A}$:

– If there are entries $(\mathsf{Session}, trid, U, M, B, tr, ps, pos)$ and $(\mathsf{User\ state}, \mathsf{Entering}, trid)$ stored, continue, otherwise stop.

– If there is an entry $(\mathsf{Entered}, M, trid')$ already stored, replace the user state by $(\mathsf{User\ state}, \mathsf{Unavailable}, trid)$ and hand over $(\mathsf{Merchant\ unavailable}, trid)$ to $\mathcal{A}$.

– Otherwise store $(\mathsf{Entered}, M, trid)$, replace the user state by $(\mathsf{User\ state}, \mathsf{Entered}, trid)$ and store $(\mathsf{Merchant\ state}, \mathsf{Entering}, trid)$. Hand over $(\mathsf{Enter\ successful}, trid)$ to $\mathcal{A}$.

Fig. 13. Mobile payment functionality $\mathcal{F}_{\mathsf{MP}}$ part I

On (Conclude enter, $trid$) from $\mathcal{A}$:
- Stop if there is no entry (Session, $trid, U, M, B, tr, ps, pos$) stored.
- If (User state, Unavailable, $trid$) is stored, remove every entry containing $trid$ and send (Merchant unavailable, M) to U.
- If (Merchant state, Entering, $trid$) is stored, replace it by (Merchant state, Entered, $trid$) and send (Entered, tr, B) to M.
- Return control to $\mathcal{A}$.

On (Charge, tr) from M:
- If (Session, $trid, U, M, B, tr, ps, pos$) and (Merchant state, Entered, $trid$) is stored, replace the merchant state by (Merchant state, Challenge, $trid$) and hand it over to $\mathcal{A}$.

On (Challenge, $trid$) from $\mathcal{A}$:
- If (User state, Entered, $trid$) and (Mechant state, Challenge, $trid$) are stored, then replace the user state by (User state, Request, $trid$) and return control to $\mathcal{A}$.

On (Withdraw, $trid$) from $\mathcal{A}$:
- If (Session, $trid, U, M, B, tr, ps, pos$) and (User state, Request, $trid$) are stored and there are no bank states containing $trid$, store (Bank state, Pending, $trid$), send (Request, U) to B and return control to $\mathcal{A}$.

On $m \in \{(\text{Accept}, U), (\text{Reject}, U)\}$ from B:
- If session entry (Session, $trid, U, M, B, tr, ps, pos$) and bank state entry (Bank state, Pending, $trid$) are stored then replace the bank state entry by (Bank state, Accept, $trid$) or (Bank state, Reject, $trid$) and hand it over to $\mathcal{A}$.

On (Output user, $trid$) from $\mathcal{A}$:
- If (Session, $trid, U, M, B, tr, ps, pos$), (User state, Request, $trid$) are stored continue, otherwise return control to $\mathcal{A}$.
- If (Bank state, Accept, $trid$) is stored, replace the user state by (User state, Accept, $trid$), send (Payment complete, tr) to U and return control to $\mathcal{A}$.
- Else if (Bank state, Reject, $trid$) is stored, replace the user state by (User state, Reject, $trid$) and send (Payment rejected, tr) to U and return control to $\mathcal{A}$.

On (Output merchant, $trid$) from $\mathcal{A}$:
- If (Session, $trid, U, M, B, tr, ps, pos$), (User state, Accept, $trid$),(Merchant state, Challenge, $trid$) and (Bank state, Accept, $trid$) are stored, replace the merchant state by (Merchant state, Done, $trid$), store (Claim, M, B), send (Payment complete, B, tr) to M and return control to $\mathcal{A}$.

On (Claim, B) from M:
- Replace any entry (Claim, M, B) by an entry (Claimed, M, B) and hand over (Claim, M, n) to $\mathcal{A}$, where n is the number of such entries.

On (Claim, M, B) from $\mathcal{A}$:
- If there is an entry (Claimed, M, B) stored, remove one such entry, send (Valid claim, M) to B and return control to $\mathcal{A}$.

On (Corrupt Pay, $trid, U, M, B, tr, ps, pos$) from A:
- If U is a honest user or $trid$ is already in use, stop. Otherwise store the session (Session, $trid, U, M, B, tr, ps, pos$) and return control to $\mathcal{A}$.

Fig. 13. Mobile payment functionality $\mathcal{F}_{\mathsf{MP}}$ part II

On $(\mathsf{Corrupt\ switch}, trid, trid')$ from $\mathcal{A}$:
- Stop if there are no entries $(\mathsf{Session}, trid, U, M, B, tr, ps, pos)$ and $(\mathsf{Session}, trid', U', M', B, tr', ps', pos')$ stored.
- If U and U' are corrupt or $\perp$, and $(\mathsf{Merchant\ state}, \mathsf{Challenge}, trid)$ and $(\mathsf{Bank\ state}, \mathsf{Accept}, trid')$ are stored then delete every entry with $trid$, replace the session entry corresponding to $trid'$ by $(\mathsf{Session}, trid', U', M, B, tr, ps', pos')$ and the merchant state by $(\mathsf{Merchant\ state}, \mathsf{Done}, trid')$, store (Claim, M, B) and, send $(\mathsf{Payment\ complete}, B, tr)$ to M and return control to $\mathcal{A}$.

On (Listen, pos) from $\mathcal{A}$:
- Make a *leak* list as $\mathcal{F}_{\mathsf{AIA}}$ except that the list additionally contains the leakage as in the table of Fig. 12 for every entry $(\mathsf{Session}, trid, U, M, B, tr, ps, pos)$.

On (Hack, M) from $\mathcal{A}$:
- Record (Hack, M) and for every entry $(\mathsf{Session}, trid, U, M, B, tr, ps, pos)$ generate *leak* as in the table of Fig. 12 and hand over $(\mathsf{Hack}, M, \{leak\})$ to $\mathcal{A}$.

On $(\mathsf{Corrupt\ state}, P, X, trid)$ from $\mathcal{A}$:
- Stop if P is not a corrupt player or there is no entry $(\mathsf{Session}, trid, U, M, B, tr, ps, pos)$ stored.
- If $P = \mathsf{User}$ replace the user state by $(\mathsf{User\ state}, X, trid)$,
- If $P = \mathsf{Merchant}$ replace the merchant state by $(\mathsf{Merchant\ state}, X, trid)$,
- If $P = \mathsf{Bank}$ replace the bank state by $(\mathsf{Bank\ state}, X, trid)$.
- In either case return control to $\mathcal{A}$.

Fig. 13. Mobile payment functionality $\mathcal{F}_{\mathsf{MP}}$ part III

attack in corrupted positions, but as this is a very noticeable attack it is not a serious breach to the privacy. Lastly transaction data only leaks when M or Nf is corrupted appart from the data that can be obtained by traffic analysis of the communication channels.

5 Simulation

In this section we sketch for the different corruption scenarios how the protocol can be simulated from the functionality.

Setup:

In the setup phase before the actual protocol runs the simulator generates signature and encryption keys for all the honest players and registers them. This especially means that the simulator can read messages encrypted to honest banks and simulate perfectly signatures and ciphertexts from honest players.

Honest users:

In the sessions where the user is honest the simulation is mostly straight forward. The simulator runs the protocol using fake values when they are unknown to him and keeping the functionality $\mathcal{F}_{\mathsf{MP}}$ updated on the progression by sending messages like $(\mathsf{Challenge}, trid)$, $(\mathsf{Withdraw}, trid)$ etc. If the bank or merchant is corrupted the simulator can force the session to proceed without waiting for input from the environment by using "$\mathsf{Corrupted\ state}$" messages. Note that any player identity, pseudonym, position or transaction identifier that would have leaked in the protocol also leaks to the simulator from the functionality and that the outcome of a session never depends on the unknown values.

If the near field or the mobile position gets corrupted during one session the simulator must replace fake values by the newly obtained ones before leaking them to the adversary.

By using UC-secure encryption and signatures we ensure that only the honest user can make requests on its own behalf and since only fresh request tokens are accepted by honest banks there can be no injection, replay or modification of messages between honest players.

Dishonest user:

In the case when the user is dishonest the protocol session may split into two parts. The dishonest user can interact with his bank (with or without a merchants involvement) or he can interact with his merchant. Since the request tokens of our blind signature functionality do not commit the user to a fixed message at the time of requesting, the simulator has no hope to match a bank interaction with a specific merchant interaction. In fact for the merchant interactions the corrupted user remains anonymous to the simulator. We can however

guarantee that a successful payment corresponds to some blinded signature issued by the bank to some dishonest user.

When a corrupt user U interacts with a bank B, the simulator receives an encrypted message of the data (U, k, ρ, σ_U) from ps at pos to B. If σ_U is fresh and $\mathsf{Ver}(vk_U, (B, k, \rho)) = 1$, the environment expects B to output $(\mathsf{Request}, U, ps)$. The simulator can achieve this by sending the messages $(\mathsf{Corrupt\ pay}, trid, U, \perp, B, \perp, ps, pos)$, $(\mathsf{Corrupt\ state}, \mathsf{User}, \mathsf{Request}, trid)$ and finally $(\mathsf{Withdraw}, trid)$ to $\mathcal{F}_{\mathsf{MP}}$. If the simulator later receives $(\mathsf{Bank\ state}, \mathsf{Accept})$ or $(\mathsf{Bank\ state}, \mathsf{Reject})$ from $\mathcal{F}_{\mathsf{MP}}$ the simulator follows the protocol and sends encryptions of $\tilde{\sigma}$ or (Reject) to U. For any successful interaction with B the simulator stores the $trid$ as a corrupted bank interaction for later use.

In an interaction between a corrupt user and a honest merchant M the user is anonymous and unknown to the simulator. Such an interaction would be initiated by the adversary sending a message (Enter, M) to the simulated $\mathcal{F}_{\mathsf{NFC}}$ functionality followed by a message (Pay, tr, B) to M through $\mathcal{F}_{\mathsf{NFC}}$. When this occurs the simulator can hand over $(\mathsf{Corrupt\ pay}, trid, \perp, M, B, tr, \perp, \perp)$, $(\mathsf{Corrupt\ state}, \mathsf{User}, \mathsf{Entering}, trid)$ and $(\mathsf{Entered}, trid)$. If later in the session the merchant receives the message $(\mathsf{Sign}, tr, \sigma, B)$ and $\mathsf{Ver}(bvk_B, \sigma, (c, M)) = 1$ where c is the simulated challenge, the simulator picks any stored $trid'$ from a successful corrupted interaction with B and sends $(\mathsf{Corrupt\ switch}, trid, trid')$ to $\mathcal{F}_{\mathsf{MP}}$. By the definition of the blind signature functionality the adversary can only generate a valid signature if the simulator has such a stored $trid'$ and the interaction with M is completed only if the adversary generates a valid blind signature.

6 Conclusions

In this paper we have defined a weaker notion of blind signatures that differs from previous UC-secure blind signatures in that the communication channel is abstracted away. We have defined a mobile payment protocol using the blind signatures and analysed it in the modified UC framework of [6].

References

1. S. Brands. Untraceable off-line cash in wallets with observers (extended abstract). In *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 302–318. Springer, 1993.
2. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145. IEEE Computer Society, 2001.
3. D. Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203. Plenum Press, New York, 1982.
4. D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *CRYPTO*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327. Springer, 1988.
5. M. Fischlin. Round-optimal composable blind signatures in the common reference string model. In *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 60–77. Springer, 2006.

6. K. Gjøsteen, G. Petrides, and A. Steine. A novel framework for protocol analysis. In *ProvSec*, volume 6980 of *Lecture Notes in Computer Science*, pages 340–347. Springer, 2011.
7. A. Juels, M. Luby, and R. Ostrovsky. Security of blind digital signatures (extended abstract). In *IN CRYPTO '97: PROCEEDINGS OF THE 17TH ANNUAL INTERNATIONAL CRYPTOLOGY CONFERENCE ON ADVANCES IN CRYPTOLOGY*, pages 150–164. Springer-Verlag, 1997.
8. L. Kråkmo. *Privacy Preserving Protocols and Security Proof Techniques*. PhD thesis, NTNU, 2009.
9. G. Petrides, K. Gjøsteen, and A. Steine. Towards privacy preserving mobile communications. Ph.D. Thesis.