Basic Research in Computer Science

Honest Verifier Zero-knowledge Arguments Applied

Jens Groth

BRICS Dissertation Series **DS-04-3**

ISSN 1396-7002 **October 2004**

See back inner page for a list of recent BRICS Dissertation Series publications. Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `DS/04/3/`

Honest Verifier Zero-Knowledge Arguments Applied

Jens Groth

PhD Dissertation

BRICS
Department of Computer Science
University of Aarhus
Denmark

Honest Verifier Zero-Knowledge Arguments Applied

A Dissertation
Presented to the Faculty of Science
of the University of Aarhus
in Partial Fulfilment of the Requirements for the
PhD Degree

by
Jens Groth
June 6, 2004

Abstract

We apply honest verifier zero-knowledge arguments to cryptographic protocols for non-malleable commitment, voting, anonymization and group signatures. For the latter three the random oracle model can be used to make them non-interactive. Unfortunately, the random oracle model is in some ways unreasonable, we present techniques to reduce the dependence on it.

Several voting schemes have been suggested in the literature, among them a class of efficient protocols based on homomorphic threshold encryption. Voters encrypt their votes, and using the homomorphic property of the cryptosystem we can get an encryption of the result. A group of authorities now makes a threshold decryption of this ciphertext to get the result. We have to protect against voters that cheat by encrypting something that is not a valid vote. We therefore require that they make a non-interactive zero-knowledge argument of correctness of the encrypted vote. Using integer-commitments, we suggest efficient honest verifier zero-knowledge arguments for correctness of a vote, which can be made non-interactive using the Fiat-Shamir heuristic. We suggest arguments for single-voting, multi-way voting, approval voting and shareholder voting, going from the case where the voter has a single vote to the case where a voter may cast millions of votes at once.

For anonymization purposes, one can use a mix-net. One way to construct a mix-net is to let a set of mixers take turns in permuting and reencrypting or decrypting the inputs. If at least one of the mixers is honest, the input data and the output data can no longer be linked. For this anonymization guarantee to hold we do need to ensure that all mixers act according to the protocol. For use in reencrypting mix-nets we suggest an honest verifier zero-knowledge argument of a correct reencrypting shuffle that can be used with a large class of homomorphic cryptosystems. For use in decrypting mix-nets, we offer an honest verifier zero-knowledge argument for a decrypting shuffle.

For any NP-language there exists, based on one-way functions, a 3-move honest verifier zero-knowledge argument. If we pick a suitable hard statement and a suitable 3-move argument for this statement, we can use the protocol as a commitment scheme. A commitment to a message is created by using a special honest verifier zero-knowledge property to simulate an argument with the message as challenge. We suggest picking the statement to be proved in a particular

way related to a signature scheme, which is secure against known message attack. This way we can create many commitments that can be trapdoor opened, yet other commitments chosen by an adversary remain binding. This property is known as simulation soundness. Simulation soundness implies non-malleability, so we obtain a non-malleable commitment scheme based on any one-way function. We also obtain efficient non-malleable commitment schemes based on the strong RSA assumption.

We investigate the use of honest verifier zero-knowledge arguments in signature schemes. By carefully selecting the statement to be proven and finding an efficient protocol, we obtain a group signature scheme that is very efficient and has only weak set-up assumptions. Security is proved in the random oracle model. The group signature scheme can be extended to allow for dynamic join of members, revocation of members and can easily be transformed into a traceable signature scheme. We observe that traceable signatures can be built from one-way functions and non-interactive zero-knowledge arguments, which seems to be weaker than the assumptions needed to build group signatures.

Acknowledgements

The three years of work leading to this dissertation have been tremendously interesting. I would like to thank advisors, co-authors and others that I have interacted with.

First, I would like to thank Ivan Damgård for always taking the time to give sound advice when needed. I owe this dissertation to him; had it not been for his encouragement I might not even have embarked on this path in the first place. Tusind tak!

Thanks also go to Gorm Salomonsen for showing me real life project management and development. He has the ability to come up with clever ideas and several of the papers I have authored or co-authored are due to our cooperation on developing a working voting system.

Many other people helped me in one way or another. Moni Naor invited me to the Weizmann Institute and I had an enjoyable and productive time there; my only regret is that I could not stay longer. Jan Camenisch worked and is working with me on group signatures, he has kindly allowed me to include some of his ideas in the dissertation. Anders Bjarklev much to my surprise actually read and commented on my quarterly reports to ATV :-).

Students, staff and visitors at BRICS, students at the Weizmann Institute and coworkers at Cryptomathic have all been friendly and helpful. In particular I would like to thank Douglas Wikström for cooperation, and Philip MacKenzie, Felix Brandt, Helger Lipmaa, Berry Schoenmakers, Mads Jurik, Jesper Buus Nielsen, Serge Fehr, Maciej Koprowski, Saurabh Agarwal, Kristian Gjøsteen, Kirill Morozov, Alon Rosen, Adi Akavia, Danny Harnik, Eran Ofek, Udi Wieder, Boaz Barak, Huafei Zhu, Dörte Rappe, Jesùs F. Almansa, Ronald Cramer, Louis Salvail and Martijn Stam for interesting discussions.

Jens Groth,
Århus, June 6, 2004.

What to Read in this Dissertation

You are now sitting with the final version of the dissertation that I successfully defended on October 15, 2004. I was planning to remove some of the typos and brush it up a little before sending it to print, but have realized that I will most likely never have the time to do so. Instead I bring you the original dissertation with this little guide to what to read and not to read.

Chapter 4: Validating Votes. This chapter appears in an updated version at ACNS 2005 under the title "Non-interactive Zero-knowledge Arguments for Voting". My advice is to download the full paper from my homepage instead of reading this chapter.

Chapter 5: Verifying Shuffles. I am trying to turn this chapter into a full paper version of [50]. Several changes and simplifications have already been made. Before reading this chapter you may want to look at my homepage for a newer version.

Chapter 6: Non-malleable Commitment. This version is more recent than [27], so I think this is the only chapter I might actually recommend reading.

Chapter 7: Group Signature. Jan Camenisch and I have published the results presented in this chapter at SCN 2003 [14]. I recommend getting the full version of that paper from my homepage instead of reading this chapter.

Jens Groth,
Los Angeles, April 6, 2005.

Contents

Chapter 1

Introduction

Honest verifier zero-knowledge arguments have many applications in cryptographic protocols. We suggest efficient honest verifier zero-knowledge (HVZK) arguments for voting, efficient HVZK arguments for shuffling, use HVZK argument to construct non-malleable commitments, and use HVZK arguments to construct efficient group signatures.

Honest verifier zero-knowledge arguments. Consider two parties, a prover and a verifier. They both have input x, and P wants to prove that $x \in L$, where L is some NP-language. In other words P knows a witness w for $x \in L$, but he does not want V to obtain anything else than the single bit of information that $x \in L$.

To accomplish this they may carry out an interactive protocol. Most honest verifier zero-knowledge protocols in use are efficient 3-move protocols, called Σ-protocols. The prover sends an initial message a, the verifier chooses a random challenge e, and the prover sends an answer z. Given (x, a, e, z) the verifier can now decide whether to accept or reject. A Σ-protocol has the following three properties.

Completeness: If indeed $x \in L$ and the prover knows a witness w, then the verifier will accept this at the end of the protocol.

Special soundness: Given acceptable arguments (x, a, e, z) and (x, a, e', z'), where $e \neq e'$ we can extract the witness w for $x \in L$.

Special honest verifier zero-knowledge: Given any challenge e we can simulate an argument (a, e, z) with this challenge, which is indistinguishable from a real argument with challenge e.

Special soundness implies both standard soundness and knowledge.

It is possible to make a Σ-protocol non-interactive through the Fiat-Shamir heuristic. Instead of letting the verifier choose the challenge, we compute $e = \text{hash}(x, a)$. This way the verifier need not be involved at all, the argument is

non-interactive. We can hope that the argument is still sound and that it does not reveal any useful information. This hope is expressed in the random oracle model, a heuristic model where we in security proofs replace the hash-function with a random oracle. The random oracle assigns to each input (x, a) a random string e. Furthermore, to argue zero-knowledge we let the simulator program the random oracle, i.e., the simulator can choose an e to assign to some specific input (x, a).

Most of the honest verifier zero-knowledge arguments we suggest in this dissertation are Σ-protocols. For shuffling, we do use a 7-move argument though, and in some cases we do not have the special soundness property. We therefore define more carefully the notion of an argument of knowledge and the class of honest verifier zero-knowledge arguments that we are investigating in Chapter 2.

NIZK arguments in voting. In voting we have the seemingly conflicting requirements of keeping votes secret and at the same time having to count them. Cramer, Gennaro and Schoenmakers [23] suggest using a homomorphic threshold cryptosystem to solve this problem. Suppose we consider yes/no-voting, the voter can then encrypt 1 to vote yes and encrypt 0 to vote no. The homomorphic property of the cryptosystem states that

$$E_{pk}(m_1 + m_2) = E_{pk}(m_1)E_{pk}(m_2).$$

Taking the product of all encrypted votes, we get an encryption of the sum of all votes, i.e., an encryption of the number of yes-votes. We now let a group of authorities hold a secret sharing of the decryption key for the cryptosystem. They can use multi-party computation to decrypt the ciphertext and get the result. We can therefore count the votes if a large enough group of authorities cooperate, on the other hand no individual vote is revealed if a large enough group of authorities is honest and does not participate in a decryption protocol for an individual voter's ciphertext.

A major problem to resolve in the protocol suggested above is to ensure that voters do not send in invalid ballots. Suppose a voter sends in $E_{pk}(100)$, then effectively he has cast a 100 yes-votes. To avoid this kind of attack we demand that a voter forms a NIZK argument for his ciphertext c containing a valid vote. General NIZK arguments [38, 56] are very inefficient so in practice we use the Fiat-Shamir heuristic to make an HVZK argument for the correctness of a vote non-interactive. We can then argue security in the random oracle model. More precisely we can show that this type of voting scheme realizes an ideal voting functionality in the universal composability framework, see [51] for details.

Voting is a touchy subject and we may want to guarantee secrecy of ballots many years into the future. This means that we have to select very large key sizes. Carrying out the encryption of the vote itself can therefore be a heavy computation. HVZK arguments for correctness of a vote suggested in [23, 7, 29, 31]

rely on the use of multiple encryptions of intermediate values too. The overhead stemming from the HVZK argument is therefore considerable. Lipmaa, Asokan and Niemi [59] suggest a more efficient HVZK argument that uses integer commitments instead of encryptions. The advantage is two-fold: A commitment only has to be computationally binding it can be much smaller than the committed integer and computationally easier to handle. Since we commit to integers, we can also take advantage of special properties of the integers, among them the property that any integer has a unique prime factorization.

In [28] we also use integer commitments for the HVZK arguments of correctness of a vote. [59] use among other things an HVZK argument of a committed integer being positive, we are able to avoid this argument since we ensure that any valid vote is a square and thereby automatically positive. Another contribution of [28] is efficient HVZK arguments for multi-way voting, where each voter may cast a specified number of votes. In this dissertation, we improve on this HVZK argument.

Different types of elections exist. In approval voting each voter can vote for as many of the candidates as he likes. We suggest an HVZK argument for this kind of election too. Some elections allow weighted votes, the voter orders the candidates in order of preference and gives 0 votes to the worst candidate, 1 vote to the second worst, etc. Interestingly, the techniques we use for shuffling allow us to make an efficient HVZK argument in this kind of election.

In shareholder elections, each voter may have many shares and thus many votes to cast. If a voter has a million shares, it would be inconvenient to cast a million single-votes and it would also be inconvenient to use our multi-way HVZK argument. Ishida, Matsuo and Ogata [53] suggest the notion of divisible voting, meaning that the voter can divide his million votes one several candidates. Their scheme does not take advantage of integer commitments though, we suggest a significantly more efficient scheme for this type of election.

HVZK argument for correctness of a shuffle. Suppose we have a homomorphic cryptosystem, for instance ElGamal encryption. The homomorphic property means that we can rerandomize a ciphertext. Given a ciphertext c we can create a new ciphertext $C \leftarrow E_{pk}(0)c$. Now C contains the same plaintext as c, however this fact may be hard to verify for an outsider. Indeed shuffling builds on the hardness of linking rerandomized ciphertexts to the original ciphertexts.

In a shuffle, we get as input a set of ciphertexts $c_1, \ldots, c_n$. We select a permutation $\pi \in \Sigma_n$ at random. We compute a new set of ciphertexts $C_1 \leftarrow c_{\pi(1)}E_{pk}(0), \ldots, C_n \leftarrow c_{\pi(n)}E_{pk}(0)$. We say $c_1, \ldots, c_n$ have been shuffled into $C_1, \ldots, C_n$. Using a suitable homomorphic cryptosystem for this purpose, an external party will have no clue about the permutation π that has been used.

Shuffles constitute a building block in many mix-nets. The idea is that a group of mixers wants to shuffle a set of ciphertexts. To do this they take turns

in shuffling the ciphertexts. When the last mixer finishes, we have a shuffle of the input ciphertexts if all mixers follow the protocol. Furthermore, if at least one of the mixers is honest and does not reveal the permutation he has used, then the permutation in the entire mix is secret.

This type of mix-net requires that indeed the parties follow the protocol, otherwise many attacks exist. HVZK arguments for correctness of a shuffle have been suggested in [44, 66, 50]. We use techniques related to the latter paper but are able to obtain a considerable reduction in the size of the HVZK argument.

In a decrypting mix-net, the mixers must both shuffle and decrypt the incoming ciphertexts. In the case of ElGamal encryption, a method would be the following. The mixers share the secret key. Each mixer performs a shuffle of the ciphertexts and subsequently makes a partial decryption according to its share of the decryption key. With the fast shuffle arguments mentioned above, the relative cost of making the decryption argument is significant. [43, 42] therefore propose arguments that prove directly a correct relation between input ciphertexts and output shuffled and partially decrypted ciphertexts. Unfortunately, these arguments are not zero-knowledge. We suggest modifying our HVZK shuffle argument to an HVZK shuffle-and-decrypt argument at virtually no loss of efficiency.

Our approach is very general, we can make HVZK arguments for correct shuffles for almost any type of homomorphic encryption when we know the order of the message space. We do suggest a method that can be used also for message spaces of unknown order and which gives an advantage when using large message space. The size of this type of HVZK argument is independent of the order of the message space that we are shuffling.

Non-malleable commitments. Consider the case of fair contract bidding. Each bidder commits to his bid. When everybody has made a commitment, the bidders open their commitments and we can determine the winner. Unfortunately this scheme does not always work. Many commitment schemes are malleable, seeing the commitment of one party we may make a commitment to a related value. For instance, if party A bids m then party B may bid to $m-1$. At the time of making the commitment B does not know exactly what he bid, however, when A opens his commitment to m, party B can use this information and open his own commitment to $m-1$. To prevent this kind of attack we can use non-malleable commitment schemes.

Non-malleability was first introduced in [37]. Essentially the definition compares two protocols and non-malleability is when the adversary cannot gain an advantage in a man-in-the-middle attack. [37] also suggested an interactive non-malleable commitment scheme. A few other interactive non-malleable commitment schemes have been suggested [41, 4]. We focus on non-interactive commitment schemes. A non-malleable cryptosystem can often serve as an unconditionally binding non-malleable commitment scheme too. Better options may exist

though, both in term of assumptions and efficiency. [35] suggest a non-malleable commitment scheme based on one-way functions, and [36] suggest an unconditionally hiding commitment scheme based on a DDH assumption. Both these latter schemes only give non-malleability in the case where the adversary sees a single commitment though. In [27], which this dissertation builds upon, we obtain a scheme which is non-malleable in the case where the adversary sees an unbounded number of inputs. Actually we present a general construction of a commitment scheme based on HVZK arguments, which can either be instantiated using the minimal assumption of one-way functions, or be instantiated efficiently using stronger assumption, for instance the strong RSA assumption. Independently of this work [45, 61] have suggested the related notion of simulation sound commitments, which implies non-malleability. They use similar techniques to construct simulation sound commitments. Finally, we should mention the notion of universally composable commitments [18, 21], which imply non-malleability. Constructions of these commitments rely on both strong assumptions and are not so efficient. [32] suggest an interactive universally composable commitment scheme, it turns out that we can actually apply our non-malleable commitment scheme to improve their scheme.

Let us describe how this problem has any relation with HVZK arguments. Consider a hard membership problem $x \in L$. Consider furthermore a Σ-protocol for membership of this language. It is possible to use this Σ-protocol to make a commitment. To commit to message m we use the special honest verifier zero-knowledge property to simulate an argument (a, m, z) for $x \in L$. The commitment consists of (x, a), while we open the commitment by showing (m, z). The commitment scheme is binding, because if we could open the commitment to two different messages m, m', then the special soundness property means that we can compute a witness w for $x \in L$, which we assumed was a hard language. On the other hand, it is hiding because just seeing a gives us no clue about which message m we can answer correctly.

If we select the problem x with care we actually get a reusable non-malleable commitment scheme. The main idea is to pick it such that the witness for x is a signature on x. If the signature scheme we choose is secure against known message attack, we can actually make many different commitments with different x's and the problem of finding a signature on a new x will still be hard. We force the adversary to use a new x when it makes its commitments, and therefore the adversary's commitments are binding. They are binding even though we can set it up such that we know signatures on the x's of the input and thus can open those commitments to any message we like. Using rewinding techniques, it is now simple to extract from the adversary the contents of its commitments.

We find this application of an HVZK argument interesting because here we do not use the random oracle model to make it non-interactive, something that we do or at least can do in the other applications in this dissertation.

Group signatures from HVZK arguments. The Fiat-Shamir heuristic [39] was introduced to transform an HVZK argument into a signature scheme. There is thus a clear link between HVZK arguments and signatures.

We shall look at a complex type of signature scheme, namely a group signature scheme. A group signature scheme allows members to make signatures on behalf of the group. such a group signature only shows that somebody from the group signed the message, not which member of the group actually made the signature. One reason to use group signatures can be to hide organizational structure, consider for instance an intelligence agency that does want to ensure integrity of the information it is getting, however, does not want the enemy to be able to associate data with particular agents. In special cases of abuse, however, we want to be able to find out who signed a particular message. Consider for instance a double agent that continuously supplies false information, we want to be able to identify that agent and "silence" him. For this purpose, we have a group manager that holds a master key that allows him to open signatures and see who actually signed the message. We may require also that the group signature is dynamic, i.e., the group manager can let new members join and/or revoke memberships.

To create such a signature scheme we set up a problem x such that only a member of the group can know the witness w. A group signature then consists of an HVK argument of knowledge of w. We make this argument non-interactive by selecting the challenge $e = \text{hash}(x, a, m)$, where m is the message that we want to sign. Let us briefly describe the kind of problem x that the member needs to make a group signature. It must be a problem that can be randomized such that x itself does not reveal the identity of the signer. Typically, this is done using encryption, so the problem is of the nature: "I know that x is an encryption of...". The plaintext must also be of a special nature. We want two things from it: It should contain something known only to the signer, to avoid that the group manager or somebody else frames the member by making a signature in his name. It should also contain something that ties the connection with the group. In our scheme, this is a digital signature created by the group manager. Finally, the two components must be connected in some way, otherwise a member could start issuing keys himself.

Several group signature schemes have been suggested in the literature, the most efficient being [1]. In [15] we suggest a more efficient group signature scheme using a new type of membership problem x and an efficient HVZK argument for it. We prove security according to the [10] model but also note that our group signature has several additional nice properties not covered by this definition. In particular, we support dynamically joining new members, revocation of membership and can transform it into a traceable signature scheme.

Chapter 2

Preliminaries

2.1 Notation

Unless otherwise noted the algorithms and adversaries we consider are modeled as probabilistic polynomial time Turing machines or interactive probabilistic polynomial time Turing machines. In general they get a security parameter 1^k as input, however, we will usually not write this explicitly. Whenever we speak of an adversary or a distinguisher unless otherwise stated they get an auxiliary input that is bounded by some polynomial in the security parameter. This makes them non-uniform. Also this input we will usually not state explicitly.

A function f of the security parameter is called negligible if for any $c > 0$ there is a $K > 0$ so for all $k > K$ we have $f(k) < k^{-c}$. On occasion, we will just write $negl(k)$ for some function that is negligible. If we have two functions f, g we write $f(k) \approx g(k)$ if $|f(k) - g(k)| < negl(k)$. We call a function that is not negligible for noticeable.

2.2 Commitment

A commitment scheme consists of a key generation algorithm K, a commitment algorithm *com* and a decommitment algorithm dec. The key generation algorithm produces a public key pk, which specifies a message space $\mathcal{M}_{pk}$, a randomizer space $\mathcal{R}_{pk}$, an opening space or decommitment space $\mathcal{D}_{pk}$ and a commitment space $\mathcal{C}_{pk}$. These may be included in larger spaces $\mathcal{D}_{pk} \subset \mathcal{QD}_{pk}, \mathcal{R}_{pk} \subset \mathcal{QR}_{pk}, \mathcal{C}_{pk} \subset \mathcal{QC}_{pk}$.

We require that the commitment is hiding and binding.

Hiding: For all adversaries $\mathcal{A}$, we have

$$\begin{aligned} & \Pr[pk \leftarrow K(); (m_0, m_1) \leftarrow \mathcal{A}(pk); (c, d) \leftarrow \mathrm{com}_{pk}(m_0) : \mathcal{A}(c) = 1] \\ \approx \; & \Pr[pk \leftarrow K(); (m_0, m_1) \leftarrow \mathcal{A}(pk); (c, d) \leftarrow \mathrm{com}_{pk}(m_1) : \mathcal{A}(c) = 1]. \end{aligned}$$

Binding: For all adversaries $\mathcal{A}$, we have

$$\Pr[pk \leftarrow K(); (c, d_1, d_2) \leftarrow \mathcal{A}(pk); m_1 \leftarrow \mathrm{dec}_{pk}(c, d_1); \\ m_2 \leftarrow \mathrm{dec}_{pk}(c, d_2) : \perp \neq m_1 \neq m_2 \neq \perp] \approx 0.$$

We say the commitment scheme is unconditionally hiding or unconditionally binding if $\mathcal{A}$ is unbounded in the respective experiment.

Randomness revealing commitments. In many cases, the opening of a commitment d consists of the message that we committed to and the randomness we used. I.e., $d = (m, r) \in \mathcal{M}_{pk} \times \mathcal{R}_{pk}$. We call a commitment randomness revealing if the decommitment information is on this form.

Homomorphic property. We say a randomness revealing commitment scheme is homomorphic if the message space, the randomizer space and the commitment space are abelian groups and

$$\forall (m_1, r_1), (m_2, r_2) \in \mathcal{M}_{pk} \times \mathcal{R}_{pk} : \\ \mathrm{com}_{pk}(m_1 + m_2; r_1 + r_2) = \mathrm{com}_{pk}(m_1; r_1)\mathrm{com}_{pk}(m_2; r_2).$$

Root extraction property. Let abelian groups $\mathcal{R}_{pk} \leq \mathcal{QR}_{pk}$ and $\mathcal{C}_{pk} \leq \mathcal{QC}_{pk}$ be associated with the public key. We demand that also with respect to these groups the homomorphic property, as well as the hiding and binding property holds.

A homomorphic commitment scheme may have the following root extraction property. There exists and extractor E such that for all adversaries $\mathcal{A}$ we have

$$\Pr[pk \leftarrow K(); (c, e, M, R) \leftarrow \mathcal{A}(pk); (m, r) \leftarrow E(c, e, M, R) : \\ \text{if } c \in \mathcal{QC}_{pk}, M \in \mathcal{M}_{pk}, R \in \mathcal{R}_{pk}, \gcd(e, |\mathcal{C}_{pk}|) = 1, \\ c^e = \mathrm{com}_{pk}(M; R) \text{ then } c = \mathrm{com}_{pk}(m; r)] \approx 1.$$

We call an element $c \in \mathcal{QC}_{pk}$ a quasi-commitment, and $(m, r) \in \mathcal{M}_{pk} \times \mathcal{QR}_{pk}$ a quasi-opening.

Example. Consider a Pedersen commitment with q, p being primes so that $q|p-1$. We have a group $G_q \leq \mathbb{Z}_p^*$ of order q, and select two random generators for this group, g, h. We use $\mathcal{M}_{(p,q,g,h)} = \mathcal{R}_{(p,q,g,h)} = \mathbb{Z}_q$ and $\mathcal{C}_{pk} = G_q$. To commit to m we pick $r \in \mathbb{Z}_q$ at random and compute $c = \mathrm{com}(m; r) = g^m h^r \bmod p$. The opening is $d = (m, r)$. To open c we simply check whether $c = \mathrm{com}(m; r)$ and in that case output m.

To formulate the root extraction property we consider $\mathcal{QC}_{pk} = \mathbb{Z}_p^*$ and $\mathcal{QR}_{pk} = \mathbb{Z}_q \times \mathbb{Z}_p^*/G_q$. Given e which is prime to q and an element $c \in \mathbb{Z}_p^*$, $M \in \mathbb{Z}_q$, $R \in \mathbb{Z}_q$ so $c^e = g^M h^R \bmod p$, we get $c = a g^m h^r$, where $m = Me^{-1} \bmod q$, $r = Re^{-1} \bmod q$ and $a^e = 1 \bmod p$, which shows that $a \in \mathbb{Z}_p^*/G_q$.

Example. Consider $n = pq$ where p, q are safe primes with $p'|p-1$ and $q'|q-1$. Reasonable sizes would be $|p| = |q| = 512$. The public key consists of (n, g, h) where g, h are randomly chosen generators of QR_n. The message space is $\mathbb{Z}$ and the randomizer space is $\mathbb{Z} \times \{-1, 1\}$. To commit to an integer m with randomness (r, b) we compute $c = bg^m h^r \bmod n$. To open c we simply reveal $d = (m, (r, b))$. The commitment scheme is obviously homomorphic and randomness revealing. The hiding property, the binding property and the root extraction property have been proved in [26].

Multi-commitment. Quite often, we need to commit to many elements at once. In the examples above, this is simple. If we have keys $g_1, \ldots, g_n, h$, we can let a commitment be computed as $g_1^{m_1} \cdots g_n^{m_n} h^r \bmod p$ for the Pedersen commitment. We could of course do something similar with respect to the integer commitment scheme. Of course, this can just be seen as a homomorphic commitment with message space $\mathbb{Z}_q \times \cdots \times \mathbb{Z}_q$. We shall sometimes call such a commitment a multi-commitment, and we write $c \leftarrow \mathrm{mcom}_{pk}(m_1, \ldots, m_n)$ for a commitment to n elements.

2.2.1 Equivocable Commitment

Equivocable commitment schemes, also known as trapdoor commitments, are a special type of commitment schemes where we can generate the public key in a special way getting some equivocation information about the public key. This extra information, the equivocation key, allows us to violate the binding property. With it we are able to generate commitments that we can open as containing any message we wish, without the adversary being able to notice the deceit.

We generate the public key and the equivocation key ek using a modified key generator $\widehat{K}$. We create an equivocable commitment by running an algorithm $\widehat{\mathrm{com}}_{pk}$ on ek. This produces a commitment c and some associated equivocation information e. We open c as containing any message $m \in \mathcal{M}_{pk}$ by running $\mathrm{equiv}_{pk,ek}(c, e, m)$.

Definition 2.1 *We say the commitment scheme is* equivocable *if for any distinguisher D we have*

$$\begin{aligned} &\Pr[(pk, ek) \leftarrow \widehat{K}() : D^{\widehat{\mathcal{O}}}(pk) = 1] \\ &< \Pr[pk \leftarrow K() : D^{\mathcal{O}}(pk) = 1] + negl(k). \end{aligned}$$

Here the oracles are the following

- $\widehat{\mathcal{O}}$ *on query $m \in \mathcal{M}_{pk}$ returns (c, d), where $(c, e) \leftarrow \widehat{\mathrm{com}}_{pk}(ek)$ and $d \leftarrow \mathrm{equiv}_{pk,ek}(c, e, m)$.*
- $\mathcal{O}$ *on query $m \in \mathcal{M}_{pk}$ returns $(c, d) \leftarrow \mathrm{com}_{pk}(m)$.*

There are several ways to strengthen the notion of equivocability. We could for instance require that the distributions of public keys are identical, not just indistinguishable. Another strengthening would be to require that we can equivocate using just the equivocation information produced by $\widehat{K}$, i.e., that we do not need e at all.

2.3 Public Key Encryption

A public key cryptosystem is a triple of algorithms (K, E, D). K is the key generation algorithm producing a public key pk and a secret key sk. We associate with the public key a message space $\mathcal{M}_{pk}$, a randomizer space $\mathcal{R}_{pk}$ and a ciphertext space $\mathcal{C}_{pk}$ such that

$$\Pr[(pk, sk) \leftarrow K() : \Pr[\forall (m, r) \in \mathcal{M}_{pk} \times \mathcal{R}_{pk} : D_{sk}(E_{pk}(m; r)) = m]] \approx 1.$$

We assume there is a probability distribution associated with $\mathcal{R}_{pk}$ and write $E \leftarrow E_{pk}(m)$ for the process $r \leftarrow \mathcal{R}_{pk}; E = E_{pk}(m; r)$.

A public key cryptosystem can be seen as a commitment scheme with an extraction property. The hiding property of a commitment scheme is usually called semantic security when we talk about cryptosystems.

We will give a couple of examples of homomorphic cryptosystems with root extraction.

Example. Let q, p be primes as in the Pedersen commitment scheme. The message space is G_q, the randomizer space is $\mathbb{Z}_q$ and the ciphertext space is $G_q \times G_q$.

The key generation algorithm outputs two generators g, h for G_q such that $h = g^x \bmod p$. The public key is $pk = (p, q, g, h)$ and the secret key is $sk = x$. To encrypt a message we compute $c = (u, v) = (g^r \bmod p, h^r m \bmod p)$. The decryption algorithm outputs $m = vu^{-x} \bmod p$.

We define $\mathcal{QR}_{(p,q,g,h)} = \mathbb{Z}_q \times \mathbb{Z}_p^*/G_q \times \mathbb{Z}_p^*/G_q$ and $\mathcal{QC}_{(p,q,g,h)} = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$. Given e which is coprime to q and $c \in \mathcal{QC}_{(p,q,g,h)}, R \in \mathbb{Z}_q, M \in G_q$ so $c^e = (g^R \bmod p, h^R M \bmod p)$ we can compute m, r so $R = er \bmod q$ and $M = m^e \bmod p$. We then have $c = (ag^r \bmod p, bh^r \bmod p)$, where $b^e = a^e = 1 \bmod p$, i.e., $a, b \in \mathbb{Z}_p^*/G_q$.

Example. We take the following example of a semantically secure homomorphic cryptosystem with root extraction from [30].

Key generation: Choose $n = pq = (2p' + 1)(2q' + 1)$ as a product of two safe primes and let s be a small number. Let g be a generator of QR_n and select $h = g^{n^s x} \bmod n^{s+1}$ for x chosen at random in $\mathbb{Z}_n$.

The public key is $pk = (n, s, g, h)$, the secret key is $sk = x$.

The message space is $\mathcal{M}_{pk} = \mathbb{Z}_{n^s}$, the randomizer space is $\mathcal{R}_{pk} = \mathbb{Z} \times \{-1, 1\} \times \{-1, 1\}$ and the ciphertext space is $\mathbb{Z}_n \times \mathbb{Z}_{n^{s+1}}$.

Encryption: To encrypt m with randomness (r, b_0, b_1) compute $c = (u, v) = E_{(n,s,g,h)}(m; r, b_0, b_1) = (b_0 g^r \bmod n^{s+1}, b_1 h^r (1+n)^m \bmod n^{s+1})$. We associate the following randomness distribution with $\mathcal{R}_{pk}$: Pick r as a random $|n|/2$-bit integer and use $(r, 1, 1)$ as the randomizer when encrypting.

Decryption: To decrypt ciphertext $c \in \mathcal{C}_{(n,s,g,h)}$ compute $(1+n)^{2m} = (vu^{-x})^2 \bmod n^{s+1}$. Computing the discrete logarithm $2m$ base $1+n$ is easy [29, 70] and then we can output m. If anything fails, output $\perp$.

Semantic security of the scheme follows from the DDH assumption in QR_n and the DCR assumption [30]. It is obvious that the cryptosystem has a homomorphic property. Left is to argue that it has the root extraction property. Suppose the adversary has generated $c^e = (u^e \bmod n, v^e \bmod n^{s+1}) = (\pm g^R \bmod n, \pm h^R (1+n)^M \bmod n^{s+1})$ for some e coprime to n. Define $m = Me^{-1} \bmod n^s$. Furthermore, from $u^{2e} = g^{2R} \bmod n$ we get from the strong RSA assumption, see Lemma 7.2, that $R = er$. We conclude that $(u, v) = (\pm g^r \bmod n, \pm h^r (1+n)^m \bmod n^{s+1})$.

2.3.1 Threshold Encryption

In a (t, N)-threshold cryptosystem we have N authorities that each get a share of the secret key. We require that the semantic security property holds even against an adversary that knows t of the secret shares. On the other hand, $t+1$ of the authorities may run a multi-party computation protocol to decrypt ciphertexts.

Key generation: The key generation algorithm produces N secret shares $sk_1, \ldots, sk_N$.

Threshold decryption: To decrypt a ciphertext c the authorities compute decryption shares $ds_1, \ldots, ds_N$. Given just $t+1$ correctly computed decryption shares we can extract the plaintext.

Simulated threshold decryption: Given a ciphertext c, a plaintext m, and t secret shares we can run a simulation algorithm S to get simulated decryption shares for all the remaining authorities that makes it look like c decrypts to m.

[30] also suggest a threshold version of their cryptosystem.

Key generation: Select $n = pq = (2p'+1)(2q'+1)$ as a product of two large random safe primes of equal length. Let s be a small integer. Choose g as a generator of QR_n, for instance choose $g = 4$. Define $\Delta = N!$.

Choose $x \in \mathbb{Z}_{p'q'}$ at random and let $h = g^{4\Delta^2 n^s} \bmod n^{s+1}$. Choose at random a polynomial $f \in \mathbb{Z}_{p'q'}[X]$ of degree t such that $f(0) = x$. Let the secret shares be $s_1 = f(1), \ldots, s_N = f(N)$. Let the verification keys be $vk_1 = g^{s_1} \bmod n, \ldots, vk_N = g^{s_N} \bmod n$.

The public key is $pk = (n, s, g, h, vk_1, \ldots, vk_N)$, while the secret keys are $sk_1 = s_1, \ldots, sk_N = s_N$. The message space, randomizer space and ciphertext space is as before.

Encryption: Encryption works as in the basic scheme.

Decryption share: Given a ciphertext $(u, v) = E_{pk}(m; (r, 1, 1))$ we can compute the decryption share $ds_i = u^{2\Delta s_i} \bmod n$. In addition, the decryption share may contain a non-interactive zero-knowledge argument that ds_j has been computed correctly. This is an argument that $\log_g(vk_i) = \log_{u^{4\Delta}}(ds_i^2)$.

Combining shares: Given a ciphertext (u, v) and a set S of $t+1$ indices such that for $i \in S$ we have ds_i is a correctly computed decryption share, we can combine them to obtain the plaintext using Lagrange interpolation.

We compute

$$d = \prod_{i \in S} ds_i^{2\lambda_i^S} = u^{4\Delta^2 x} \bmod n, \text{ where } \lambda_i^S = \prod_{j \in S \setminus \{i\}} \Delta \frac{j}{j-i}.$$

We now have $(vd^{-n^s})^2 = (1+n)^{2m} \bmod n^{s+1}$. From this we can find m.

2.4 Signature

A signature scheme is a triple of algorithms (K, S, V). K is the key generation algorithm, which outputs a verification key vk and a signing key sk. We associate a message space $\mathcal{M}_{vk}$, a randomizer space $\mathcal{R}_{vk}$ and a signature space $\mathcal{S}_{vk}$ with the verification key.

We require the following correctness property for all adversaries $\mathcal{A}$,

$$\Pr[(vk, sk) \leftarrow K(); (m, r) \leftarrow \mathcal{A}(sk, vk); s \leftarrow S_{sk}(m; r) : (m, r) \notin \mathcal{M}_{vk} \times \mathcal{R}_{pk} \text{ or } V_{vk}(m, s) = 1],$$

where we demand that $\mathcal{A}$ does not output a message m that the oracle has signed.

We define security against existential forgery under various attacks by demanding that for all adversaries $\mathcal{A}$, we have

$$\Pr[(vk, sk) \leftarrow K(); (m, s) \leftarrow \mathcal{A}^{\mathcal{O}}(vk) : V_{vk}(m, s) = 0] \approx 1,$$

with the oracle and adversary further specified below.

Known/Chosen message attack: In a chosen message attack, $\mathcal{A}$ can input messages to the oracle and get back signatures on those messages. In a known message attack, $\mathcal{A}$ does not choose the messages. Instead, we have a message generator M that outputs messages. We run $(m, h) \leftarrow M(vk); s \leftarrow S_{sk}(m)$ and return (m, h, s) to $\mathcal{A}$. Here h represents some history about the message m.

Strong signatures: We can strengthen the definition of a signature scheme by allowing $\mathcal{A}$ to output a message m that the oracle has signed but just demanding that the signature on m has not been generated by the oracle.

One-time signature: We call it a one-time signature scheme if we restrict $\mathcal{A}$ to only make one query to the oracle.

Let us give an example of a strong signature scheme that is secure against chosen message attack from [77].

Key generation: Pick $n = pq$ as a product of two safe primes. Pick at random $a, g, h \in \mathrm{QR}_n$.

The verification key is $vk = (n, a, g, h)$, while the secret key is $sk = (p, q)$.

Signing: To sign a message $m \in (-2^{\ell_m}; 2^{\ell_m})$ we pick at random a $\ell_m + 2$-bit prime e. Choose at random $r \in \mathbb{Z}_e$ and compute $y = (ag^m h^r)^{e^{-1}} \bmod n$.

The signature is $s = (y, e, r)$.

Verification: To verify signature $s = (y, e, r)$ on m first check whether e is an odd $t + 1$-bit number and that $r \in \mathbb{Z}_e$. Next check that indeed $y^e = ag^m h^r \bmod n$.

2.5 Special Honest Verifier Zero-Knowledge Argument

Consider a pair of interactive algorithms (P, V) called the prover and the verifier. They may have access to a common reference string σ generated by a key generation algorithm Σ. We consider a relation R that can be evaluated in polynomial time and which may depend on σ. For an element x we call w a witness if $(\sigma, x, w) \in R$. We define a corresponding language L_σ consisting of elements that have a witness. If σ is the empty string, this is of course the standard definition of an NP-language. We write view $\leftarrow< P(x), V(y) >$ for the public view produced by P and V when interacting on inputs x and y. This view ends with V either accepting or rejecting. We sometimes shorten the notation by saying $< P(x), V(y) >= b$ if V ends by accepting $b = 1$ or rejecting $b = 0$.

Definition 2.2 (Argument) *The triple (K, P, V) is called an argument for relation R if for all adversaries $\mathcal{A}$ it has the following properties*

Completeness:

$$\Pr[\sigma \leftarrow K(); (x, w) \leftarrow \mathcal{A}(\sigma) : (\sigma, x, w) \notin R \text{ or } < P(\sigma, x, w), V(\sigma, x) >= 1] \approx 1.$$

Soundness: $\Pr[\sigma \leftarrow K(); x \leftarrow \mathcal{A}(\sigma) : x \notin L_\sigma \text{ and } < \mathcal{A}, V(\sigma, x) >= 1] \approx 0.$

We define special honest verifier zero-knowledge in the following way. We restrict the verifier to a public coin verifier that generates the challenges obliviously of the inputs it receives. We define special honest verifier zero-knowledge as the ability to simulate the view with any set of challenges produced by V, but without access to the witness.

Definition 2.3 (Special honest verifier zero-knowledge) *The quadruple (K, P, V, S) is called a special honest verifier zero-knowledge argument for R if for all adversaries $\mathcal{A}$ we have*

$$\begin{aligned} & \Pr[\sigma \leftarrow K(); (x, w, s) \leftarrow \mathcal{A}(\sigma); \text{view} \leftarrow < P(\sigma, x, w), V(\sigma, x) >: \\ & \qquad D(\sigma, x, \text{view}) = 1] \\ \approx\ & \Pr[\sigma \leftarrow K(); (x, w, s) \leftarrow \mathcal{A}(\sigma); \text{challenges} \leftarrow V(\sigma, x); \\ & \qquad \text{view} \leftarrow S(\sigma, x, \text{challenges}) : D(\sigma, x, \text{view}) = 1], \end{aligned}$$

where require that $\mathcal{A}$ does indeed produce (x, w) so $(\sigma, x, w) \in R$ and that S must use the challenges produced by V when generating the view.

The definition can be strengthened to say that we can simulate given any set of challenges, not just challenges generated by V. Most of the SHVZK arguments that we know of actually have this stronger SHVZK property.

2.5.1 Argument of Knowledge

Witness-extended emulation. The standard definition of a system for proof of knowledge does not work in our setting since we set up some public keys before making the argument of knowledge. Therefore, a cheating prover may have non-zero probability of computing some trapdoor from the public keys and use that information in the argument. In this case, it may be impossible to extract a witness. But the standard definition calls for 100% probability of extracting the witness.

We shall define an argument knowledge through what we call witness-extended emulation, inspired by the eponymous notion in [57]. This definition says that given an adversary that produces a argument with some probability, there exists

an extractor that produces a similar argument but also extracts a witness from the prover with almost the same probability. A strategy for proving security under this notion is to run the adversary and see if it produces an acceptable argument. In case it does produce an acceptable argument, we then use rewinding techniques to extract the witness.

Definition 2.4 (Witness extended emulation) *We say the argument has witness-extended emulation for all deterministic polynomial time P^* there exists an expected polynomial time emulator E such that for all adversaries $\mathcal{A}$ and distinguishers D we have*

$$\begin{aligned} & \Pr[\sigma \leftarrow K(); (x, s, h) \leftarrow \mathcal{A}(\sigma); \text{view} \leftarrow< P^*(\sigma, x, s), V(\sigma, x) >: \\ & \qquad\qquad D(\text{view}, h) = 1] \\ \approx\ & \Pr[\sigma \leftarrow K(); (x, s, h) \leftarrow \mathcal{A}(\sigma); (\text{view}, w) \leftarrow E(\sigma, x, s) : \\ & \qquad D(\text{view}, h) = 1 \text{ and (if view is accepting then } (\sigma, x, w) \in R)]. \end{aligned}$$

Some remarks on the definition are in place. We think of s as being the state of P^*, including the randomness. Then we have an argument of knowledge in the sense that from this state s and the public data σ, x the emulator should be able to extract a witness whenever P^* is able to make a convincing argument. This shows that the definition implies soundness.

We let the statement x be generated by an adversary $\mathcal{A}$. The string h is the history of this generation, i.e., it is quite possible that some auxiliary information about x is made public or otherwise known to somebody. We give this h to the distinguisher D. The emulator E must produce a correct looking view, and even with knowledge of h should it be impossible for the distinguisher to tell the difference.

A generalization of the definition would allow for a different kind of key generator $\hat{K}$ in the second probability that also generated a secret key. In that case we can make straight-line extraction of the witness and E can be independent $\mathcal{P}^*$. This is the definition in [45]. A possible restriction is to demand that E runs in polynomial time, using techniques of Barak [4] it is actually possible to make strict polynomial time extraction. If we remove the key generation step, we actually have the definition of a proof of knowledge that is presented in [4]. In comparison with the witness-extended emulation definition [57], our definition is weaker, since we do not demand that the distribution of the prover's view is identical to the distribution of the view produced by E. In the construction we use it is actually the case that the two views are perfectly indistinguishable though. The usual definition of an argument of knowledge uses a black-box simulator with oracle access to P^*. In the constructions, we provide in the following we actually prove this stronger statement of black-box emulation.

Damgård and Fujisaki [26] have also suggested a definition of argument of knowledge in the presence of a public key. Their definition is a black-box def-

inition. In the following, we will sketch a proof that our definition of witness-extended emulation implies their definition of argument of knowledge. We start by restating their definition using our own notation.

Definition 2.5 (Knowledge soundness) *A protocol (P, V) with the knowledge completeness property*[1] *is called a computationally convincing argument of knowledge for the ternary relation R with knowledge error $\kappa(k)$ and failure probability $\nu(k)$ if there is a polynomial $p(k)$ and a probabilistic machine M such that for all deterministic polynomial time provers P^* and all probabilistic polynomial time adversaries $\mathcal{A}$ we have*

$$\Pr[\sigma \leftarrow K(); (x, s) \leftarrow \mathcal{A}(\sigma) : \epsilon_{P^*}(\sigma, x, s) > \kappa(k) \text{ and the expected running time of } M^{P^*(\sigma,x,s)}(\sigma, x) \text{ is larger than } \frac{p(k)}{\epsilon_{P^*}(\sigma, x, s) - \kappa(k)}] < \nu(k),$$

where $\epsilon_{P^*}(\sigma, x, s)$ is the probability that $P^*(\sigma, x, s)$ makes an honest verifier accept on input σ, x, and we define the running time of $M^{P^*(\sigma,x,s)}(\sigma, x)$ to be ∞ if there is non-zero probability that it does not output a witness w such that $(\sigma, x, w) \in R$.

Theorem 2.1 *An argument with black-box witness-extended emulation for ternary relation R is knowledge sound with negligible knowledge error and negligible failure probability.*

Sketch of Proof. First we argue as in the proof of Theorem 4.2 from [8] that Theorem 3.7 of [8] shows that black-box witness extended emulation implies that there exists a negligible function $\delta(k)$ and an emulator E such that for all deterministic polynomial time P^* we have E^{P^*} runs in expected polynomial time and for all adversaries $\mathcal{A}$ and distinguishers D we have

$$\begin{aligned} &\Pr[\sigma \leftarrow K(); (x, s, h) \leftarrow \mathcal{A}(\sigma); \text{view} \leftarrow< P^*(\sigma, x, s), V(\sigma, x) >: \\ &\qquad\qquad D(\text{view}, h) = 1] \\ -\ &\Pr[\sigma \leftarrow K(); (x, s, h) \leftarrow \mathcal{A}(\sigma); (\text{view}, w) \leftarrow E^{P^*(\sigma,x,s)}(\sigma, x) : \\ &\quad D(\text{view}, h) = 1 \text{ and if view is accepting then } (\sigma, x, w) \in R)] < \delta(k) \end{aligned}$$

Setting D to be an algorithm that outputs 1 if and only if view shows that the verifier accepts we have

$$\begin{aligned} &\Pr[\sigma \leftarrow K(); (x, s, h) \leftarrow \mathcal{A}(\sigma) : P^*(\sigma, x, s) \text{ makes } V(\sigma, x) \text{ accept}] \\ -\ &\Pr[\sigma \leftarrow K(); (x, s, h) \leftarrow \mathcal{A}(\sigma) : (\text{view}, w) \leftarrow E^{P^*(\sigma,x,s)}(\sigma, x) : \\ &\qquad\qquad \text{view is accepting and } (\sigma, x, w) \in R)] < \delta(k) \end{aligned}$$

We ignore the h in the output of $\mathcal{A}$ since it is not used. We can also let E test itself whether the view it produces is accepting or not. If it is not accepting then

[1] See [26] for their straightforward definition of knowledge completeness.

it simply outputs $w = \perp$. We can now ignore this part of the output. Writing out the probabilities, we get

$$\sum_{\sigma,x,s} \Pr[\Sigma \leftarrow K(); (X,S) \leftarrow \mathcal{A}(\Sigma) : \Sigma = \sigma, X = x, S = s] \cdot \epsilon_{P^*}(\sigma, x, s)$$
$$- \sum_{\sigma,x,s} \Pr[\Sigma \leftarrow K(); (X,S) \leftarrow \mathcal{A}(\Sigma) : \Sigma = \sigma, X = x, S = s] \cdot$$
$$\Pr[w \leftarrow E^{P^*(\sigma,x,s)}(\sigma, x) : (\sigma, x, w) \in R)] \qquad < \delta(k)$$

Defining $\kappa(k) = \nu(k) = \sqrt{\delta(k)}$ and $\alpha(\sigma, x, s) = \Pr[\Sigma \leftarrow K(); (X,S) \leftarrow \mathcal{A}(\Sigma) : \Sigma = \sigma, X = x, S = s]$ and $\phi(\sigma, x, s) = \Pr[w \leftarrow E^{P^*(\sigma,x,s)}(\sigma, x) : (\sigma, x, w) \in R)]$ we have

$$\sum_{\sigma,x,s:\epsilon_{P^*}(\sigma,x,s)-\phi(\sigma,x,s)>\kappa(k)} \alpha(\sigma, x, s)(\epsilon_{P^*}(\sigma, x, s) - \phi(\sigma, x, s))$$
$$+ \sum_{\sigma,x,s:\epsilon_{P^*}(\sigma,x,s)-\phi(\sigma,x,s)\leq\kappa(k)} \alpha(\sigma, x, s)(\epsilon_{P^*}(\sigma, x, s) - \phi(\sigma, x, s)) < \delta(k)$$

This implies $\sum_{\sigma,x,s:\epsilon_{P^*}(\sigma,x,s)-\phi(\sigma,x,s)>\kappa(k)} \alpha(\sigma, x, s)\kappa(k) < \delta(k)$, so the probability of $\mathcal{A}$ producing σ, x, s where $\epsilon_{P^*}(\sigma, x, s) - \Pr[w \leftarrow E^{P^*(\sigma,x,s)}(\sigma, x) : (\sigma, x, w) \in R)] > \kappa(k)$ is less than $\frac{\delta(k)}{\kappa(k)} = \nu(k)$.

On the other hand, consider σ, x, s where $\epsilon_{P^*}(\sigma, x, s) - \kappa(k) \leq \Pr[w \leftarrow E^{P^*(\sigma,x,s)}(\sigma, x) : (\sigma, x, w) \in R)]$. We define $M^{P^*(\sigma,x,s)}(\sigma, x)$ to be the machine that repeatedly runs $E^{P^*(\sigma,x,s)}(\sigma, x)$ until it gets a witness w so $(\sigma, x, w) \in R$. It takes no more than an expected number of $\frac{1}{\epsilon_{P^*}(\sigma,x,s)-\kappa(k)}$ invocations to make $E^{P^*(\sigma,x,s)}(\sigma, x)$ output the witness, provided $\epsilon_{P^*}(\sigma, x, s) > \kappa(k)$. □

2.5.2 Σ-protocol

Most of the SHVZK arguments in use are 3-move arguments. The prover consists of two algorithms A, Z. He sends an initial message $(a, s) \leftarrow A(\sigma, x)$, receives a random challenge e, and answers with $z \leftarrow Z(e, s)$. Quite often, these arguments have a strong soundness property called special soundness. Special soundness says that there exists an extractor E such that for all adversaries $\mathcal{A}$ we have

Special soundness:

$$\Pr[\sigma \leftarrow K(); (x, a, e, z, e', z') \leftarrow \mathcal{A}(\sigma); w \leftarrow E(\sigma, x, a, e, z, e', z') :$$
$$V(\sigma, x, a, e, z) = 0 \text{ or } V(\sigma, x, a, e', z') = 0 \text{ or } (\sigma, x, w) \in R] \approx 1,$$

where we demand that $e \neq e'$.

A Σ-protocol is a 3-move SHVZK argument with the special soundness property. It is not hard to see that special soundness implies both soundness and witness extended emulation, so a Σ-protocol is a 3-move SHVZK argument of knowledge with witness extended emulation.

2.5.3 The Random Oracle Model

Special honest verifier zero-knowledge arguments can be made non-interactive using the Fiat-Shamir heuristic. This means that we choose the challenge e as a hash-value $e = \text{hash}(x, a, aux)$, where aux is some auxiliary input, for instance the empty string or the identity of the prover. Using a cryptographic hash-function, we hope that it messes up the input in an intractable way such that the resulting non-interactive argument is sound. Furthermore, we can also hope that e chosen this way is sufficiently garbled that the following answer z does not reveal anything important about the witness w.

To formalize this heuristic the so-called random oracle model [11] replaces the query to the hash-function with a query to a random oracle that outputs ℓ-bit strings, for some suitably large ℓ.

Definition 2.6 (Random oracle) *A random oracle $\mathcal{O}$ works the following way*

- *On input x it has not seen before, it returns a random value $y \leftarrow \{0,1\}^\ell$ and stores (x, y).*
- *On input x where a pair (x, y) is stored, it returns (x, y).*
- *In simulations, we can program the random oracle, i.e., for some x not stored, we can choose a value y and store (x, y).*

Combining a SHVZK argument with a random oracle $\mathcal{O}$ to compute the challenges yields a non-interactive zero-knowledge argument.

Chapter 3

A Note on Moving from the Random Oracle Model to the Standard Model

3.1 Introduction

Non-interactive zero-knowledge. In order to minimize interaction of zero-knowledge proofs and arguments Blum, Feldman and Micali [12] introduced a model in which the parties have access to a common reference string (CRS), sometimes called a uniform random string (URS) if it chosen with a uniform distribution, that is guaranteed to be drawn from a certain distribution. With this help, it is possible to make non-interactive zero-knowledge arguments for membership of any NP-language.

Since then, more sophisticated NIZK proofs and arguments have been introduced. Some salient properties that can be obtained are simulation soundness, non-malleability and proofs or arguments of knowledge. The existence of trapdoor permutations is sufficient to give us such NIZK arguments and proofs, but the known constructions are very inefficient.

NIZK arguments in the random oracle model. As we have seen, we can use the Fiat-Shamir heuristic to make a Σ-protocol non-interactive. We can argue in the random oracle model that such an argument has several salient properties: It is a non-interactive zero-knowledge argument. If the Σ-protocol is designed in such a way that to any challenge there is only one unique answer then the NIZK argument is simulation sound, meaning that it is sound even if the adversary has access to simulated arguments. It is a NIZK argument of knowledge, we can rewind, give the adversary new challenges and use the special soundness property to extract a witness.[1] Finally, since we have access to a random oracle we can create NIZK arguments without a CRS. Since many Σ-protocols are very efficient,

[1] See however [74] for a warning that it may cause problems that the witnesses are not straight-line extractable.

these NIZK arguments are also very efficient. Furthermore, since the Σ-protocols are often statistical zero-knowledge we get statistical NIZK arguments.

Criticism of the random oracle model. It seems strange to replace a deterministic hash-function with a random oracle. A range of papers [19, 9, 49, 20] present protocols that are secure in the random oracle model but not secure when the random oracle is replaced with any real hash-function. These constructions use the fact that we cannot predict how a random oracle may react on a given input, but we can certainly predict a deterministic function's behavior.

Nielsen [67] demonstrates that the simulator's ability to program the random oracle makes it possible to construct a non-interactive non-committing encryption protocol, yet in real world non-interactive non-committing encryption is impossible. He suggests that to be more realistic we should work in a non-programmable random oracle model.

Since the Σ-protocols are only honest verifier zero-knowledge, it may be problematic that we determine the challenge as a hash-value, which certainly is not random. Indeed, in many protocols we do not see any way to simulate such a non-interactive argument. The random oracle method may be said to give some heuristic argument of a weaker property of hiding the content, one might consider it a good heuristic argument for witness hiding for instance.

Another problem with the random oracle arises when we use rewinding techniques. This type of argument assumes not only that we can change the behavior of a deterministic function but also that we always know the input to the hash-function. In [76] such a function is called sole-samplable. [67] call this property evaluation point knowledge. Naor [64] has criticized proposals for sole-samplable function assumptions due to their non-falsifiable nature. Pass [71] suggests a method to extract witnesses without rewinding, however, this method relies heavily on evaluation point knowledge.

[47, 46] show that it is impossible to make zero-knowledge proofs or arguments in less than 3 moves if we do not have a CRS. This does not rule out that one might consider the description of the hash-function, which is hidden in the random oracle model, as the CRS and argue zero-knowledge this way. No such argument is known though.

Countering the criticism. While [19, 9, 49, 20] present protocols that warn us about the limitations of the random oracle model, the examples are artificial and not likely to cause trouble in practice.

With respect to the zero-knowledge property, we can restrict ourselves to protocols that are real zero-knowledge, not just honest verifier zero-knowledge. This way the result of applying the Fiat-Shamir heuristic is a non-interactive zero-knowledge protocol. [25, 54, 45] all suggest methods to turn Σ-protocols into protocols that really are zero-knowledge.

With respect to knowledge, a standard technique solves the problem. We let the CRS include a key for a public key cryptosystem. Then we let the prover encrypt his witness w for the statement x he wishes to prove. Instead of proving $x \in L$ directly, he can now prove that he has encrypted w for $x \in L$. This proves $x \in L$ and allows us to set up the CRS in a way such that we can extract the witness.

Barak and Pass [6] suggest a way to get a non-interactive zero-knowledge argument that is sound against uniform adversaries and has a simulator that runs in quasi-polynomial time. This circumvents the theorems of [47, 46].

Our contributions. We suggest a new method, slightly different from [25, 54] and slightly more efficient, to convert Σ-protocols into real zero-knowledge protocols. Our idea is to form the initial message a and make a trapdoor commitment c to a nonce e_1. We compute $e_2 = \text{hash}(x, a, c)$ and use $e = e_1 \oplus e_2$ as challenge. We then open the commitment and answer the challenge. To argue zero-knowledge we use the trapdoor property of the commitment and therefore we do not need to program the random oracle, neither do we need the oracle to have any special output distribution.

We do not see a way to reduce the soundness of the resulting NIZK argument to a well-known intractability assumption such as computation of discrete logarithms or factoring. On the other hand, for most concrete Σ-protocols it seems very plausible that we do have soundness. We shall therefore simply for any concrete instance make an intractability assumption that the scheme is sound. Under this assumption, we have a NIZK argument. As a heuristic argument, we can argue soundness in the random oracle model.

If we have a Σ-protocol with a single possible answer to any challenge, and use a commitment scheme that has a single way to open the commitment, then we can make the intractability assumption that the resulting non-interactive argument is a NIZK argument. Also for these assumptions we could as a heuristic argument use the random oracle model to prove unbounded zero-knowledge and unbounded simulation soundness.

With respect to an argument of knowledge, we will use the standard technique of encrypting the witness. This does cause some trouble though since we cannot argue statistical zero-knowledge then. Indeed the definition of a NIZK proof of knowledge in [34, 33] does not even allow for such a thing as a statistical zero-knowledge proof of knowledge. In our opinion this is a flaw in the definition, we therefore first redefine NIZK argument of knowledge and then suggest NIZK arguments of knowledge. Using the DDH assumption we obtain a statistical NIZK argument of knowledge in the URS model.

If one uses the CRS model, a more efficient statistical NIZK argument of knowledge can be obtained by making the [32] universally composable zero-knowledge proof non-interactive with the Fiat-Shamir heuristic. Actually, this

gives us a universally composable NIZK argument under reasonable intractability assumptions. Due to lack of time, we do not include this in the present dissertation.

We look at the question of obtaining a NIZK argument in the plain model where we do not have a CRS. We obtain an efficient non-interactive protocol with soundness holding against uniform adversaries and with a non-uniform simulator. Unlike [6] we do not allow the simulator to run in quasi-polynomial time.

3.2 From Random Oracle Zero-Knowledge to Standard Zero-Knowledge

Non-interactive zero-knowledge.

Definition 3.1 (Zero-knowledge) *(K, P, V, S_1, S_2) is an (unbounded adaptive) NIZK argument for relation R if (K, P, V) is a non-interactive argument for R and for all adversaries $\mathcal{A}$ and distinguishers D we have*

$$\begin{aligned} & \Pr[\Sigma \leftarrow K(); s \leftarrow \mathcal{A}^{P(\Sigma,\cdot,\cdot)}(\Sigma, z) : D(s) = 1] \\ \approx \; & \Pr[(\Sigma, \tau) \leftarrow S_1(); s \leftarrow \mathcal{A}^{S'(\Sigma,\tau,\cdot,\cdot)}(\Sigma) : D(s) = 1], \end{aligned}$$

where we define $S'(\Sigma, \tau, x, w) = S_2(\Sigma, \tau, x)$, and demand that $\mathcal{A}$ in both expressions only queries (x, w) so $(\Sigma, x, w) \in R$.

If the equality holds also for unbounded distinguishers D, we say that (K, P, V, S_1, S_2) is a statistical NIZK argument for R.

Let (K, A, Z, V, S) be a Σ-protocol for R and $(K_{\mathrm{com}}, \mathrm{com}, \widehat{\mathrm{com}}, \mathrm{dec}, \mathrm{equiv})$ be a statistically hiding trapdoor commitment scheme. In Figure 3.1, we present a NIZK argument (K, P, V, S_1, S_2) for R based on these primitives.

Theorem 3.1 *(K, P, V, S_1, S_2) described in Figure 3.1 is an unbounded NIZK argument for R. If the Σ-protocol is statistical zero-knowledge, then (K, P, V, S_1, S_2) is statistical zero-knowledge.*

Proof.
Completeness. Completeness is obvious.
Soundness. We make the intractability assumption that the scheme is sound. As a heuristic argument that this assumption is reasonable consider an adversary $\mathcal{A}$ that with noticeable probability produces (x, p) such that the verifier accepts and $x \notin L_\sigma$. To do so it must make queries $e_2 = \mathcal{O}(x, c, a)$ and use one of them in the argument with noticeable probability.

We will construct an algorithm $\mathcal{B}$ that can break the commitment scheme. $\mathcal{B}$ runs $\mathcal{A}$ and answers the oracle queries that $\mathcal{A}$ makes with random values. When $\mathcal{A}$ outputs the argument, $\mathcal{B}$ rewinds it to the point where it made the query and

NIZK Argument

Reference string: Generate $pk \leftarrow K_{\mathrm{com}}()$ and $\sigma \leftarrow K()$.

Return $\Sigma = (pk, \sigma)$.

Argument: Given input (x, w) so $(\sigma, x, w) \in R$

Choose at random e_1. Set $(c, d) \leftarrow \mathrm{com}_{pk}(e_1)$. Set $(a, s) \leftarrow A(\sigma, x, w)$. Set $e_2 \leftarrow \mathcal{O}(x, c, a)$. Let $e = e_1 \oplus e_2$. Set $z \leftarrow Z(s, e)$.

Return $p = (c, d, a, z)$.

Verification: Given input x and $p = (c, d, a, z)$.

Compute $e_1 \leftarrow \mathrm{dec}_{pk}(c, d)$ and $e_2 = \mathcal{O}(x, c, a)$. Set $e = e_1 \oplus e_2$.

Return $V(\sigma, x, a, e, z)$.

Simulation: S_1 runs $(pk, ek) \leftarrow \widehat{K}_{\mathrm{com}}(); \sigma \leftarrow K()$ and sets $\tau = ek$. It returns (Σ, τ).

S_2 on input (Σ, τ, x) selects e at random and simulates an argument (a, e, z) for $(\sigma, x, w) \in R$. It sets $(c, tk) \leftarrow \widehat{com}_{pk}(ek)$ and $e_2 = \mathcal{O}(x, c, a)$. Now it simulates an opening of the commitment to $e_1 = e \oplus e_2$, $d' \leftarrow \mathrm{equiv}_{pk,ek}(c, tk, e_1)$.

Return $p = (c, d', a, z)$.

Figure 3.1: NIZK Argument

gives it a different answer e_2'. With noticeable probability, $\mathcal{A}$ can also use this query to make a valid argument. However, by the special soundness $\mathcal{A}$ cannot answer two different challenges e and e' so $\mathcal{A}$ must open the commitment c in two different ways.

Unbounded zero-knowledge. We have to argue that the distinguisher cannot guess whether arguments are produced by a prover P with access to a witness, or whether they are produced by the simulator with access to the equivocation key ek for the equivocable commitment scheme.

The reference string σ is generated the same way by K and S_1. Moreover, since the commitment scheme is statistically hiding we also have statistical indistinguishability between pk in the two generations. This means that Σ as generated by the two algorithms is statistically indistinguishable.

Consider a hybrid algorithm H replacing S_2 that takes as input (Σ, ek, x, w). H selects e at random and forms an argument (a, e, z) using its knowledge of the witness w. It also forms a commitment $(c, tk) \leftarrow \widehat{\mathrm{com}}_{pk}(ek)$. It computes $e_2 = \mathcal{O}(x, c, a)$ and sets $e_1 = e \oplus e_2$. Finally, it sets $d' \leftarrow \mathrm{equiv}_{pk,ek}(c, tk, e_1)$ and returns the proof (c, d', a, z).

This hybrid experiment is statistically indistinguishable from a real argument, otherwise we would be able to distinguish a real opening of a commitment from a simulated opening of a commitment.

On the other hand, we also find the hybrid experiment to be indistinguishable from a simulation. If we could distinguish between the hybrid argument and a simulated argument then that would violate the special honest verifier zero-knowledge property of the Σ-protocol. If the Σ-protocol is statistical zero-knowledge then we also get statistical indistinguishability between the hybrid experiment and the simulation experiment. □

Simulation Soundness

Definition 3.2 (Simulation sound zero-knowledge) *(K, P, V, S_1, S_2) is an (unbounded) simulation sound NIZK argument for R if for all adversaries $\mathcal{A}$ we have*

$$\Pr[(\Sigma, \tau) \leftarrow S_1(); (x, p) \leftarrow \mathcal{A}^{S_2(\Sigma,\tau,\cdot)}(\Sigma) : p \notin Q \wedge x \notin L_\Sigma \wedge V(\Sigma, x, p) = 1] \approx 0,$$

where Q is the list of arguments given by the oracle $S_2(\Sigma, \tau, \cdot)$,

Theorem 3.2 *(K, P, V, S_1, S_2) described in Figure 3.1 is a simulation sound NIZK argument if the Σ-protocol has a unique answer to any challenge and it is infeasible to construct a commitment with two different valid decommitments, i.e., we require that the binding property holds for both the message and the decommitment. If the Σ-protocol is statistical zero-knowledge, then (K, P, V, S_1, S_2) is statistical zero-knowledge too.*

Proof. We simply make the intractability assumption that the NIZK argument is simulation sound.

To give a heuristic argument in the random oracle model imagine $\mathcal{A}$ with noticeable probability produces a new argument (c, d, a, z) for $x \in L_\sigma$ that is acceptable to the verifier.

Since we have unique opening of the commitment scheme and unique answer for the Σ-protocol we must have that c or a are new. Furthermore, c is binding even if $\mathcal{A}$ received it in one of the simulation queries, because otherwise it would be able to distinguish simulated openings from real openings.

Let us look at the oracle query (x, c, a) made by $\mathcal{A}$ and which it is going to use later for the argument. To be successful it must be able to answer a noticeable fraction of the possible e_2's that $\mathcal{O}(x, c, a)$ can produce.

This means that $e = e_1 \oplus e_2$ can attain a large number of possible values. If $x \notin L_\sigma$ then the special soundness of the Σ-protocol implies that $\mathcal{A}$ can only answer the challenge e with negligible probability. □

If the Σ-protocol does not have the property that a challenge has a unique good answer, then it is easy to modify it into one that does using a strong one-time signature as shown in [45].

3.2.1 Argument of Knowledge

Definition 3.3 (Argument of knowledge) *(K, P, V, E_1, E_2) is an argument of knowledge for R if (K, P, V) is a non-interactive argument for R, and for all adversaries $\mathcal{A}$ and distinguishers D we have*

- Reference-string indistinguishability:

$$\Pr[\Sigma \leftarrow K() : D(\Sigma) = 1] \approx \Pr[(\Sigma, \tau) \leftarrow E_1() : D(\Sigma) = 1].$$

- Witness extractability:

$$\begin{aligned} & \Pr[\Sigma \leftarrow K(); (x, p) \leftarrow \mathcal{A}(\Sigma) : V(\Sigma, x, p) = 1] \\ \approx\ & \Pr[(\Sigma, \tau) \leftarrow E_1(); (x, p) \leftarrow \mathcal{A}(\Sigma); w \leftarrow E_2(\Sigma, \tau, x, p) : (\Sigma, x, w) \in R]. \end{aligned}$$

Remark. The usual practice in the literature is to demand that the reference string generated by E_1 is statistically indistinguishable from the reference string generated by K. Using such a definition it is impossible to make a statistical NIZK argument of knowledge for a non-trivial language.

To see this, consider an unbounded distinguisher trying to tell whether it is seeing a simulated argument or a real argument. With statistically indistinguishable reference strings, this distinguisher can compute the extraction knowledge τ and then try to extract a witness from the argument. With a real argument, this extraction is possible, but in a simulated argument, it is not possible to extract a witness.

In this paper, we wish to construct statistical NIZK arguments of knowledge and therefore we deviate from the usual practice.

Extractable commitments. A standard method for arguing knowledge is to encrypt the witness and prove that the ciphertext contains a witness for $x \in L_\Sigma$. An extractor can set up the common reference string with a public key for the cryptosystem and know the corresponding secret key itself. This way a simple decryption gives the extractor a witness of the statement. We will employ this trick through the related notion of extractable commitments.

Definition 3.4 (Extractable commitment) *We say* $(\text{Egen}, \text{Ecom}, \text{Edec}, \text{ExtGen}, \text{Extract})$ *is an extractable commitment scheme if* $(\text{Egen}, \text{Ecom}, \text{Edec})$ *is a commitment scheme and we have the following*

- Key indistinguishability *For all $\mathcal{A}$ we have*

$$\Pr[pk \leftarrow \text{Egen}() : \mathcal{A}(pk) = 1] \approx \Pr[(pk, ek) \leftarrow \text{ExtGen}() : \mathcal{A}(pk) = 1].$$

- Extraction *For all $\mathcal{A}$ we have*

$$\begin{aligned} \Pr[(pk, ek) \leftarrow \text{ExtGen}(); (c, d) \leftarrow \mathcal{A}(pk); m \leftarrow \text{Extract}_{ek}(c) : \\ \text{Edec}_{pk}(c, d) \in \{\perp, m\}] \approx 1 \end{aligned}$$

Example of an extractable commitment. In the key generation phase we pick suitable primes q, p such that $q|p-1$. We will work in a group $G_q \leq \mathbb{Z}_p^*$ of order q. We pick at random elements $g_1, g_2, h_1, h_2 \in G_q$.

A commitment to $m \in G_q$ is generated by picking r_1, r_2 at random from $\mathbb{Z}_q$ and setting $c = (g_1^{r_1} g_2^{r_2} \bmod p, h_1^{r_1} h_2^{r_2} m \bmod p)$. The corresponding opening d is (m, r_1, r_2).

This commitment is statistically hiding. To see this let x_1, x_2 be such that $h_1 = g_1^{x_1} \bmod p$ and $h_2^{x_2} \bmod p$ and let y be such that $g_2 = g_1^y \bmod p$. Now, m can be described as $g_1^{\log(m)} \bmod p$. We see that given a commitment c constructed as described above and knowledge of $\log(m)$, we may choose any other $\log(m') \in \mathbb{Z}_q$ and open the commitment as $(m', r_1' = r_1 + (r_2 - r_2')y \bmod q, r_2' = r_2 + \frac{\log(m) - \log(m')}{y(x_2 - x_1)} \bmod q)$.

However, when running ExtGen we may deliberately pick h_1, h_2 such that $h_1 = g_1^x \bmod p$ and $h_2 = g_2^x \bmod p$, with the extraction key $ek = x$. It follows from the DDH assumption that this public key is indistinguishable from a real key. Furthermore, commitment to a message now corresponds to making an ElGamal encryption, so with knowledge of x it is easy to extract the message.

The protocol in Figure 3.2 is a combination of the idea of making an extractable commitment to the witness and the simulation sound NIZK argument presented earlier.

Theorem 3.3 *$(K, P, V, S_1, S_2, E_1, E_2)$ described in Figure 3.2 is a NIZK argument of knowledge for R. If the extractable commitment scheme is statistically hiding and the Σ-protocol for R is statistical zero-knowledge, then (K, P, V, S_1, S_2) is statistical zero-knowledge.*

Proof. Completeness, soundness and (statistical) zero-knowledge follows from a proof similar to the one we made for the NIZK argument and the simulation sound NIZK argument.

By the indistinguishability of public keys produced by Egen and ExtGen we see that $\mathcal{A}$ will output (x, p) that is indistinguishable from (x, p) output by $\mathcal{A}$ with a real CRS. Left is the question of E_2 being able to extract a witness for this x.

Here we make the intractability assumption that it is infeasible to produce a valid argument unless indeed C has been produced by a valid commitment to the witness.

As a heuristic argument for the reasonability of this assumption suppose the argument p is valid. We will argue that the commitment C in the argument is actually an encryption of w such that $(\sigma, x, w) \in R$. To see this consider the state of $\mathcal{A}$ when it queries (x, C, c, a) to $\mathcal{O}$. To have noticeable chance of success it must be able to use a noticeable fraction of all possible responses e_2 that $\mathcal{O}$ can return, i.e., in particular it can use two responses e_2 and e_2'. The trapdoor commitment c is binding since we only simulate openings to random values. This means that

NIZK Argument of Knowledge

Reference string: Set $pk \leftarrow K_{\text{com}}(), PK \leftarrow \text{Egen}()$ and $\sigma \leftarrow K()$.

Return $\Sigma = (pk, PK, \sigma)$.

Argument: On input (x, w) so $(\sigma, x, w) \in R$.

Set $(C, D) \leftarrow \text{Ecom}_{PK}(w)$. Pick e_1 at random and run $(c, d) \leftarrow \text{com}_{pk}(e_1)$. Let $(a, s) \leftarrow A(\sigma, (x, PK, C), (w, D))$. Let $e_2 = \mathcal{O}(x, C, c, a)$ and compute $e = e_1 \oplus e_2$. The response is $z \leftarrow Z(s, e)$.

Return $p = (C, c, d, a, z)$.

Verification: On input x and $p = (C, c, d, a, z)$.

Compute $e_1 = \text{dec}_{pk}(c, d)$ and $e_2 = \mathcal{O}(x, C, c, a, vk)$. Let $e = e_1 \oplus e_2$. Verify the argument (a, e, z) for (x, PK, C) being such that $C \in \text{Ecom}_{PK}(w)$ and $(\sigma, x, w) \in R$.

Return 1 if everything works out, else return 0.

Simulation: S_1 runs $(pk, ek) \leftarrow \widehat{K}_{\text{com}}()$ and the rest of the key generation protocol but returns in addition to Σ also $\tau = ek$.

S_2 on input x and τ has to simulate an argument for $x \in L_\sigma$. First, it sets $(C, D) \leftarrow \text{Ecom}_{PK}(0)$ and $(c, tk) \leftarrow \widehat{com}_{pk}(ek)$. Selecting e at random it now simulates an argument (a, e, z) for (x, PK, C) being such that $C \in \text{Ecom}(w)$, where $(\sigma, x, w) \in R$. Having done this, it may compute $e_2 = \mathcal{O}(x, C, c, a)$ and $e_1 = e \oplus e_2$. Finally, it computes $d' \leftarrow \text{equiv}_{pk,ek}(c, tk, e_1)$.

It returns $p = (C, c, d', a, z)$.

Extraction: E_1 generates $(pk, ek) \leftarrow \widehat{K}_{\text{com}}(), (PK, EK) \leftarrow \text{ExtGen}(), \sigma \leftarrow K()$. It sets $\Sigma = (pk, PK, \sigma)$ and $\tau = ek$.

We give Σ to $\mathcal{A}$ that produces an output (x, p), where $p = (C, c, d, a, z)$. E_2 computes $w \leftarrow \text{Extract}_{EK}(C)$.

It returns (x, w, s).

Figure 3.2: NIZK Argument of Knowledge

$\mathcal{A}$ can answer two challenges e and e'. By the special soundness property of the Σ-protocol we have (x, PK, C) such that $C \in \text{Ecom}_{PK}(w)$, where $(\sigma, x, w) \in R$.

We picked PK as a public key for a commitment scheme where it is possible to extract the content. Therefore, when E_2 computes $w = \text{Extract}_{EK}(C)$ it gets a witness for $x \in L_\sigma$. □

Uniform random string. If we set this argument up with the Pedersen commitment scheme and the extractable commitment scheme suggested above, and σ generated by K is either empty or uniformly random, then we can set up the entire argument in the uniform random string model.

Non-malleability. Non-malleability [73, 33] is a strong security definition that in particular implies knowledge of the witness. If the Σ-protocol for commitment to a witness has unique answer and the commitment scheme has unique decommitment then the scheme above is actually non-malleable. We do not prove that here.

3.3 Bare Bones Non-uniform Zero-Knowledge

Problems with the bare bones model. In the random oracle model, we may make NIZK arguments without having a CRS. However, as proved by [46] it is impossible to make NIZK arguments for languages outside BPP in real life without a CRS. Sometimes less is needed, for instance, witness hiding may be sufficient and this may be obtainable in the bare bones model. In this section, we will try, however, to see whether it is possible to make non-interactive arguments that have some sort of "zero-knowledge" property.

Trying to overcome the lack of a CRS, we may try the following. Suppose we use some well-known number, which seemingly does not have anything to do with our problem at hand as common reference string. We could for instance pick the binary expansion of π or e, or we could expand a known number with a pseudorandom number generator hoping that it does not have a "malicious structure".

The problem is that we since we do not pick the CRS according to our own desire we cannot plant trapdoors in it; we work in a model with *no programmability at all*. It might be that the string already has trapdoors in it, but we just do not know how to find them.

Non-interactive uniform soundness non-uniform zero-knowledge. Finding the trapdoors is only a problem when running a uniform algorithm. Indeed non-uniform algorithms may simply have the trapdoor information built into them. This gap of uniformity and non-uniformity may be used to our advantage. More specifically, we will hint at the possibility of making arguments that are sound against uniform adversaries and which have a non-uniform simulator.

Using ideas in the paper, this is easy to implement. Consider the unbounded zero-knowledge argument in Figure 3.1 with the Pedersen commitment. This argument needs a short CRS consisting of (q, p, g, h) where $q|p-1$ are primes, $h = g^x \bmod p$ and g is an element of order q in $\mathbb{Z}_p^*$. Making some reasonable intractability assumption about, say, the SHA-1 hash-function, we may construct

some interesting zero-knowledge arguments on this basis. Consider now the binary expansion of π. It is easy to come up with an encoding such that the first bits of π can be interpreted as elements q, p, g, h of suitable lengths depending on the security parameter. It is reasonable to assume that a uniform algorithm cannot find x such that $h = g^x \bmod p$. On the other hand, it is also easy to see that a non-uniform algorithm with x hardwired into it may prove anything it wants to.

Definition 3.5 (NUISNUZK argument) (K, P, V, S_1, S_2) *is a non-interactive uniform soundness non-uniform zero-knowledge argument for* R *if it is complete, sound against* uniform *adversaries and is zero-knowledge against* non-uniform *adversaries.* S_1 *is allowed to be non-uniform.*

Caveat. To evaluate the potential usefulness of this notion we first note that care must be taken when employing NIUSNUZK arguments. Assume that we use the bits of π to give us elements $(q, p, g, h,)$ for a Pedersen commitment as suggested above. Consider now a problem (u, v) where we want to prove that there exists some $r \in Z_q$ and m belonging to some set S, such that $(u, v) = (g^r \bmod p, h^r m \bmod p)$. Since there exists a non-uniform simulator that knows x it is easy for this simulator to decrypt the ciphertext and obtain m. Therefore, as part of the non-interactive argument the prover may send m in clear. However, intuitively this seems like a violation of the essence of zero-knowledge. The lesson is that we must take care not to use this method on problems closely related to the fixed CRS.

Example of using NIUSNUZK arguments. On the other hand, there may be scenarios where the notion is useful. Consider as a toy example the Naor-Yung [65] construction of IND-CCA1 secure encryption. In this construction, the public key consists of two randomly chosen keys for a semantically secure cryptosystem and a CRS. To encrypt a value the sender makes two ciphertexts corresponding to the two public keys and gives a NIZK argument that both of them hold the same plaintext. A hybrid argument is used to argue that this cryptosystem is secure against lunch-break attacks.

Suppose we let the public keys be selected at random as ElGamal keys $(g, h_1 = g^{x_1 \bmod p})$ and $(g, h_2 = g^{x_2} \bmod p)$ where x_1, x_2 are the secret keys known only to the receiver. We encrypt a message m by encrypting it under both keys and use a NIUSNUZK argument to prove that both ciphertexts contain the same message. Under the DDH assumption against *non-uniform* adversaries, we can argue that no *uniform* adversary can break this cryptosystem under a lunch-break attack.

Chapter 4

Validating Votes

4.1 Voting Based on Homomorphic Encryption

Election parameters M, L, N. In this section, we describe efficient voting schemes based on homomorphic encryption. We will throughout the section assume that we have a group of voters that can choose between L candidates, which may include choices such as a blank vote or an invalid vote. A drawback of this type of election scheme is that the number of candidates is fixed, we do not allow write-in votes. We denote by M a strict upper bound on the number of votes any candidate can receive. In particular, if each voter has one vote then M is a strict upper bound on the number of voters. A third parameter characterizing the elections is the number of votes the voter can cast, denoted by N. We will suggest four types of schemes, depending on the size of N. First we consider the simple case where $N = 1$, second we consider the case where N is small but may be larger than 1. Third, we consider approval voting where the voter may vote for as many candidates as he wishes. Finally, we consider elections where voters have many votes to cast, as it is the case in shareholder elections for instance.

Encoding votes. In the introduction we sketched how to base voting protocols on homomorphic encryption. Let us offer some more details. The basic ingredient is a homomorphic threshold public-key cryptosystem. We will generate a public key pk for this cryptosystem, and the secret key sk will be threshold secret shared between a group of n authorities.

We assume that the message space $\mathcal{M}_{pk}$ is on the form $\mathbb{Z}_n$. We require that n does not have prime factors smaller than 2^t and that $M^L \leq n$. We represent candidates with numbers $0, \ldots, L-1$ and encode a vote on candidate j as M^j.[1] Summing many such encodings gives us an M-addic representation of the result, $\sum_{j=0}^{L-1} v_j M^j$, where v_j is the number of votes on candidate j.

Representing votes this way, it is straightforward to encrypt a vote on candi-

[1] As an alternative Lipmaa [58] has suggested to encode votes as Lucas numbers.

date j as $E_{pk}(M^j)$. Having received many such encrypted votes we may by the homomorphic property of the cryptosystem multiply all the ciphertexts and get a new ciphertext $E = E_{pk}(\sum_{j=0}^{L-1} v_j M^j)$. We threshold decrypt this ciphertext and now it is straightforward to extract the result from the plaintext.

Setup and parameters. In this chapter, we will make use of a homomorphic threshold cryptosystem. We assume that $\mathcal{M}_{pk} = \mathbb{Z}_n$ and $\mathcal{R}_{pk} = \mathbb{Z}$. The latter assumption is purely out of notational convenience, there would be no problem in using a cryptosystem where the randomness is some finite group, for instance to use threshold Paillier encryption.

We also make use of a homomorphic integer commitment scheme. In addition, we assume that the randomizer space is $\mathbb{Z}$. Actually, we only know of homomorphic commitment schemes with randomizer space $\mathbb{Z}$, but it would not be a problem to use a finite randomizer space if such a thing exists.

We define the following parameters. $\ell_M = 2\ell_m = \lceil L \log(M) \rceil$ is the maximal bit-length of a vote. We assume that the distribution of the randomizer space of the cryptosystem is to pick a random ℓ_R-bit randomizer. Similarly for integer commitments we pick a random ℓ_r-bit number as randomizer. We will also need a couple of extra security parameters. The verifier will pick a random ℓ_e-bit number e. Furthermore, we need a security parameter ℓ_s, such that for any value a we have that $a + r$ and r are indistinguishable, where r is a random $|a| + \ell_s$-bit number. It is quite common to let $\ell_e = 160$, the output-length of some cryptographic hash-functions. We may suggest $\ell_s = 40$.

Proving correctness of encrypted votes. As already mentioned in the introduction we need to ensure that only valid votes are submitted. We therefore demand that each voter submits a zero-knowledge argument of the ciphertext containing a valid vote. Damgård and Jurik [29, 31] and Baudron et al. [7] propose Σ-protocols that achieve this goal, and using the Fiat-Shamir heuristic, the arguments can be made non-interactive.

Unfortunately the zero-knowledge arguments in [31] and [7] are not very efficient. They both prove that the vote is correct by encrypting each bit of j, and then proving that the vote can be written as $M^j = M^{j_0} \cdots M^{j_k}$, where $k = \lfloor \log(L-1) \rfloor$.

Efficiency is further hampered if we demand that the voting scheme must be universally verifiable. In a universally verifiable voting scheme, the encrypted votes as well as the zero-knowledge arguments are public, making it possible for anybody to verify that the votes have been tallied correctly. Supposing that we want the votes to be secret for many years to come, this forces us to use a very large security parameter. Each encryption is therefore an expensive operation.

Lipmaa, Asokan and Niemi [59] suggest a more efficient method for proving the correctness of a vote using integer commitments. Namely, commit to the

vote and prove that the ciphertext and the commitment have the same content. Now, if we can prove knowledge of the integer commitment holding a valid vote, then we are done. The advantage of this technique is that commitments can be statistically hiding, therefore they do not reveal anything about the vote no matter how sophisticated cryptanalytic techniques may be in the future. Since the zero-knowledge arguments are checked right away, we don't need a large security parameter, we only have to worry about present day cryptanalytic techniques for soundness. From a situation where the zero-knowledge arguments were very heavy to carry out, we can therefore move to a situation where the expensive part of casting a vote is to encrypt it. The rest of this section is dedicated to demonstrating some very efficient zero-knowledge arguments for correct votes using integer commitments.

4.1.1 Single Vote

[59] suggests setting M to be a prime. So we have a commitment to M^j and have to prove that j is on the correct form. To do so we also commit to M^{L-j-1} and using a multiplication argument we show that the product of the contents of the two commitments is M^{L-1}. Since we are working over the integers this shows that the vote is on the form $\pm M^j$, with $0 \leq j < L$. What is left is to prove that the vote is a positive integer. This can be done using the argument from Boudot [13] or the method suggested in [58].

We improve the efficiency of this zero-knowledge argument through the observation that a square is always positive. Instead of selecting M as a prime we can select it as the square of a prime, $M = p^2$. We still commit to M^j, but we also commit to p^j and p^{L-j-1}. Then we show that the contents of the two latter commitment when multiplied give p^{L-1}. The verifier concludes that the second commitment must therefore contain $\pm p^j$, with $0 \leq j < L$. The second step is to show that the content of the first commitment is the square of the content of the second commitment. It is on the form $M^j = p^{2j}$, with $0 \leq j < L$. This way we do not have to perform the expensive part of the [59]-argument, namely to show that a committed number is non-negative.

It turns out that there are several ways to implement the zero-knowledge argument. For completeness, we give three different protocols that differ in the detail. Depending on a variety of parameters one or another of them may be the most efficient choice for a particular election.

Theorem 4.1 *The protocol in Figure 4.1 is a 3-move public coin special honest verifier zero-knowledge argument of knowledge with witness extended emulation for E encrypting a ciphertext on the form M^j where $0 \leq j < L$. If the commitment scheme is statistically hiding, then the argument is statistical honest verifier zero-knowledge.*

Zero-Knowledge Argument for Correctness of a Vote I

Common input: Ciphertext E and public keys.

Private input: $0 \leq j < L$ and $R \in \{0,1\}^{\ell_R}$ such that $E = E_{pk}(M^j; R)$.

Initial message: Choose $r_a, r_b \in_R \{0,1\}^{\ell_r}$ and set $c_a = \mathrm{com}(p^j; r_a)$ and $c_b = \mathrm{com}(p^{L-j-1}; r_b)$. Choose $d \in_R \{0,1\}^{\ell_m+\ell_e+\ell_s}$ and choose $D \in_R \{0,1\}^{\ell_M+\ell_e+\ell_s}$. Choose $r_d \in_R \{0,1\}^{\ell_r+\ell_e+\ell_s}$ and $r_{db}, r_D \in_R \{0,1\}^{\ell_m+\ell_r+\ell_e+2\ell_s}$. Set $c_d = \mathrm{com}(d; r_d), c_{db} = \mathrm{com}(-dp^{L-j-1}; r_{db}), c_D = \mathrm{com}(D - dp^j; r_D)$. Choose $R_D \in_R \{0,1\}^{\ell_R+\ell_e+\ell_s}$ and set $E_D = E_{pk}(D; R_D)$.

Send $c_a, c_b, c_d, c_{db}, c_D, E_D$ to the verifier.

Challenge: Select $e \in_R \{0,1\}^{\ell_e}$.

Answer: Send $f = ep^j + d, z_a = er_a + r_d, z_b = fr_b + r_{db}, z_D = fr_a + r_D, F = eM^j + D$ and $Z = eR + R_D$ to the verifier.

Verification: Verify that $c_a^e c_d = \mathrm{com}(f; z_a), c_b^f c_{db} = \mathrm{com}(ep^{L-1}; z_b), c_a^f c_D = \mathrm{com}(F; z_D)$ and $E^e E_D = E_{pk}(F; Z)$.

Figure 4.1: Single-Vote Argument I.

Proof. It is easy to see that we have a 3-move public coin protocol. It is straightforward to verify that the protocol is complete. Left is to argue special honest verifier zero-knowledge and witness extended emulation.

Special honest verifier zero-knowledge. Given a challenge e we simulate an argument in the following way. Set $c_a \leftarrow \mathrm{com}(0), c_b \leftarrow \mathrm{com}(0)$. Pick f, z_a, z_b, z_D, Z at random. Set $c_d = \mathrm{com}(f; z_a)c_a^{-e}, c_{db} = \mathrm{com}(ep^{L-1}; z_b)c_b^{-f}, c_D = \mathrm{com}(F; z_D)c_a^{-f}$ and $E_D = E_{pk}(F; Z)E^{-e}$.

To see that the simulated argument is indistinguishable from a real argument, consider the following hybrid argument. We proceed as in the simulation expect we set $c_a \leftarrow \mathrm{com}(p^j), c_b \leftarrow \mathrm{com}(p^{L-j-1})$. The hybrid argument is statistically indistinguishable from a real argument, the only difference is in the order and manner we compute the elements. On the other hand, the hiding property of the commitment scheme implies that the hybrid argument is indistinguishable from a simulated argument. Moreover, if the commitment scheme is statistically hiding then we have statistical indistinguishability between the hybrid argument and the simulated argument.

Witness-extended emulation. We start by running the adversarial prover P^* with a random challenge e. This is the argument we output. If it is not an accepting argument then we halt. However, if it is an accepting argument then we must extract a witness. To do so we run P^* with randomly chosen challenges until we obtain a new accepting argument. The obtainment of these two arguments takes expected polynomial time.

Consider the two arguments $c_a, c_b, c_d, c_{db}, c_D, E_D, e, f, z_a, z_b, z_D, Z$ and $c_a, c_b, c_d, c_{db}, c_D, E_D, e', f', z'_a, z'_b, z'_D, Z'$. With overwhelming probability, we have $e \neq e'$. This gives us $E^{e-e'} = E_{pk}(F - F'; Z - Z')$. Using the root extraction property of the cryptosystem we can therefore with overwhelming probability extract m, R such that $E = E_{pk}(m; R)$. If $m = M^j$ then it is easy to compute j and output the witness.

Left is to argue that indeed $m = M^j$ for some $0 \leq j < L$. We have $c_a^e c_d = \mathrm{com}(f; z_a)$ and $c_a^{e'} c_d = \mathrm{com}(f'; z_a)$. This gives us $c_a^{e-e'} = \mathrm{com}(f - f'; z_a - z'_a)$ and by the root extraction property we then have $e - e' | f - f'$. Let $a = \frac{f-f'}{e-e'}$. Now $c_b^{f-f'} = \mathrm{com}((e-e')p^{L-1}; z_b - z'_b)$ and the root extraction property gives us $a | p^{L-1}$. Next we look at $\mathrm{com}(F - F'; z_D - z'_D) = c_a^{f-f'} = c_a^{(e-e')a} = \mathrm{com}(a^2(e - e'); a(z_a - z'_a))$, so $F - F' = (e - e')a^2$. Finally, note that $E^{e-e'} = E_{pk}((e - e')a^2; Z - Z')$ so the root extraction property gives us that the plaintext of E is a^2. This shows that with overwhelming probability the witness we extracted is on the right form. □

Theorem 4.2 *The protocol in Figure 4.2 is a 3-move public coin special honest verifier zero-knowledge argument of knowledge with witness-extended emulation for E encrypting a valid vote. If the commitments are statistically hiding then the argument is statistical honest verifier zero-knowledge.*

Proof. It is obvious that we have a 3-move public coin protocol. It is straightforward to verify that the protocol is complete. Left is to argue special honest verifier zero-knowledge and witness-extended emulation.

Special honest verifier zero-knowledge. Given challenge e, we can simulate an argument as follows. Choose $c \leftarrow \mathrm{mcom}(0, 0)$. Pick f, f_b, z, z_D, Z at random. Set $c_d = \mathrm{mcom}(f, f_b; z)c^{-e}$ and $c_D = \mathrm{mcom}(F, ep^{L-1}; z_D)c^{-f}$. Set $E_D = E_{pk}(F; Z)E^{-e}$.

To argue that the simulated argument is indistinguishable from a real argument, consider the hybrid argument where we pick $c \leftarrow \mathrm{mcom}(p^j, p^{L-j-1})$ and otherwise follow the simulation above. The hybrid argument is statistically indistinguishable from a real argument. On the other hand, the hiding property of the commitment scheme implies that the hybrid argument is indistinguishable from a simulated argument. If the commitment scheme is statistically hiding, then the hybrid argument is actually statistically indistinguishable from a simulated argument.

Zero-Knowledge Argument for Correctness of a Vote II

Common input: Ciphertext E and public keys.

Private input: $0 \leq j < L$ and $R \in \{0,1\}^{\ell_R}$ such that $E = E_{pk}(M^j; R)$.

Initial message: Choose $r \in_R \{0,1\}^{\ell_r}$ and set $c = \mathrm{mcom}(p^j, p^{L-j-1}; r)$. Choose $d, d_b \in_R \{0,1\}^{\ell_m+\ell_e+\ell_s}$ and choose $D \in_R \{0,1\}^{\ell_M+\ell_e+\ell_s}$. Choose $r_d \in_R \{0,1\}^{\ell_r+\ell_e+\ell_s}$ and $r_D \in_R \{0,1\}^{\ell_m+\ell_r+\ell_e+s\ell_s}$. Set $c_d = \mathrm{mcom}(d, d_b; r_d), c_D = \mathrm{mcom}(D - dp^j, -dp^{L-j-1}; r_D)$. Choose $R_D \in_R \{0,1\}^{\ell_R+\ell_e+\ell_s}$ and set $E_D = E_{pk}(D; R_D)$.

Send c, c_d, c_D, E_D to the verifier.

Challenge: Select $e \in_R \{0,1\}^{\ell_e}$.

Answer: Send $f = ep^j + d, f_b = ep^{L-j-1} + d_b, z = er + r_d, z_D = fr + r_D, F = eM^j + D$ and $Z = eR + R_D$ to the verifier.

Verification: Verify that $c^e c_d = \mathrm{mcom}(f, f_b; z), c^f c_D = \mathrm{mcom}(F, ep^{L-1}; z_D)$ and $E^e E_D = E_{pk}(F; Z)$.

Figure 4.2: Single-Vote Argument II.

Witness-extended emulation. The emulator runs P^* with a randomly chosen challenge e. It outputs the resulting argument. If the argument is not acceptable then we can stop here, otherwise we have to extract a witness j, R. To find the witness we run P^* again on randomly chosen challenges until an accepting argument occurs. This takes expected polynomial time. Call the two acceptable arguments $c, c_d, c_D, E_D, e, f, f_b, z, z_D, F, Z$ and $c, c_d, c_D, E_D, e', f', f'_b, z', z'_D, Z'$. We have $E_{pk}(F - F'; Z - Z') = E^{e-e'}$. With overwhelming probability $e \neq e'$ and we can use the root extraction property of the cryptosystem to extract (m, R) such that $E = E_{pk}(m; R)$. Now we just need to show that $m = M^j$ for some $0 \leq j < L$.

We have $c^{e-e'} = \mathrm{mcom}(f - f', f_b - f'_b; z - z')$, implying $e - e' | f - f'$ and $e - e' | f_b - f'_b$. Define $a = \frac{f-f'}{e-e'}$ and $b = \frac{f_b - f'_b}{e-e'}$. Now $\mathrm{mcom}(F - F'; (e-e')p^{L-1}; z_D - z'_D) = c^{f-f'} = c^{(e-e')a} = \mathrm{mcom}((e-e')a^2, (e-e')ab; a(z_d - z'_d))$ so $ab = p^{L-1}$ and $(e-e')a^2 = F - F'$. This implies that $a^2 = M^j$ for some $0 \leq j < L$. Finally, we have $E^{e-e'} = E_{pk}((e-e')a^2; Z - Z')$ which by the root extraction property shows that E has a^2 as plaintext. □

Theorem 4.3 *The protocol in Figure 4.3 is a 3-move public coin special honest verifier zero-knowledge argument of knowledge with witness-extended emulation for E containing a valid vote. If the commitment scheme is statistically hiding,*

Zero-Knowledge Argument for Correctness of a Vote III

Common input: Ciphertext E and public keys.

Private input: $0 \leq j < L$ and $R \in \{0,1\}^{\ell_R}$ such that $E = E_{pk}(M^j; R)$.

Initial message: Choose $d, d_b \in_R \{0,1\}^{\ell_m+\ell_e+\ell_s}$ and $D \in_R \{0,1\}^{\ell_M+\ell_e+\ell_s}$. Choose $r \in_R \{0,1\}^{\ell_r}$ at random from $\mathcal{R}_K$ and set $c = \mathrm{mcom}(p^j, p^{L-j-1}, p^j d_b + p^{L-j-1} d, 2p^j d - D; r)$. Choose $r_d \in_R \{0,1\}^{\ell_r+\ell_e+\ell_s}$. Set $c_d = \mathrm{mcom}(d, d_b, dd_b, d^2; r_d)$. Choose $R_D \in_R \{0,1\}^{\ell_R+\ell_e+\ell_s}$ and set $E_D = E_{pk}(D; R_D)$.

Send c, c_d, E_D to the verifier.

Challenge: Select $e \in_R \{0,1\}^{\ell_e}$.

Answer: Send $f = ep^j + d$, $f_b = ep^{L-j-1} + d_b$, $f_\delta = e(p^j d_b + p^{L-j-1} d) + dd_b$, $f_\Delta = e(2p^j d - D) + d^2$, $z = er + r_d$, $F = eM^j + D$ and $Z = eR + R_D$ to the verifier.

Verification: Verify that $c^e c_d = \mathrm{mcom}(f, f_b, f_\delta, f_\Delta; z)$ and $E^e E_D = E_{pk}(F; Z)$. Check that $f f_b = e^2 p^{L-1} + f_\delta$ and $f^2 = eF + f_\Delta$.

Figure 4.3: Single-Vote Argument III.

then the argument is statistical honest verifier zero-knowledge.

Proof. Obviously, we have a 3-move public coin protocol. It is straightforward to see that the protocol is complete. It remains to argue special honest verifier zero-knowledge and witness-extended emulation.

Special honest verifier zero-knowledge. Given challenge e, let us describe how to simulate the argument. We set $c \leftarrow \mathrm{mcom}(0,0,0,0)$. We choose f, f_b, z, F, Z at random. From that we can compute $f_\delta = f f_b - e^2 p^{L-1}$ and $f_\Delta = f^2 - eF$. We set $c_d = \mathrm{mcom}(f, f_b, f_\delta, f_\Delta; z)c^{-e}$ and $E_D = E_{pk}(F; Z)E^{-e}$.

To argue that the simulated argument is indistinguishable from a real argument, consider the following hybrid argument. We proceed as in the simulation except we compute $d = f - ep^j, d_b = f_b - ep^{L-j-1}, D = F - eM^j$ and set $c \leftarrow \mathrm{mcom}(p^j, p^{L-j-1}, p^j d_b + p^{L-j-1} d, 2p^j d - D)$. The hybrid argument is statistically indistinguishable from a real argument. On the other hand, the hiding property of the commitment scheme implies that the hybrid argument is indistinguishable from a simulated argument. If the commitment scheme is statistically hiding then the hybrid argument is statistically indistinguishable from a simulated argument.

Witness-extended emulation. The emulator starts by running P^* with a randomly chosen challenge e. We output this argument. If the argument is acceptable then we have to extract a witness (j, R). To do so we run P^* with randomly chosen challenges until we get an acceptable argument. This takes expected polynomial time. Call the two acceptable arguments $c, c_d, E_D, e, f, f_b, f_\delta, f_\Delta, z, F, Z$ and $c, c_d, E_D, e', f', f'_b, f'_\delta, f'_\Delta, F', Z'$. Therefore $E^{e-e'} = E_{pk}(F - F'; Z - Z')$. With overwhelming probability we have $e \neq e'$ and we can use the root extraction property of the cryptosystem to get (m, R) such that $E = E_{pk}(m, R)$.

It remains to argue that m is on the form M^j for some $0 \leq j < L$. From $c^{e-e'} = \mathrm{mcom}(f - f', f_b - f'_b, f_\delta - f'_\delta, f_\Delta - f'_\Delta; z - z')$ we get elements a, b, δ, Δ, r so $c = \mathrm{mcom}(a, b, \delta, \Delta; r)$. From $c_d = \mathrm{mcom}(f, f_b, f_\delta, f_\Delta; z)c^{-e}$, we then get values $d, d_b, d_\delta, d_\Delta, r_d$ so that $f = ea + d, f_b = eb + d_b, f_\delta = e\delta + d_\delta, f_\Delta = e\Delta + d_\Delta, z = er + r_d$. These values constitute a quasi-opening of c_d. Similarly we have $F = em + D, Z = eR + R_D$, where $E_D = E_{pk}(D; R_D)$.

Consider now an adversarial P^* with noticeable chance of producing an acceptable argument. Given a random challenge it must use $f = ea + d, f_b = eb + d_b, f_\delta = e\delta + d_\delta, f_\Delta = e\Delta + d_\Delta, z = er + r_d, F = em + D, Z = eR + R_D$. The equation $ff_b = e^2p^{L-1} + f_\delta$ then becomes $e^2ab + e(ad_b + bd) + dd_b = e^2p^{L-1} + e\delta + d_\delta$. By the few polynomial roots assumption we conclude that to have noticeable chance of succeeding in this we have that the two polynomials are identical, in particular $ab = p^{L-1}$. This shows that $a | p^{L-1}$. The equation $f^2 = eF + f_\Delta$ becomes $e^2a^2 + e(2ad) + d^2 = e^2m + e(D + \Delta) + d_\Delta$. Again to have noticeable chance of success it must be the case that $m = a^2$. □

Optimizing the Arguments

Size of arguments. All the arguments have size $\mathcal{O}(k + L \log M)$. To be more precise when making estimates let us look at a concrete example. We assume there are a hundred candidates, $L = 100$, and less than a million voters $M = 1,000,000$. We use Damgaard-Fujisaki [26] integer commitments with a security parameter of 1500 bits. To encrypt votes we use the ElGamal-style cryptosystem from [28] with a security parameter of 3000 bits. Using these parameters, we have public keys of size around 2kB for all three schemes. The votes, consisting of ciphertext E and the argument, have sizes around 3kB. Some improvements can be made, for instance in scheme III we can compute f_δ, f_Δ from f, f_b, F and therefore we do not need to transmit it. Furthermore, when we make the argument non-interactive by computing e as a hash-value, we could send just e and then compute $c_d = \mathrm{mcom}(f, f_b, f_\delta, f_\Delta; z)c^{-e}$ and $E_D = E_{pk}(F; Z)E^{-e}$. Using the hash-function, we can check whether it is the correct e that has been sent. Therefore, if memory was short we could avoid transmitting c_d and E_D. This latter technique might make verification more cumbersome from a computational point of view, though. For realistic settings, a total of 5kB, for transmitting the vote makes it possible

to use even low bandwidth devices. There is therefore not really much need to make efforts in terms of reducing the size of the arguments and we do not see this as a distinguishing feature when having to select, which of the schemes to use.

The voter's computation. On first glance, we might consider scheme II the best since it uses one less exponentiation than the other two. When we take into account the length of the exponents, the picture becomes less clear. All three schemes share the same computational complexity when it comes to computing the ciphertexts E and E_D. With the parameters above it corresponds to making four exponentiations of roughly 2000 bits over a modulus of size 6000 bits (which is a square of an RSA-modulus of 3000 bits). The three schemes differ in the commitments but the sum of the lengths of the exponents is approximately the same, 14000 bits - 17000 bits. If we do not have access to multi-exponentiation techniques, for instance if we use the Java bigInteger class to make the computations, then the three schemes are pretty similar. The tendency is that the larger the election, the less does the randomness matter and the better does scheme I become. On the other hand, the smaller the election the more does randomness matter and the better does scheme III become. If we use multi-exponentiation techniques then scheme III has additional advantages over schemes I and II. The expensive computation, however, is the computation of the ciphertexts and therefore in most scenarios it does not make much difference to the voter whether he must use scheme I, II or III.

Verifying a vote. Using the cryptosystem from [28] we have ciphertexts $E = (U, V)$ and $E_D = (U_D, V_D)$ in $\mathbb{Z}^*_{n^2} \times \mathbb{Z}^*_{n^2}$. Instead of verifying $U^e U_D = G^Z \bmod n^2, V^e V_D = H^Z(1+n)^F \bmod n^2$ we can pick s as small random exponent, say 80 bits, and verify that $(U^s V)^e U_D^s V_D = (G^s H)^Z (1+n)^F \bmod n^2$. This way we almost halve the computation needed for these verifications.

We can use similar randomization techniques on the commitments to reduce the computational effort needed in schemes I and II. Using integer commitments over modulus n' we can replace the three equations in scheme I with choosing s_1, s_2 at random and checking whether $(c_a^{s_1} c_b^{s_2})^f (c_a^e c_d)^{s_1} c_{db}^{s_2} c_D = \mathrm{com}(s_1 f + s_2 e p^{L-1} + s_3 F; s_1 z_a + s_2 z_b + z_D) \bmod n'$. This almost halves the computational effort needed.

Fixed-base comb techniques [62] make it possible to compute exponentiations of G, H and g, h with great efficiency. In comparison, it can be expensive to compute exponentiations of new elements, like the exponentiation to f. Therefore, even though scheme III uses more exponentiations than the randomized versions of schemes I and II, it may be more efficient. Actually, using fixed-base comb techniques the most expensive operation may be the large exponentiation of a variable base in schemes I and II.

Verifying many votes. When we have many votes to verify we can use randomization techniques to even greater effect. Consider for instance scheme III with votes $\{(E_1, c_1, c_{d,1}, E_{D,1}, e_1, f_1, f_{b,1}, f_{\delta,1}, f_{\Delta,1}, z_1, F_1, Z_1)\}_{i=1}^n$. We can pick $s_1, \ldots, s_n$ at random and verify that $\prod_{i=1}^n (E_i^{e_i} E_{D,i})^{s_i} = E_{pk}(\sum_{i=1}^n s_i F_i; \sum_{i=1}^n s_i Z_i)$ and $\prod_{i=1}^n (c_i^{e_i} c_{d,i})^{s_i} = \mathrm{mcom}(\sum_{i=1}^n s_i f_i, \sum_{i=1}^n s_i f_{b,i}, \sum_{i=1}^n s_i f_{\delta,i}, \sum_{i=1}^n s_i f_{\Delta,i}; \sum_{i=1}^n s_i z_i)$. This way we share the expenses for encrypting and committing between all the verifications. The price for each vote to be verified is now only a few exponentiations of small size, and even this price can be reduced using multi-exponentiation techniques.

4.1.2 Multi-way Vote

In some elections, voters are allowed to vote multiple times, say N times. It may be a requirement that they use all their votes on different candidates, or alternatively they may be permitted to spend several votes on the same candidates. We will present a protocol for the former case, it is easy to modify the protocol into one that handles the latter case.

The voter will encode his vote as $M^{j_1} + \cdots + M^{j_N}$, where $0 \leq j_1 < \ldots < j_N < L$. We can use the method from before, encrypt the vote and make an integer commitment to it as well. Since we can prove that the integer commitment and the ciphertext contain the same plaintext, we are now left with the task of showing that the integer commitment contains a legal vote.

We will again choose $M = p^2$, where p is a prime. As before the strategy is to show that the committed vote is on the form $p^{2j_1} + \cdots + p^{2j_N}$. Note that $pp^{j_1}p^{j_2-j_1-1} = p^{j_2}, \ldots, pp^{j_N}p^{L-j_N-1} = p^L$. We can therefore commit to $p^{j_1}, p^{j_2-j_1-1}, \ldots, p^{j_N}, p^{L-j_N-1}$, and then prove the relationship that the content of the first commitment, multiplied with p and multiplied with the contents of the second commitment gives the contents of the third commitment, etc. This proves that the contents of the penultimate commitment contains something that when multiplied with p divides, p^L. I.e., a commitment on the form $\pm p^{j_N}$ where $0 \leq j_N < L$. It furthermore shows that the fourth-last commitment contains $\pm p^{j_{N-1}}$ where $0 \leq j_{N-1} < j_N$, etc.

This idea shows how to prove the correctness of a multi-way vote. What is left is to cut away redundant commitments. We will suggest a protocol where we commit to all the $p^{j_1}, p^{j_2-j_1-1}, \ldots, p^{j_N}, p^{L-j_N-1}$ at the same time and prove in a direct way that the encrypted vote is one the correct form. In comparison with the protocol in [28] this allows us to save several exponentiations.

Theorem 4.4 *The protocol in Figure 4.4 is a 3-move public coin special honest verifier zero-knowledge argument of knowledge with witness-extended emulation for E encrypting a correctly formed multi-way vote. If the commitment scheme is statistically hiding then the argument is statistical honest verifier zero-knowledge.*

Zero-Knowledge Argument for Correctness of a Multi-way Vote

Common input: Ciphertext E and public keys.

Private input: $0 \leq j_1 < \cdots < j_N < L$ and $R \in \{0,1\}^{\ell_R}$ such that $E = E_{pk}(\sum_{i=1}^{N} M^{j_i}; R)$.

Initial message: Choose $D \in_R \{0,1\}^{\ell_M+\ell_e+\ell_s}$ and $d_1, \ldots, d_N, \hat{d}_1, \ldots, \hat{d}_N \in_R \{0,1\}^{\ell_m+\ell_e+\ell_s}$. Define $j_{N+1} = L$ and $d_{N+1} = 0$.
$c = \mathrm{mcom}(p^{j_1}, p^{j_2-j_1-1}, d_1 p^{j_2-j_1} + \hat{d}_1 p^{j_1+1} - d_2, \ldots, p^{j_N}, p^{L-j_N-1}, d_N p^{L-j_N} + \hat{d}_N p^{j_N+1} - d_{N+1}, 2d_1 p^{j_1} + \cdots + 2d_N p^{j_N} - D; r)$.

$c_d = \mathrm{mcom}(d_1, \hat{d}_1, d_1\hat{d}_1, \ldots, d_N, \hat{d}_N, d_N\hat{d}_N, d_1^2 + \cdots + d_N^2; r_d)$.

Choose $R_D \in_R \{0,1\}^{\ell_R+\ell_e+\ell_s}$ and set $E_D = E_{pk}(D; R_D)$.

Send c, c_d, E_D to the verifier.

Challenge: Select $e \in_R \{0,1\}^{\ell_e}$.

Answer: Send $f_1 = ep^{j_1} + d_1$, $\hat{f}_1 = ep^{j_2-j_1-1} + \hat{d}_1$, $f_{\delta_1} = e(d_1 p^{j_2-j_1} + \hat{d}_1 p^{j_1+1} - d_2) + pd_j\hat{d}_j, \ldots, f_N = ep^{j_N} + d_N$, $\hat{f}_N = ep^{L-j_N-1} + \hat{d}_N$, $f_{\delta_N} = e(d_N p^{L-j_N} + \hat{d}_N p^{j_N+1} - d_{N+1}) + pd_N\hat{d}_N$, $f_\Delta = e(2d_1 p^{j_1} + \cdots + 2d_N p^{j_N} - D) + d_1^2 + \cdots + d_N^2$, $z = er + r_d$, $F = e\sum_{i=1}^{N} M^{j_i} + D$ and $Z = eR + R_D$ to the verifier.

Verification: Define $f_{N+1} = ep^L$.
Verify that $c^e c_d = \mathrm{mcom}(f_1, \hat{f}_1, f_{\delta_1}, \ldots, f_{\delta_N}, f_\Delta; z)$ and $E^e E_D = E_{pk}(F; Z)$. Check for $j = 1, \ldots, N$ that $pf_j\hat{f}_j = ef_{j+1} + f_{\delta_j}$. Check that $f_1^2 + \cdots + f_N^2 = eF + f_\Delta$.

Figure 4.4: Multi-way Vote Argument.

Proof. Obviously, we have a 3-move public coin protocol. It is straightforward to verify that it is complete. Left is to argue special honest verifier zero-knowledge and witness-extended emulation.

Special honest verifier zero-knowledge. Given a challenge e we make a simulation like this. We pick at random $f_1, \hat{f}_1, \ldots, f_N, \hat{f}_N, z, F, Z$. We compute $f_{\delta_1}, \ldots, f_{\delta_N}$ such that for all $j = 1, \ldots, N$ we have $pf_j\hat{f}_j = ef_{j+1} + f_{\delta_j}$. We pick f_Δ such that $f_\Delta = f_1^2 + \cdots + f_N^2 - eF$. We set $c \leftarrow \mathrm{mcom}(0, \ldots, 0)$. Then we compute $c_d = \mathrm{mcom}(f_1, \hat{f}_1, f_{\delta_1}, \ldots, f_{\delta_N}, f_\Delta; z)c^{-e}$ and $E_D = E_{pk}(F; Z)E^{-e}$.

To argue that the simulated argument is indistinguishable from a real argument, consider the following hybrid argument. We proceed as in the simulation but define $d_1 = f_1 - ep^{j_1}, \hat{d}_1 = \hat{f}_1 - ep^{j_2-j_1-1}, \ldots, d_N = f_N - ep^{j_N}, \hat{d}_N = \hat{f}_N - ep^{j_N}$ and $D = F - e\sum_{i=1}^{N} M^{j_i}$. We set $c \leftarrow \mathrm{mcom}(p^{j_1}, p^{j_2-j_1}, d_1 p^{j_2-j_1} +$

$\hat{d}_1 p^{j_1+1}, \ldots, p^{j_N}, p^{L-j_N-1}, d_N p^{L-j_N} + \hat{d}_N p^{j_N+1}, 2d_1 p^{j_1} + \cdots + 2d_N p^{j_N} - D)$. The rest of the hybrid argument is carried out as in the simulation.

The hybrid argument is statistically indistinguishable from a real argument, all that is changed is the order in which elements are chosen. On the other hand, the only difference from a simulated argument is in the computation of the commitment c. The commitment scheme's hiding property shows that the hybrid argument is statistically indistinguishable from a simulated argument. Moreover, if the commitment scheme is statistically hiding then the hybrid argument is statistically indistinguishable from the simulated argument.

Witness-extended emulation. The emulator starts by running P^* on a random challenge e. It outputs the resulting argument. If the argument is acceptable then we have to extract a witness. To do so we feed P^* with random challenges until we get a new acceptable argument. This takes expected polynomial time. Let us call the two acceptable arguments $c, c_d, E_D, e, f_1, \hat{f}_1, f_{\delta_1}, \ldots, f_N, \hat{f}_N, f_{\delta_N}, f_\Delta, z, F, Z$ and $c, c_d, E_D, e', f'_1, \hat{f}'_1, f'_{\delta_1}, \ldots, f'_N, \hat{f}'_N, f'_{\delta_N}, f'_\Delta, z', F', Z'$. This gives us $E^{e-e'} = E_{pk}(F - F'; Z - Z')$. With overwhelming probability we have $e \neq e'$ and using the root extraction property of the cryptosystem we can try to extract (m, R) so $E = E_{pk}(m; R)$.

It remains to argue that m is a message on the form $\sum_{i=1}^{N} M^{j_i}$ for $0 \leq j_1 < \cdots j_N < L$. From $c^{e-e'} = \mathrm{mcom}(f_1 - f'_1, \hat{f}_1 - \hat{f}'_1, f_{\delta_1} - f'_{\delta_1}, \ldots, f_N - f'_N, \hat{f}_N - \hat{f}'_N, f_{\delta_N} - f'_{\delta_N}, f_\Delta - f'_\Delta; z - z')$ we get a quasi-opening $(a_1, b_1, \delta_1, \ldots, a_N, b_N, \delta_N, \Delta, r)$ of c. From $c_d = \mathrm{mcom}(f_1, \hat{f}_1, f_{\delta_1}, \ldots, f_{\delta_N}, f_\Delta; z)c^{-e}$ we then get a quasi-opening $(d_1, \hat{d}_1, d_{\delta_1}, \ldots, d_N, \hat{d}_N, d_{\delta_N}, d_\Delta, r_d)$ of c_d. Additionally, define $D = F - em$, $R_D = Z - eR$ and we have $E_D = E_{pk}(D; R_D)$.

Consider now P^* having noticeable probability of making an acceptable argument. It must use $f_1 = ea_1 + d_1, \hat{f}_1 = eb_1 + \hat{d}_1, f_{\delta_1} = e\delta_1 + d_{\delta_1}, \ldots, f_N = ea_N + d_N, \hat{f}_N = eb_N + \hat{d}_N, f_{\delta_N} = e\delta_N + d_{\delta_N}, f_\Delta = e\Delta + d_\Delta$ and $F = em + D$. For $j = 1, \ldots, N$ we have equations $pf_j\hat{f}_j = ef_{j+1} + f_{\delta_j}$ implying $e^2 pa_j b_j + ep(d_j b_j + \hat{d}_j a_j) + pd_j\hat{d}_j = e^2 a_{j+1} + e\delta_j + d_{\delta_j}$. Unless $a_j b_j p = a_{j+1}$ this has negligible chance of being true. Since $a_{j+1} = p^L$ we see that $pa_N | p^L, pa_{N-1} | a_N, \ldots, pa_1 | a_2$. In other words, there exists $0 \leq j_1 < \ldots < j_N < L$ such that $a_1 = \pm p^{j_1}, \ldots, a_N = \pm p^{j_N}$. From the last equation $f_1^2 + \cdots + f_N^2 = eF + f_\Delta$ we see that $e^2(a_1^2 + \cdots + a_N^2) + e(2a_1 d_1 + \cdots + 2a_N d_N) + (d_1^2 + \cdots d_N^2) = e^2 m + e(D + \Delta) + d_\Delta$. P^* can only have noticeable chance of success in constructing a valid argument if indeed $m = a_1^2 + \cdots + a_N^2 = p_1^{2j_1} + \cdots + p^{2j_N} = M^{j_1} + \cdots + M^{j_N}$ with $0 \leq j_1 < \cdots < j_N < L$. □

Potential speedup with many votes. Instead of using $M = p^2$, use $M = p^4$, or even higher even powers. This way we reduce the computational effort in

proving that $0 \leq j_1 < \cdots < L$ but increase computational effort where we prove that the plaintext contains sums of quartics. If each voter has many votes to spend, this trade-off may be good.

4.1.3 Approval Voting

Increasing the number of votes a voter may cast, we come to approval voting where the voter can vote for as many different candidates as he likes. The advantage of this kind of voting system is that the voter does not waste votes by selecting all the candidates he likes. In instant-runoff voting it may be foolish business to cast a vote for the candidate you prefer if the chances of the candidate winning the election are slim. Note, in this kind of election the number N_i of votes cast by voter i may be anywhere between 0 and L.

Define $\delta_j = 1$ if the voter wishes to vote for candidate j and $\delta_j = 0$ if he does not. The vote is then on the form $\sum_{j=0}^{L-1} \delta_j M^j$. Again we can form a commitment to $\sum_{j=0}^{L-1} \delta_j M^j$, and then we have to prove that the commitment is on the right form. To do so we can let $c_0, \ldots, c_{L-1}$ be commitments to $\delta_0, \ldots, \delta_{L-1}$. We then have a bunch of commitment and have to prove that all of them contain 0 or 1. The method we will use for this purpose is to show for each of them that the content equals the square of the contents. Since the only solution to the equation $x^2 = x$ are 0 and 1 this gives us the required result. The homomorphic property then gives us that $\prod_{j=0}^{L-1} c_j^{M^j}$ contains $\sum_{j=0}^{L-1} \delta_j M^j$. By committing to all the δ_j's in one commitment instead of committing to them all at once we may save some computational effort.

Theorem 4.5 *The protocol in Figure 4.5 is a 3-move public coin special honest verifier zero-knowledge argument of knowledge with witness extended emulation for E containing a correctly formed approval vote. If the commitment scheme is statistically hiding then the argument is statistical honest verifier zero-knowledge.*

Proof. Obviously, we have a 3-move public coin protocol. It is straightforward to verify completeness. Left is to argue that it is special honest verifier zero-knowledge and has witness-extended emulation.

Special honest verifier zero-knowledge. Given challenge e, we simulate an argument as follows. We pick $f_0, \ldots, f_{L-1}, z, Z$ at random and compute $f = \sum_{j=0}^{L-1} f_j^2 - e\sum_{j=0}^{L-1} f_j$ and $F = \sum_{j=0}^{L-1} f_j M^j$. We set $c \leftarrow \mathrm{mcom}(0, \ldots, 0)$. We compute $c_d = \mathrm{mcom}(f_0, \ldots, f_{L-1}, f; z)c^{-e}$ and $E_D = E_{pk}(F; Z)E^{-e}$.

To argue that the simulated argument is indistinguishable from a real argument, consider the following hybrid argument. We generate $f_0, \ldots, f_{L-1}$ at random and define $d_0 = f_0 - e\delta_0, \ldots, d_{L-1} = f_{L-1} - e\delta_{L-1}$. We set $c \leftarrow \mathrm{mcom}(\delta_0, \ldots, \delta_{L-1}, (2\delta_0 - 1)d_0 + \cdots + (2\delta_{L-1} - 1)d_{L-1})$. The rest of the hybrid argument is generated as in the simulation.

Zero-Knowledge Argument for Correctness of an Approval Vote

Common input: Ciphertext E and public keys.

Private input: $\delta_0, \ldots, \delta_{L-1} \in \{0,1\}$ and $R \in_R \{0,1\}^{\ell_R}$ such that $E = E_{pk}(\sum_{j=0}^{L-1} \delta_j M^j; R)$.

Initial message: Choose $d_0, \ldots, d_{L-1} \in_R \{0,1\}^{\ell_e+\ell_s}$. Choose $r \in_R \{0,1\}^{\ell_r}$ at random. Set $c = \mathrm{mcom}(\delta_0, \ldots, \delta_{L-1}, (2\delta_0 - 1)d_0 + \cdots + (2\delta_{L-1} - 1)d_{L-1}; r)$. Choose $r_d \in_R \{0,1\}^{\ell_r+\ell_e+\ell_s}$ and set $c_d = \mathrm{mcom}(d_0, \ldots, d_{L-1}, d_0^2 + \cdots + d_{L-1}^2; r_d)$. Define $D = d_0 M^0 + \cdots + d_{L-1} M^{L-1}$. Choose $R_D \in_R \{0,1\}^{\ell_R+\ell_e+\ell_s}$. Set $E_D = E_{pk}(D; R_D)$.

Send c, c_d, E_D to the verifier.

Challenge: Select $e \in_R \{0,1\}^{\ell_e}$.

Answer: Send $f_0 = e\delta_0 + d_0, \ldots, f_{L-1} = e\delta_{L-1} + d_{L-1}$, $f = e((2\delta_0 - 1)d_0 + \cdots + (2\delta_{L-1} - 1)d_{L-1}) + d_0^2 + \cdots + d_{L-1}^2$, $z = er + r_d$, $F = e\sum_{j=0}^{L-1} \delta_j M^j + D$ and $Z = eR + R_D$ to the verifier.

Verification: Verify that $c^e c_d = \mathrm{mcom}(f_0, \ldots, f_{L-1}, f; z)$ and $E^e E_D = E_{pk}(F; Z)$. Check that $\sum_{j=0}^{L-1} f_j^2 = f + e\sum_{j=0}^{L-1} f_j$ and $\sum_{j=0}^{L-1} f_j M^j = F$.

Figure 4.5: Approval Vote Argument.

The hybrid argument is statistically indistinguishable from a real argument. On the other hand, the only difference from a simulated argument is in the generation of the commitment c. By the hiding property of the commitment scheme, we get indistinguishability between the hybrid argument and the simulated argument. Moreover, if the commitment scheme is statistically hiding then the hybrid argument is statistically indistinguishable from a simulated argument.

Witness-extended emulation. The emulator runs P^* on a random challenge e. It outputs the resulting argument. If the argument is not acceptable then we are done, otherwise we must extract a witness $(\delta_0, \ldots, \delta_{L-1}, R)$. We run P^* with random challenges until we get an acceptable argument. This takes expected polynomial time. Call the two acceptable arguments $c, c_d, E_D, e, f_0, \ldots, f_{L-1}, f, z, F, Z$ and $c, c_d, E_D, e', f_0', \ldots, f_{L-1}', f', z', F', Z'$. We have $E^{e-e'} = E_{pk}(F - F'; Z - Z')$. With overwhelming probability $e \neq e'$ and we can use the root extraction property of the cryptosystem to find (m, R) so $E = E_{pk}(m; R)$.

The remaining question is whether $m = \sum_{j=0}^{L-1} \delta_j M^j$ for $\delta_0, \ldots, \delta_{L-1} \in \{0, 1\}$. From $c^{e-e'} = \text{mcom}(f_0 - f'_0, \ldots, f_{L-1} - f'_{L-1}, f - f'; z - z')$ we can extract a quasi-opening $(\delta_0, \ldots, \delta_{L-1}, \Delta, r)$ of c. Then $d_0 = f_0 - e\delta_0, \ldots, d_{L-1} = f_{L-1} - e\delta_{L-1}, d_\Delta = f - e\Delta, r_d = z - er$ constitute a quasi-opening of c_d. We also have $D = F - em, R_D = Z - eR$ satisfying $E_D = E_{pk}(D; R_D)$.

P^* with noticeable chance of producing acceptable arguments must therefore on a random challenge e use $f_0 = e\delta_0 + d_0, \ldots, f_{L-1} = e\delta_{L-1} + d_{L-1}, f = e\Delta + d_\Delta, F = em + D$. The first equation $\sum_{j=0}^{L-1} f_j^2 = f + e\sum_{j=0}^{L-1} f_j$ can be rewritten as $e^2(\delta_0^2 + \cdots + \delta_{L-1}^2) + e(2\delta_0 d_0 + \cdots + 2\delta_{L-1} d_{L-1}) + (d_0^2 + \cdots + d_{L-1}^2) = e^2(\delta_0 + \ldots + \delta_{L-1}) + e(\Delta + d_0 + \ldots + d_{L-1}) + d_\Delta$. Only if the two polynomials in e are equal can P^* have noticeable chance of success, so $\sum_{j=0}^{L-1}(\delta_j^2 - \delta_j) = 0$. This implies $\delta_j \in \{0, 1\}$ for $j = 0, \ldots, L-1$. The second equation says $\sum_{j=0}^{L-1} f_j M^j = F$, which means $e\sum_{j=0}^{L-1} \delta_j M^j + \sum_{j=0}^{L-1} d_j M^j = em + D$. Again, to have noticeable chance of success in creating an acceptable argument we conclude that $m = \sum_{j=0}^{L-1} \delta_j M^j$. □

Multi-way vote with large N. It is easy to add another condition so we can verify that $\sum_{j=0}^{L-1} \delta_j = N$ for some known N. It is therefore also possible to use the above argument as a multi-way vote argument.

In comparison with the multi-way vote argument, the f_j's are of small size, while the multi-way vote argument may use very large exponents when we have many candidates. The multi-way vote argument is thus suitable when N is very small in comparison with L, while for large N it is better to use a variation of the above argument.

4.1.4 Divisible Voting

Left is the case where each voter may have an enormous amount of votes. Consider for instance a company where each share gives the right to cast a vote. It would be highly impractical for the shareholders to use the before-mentioned techniques since it would force them to make a huge number of encryptions. Rather we want to be able to prove in a direct manner that the ciphertext contains a vote on the form $\sum_{j=0}^{L-1} v_j M^j$, where v_j is the number of votes on candidate j.

In [53] they call this divisible voting and they suggest a protocol. We will suggest an alternative protocol here that takes full advantage of integer commitments. In comparison with [53] we save a factor $\log(N_i)$ in complexity, and we carry out the protocol using integer commitments instead of encryptions giving a further substantial saving.

The idea is very simple. Commit to $v_0, \ldots, v_{L-1}$. We then prove that all these elements are positive, and furthermore we prove that their sum is N_i, where N_i is the number of votes that voter i may cast.

To prove that an element is positive we could use Boudot's argument that essentially writes $2^T v_j$ as the sum of a square and a small number. With T large enough, this proves that v_j must be positive. Or we could use [59]'s argument where v_j is proven to a sum of four squares, using Lagrange's theorem that any non-negative integer can be written as the sum of four squares.

We offer a variation over the latter idea. It is a well-known fact from number theory that the only numbers that cannot be written as the sum of three squares are on the form $4^n(8k+7)$. It therefore suffices to show that $4v_j+1$ is a sum of three squares, which would imply that v_j is non-negative. Rabin and Shallit [72] offer an efficient algorithm for finding three such squares. This algorithm does rely on some non-standard assumptions though. However, in our case the numbers are still fairly small for a computer, i.e., we cannot imagine elections where voters would have more than, say, a million votes. It is therefore possible to find the three squares using brute force.

Theorem 4.6 *The protocol above is a 3-move public coin special honest verifier zero-knowledge argument of knowledge with witness-extended emulation for a ciphertext containing a specified number of votes. If the commitment scheme is statistically hiding then the argument is statistical honest verifier zero-knowledge.*

Proof. Obviously, we have a 3-move protocol that is public coin. It is straightforward to verify completeness. Left is to argue that the protocol is special honest verifier zero-knowledge and has witness-extended emulation.

Special honest verifier zero-knowledge. Given challenge e, we have to simulate an argument. Pick $f_0, f_{x_0}, f_{y_0}, f_{z_0}, \ldots, f_{L-1}, f_{x_{L-1}}, f_{y_{L-1}}, f_{z_{L-1}}, z, Z$ at random. For $j = 0, \ldots, L-1$ compute $f_{\Delta_j} = 4(f_j + e) - f_{x_j}^2 - f_{y_j}^2 - f_{z_j}^2$. Set $F = \sum_{j=0}^{L-1} f_j M^j$. Set $c \leftarrow \mathrm{mcom}(0, \ldots, 0)$. Let $d = f_0 + \cdots + f_{L-1} - eN$. Set $c_d = \mathrm{mcom}(f_0, \ldots, f_{\Delta_{L-1}}; z)c^{-e}$ and $E_D = E_{pk}(F; Z)E^{-e}$.

To argue that the simulated argument is indistinguishable from a real argument, we consider the following hybrid argument. For $j = 0, \ldots, L-1$ we find x_j, y_j, z_j so $4v_j + 1 = x_j^2 + y_j^2 + z_j^2$ in the same manner as we do in a real argument. For $j = 0, \ldots, L-1$ we pick $f_j, f_{x_j}, f_{y_j}, f_{z_j}$ at random and define $d_j = f_j - ev_j, d_{x_j} = f_{x_j} - ex_j, d_{y_j} = f_{y_j} - ey_j, d_{z_j} = f_{z_j} - ez_j$. We now compute $c \leftarrow \mathrm{mcom}(v_0, x_0, y_0, z_0, 4d_0 - 2x_0 d_{x_0} - 2y_0 d_{y_0} - 2z_0 d_{z_0}, \ldots, v_{L-1}, x_{L-1}, y_{L-1}, z_{L-1}, 4d_{L-1} - 2x_{L-1} d_{x_{L-1}} - 2y_{L-1} d_{y_{L-1}} - 2z_{L-1} d_{z_{L-1}})$. We then proceed as when creating a simulated argument.

The hybrid argument is statistically indistinguishable from a real argument, since the only difference is in the order in which we pick the elements. On the other hand, the only difference between the hybrid argument and the simulated argument is in the formation of c. By the hiding property of the commitment scheme we therefore get that the hybrid argument is indistinguishable from a simulated argument. If the commitment scheme is statistically hiding then the hybrid argument and the simulated argument are statistically indistinguishable.

Zero-Knowledge Argument for Correctness of a Divisible Vote

Common input: Ciphertext E, a number of votes N and public keys.

Private input: $0 \leq v_0, \ldots, v_{L-1}$ and $R \in \{0,1\}^{\ell_R}$ such that $N = \sum_{j=0}^{L-1} v_j$ and $E = E_{pk}(\sum_{j=0}^{L-1} v_j M^j; R)$.

Initial message: Find $x_0, y_0, z_0, \ldots, x_{L-1}, y_{L-1}, z_{L-1}$ such that $4v_j + 1 = x_j^2 + y_j^2 + z_j^2$ for all j. Choose $d_j, d_{x_j}, d_{y_j}, d_{z_j} \in_R \{0,1\}^{\log(N)+2+\ell_e+\ell_s}$. Choose $r \in_R \{0,1\}^{\ell_r}$ and set $c = \mathrm{mcom}(v_0, x_0, y_0, z_0, 4d_0 - 2x_0 d_{x_0} - 2y_0 d_{y_0} - 2z_0 d_{z_0}, \ldots, v_{L-1}, x_{L-1}, y_{L-1}, z_{L-1}, 4d_{L-1} - 2x_{L-1} d_{x_{L-1}} - 2y_{L-1} d_{y_{L-1}} - 2z_{L-1} d_{z_{L-1}}; r)$. Choose $r_d \in_R \{0,1\}^{\ell_r}$ and set $c_d = \mathrm{mcom}(d_0, d_{x_0}, d_{y_0}, d_{z_0}, d_{x_0}^2 + d_{y_0}^2 + d_{z_0}^2, \ldots, d_{L-1}, d_{x_{L-1}}, d_{y_{L-1}}, d_{z_{L-1}}, d_{x_{L-1}}^2, d_{y_{L-1}}^2, d_{z_{L-1}}^2; r_d)$. Define $D = d_0 M^0 + \cdots + d_{L-1} M^{L-1}$. Choose $R_D \in_R \{0,1\}^{\ell_R+\ell_e+\ell_s}$. Set $E_D = E_{pk}(D; R_D)$. Define $d = d_0 + \cdots + d_{L-1}$.

Send c, c_d, E_D, d to the verifier.

Challenge: Select $e \in_R \{0,1\}^{\ell_e}$.

Answer: Send $f_j = ev_j + d_j, f_{x_j} = ex_j + d_{x_j}, f_{y_j} = ey_j + d_{y_j}, f_{z_j} = ez_j + d_{z_j}, f_{\delta_j} = e(4d_j - 2x_j d_{x_j} - 2y_j d_{y_j} - 2z_j d_{z_j}) - d_{x_j}^2 - d_{y_j}^2 - d_{z_j}^2$ to the verifier for all j. Send also $z = er + r_d, F = e\sum_{j=0}^{L-1} v_j M^j + D, Z = eR + R_D$ to the verifier.

Verification: Verify that $c^e c_d = \mathrm{mcom}(f_0, \ldots, f_{\Delta_{L-1}}; z)$ and $E^e E_D = E_{pk}(F; Z)$. Check for all j that $e(4f_j + e) = f_{x_j}^2 + f_{y_j}^2 + f_{z_j}^2 + f_{\Delta_j}$. Check that $\sum_{j=0}^{L-1} f_j = eN + d$. Check that $\sum_{j=0}^{L-1} f_j M^j = F$.

Figure 4.6: Divisible Vote Argument.

Witness-extended emulation. We first run P^* on a random challenge e. We output the resulting argument. If the argument is acceptable then we have to compute a witness $(v_0, \ldots, v_{L-1}, R)$. We run P^* on randomly chosen challenges e' until we get another acceptable argument. This takes expected polynomial time. We call the two accepting arguments $c, c_d, E_D, d, e, f_0, f_{x_0}, f_{y_0}, f_{z_0}, f_{\Delta_0}, \ldots, f_{L-1}, f_{x_{L-1}}, f_{y_{L-1}}, f_{z_{L-1}}, f_{\Delta_{L-1}}, z, F, Z$ and $c, c_d, E_D, d, e', f'_0, f'_{x_0}, f'_{y_0}, f'_{z_0}, f'_{\Delta_0}, \ldots, f'_{L-1}, f'_{x_{L-1}}, f'_{y_{L-1}}, f'_{z_{L-1}}, f'_{\Delta_{L-1}}, z', F', Z'$. Since the arguments are acceptable we have $E^{e-e'} = E_{pk}(F - F'; Z - Z')$. With overwhelming probability we have $e \neq e'$ and we can use the root extraction property of the cryptosystem to extract (m, R) so $E = E_{pk}(m; R)$.

Now we need to argue that $m = \sum_{j=0}^{L-1} v_j M^j$ and $0 \leq v_0, \ldots, v_{L-1}$ and $N = \sum_{j=0}^{L-1} v_j$. From $c^{e-e'} = \mathrm{mcom}(f_0 - f'_0, \ldots, f_{\Delta_{L-1}} - f'_{\Delta_{L-1}}; z - z')$ we can use the root extraction property to get a quasi-opening $(v_0, x_0, y_0, z_0, \Delta_0, \ldots, v_{L-1}, x_{L-1}, y_{L-1}, z_{L-1}, \Delta_{L-1}, r)$ of c. Defining $d_0 = f_0 - ev_0, \ldots, d_{\Delta_{L-1}} = f_{\Delta_{L-1}} - e\Delta_{L-1}, r_d = z - er$ we get a quasi-opening of c_d. Setting $D = F - em, R_D = Z - eR$ we get $E_D = E_{pk}(D; R_D)$.

For any randomly chosen challenge e that P^* has noticeable chance of creating a successful argument we therefore have $f_0 = ev_0 + d_0, \ldots, f_{\Delta_{L-1}} = e\Delta_{L-1} + d_{\Delta_{L-1}}$ and $F = em + D$. Consider first the equation $eN + d = \sum_{j=0}^{L-1} f_j = e\sum_{j=0}^{L-1} v_j + \sum_{j=0}^{L-1} d_j$. With overwhelming probability over e this does not old unless $N = \sum_{j=0}^{L-1} v_j$ as we wanted. Consider then for $j = 0, \ldots, L-1$ the equation $e(4f_j + e) = f_{x_j}^2 + f_{y_j}^2 + f_{z_j}^2 + f_{\Delta_j}$, which can be rewritten as $e^2(4v_j + 1) + ed_j = e^2(x_j^2 + y_j^2 + z_j^2) + e(2x_j d_{x_j} + 2y_j d_{y_j} + 2z_j d_{z_j} + \Delta) + d_{x_j}^2 + d_{y_j}^2 + d_{z_j}^2 + d_\Delta$. By the few polynomial roots assumption this has only negligible chance of being satisfied unless the two polynomials in e are identical, in particular $4v_j + 1 = x_j^2 + y_j^2 + z_j^2$. This shows that for $j = 0, \ldots, L-1$ we have $0 \leq v_j$. Finally, we have $\sum_{j=0}^{L-1} f_j M^j = F$, which can be rewritten as $e\sum_{j=0}^{L-1} v_j M^j + \sum_{j=0}^{L-1} d_j M^j = em + D$. Again, with overwhelming probability over e this can only happen if $m = \sum_{j=0}^{L-1} v_j M^j$. $\square$

Chapter 5

Verifying Shuffles

5.1 Introduction

Shuffles and mix-nets. A shuffle of ciphertexts $e_1, \ldots, e_n$ is a new set of ciphertexts $E_1, \ldots, E_n$ so that both sets of ciphertexts have the same plaintexts. If we are working with a homomorphic cryptosystem with encryption algorithm $E_{pk}(\cdot)$, we may shuffle $e_1, \ldots, e_n$ by selecting a permutation $\pi \in \Sigma_n$ and setting $E_1 \leftarrow e_{\pi(1)}E_{pk}(0), \ldots, E_n \leftarrow e_{\pi(n)}E_{pk}(0)$. If the cryptosystem is semantically secure, nothing is revealed about the permutation by publishing $E_1, \ldots, E_n$. On the other hand, this also means that nobody else can verify directly whether we shuffled correctly. Our goal in this chapter is to construct an efficient honest verifier zero-knowledge argument for the correctness of a shuffle.

Shuffles can be used to build mix-nets. A mix-net is a multi-party protocol to shuffle elements so that neither of the parties knows the permutation linking the input and output. To shuffle ciphertexts we may let the parties one after another make a shuffle with a random permutation and prove correctness of it. The arguments of correctness allow us to catch any cheater, and if at least one party is honest, it is impossible to link the input and output. In this role, shuffling constitute an important building block in anonymization protocols and voting schemes.

Another type of mix-net is a decrypting mix-net. Here the input is a set of ciphertexts, and the output is the corresponding plaintexts in random order. We must of course assume that the parties performing the mix have a secret sharing of the decryption key. To implement a decrypting mix-net we could of course first perform the mix and then do decryption afterwards. However, it can be more efficient to implement the mix-net by letting the parties take turns in shuffling and doing their part of the decryption. This way, each party only has to be activated once. While it is possible to let each party shuffle first, then do its partial decryption and prove correctness of both of them separately, it is more practical to do both in one step. This means that we need a zero-knowledge argument for shuffle-decryption.

Related work. Efficient schemes for proving the correctness of a shuffle of ElGamal ciphertexts can be found in [44, 66]. In [50] a slightly more general approach is taken by suggesting a method for shuffling that works with most homomorphic cryptosystems. This latter scheme is also the most efficient scheme for proving the correctness of a shuffle.

Since zero-knowledge arguments for shuffles are now reasonably efficient, the computational cost of proving correctness of a decryption has become relatively more important in decrypting mix-nets. Recent work [43, 42] has aimed at proving correctness of shuffle and decryption in one go. Combining the arguments saves some effort and allows us to cut out the intermediate ciphertexts between shuffling and decryption. Unfortunately, these arguments are not zero-knowledge, instead they have a weaker property called complete permutation hiding. Furthermore, a clever observation in [42] gives good performance but restricts the relevant class of ElGamal encryption to those over groups with order 2 mod 3. In some cases this may actually matter, for instance it is always the case that the product of two safe primes is 1 mod 3.

Our contribution. In this chapter, we improve on the work in [50] to suggest an honest verifier zero-knowledge arguments for shuffling and for shuffle-decryption. Both zero-knowledge arguments are statistical special honest verifier zero-knowledge, use seven moves, and require only few exponentiations and very little communication. They are designed such that randomization and multi-exponentiation techniques are very useful in improving performance further.

The papers mentioned before suggest using ElGamal encryption with a subgroup $G_q \leq \mathbb{Z}_p^*$ of order q. Here q is a 160-bit prime and p a 1024-bit prime. Nowadays, these security parameters seem too small, in particular when used for voting where privacy must last for many years. However, to make comparison easy we will use these parameters when analyzing performance of our schemes. We compare the most efficient schemes for shuffling n ElGamal ciphertexts in Table 5.1.

	Furu.-Sako	Groth	Furukawa	proposed
Shuffle P (expo.)	8n	6n		6n
Shuffle V (expo.)	10n	6n		6n
Shuffle-decrypt P (expo.)			8n	7n
Shuffle-decrypt V (expo.)			6n	6n
Communication (bits)	5280n	1344n	1504n	480n
Rounds	3	7	5	7
Privacy	SHVZK	SHVZK	Perm. hiding	SHVZK

Table 5.1: Comparison of shuffle arguments

For voting the arguments will often be made non-interactive using the Fiat-

Shamir heuristic, this way anybody can verify them. This implies that efficiency in verification matters more than efficiency in proving. It also means that we do not need to transmit the challenges, so all computational complexities should be reduced with 160n bits.

Setup and parameters. We assume we have public keys for a cryptosystem and a multi-commitment scheme. There are many ways to select these and our arguments work for a wide range of parameters. For notational purposes, we will in our choice of parameters think about them as ElGamal encryption and Pedersen multi-commitment, with parameters p, q, where $q|p-1$. The randomizer space for both encryption and commitment is then $\mathbb{Z}_q$. We also use a couple of other security parameters ℓ_e is the bit-length of some challenges that the verifier selects. ℓ_s is a parameter such that for any value a we have $a + r$ and r are indistinguishable when we pick r as a $|a| + \ell_s$-bit random number.

Shuffling encryptions of large messages. The estimates above works with a small message space of 160 bits. In real life one does, however, often encounter homomorphic ciphertexts with messages spaces of large order or unknown order. Consider for instance Paillier encryption [69], which has message space $\mathbb{Z}_n$, where n is an RSA-modulus or Okamoto-Uchiyama [68] encryption where the message space is $\mathbb{Z}_p$ for an unknown p. Our shuffling technique can be modified to handle these message spaces at a limited cost, in particular the communication complexity is independent of the size of the message space.

An application. To demonstrate that shuffling is also interesting in other contexts than mix-nets we offer a SHVZK argument of the correctness of a vote in an election where candidates are given points according to their order of preference.

5.2 Shuffle of Known Content

Committing to a permutation of values. Before looking into the question of shuffling ciphertexts, we will look at a simpler problem. Imagine we have a set of messages $m_1, \ldots, m_n$. It is easy enough to pick a permutation π and a randomizer r and set $c = \mathrm{mcom}(m_{\pi(1)}, \ldots, m_{\pi(n)}; r)$. Can we prove knowledge of the permutation π and the randomizer r such that indeed c has been computed this way?

In this section, we present an argument for a multi-commitment containing a permutation of a set of messages. The main idea is from Neff [66], namely that a polynomial $p(X) = \prod_{i=1}^{n}(m_i - X)$ is stable under permutation of the roots, i.e., for any permutation π we have $p(X) = \prod_{i=1}^{n}(m_{\pi(i)} - X)$. A way to test whether two polynomials $p(X), q(X)$ are identical is to choose a random point x

and evaluate whether $p(x) = q(x)$. Going the other direction, if two polynomials are identical over a field $\mathbb{Z}_q$ then they have the same roots.

Using this idea, what we will do to prove that c contains a shuffle of the messages $m_1, \ldots, m_n$ is to prove that for a randomly chosen x we have that $\prod_{i=1}^{n}(m_i - x)$ is equal to the product of the messages in c subtracted by x. This argument of knowledge can be performed using standard honest verifier zero-knowledge argument of multiplication of committed values. We suggest an efficient argument in Figure 5.1.

Theorem 5.1 *The protocol in Figure 5.1 is a 5-move public coin special honest verifier zero-knowledge argument of knowledge with witness extended emulation for c being a commitment to a permutation of the messages $m_1, \ldots, m_n$. If the commitment scheme is statistically hiding then the argument is statistical honest verifier zero-knowledge.*

Proof. It is obvious that we are dealing with a 5-move public coin protocol. Completeness is trivial to verify. Remaining is to prove special honest verifier zero-knowledge and witness extended emulation.

Special honest verifier zero-knowledge. We first describe how to simulate an argument given challenges x, e. Pick $f_1, \ldots, f_n, z$ at random, and pick $f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}, z_\Delta$ at random. Let $c_a \leftarrow \text{mcom}(0, \ldots, 0)$. Set $c_d = \text{mcom}(f_1, \ldots, f_n; z)c^{-e}$ and $c_\Delta = \text{mcom}(f_{\Delta_1}, \ldots, \Delta_{n-1}; z_\Delta)c_a^{-e}$. The simulated argument is $c_d, c_\Delta, x, c_a, e, f_1, \ldots, f_n, z$,
$f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}, z_\Delta$.

To see that the simulated argument is indistinguishable from a real argument, consider the following hybrid argument. We proceed as in the simulation choosing $f_1, \ldots, f_n, z, f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}, z_\Delta$ at random. Now compute $d_1 = f_1 - em_{\pi(1)}, \ldots, d_n = f_n - em_{\pi(n)}$. Compute $F_1, \ldots, F_n$ such that $F_n = e\prod_{i=1}^{n}(m_i - x)$, $F_1 = f_1$ and for $i = 1, \ldots, n-1$ we have $eF_{i+1} = F_i(f_{i+1} - ex) + f_{\Delta_i}$. Define $\Delta_i = F_i - ea_i$. Now pick r_a at random and set $c_a = \text{mcom}(\Delta_2 - (m_{\pi(1)} - x)\Delta_1 - a_1 d_2, \ldots, \Delta_n - (m_{\pi(n)} - x)\Delta_{n-1} - a_{n-1}d_n; r_a)$. Continue the simulation by setting $c_d = \text{mcom}(f_1, \ldots, f_n; z)c^{-e}$ and $c_\Delta = \text{mcom}(f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}; z_\Delta)c_a^{-e}$.

The hybrid argument is statistically indistinguishable from a real argument, since everything is generated the same way. On the other hand, in comparison with the simulated argument the only difference is the content of commitment c_a. The hiding property of the commitment scheme therefore gives us indistinguishability between the hybrid argument and the simulated argument. If the commitment scheme is statistically hiding then the argument is statistical honest verifier zero-knowledge.

Witness extended emulation. The emulator first runs P^* with the algorithm of the real verifier. This is the view that E outputs. If the view is rejecting, then

Shuffle of Known Content Argument

Common input: $c, m_1, \ldots, m_n$ and public keys.
Prover's input: A permutation $\pi \in \Sigma_n$ and a randomizer $r \in \mathbb{Z}_q$ such that $c = \text{mcom}(m_{\pi(1)}, \ldots, m_{\pi(n)}; r)$.

Initial message: Pick $d_1, \ldots, d_n$ and r_d at random from $\mathbb{Z}_q$ and let $c_d = \text{mcom}(d_1, \ldots, d_n; r_d)$.

Pick $\Delta_2, \ldots, \Delta_{n-1} \in_R \mathbb{Z}_q$ and define $\Delta_1 = d_1, \Delta_n = 0$. Choose $r_\Delta \in_R \mathbb{Z}_q$ and set $c_\Delta = \text{mcom}(-d_2\Delta_1, \ldots, -d_n\Delta_{n-1}; r_\Delta)$.

Send c_d, c_Δ to the verifier.

First challenge: $x \in \{0,1\}^{\ell_e}$.

First answer: For $j = 1, \ldots, n$ define $a_j = \prod_{i=1}^{j}(m_{\pi(i)} - x)$.

Pick $r_a \in \mathbb{Z}_q$ and set $c_a =$
$\text{mcom}(\Delta_2 - (m_{\pi(1)} - x)\Delta_1 - a_1 d_2, \ldots, \Delta_n - (m_{\pi(n)} - x)\Delta_{n-1} - a_{n-1}d_n; r_a)$.

Send c_a to the verifier.

Final challenge: $e \in_R \{0,1\}^{\ell_e}$.

Final answer: Set $f_1 = e(m_{\pi(1)}) + d_1, \ldots, f_n = e(m_{\pi(n)}) + d_n$. Set $z = er + r_d$. Let $f_{\Delta_1} = e(\Delta_2 - (m_{\pi(2)} - x)\Delta_1 - a_1 d_2) - \Delta_1 d_2, \ldots, f_{\Delta_{n-1}} = e(\Delta_n - (m_{\pi(n)} - x)\Delta_{n-1} - a_{n-1}d_n) - \Delta_{n-1}d_n$ and $z_\Delta = er_a + r_\Delta$.

Send $f_1, \ldots, f_n, z, f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}, z_\Delta$ to the verifier.

Verification: Check that $\text{mcom}(f_1, \ldots, f_n; z) = c^e c_d$.

Check that $\text{mcom}(f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}; z_\Delta) = c_a^e c_\Delta$.

Define $F_1, F_2, \ldots, F_n$ as the elements such that $F_1 = f_1 - ex, eF_2 = F_1(f_2 - ex) + f_{\Delta_1}, \ldots, eF_n = F_{n-1}(f_n - ex) + f_{\Delta_{n-1}}$.[a] Verify that $F_n = e\prod_{i=1}^{n}(m_i - x)$.

[a] The idea is that the identity $F_j = \prod_{i=1}^{j}(m_{\pi(i)} - x) + \Delta_j$ holds for all j.

Figure 5.1: Argument of Knowledge of Shuffle of Known Content.

E halts with $w = \perp$. However, if the view is accepting then E must try to find a witness itself.

To extract a witness E samples another acceptable argument. Call the two arguments $c_d, c_\Delta, x, c_a, e, f_1, \ldots, f_n, z, f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}, z_\Delta$ and $c_d, c_\Delta, x', c'_a, e', f'_1, \ldots, f'_n, z', f'_{\Delta_1}, \ldots, f'_{\Delta_{n-1}}, z'_\Delta$. If $e \neq e'$ we have $\text{mcom}(f_1, \ldots, f_n; z) = c^e c_d$ and $\text{mcom}(f'_1, \ldots, f'_n; z') = c^{e'} c_d$. This gives us

$\text{mcom}(f_1 - f'_1, \ldots, f_n - f'_n; z - z') = c^{e-e'}$. E now runs the root extraction algorithm in an attempt to learn a quasi-opening $\mu_1, \ldots, \mu_n, r$ of c. If successful E can deduce a quasi-opening $d_1, \ldots, d_n, r_d$ of c_d too.

Let us at this stage argue that E runs in expected polynomial time. If P^* is in a situation where it has probability ϵ of making the verifier accept, then the expected number of runs to get an acceptable view is $\frac{1}{\epsilon}$. Of course if P^* fails, then we do not need to sample a second run. We therefore get a total expectation of 1 query to the oracle P^*. A consequence of E using an expected polynomial number of queries to P^* is that it only has negligible probability of ending in a run where $e' = e$ or any other event with negligible probability occurs.

The next step is to argue that with overwhelming probability $\mu_1, \ldots, \mu_n$ is a permutation of $m_1, \ldots, m_n$, i.e., E has indeed found a witness. More precisely, if E is not a witness extended emulator then there is a polynomial in the security parameter $poly(k)$, such that the difference between the probabilities is larger than $\frac{2}{poly(k)}$ for an infinite number of k's.

Assume therefore that $\mu_1, \ldots, \mu_n$ is not a permutation of $m_1, \ldots, m_n$ and P^* has more than $\frac{1}{poly(k)}$ chance of producing a convincing argument. In that case we can pick a challenge x at random, and thereafter pick three random final challenges e, e', e''. With probability at least $\frac{1}{poly(k)^3}$ does P^* manage to create accepting arguments on all three of these challenges. Call the first two arguments $c_d, c_\Delta, x, c_a, e, f_1, \ldots, f_n, z, f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}, z_\Delta$ and $c_d, c_\Delta, x, c_a, e', f'_1, \ldots, f'_n, z, f'_{\Delta_1}, \ldots, f'_{\Delta_{n-1}}, z'_\Delta$. We get $\text{mcom}(f_{\Delta_1} - f'_{\Delta_1}, \ldots, f_{\Delta_{n-1}} - f'_{\Delta_{n-1}}; z_\Delta - z'_\Delta) = c_a^{e-e'}$. From this we can extract an opening $\alpha_1, \ldots, \alpha_{n-1}, r_a$. Using the homomorphic property this also gives us an opening $\delta_1, \ldots, \delta_{n-1}, r_\Delta$ of c_Δ.

Consider now the third challenge e''. Since we know openings of c_a and c_Δ, we can deduce that we have found a value e'' such that $(e'')^n \prod_{i=1}^n (\mu_i - x) = (e'')^n \prod_{i=1}^n (m_i - x) + f(e'')$, where $f(X)$ is a polynomial of degree $n-1$. Since e'' is chosen at random this implies with overwhelming probability that $\prod_{i=1}^n (\mu_i - x) = \prod_{i=1}^n (m_i - x)$. Furthermore, since x was chosen at random we have two polynomials evaluating to the same value in a random point. With overwhelming probability, they are identical. This in turn implies with overwhelming probability that $\mu_1, \ldots, \mu_n$ is a permutation of $m_1, \ldots, m_n$ as we wanted to show. □

Single commitments. We remark that we do not need to use the same public keys for the multicommitments c, c_d and c_a, c_Δ. We could therefore make a combination, where we used a multi-commitment for c_a, c_Δ, while using a bunch of single commitments to commit to the m_i's and d_i's. I.e., let $c_1 = \text{com}(m_{\pi(1)}; r_1), \ldots, c_n = \text{com}(m_{\pi(n)}; r_n)$. In some applications, this separation of the commitments to the m_i's may be useful.

5.3 Shuffle of Ciphertexts

The idea. As mentioned in the introduction it is easy to shuffle a set of homomorphic ciphertexts $e_1, \ldots, e_n$ into a new set of ciphertexts $E_1 \leftarrow e_{\pi(1)}E_{pk}(0), \ldots, E_n \leftarrow e_{\pi(n)}E_{pk}(0)$, where π is some permutation. The question is whether we can argue having done so in zero-knowledge.

We describe the idea in the protocol, which resembles the idea in [50]. Make a commitment $c_s \leftarrow \text{mcom}(\pi(1), \ldots, \pi(n))$. We send c_s to the verifier. This way we have committed to the permutation π. Of course the verifier does not yet know that we did the correct thing, we will demonstrate that later.

The second step consists in the verifier choosing at random $t_1, \ldots, t_n$ and sending them to the prover. We make another commitment $c_t \leftarrow \text{mcom}(t_{\pi(1)}, \ldots, t_{\pi(n)})$ and send it to the verifier. We will try to convince the verifier that the t_i's have been shuffled in exactly the same way as $1, \ldots, n$.

Now, what we will actually do is to let the verifier pick a challenge λ at random. We will then use the argument of the previous section to show that $c_s^{\lambda}c_t$ contain a shuffle of the elements $\lambda+t_1, \ldots, \lambda n+t_n$. Unless we did indeed commit to a permutation of $1, \ldots, n$ and $t_1, \ldots, t_n$, and used the same permutation for both sets, there is overwhelming probability over λ that we cannot create a convincing argument.

The final step is to prove that $E_1^{\lambda\pi(1)+t_{\pi(1)}} \cdots E_n^{\lambda\pi(n)+t_{\pi(n)}}$ contains the same plaintext as $e_1^{\lambda+t_1} \cdots e_n^{\lambda n+t_n}$. Since the verifier committed to a permutation π before knowing λ and the t_i's, this forces him to raise the ciphertexts to random elements $\lambda i + t_i$ completely beyond his control. Unless indeed $e_1, \ldots, e_n$ and $E_1, \ldots, E_n$ have the same set of plaintexts there is only negligible chance of succeeding with the argument.

Theorem 5.2 *The scheme in Figure 5.2 is a public coin 7-move special honest verifier zero-knowledge argument of knowledge with witness extended emulation for a correct shuffle. If the commitments are statistically hiding then the argument is statistical special honest verifier zero-knowledge.*

Proof. It is easy to see that we are dealing with a 7-move public coin protocol. Completeness follows by straightforward verification. We need to demonstrate that the protocol is special honest verifier zero-knowledge and we need to show that it has the witness-extended emulation property.

Special Honest Verifier Zero-Knowledge. We are given $t_1, \ldots, t_n, \lambda, x, e$ as input, and wish to produce something that is indistinguishable from a real argument.

Choose $c_s \leftarrow \text{mcom}(0, \ldots, 0), c_t = \text{mcom}(0, \ldots, 0), c_a = \text{mcom}(0, \ldots, 0)$. Pick $f_1, \ldots, f_n$ at random and $F_2, \ldots, F_{n-1}$ at random. Let $F_1 = f_1 - ex$ and $F_n = e\prod_{i=1}^{n}(\lambda i + t_i - x)$. Pick z, z_Δ and Z at random.

Shuffle of Ciphertexts Argument

Common input: $e_1, \ldots, e_n$ and $E_1, \ldots, E_n$ and public keys.
Prover's input: A permutation $\pi \in \Sigma_n$ and randomizers $R_1, \ldots, R_n$ satisfying $E_1 = e_{\pi(1)} E_{pk}(0; R_1), \ldots, E_n = e_{\pi(n)} E_{pk}(0; R_n)$.

Initial message: Pick $r_s \in_R \mathbb{Z}_q$ at random and set $c_s = \text{mcom}(\pi(1), \ldots, \pi(n); r_s)$.

Select $d_1, \ldots, d_n \in_R \mathbb{Z}_q$. Select $r_d \in_R \mathbb{Z}_q$ and set $c_d = \text{mcom}(d_1, \ldots, d_n; r_d)$.

Select $\Delta_2, \ldots, \Delta_{n-1} \in_R \mathbb{Z}_q$ and define $\Delta_1 = d_1, \Delta_n = 0$. Select $r_\Delta \in_R \mathbb{Z}_q$ at random and set $c_\Delta = \text{mcom}(-d_2\Delta_1, \ldots, -d_n\Delta_{n-1}; r_\Delta)$.

Select $R \in_r \mathbb{Z}_q$ and set $E = E_{pk}(0; R) E_1^{d_1} \cdots E_n^{d_n}$.

Send c_s, c_d, c_Δ, E to the verifier.

First challenge: $t_1, \ldots, t_n \in_R \{0,1\}^{\ell_e}$.

First answer: Select $r_t \in_r \mathbb{Z}_q$ and set $c_t = \text{mcom}(t_{\pi(1)}, \ldots, t_{\pi(n)}; r_t)$.

Send c_t to the verifier.

Second challenge: Choose $\lambda, x \in_R \{0,1\}^{\ell_e}$ at random.

Second answer: For $j = 1, \ldots, n$ let $a_j = \prod_{i=1}^{j} (\lambda\pi(i) + t_{\pi(i)} - x)$.

Select $r_a \in_R \mathbb{Z}_q$ and set $c_a = \text{mcom}(\Delta_2 - (\lambda\pi(2) + t_{\pi(2)} - x)\Delta_1 - a_1 d_2, \ldots, \Delta_n - (\lambda\pi(n) + t_{\pi(n)} - x)\Delta_{n-1} - a_{n-1} d_n; r_a)$.

Send c_a to the verifier.

Third challenge: Select $e \in_R \{0,1\}^{\ell_e}$.

Final answer: Set $f_1 = e(\lambda\pi(1) + t_{\pi(1)}) + d_1, \ldots, f_n = e(\lambda\pi(n) + t_{\pi(n)}) + d_n$. Set $z = e(\lambda r_s + r_t) + r_d$. Let $f_{\Delta_1} = e(\Delta_2 - (\lambda\pi(2) + t_{\pi(2)} - x)\Delta_1 - a_1 d_2) - \Delta_1 d_2, \ldots, f_{\Delta_{n-1}} = e(\Delta_n - (\lambda\pi(n) + t_{\pi(n)} - x)\Delta_{n-1} - a_{n-1} d_n) - \Delta_{n-1} d_n$ and $z_\Delta = e r_a + r_\Delta$.

Set $Z = R - e(\lambda\pi(1) + t_{\pi(1)}) R_1 - \cdots - e(\lambda\pi(n) + t_{\pi(n)}) R_n$.

Send $f_1, \ldots, f_n, z, f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}, z_\Delta, Z$ to the verifier.

Verification: Check that $\text{mcom}(f_1, \ldots, f_n; z) = (c_s^\lambda c_t)^e c_d$.

Check that $\text{mcom}(f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}; z_\Delta) = c_a^e c_\Delta$.

Define $F_1, F_2, \ldots, F_n$ as the elements such that $F_1 = f_1 - ex, eF_2 = F_1(f_2 - ex) + f_{\Delta_1}, \ldots, eF_n = F_{n-1}(f_n - ex) + f_{\Delta_{n-1}}$. Verify that $F_n = e \prod_{i=1}^{n} (\lambda i + t_i - x)$.

Check that $E_{pk}(0; Z) E_1^{f_1} \cdots E_n^{f_n} = (e_1^{\lambda 1 + t_1} \cdots e_n^{\lambda n + t_n})^e E$.

Figure 5.2: SHVZK argument of knowledge of a shuffle of ciphertexts.

Compute $f_{\Delta_1} = eF_2 - F_1(f_2 - ex), \ldots, f_{\Delta_{n-1}} = eF_n - F_{n-1}(f_n - ex)$. Define $c_d = (c_s^\lambda c_t)^{-e}\text{mcom}(f_1, \ldots, f_n; z)$ and $c_\Delta = c_a^{-e}\text{mcom}(f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}; z_\Delta)$. Set $E = E_{pk}(0; Z)E_1^{f_1} \cdots E_n^{f_n}(e_1^{\lambda+t_1} \cdots e_n^{\lambda+t_n})^{-e}$.

The simulated argument is $(c_s, c_d, c_\Delta, E, t_1, \ldots, t_n, c_t, \lambda, x, c_a, e, f_1, \ldots, f_n, z, f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}, z_\Delta, Z)$.

To see that the simulated argument is indistinguishable from a real argument, consider the following hybrid argument. We do everything as in the simulation except when computing c_s, c_t, c_a. Let $c_s \leftarrow \text{mcom}(\pi(1), \ldots, \pi(n); r_s)$ and $c_t \leftarrow \text{mcom}(t_{\pi(1)}, \ldots, t_{\pi(n)})$. Define $d_i = f_i - e(\pi(i) + t_{\pi(i)}$ and $\Delta_i = F_i - ea_i$ for $i = 1, \ldots, n$. Let $c_a \leftarrow \text{mcom}(\Delta_2 - (\lambda\pi(2) + t_{\pi(2)} - x)\Delta_1 - a_1 d_2, \ldots, \Delta_n - (\lambda\pi(n) + t_{\pi(n)} - x)\Delta_{n-1} - a_{n-1}d_n)$.

The hybrid argument is statistically indistinguishable from a real argument since we have only changed the order in which we compute the things. The hybrid argument is also indistinguishable from the simulated argument since the only difference is in the commitments c_s, c_t, c_a. If the commitments are statistically hiding then the hybrid argument and the simulated argument are statistically indistinguishable.

Witness extended emulation. First, note that as in the proof of Theorem 5.1 we can from a couple of samples with the same $t_1, \ldots, t_n$ and the same λ extract the contents of $c_s^\lambda c_t$. As in that proof, we can then argue that we know how to open $c_s^\lambda c_t$ as containing a permutation of $\lambda + t_1, \ldots, \lambda n + t_n$. With overwhelming probability over λ, the permutation π is uniquely defined.

Second, we wish to argue that c_s contains $\pi(1), \ldots, \pi(n)$. To argue this sample yet another argument, this time with a different λ' but the same e as we have used before. This gives us $(c_s^{\lambda-\lambda'})^e = \text{mcom}(f_1 - f_1', \ldots, f_n - f_n', z - z')$ and we can extract an opening of c_s. If that opening is not $\pi(1), \ldots, \pi(n)$, then there would be overwhelming probability over λ for the contents of $c_s^\lambda c_t$ not being a permutation of $\lambda + t_1, \ldots, \lambda n + t_n$.

Third, we sample $n+1$ arguments $c_s, c_d, c_\Delta, E, t_1^{(i)}, \ldots, t_n^{(i)}, c_t^{(i)}, \lambda^{(i)}, x^{(i)}, c_a^{(i)}, e^{(i)}$, $f_1^{(i)}, \ldots, f_n^{(i)}$,
$z^{(i)}, f_{\Delta_1}^{(i)}, \ldots, f_{\Delta_{n-1}}^{(i)}, z_\Delta^{(i)}, Z^{(i)}$ for $i = 0, \ldots, n$. This takes an expected number of $n + 1$ queries to P^*.

The n vectors $(e^{(0)}(\lambda^{(0)} + t_1^{(0)}) - e^{(i)}(\lambda^{(i)} + t_n^{(i)}), \ldots, e^{(0)}(\lambda^{(0)}n + t_n^{(0)}) - e^{(i)}(\lambda^{(i)}n + t_n^{(i)})$ are with overwhelming probability linearly independent. This means that from the n vectors $(Z^{(0)} - Z^{(i)}, f_1^{(0)} - f_1^{(i)}, \ldots, f_n^{(0)} - f_n^{(i)}, e^{(0)}(\lambda^{(0)} + t_1^{(0)}) - e^{(i)}(\lambda^{(i)} + t_n^{(i)}), \ldots, e^{(0)}(\lambda^{(0)}n + t_n^{(0)}) - e^{(i)}(\lambda^{(i)}n + t_n^{(i)})$, we can for any $j = 1, \ldots, n$ find a linear combination giving us the vector $(-R_i, 0 \ldots, 0, 1_{j+1}, 0, \ldots, 0, 0, \ldots, 0, 1_{\pi(j)+n+1}, 0, \ldots, 0)$. This in turn gives us $E_{pk}(0; -R_i)E_j = e_{\pi(j)}$ for $j = 1, \ldots, n$. In other words, we have found a witness. □

5.3.1 Using Small Integers to Shuffle.

When we work with small message spaces, say $\mathbb{Z}_q$ with $|q| = 160$ the schemes suggested above work fine. However, with larger message spaces we loose efficiency because the commitments have a message space of the same size. Another problem is when we are dealing with a message space of unknown size in which case the known shuffle techniques do not suffice.

Here we will suggest a variation of the argument in Figure 5.2 that handle large message spaces or message spaces of unknown size. The observation is that no matter what the message space, as long as it does not have small prime factors dividing its order, it is sufficient to prove that $E_{pk}(0;Z)E_1^{\lambda\pi(1)+t_{\pi(1)}}\cdots E_n^{\lambda\pi(n)+t_{\pi(n)}} = e_1^{\lambda+t_1}\cdots e_n^{\lambda n+t_n}E$ for integers $\lambda, t_1, \ldots, t_n$ of small size, say 80 bits for an interactive argument and 160 bits for a non-interactive argument using the Fiat-Shamir heuristic.

Now if we use a commitment scheme with message space $\mathbb{Z}_{q'}$, where q' is a prime with $|q'| > 3t$, then $e(\lambda i + t_i) + d_i$ are unlikely to be larger than q', we have no overflow. This means that we actually have a shuffle of small integers.

The consequence is that we can use a commitment scheme with relatively short message space to prove a shuffle of ciphertext that use an entirely different message space. This also has the advantage that we may use a scheme with keys that can be easily generated in a distributed way and verified efficiently, for instance Pedersen commitments over $\mathbb{Z}_{p'}^*$, no matter the message space of the cryptosystem.

5.4 Shuffle and Decryption

As mentioned in the introduction some efficiency gain can be achieved by combining the argument of a shuffle and a decryption. Here we shall look at an ElGamal-style cryptosystem. Consider a key $(G, h_1, \ldots, h_k)$ for the cryptosystem, where a group of parties hold individual shares $s_1, \ldots, s_k$ such that for all i we have $h_i = G^{s_i}$. We can encrypt a message under those keys as $(G^r, (\prod_{i=1}^k h_i)^r m)$. Party 1 can peel of a layers of encryption to give us $(G^r, (\prod_{i=2}^k h_i)^r m)$, party 2 can peel of another layer to get $(G^r, (\prod_{i=3}^k h_i)^r m)$, etc. After party k has peeled of a layer, we have the message.

We can combine this technique with the shuffling to get a decrypting shuffle. I.e., suppose (G, h, H) are publicly known where we know s such that $h = G^s$. Intuitively H is the product of the h_i's of the remaining parties. Furthermore, assume we get as input ciphertexts $(u_1, v_1), \ldots, (u_n, v_n)$ encrypted under (G, hH). We can shuffle and decrypt by picking at random a permutation π and randomizers $R_1, \ldots, R_n$ and compute $(U_1 = G^{R_1}u_{\pi(1)}, V_1 = H^{R_1}v_{\pi(1)}u_{\pi(1)}^{-s}), \ldots, (U_n, V_n) = (G^{R_n}u_{\pi(n)}, H^{R_n}v_{\pi(n)}u_{\pi(n)}^{-s})$. If the parties one after another perform this shuffle-and-decrypt operation, we get a decrypting mix-net. What we need is of course

a guarantee for them following the protocol, i.e., a zero-knowledge argument for the shuffle-and-decrypt operation.

Shuffle-and-Decrypt Argument

Common input: $(u_1, v_1), \ldots, (u_n, v_n), (U_1, V_1), \ldots, (U_n, V_n)$ and public keys, including G, H, h.
Prover's input: A permutation $\pi \in \Sigma_n$, an exponent s and randomizers $R_1, \ldots, R_n$ satisfying $G^s = h$ and
$(U_1, V_1) = (G^{R_1} u_{\pi(1)}, H^{R_1} v_{\pi(1)} u_{\pi(1)}^{-s}), \ldots, (U_n, V_n) = (G^{R_n} u_{\pi(n)}, H^{R_n} v_{\pi(n)} u_{\pi(n)}^{-s})$.

Initial message: Pick $r_s \in_r \mathbb{Z}_q$ and set $c_s = \text{mcom}(\pi(1), \ldots, \pi(n); r_s)$.

Select $d_1, \ldots, d_n \in_R \mathbb{Z}_q$. Select $r_d \in_R \mathbb{Z}_q$ and set $c_d = \text{mcom}(d_1, \ldots, d_n; r_d)$.

Select $\Delta_2, \ldots, \Delta_{n-1} \in \mathbb{Z}_q$ and set $\Delta_1 = d_1, \Delta_n = 0$. Select $r_\Delta \in_r \mathbb{Z}_q$ and set $c_\Delta = \text{mcom}(-d_2\Delta_1, \ldots, -d_n\Delta_{n-1}; r_\Delta)$.

Select $d \in_R \mathbb{Z}_q$ and set $D = G^d$. Select R at random. Set $U = G^R U_1^{d_1} \cdots U_n^{d_n}$.

Send c_s, c_d, c_Δ, D, U to the verifier.

First challenge: $t_1, \ldots, t_n \in \{0,1\}^{\ell_e}$.

First answer: Select $r_t \in_R \mathbb{Z}_q$ and set $c_t = \text{mcom}(t_{\pi(1)}, \ldots, t_{\pi(n)}; r_t)$.

Send c_t to the verifier.

Second challenge: Choose $\lambda, x \in_r \{0,1\}^{\ell_e}$.

Second answer: For $j = 1, \ldots, n$ let $a_j = \prod_{i=1}^{j} (\lambda\pi(i) + t_{\pi(i)} - x)$.

Select $r_a \in_R \mathbb{Z}_q$ and set $c_a =$
$\text{mcom}(\Delta_2 - (\lambda\pi(2) + t_{\pi(2)} - x)\Delta_1 - a_1 d_2, \ldots, \Delta_n - (\lambda\pi(n) + t_{\pi(n)} - x)\Delta_{n-1} - a_{n-1} d_n; r_a)$.

Set $V = H^R (u_1^{\lambda + t_1} \cdots u_n^{\lambda n + t_n})^d (V_1^{d_1} \cdots V_n^{d_n})$.

Send c_a, V to the verifier.

Third challenge: Select at random $e \in_R \{0,1\}^{\ell_e}$.

Final answer: Set $f_1 = e(\lambda\pi(1) + t_{\pi(1)}) + d_1, \ldots, f_n = e(\lambda\pi(n) + t_{\pi(n)}) + d_n$. Set $z = e(\lambda r_s + r_t) + r_d$. Let
$f_{\Delta_1} = e(\Delta_2 - (\lambda\pi(2) + t_{\pi(2)} - x)\Delta_1 - a_1 d_2) - \Delta_1 d_2, \ldots, f_{\Delta_{n-1}} =$
$e(\Delta_n - (\lambda\pi(n) + t_{\pi(n)} - x)\Delta_{n-1} - a_{n-1} d_n) - \Delta_{n-1} d_n$ and $z_\Delta = e r_a + r_\Delta$.
Set $Z = R - e(\lambda\pi(1) + t_{\pi(1)})R_1 - \cdots - e(\lambda\pi(n) + t_{\pi(n)})R_n$. Set $f = es + d$.

Send $f_1, \ldots, f_n, z, f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}, z_\Delta, Z, f$ to the verifier.

Verification: Check that $\text{mcom}(f_1, \ldots, f_n; z) = (c_s^\lambda c_t)^e c_d$. Check that $\text{mcom}(f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}; z_\Delta) = c_a^e c_\Delta$.

Define $F_1, F_2, \ldots, F_n$ as the elements such that
$F_1 = f_1 - ex, eF_2 = F_1(f_2 - ex) + f_{\Delta_1}, \ldots, eF_n = F_{n-1}(f_n - ex) + f_{\Delta_{n-1}}$. Verify that $F_n = e \prod_{i=1}^{n} (\lambda i + t_i - x)$.

Check that $G^Z U_1^{f_1} \cdots U_n^{f_n} = (u_1^{\lambda + t_1} \cdots u_n^{\lambda n + t_n})^e U$. Verify that $G^f = h^e D$. Check that $H^Z V_1^{f_1} \cdots V_n^{f_n} = (v_1^{\lambda + t_1} \cdots v_n^{\lambda n + t_n})^e (u_1^{\lambda + t_1} \cdots u_n^{\lambda n + t_n})^{-f} V$.

Figure 5.3: Argument of knowledge for a shuffle-and-decrypt operation.

Theorem 5.3 *The protocol in Figure 5.3 is a 7-move public coin special honest*

verifier zero-knowledge argument with witness extended emulation. If the commitment scheme is statistically hiding, then the argument is statistical honest verifier zero-knowledge.

Proof. It is easy to see that it is a 7-move public coin protocol. Completeness is straightforward to verify. Left is to argue special honest verifier zero-knowledge and witness-extended emulation.

Special honest verifier zero-knowledge. Given challenges $t_1, \ldots, t_n, \lambda, x, e$ we want to simulate an argument. To that end, we pick $f_1, \ldots, f_n, z, f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}, z_\Delta, f, Z$ at random. We set $c_s \leftarrow \text{mcom}(0, \ldots, 0), c_t \leftarrow \text{mcom}(0, \ldots, 0), c_a \leftarrow \text{mcom}(0, \ldots, 0)$. We then compute $c_d = \text{mcom}(f_1, \ldots, f_n; z)(c_s^\lambda c_t)^{-e}, c_\Delta = \text{mcom}(f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}; z_\Delta)c_a^{-e}, D = G^f h^{-e}, U = G^Z U_1^{f_1} \cdots U_n^{f_n}(u_1^{\lambda+t_1} \cdots u_n^{\lambda n+t_n})^{-e}$ and $V = H^Z V_1^{f_1} \cdots V_n^{f_n}(v_1^{\lambda+t_1} \cdots v_n^{\lambda+t_n})^{-e}$.

The simulated argument is $(c_s, c_d, c_\Delta, D, U, t_1, \ldots, t_n, \lambda, x, c_a, V, , f_1, \ldots, f_n, z, f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}, z_\Delta, f, Z)$.

To argue that this is a good simulation consider the hybrid argument that carries out the simulation procedure except generating c_s, c_t, c_a as the hybrid argument in the proof of Theorem 5.2. The hybrid argument is statistically indistinguishable from a real argument, since the only difference is in the order in which we generate the different elements. It is also indistinguishable from a simulated argument because of the hiding property of the commitments. If the commitments are statistically hiding, then the hybrid argument is statistically indistinguishable from a simulated argument.

Witness-extended emulation. To argue witness-extended emulation first note that from a couple of convincing arguments with challenge $e \neq e'$ we may extract the exponent s such that $h = G^s$.

We then proceed as in the proof of Theorem 5.2 and find linear combination of vectors giving us for $j = 1, \ldots, n$ a vector $(R_j, 0 \ldots, 1_{j+1}, 0 \ldots, 0, 0, \ldots, 0, 1_{\pi(j)+n+1}, 0, \ldots, 0)$. This gives us $G^{R_j} U_j = u_{\pi(j)}$.

At this point, we have a permutation π, randomizers $R_1, \ldots, R_n$ and the secret exponent s. All that remains is to argue that $H^{R_1} V_1 = v_{\pi(1)} u_{\pi(1)}^{-s}, \ldots, H^{R_n} V_n = v_{\pi(n)} u_{\pi(n)}^{-s}$. First, since we know $R, d_1, \ldots, d_n, \lambda, t_1, \ldots, t_n, s, d$ we see that the adversary has negligible chance of success unless indeed $V = H^R(u_1^{\lambda+t_1} \cdots u_n^{\lambda n+t_n})^d V_1^{d_1} \cdots V_n^{d_n}$. In successful arguments we therefore have $H^Z V_1^{f_1} \cdots V_n^{f_n} = ((v_1 u_1^{-s})^{\lambda+t_1} \cdots (v_n u_n^{-s})^{\lambda n+t_n})^e (H^R V_1^{d_1} \cdots V_n^{d_n})$. Using the same vectors as above we get for $j = 1, \ldots, n$ that $H^{R_j} V_j = v_{\pi(j)} u_{\pi(j)}^{-s}$, just as we wanted to show. □

5.5 Optimizations

Generating the first challenge. One trick is to generate $t_1, \ldots, t_n$ using a pseudorandom generator. This way, the verifier does not need to send all of them. Instead, he simply selects a seed and sends it to the prover. This trick is worth remembering especially when there are multiple verifiers that generate challenges jointly. In that case generating n random numbers could be a burden when n is large, but all they need to do is to jointly generate a random seed.

Reducing the size of the public key. As noted in [50] we can implement multi-commitments over n elements by choosing a smaller l and then combine several multi-commitments to l elements to give a multi-commitment to n elements. By doing this we use slightly more computational effort, but we can reduce the size of the public key by a factor $\frac{n}{l}$. In cases where we have to jointly generate the public key for the commitment schemes, this may be worthwhile. Furthermore, we may get an advantage out of it when we use randomized verification as described below.

Using different size groups. Another trick from [50] is to use different groups for the commitments and the ciphertexts. Consider shuffling ElGamal ciphertexts, where we operate over a group $G_q \leq \mathbb{Z}_p^*$, where $|p| = 3000$ and $|q| = 300$. In electronic voting such parameters are realistic since the encryption has to hold in many years to come. Instead of making Pedersen-commitments over the group G_q, we can instead use another group $G'_q \leq \mathbb{Z}_{p'}$, of order q where $|p'| = 1000$. This is because the arguments are statistical zero-knowledge so we only need that they are sound at the time of verification, we do not need long term protection.

Unconditional soundness. [66] suggests unconditional soundness. If we use unconditionally binding commitments we get an argument with unconditional soundness and computational zero-knowledge. The price we pay in terms of reduced performance is considerable.

Parallel shuffling. [66] also suggests reusing the first rounds of the protocol in case many sets of ciphertexts need to be shuffled in the same order. This remark also applies to our shuffle argument as well as the shuffle and decrypt argument.

Randomizing the verification. Instead of using a deterministic verification step, we may use randomization to our advantage.

Replace the first step of the verification with the following: Pick α as a random exponent of size, say, 80 bits. Define $f_{\Delta_n} = 0$. Check that $\text{mcom}(\alpha f_1 + f_{\Delta_1}, \ldots, \alpha f_n + f_{\Delta_n}; \alpha z + z_\Delta) = ((c_s^\lambda c_t)^e c_d)^\alpha c_a^e c_\Delta$.

If we are checking many shuffle arguments at the same time, we may repeat this trick to get down to spending no exponentiations on the first step of the verification.

If the group is of a large size, i.e., $|q| >> |\alpha|$, it may also pay of to use randomization in the last step. Suppose for instance that we have an argument for a correct shuffle of ElGamal ciphertexts. Then we just need to verify that $(G^\alpha H)^Z (U_1^\alpha V_1)^{f_1} \cdots (U_n^\alpha V_1)^{f_n} = ((u_1^\alpha v_1)^{\lambda+t_1} \cdots (u_n^\alpha v_n)^{\lambda n+t_n})^e U^\alpha V$.

In other words, we can get close to using only 3 exponentiations in the verification, and down to 2 exponentiations if we are checking multiple arguments at the same time.

5.6 Multi-exponentiation

Consider some election with a huge number of voters. In that case, we may have many votes to mix, i.e., we have to make shuffles of thousands or even millions of encrypted votes. We offer a few thoughts on some multi-exponentiation techniques that offer good performance in this scenario.

The setting is the following. We work in some abelian group with elements $g_1, \ldots, g_n$ and exponents $e_1, \ldots, e_n$ of length t. We want to compute $h = g_1^{e_1} \cdots g_n^{e_n}$.

Let s be some small number dividing t, i.e., $t = su$. A suitable size of s is something like $\lfloor \log(n) \rfloor$. We divide the e_i's into s-bit blocks: $e_i = e_{i,u-1} || \cdots || e_{i,0}$. We have $g_i^{e_i} = g_i^{e_{i,u-1} 2^{s(u-1)}} \cdots g_i^{e_{i,0} 2^0}$.

The observation is now that

$$h = \prod_{i=1}^{n} g_i^{e_i} = \prod_{i=1}^{n} \prod_{j=0}^{u-1} g_i^{e_{i,j} 2^{sj}} = \prod_{j=0}^{u-1} (\prod_{i=1}^{n} g_i^{e_{i,j}})^{2^{sj}}.$$

The inner expression can be computed in a fast way by sorting the elements according to their $e_{i,j}$-block. We have

$$\prod_{i=1}^{n} g_i^{e_{i,j}} = \prod_{k=1}^{2^s-1} (\prod_{i:e_i=k} g_i)^k.$$

Combining the two we get

$$h = \prod_{j=0}^{u-1} (\prod_{k=1}^{2^s-1} (\prod_{i:e_i=k} g_i)^k)^{2^{sj}}.$$

Further optimizations of the loop $\prod_{k=1}^{2^s-1} (\ldots)^k$ are possible since here we can easily find groups of exponents that share many bits. Due to time constraints writing this dissertation, we do not explore this further.

If we are a bit rough and count a modular squaring as a modular multiplication, we get the following estimate. To compute h we need to use approximately $t(\frac{3}{2}2^s + \frac{n}{s})$ multiplications. In comparison, a single exponentiation costs around $\frac{3}{2}t$ multiplications. Setting $s = \log(n) - \log^2(n)$ we therefore only use the computational equivalent of $\mathcal{O}(\frac{n}{\log(n)})$ exponentiations to compute h. As a concrete example, take the parameters from the voting mix-net of [43]. Let $n = 100,000, t = 160, s = 10$, then we use the equivalent of 7691 single exponentiations to compute h. Using our mix-net multiply that with 6 for ElGamal encryption shuffling to get a price of $0.7n$ single exponentiations for the prover and using randomization $0.6n$ single exponentiations for the verifier. In comparison the best parameters for the multi-exponentiation algorithm in [62] are $n = 100,000, t = 160, s = 6$ which gives approximately 20,000 single exponentiations to compute h.

5.7 Correctness of a Priority Vote

Let us work with the election scheme of the previous chapter but look at another way to form valid votes. Each voter ranks the candidates in order of preference, then he assigns 0 votes to the worst candidate, 1 vote to the second-worst candidate, ..., $L-1$ votes to the best candidate. We need a SHVZK argument for correctness of this kind of vote.

The idea is the following. A correct vote is an assignment of votes $0, \ldots, L-1$ to the L candidates. I.e., a permutation $\pi \in \Sigma_n$ uniquely determines a valid vote by assigning $\pi(0)$ votes to the first candidate, $\pi(1)$ votes to the second candidate, etc. We can commit to the vote by forming the commitment $c \leftarrow \text{mcom}(\pi(0), \ldots, \pi(L-1))$.

Using our standard techniques we can then get out $f_1 = e\pi(0)+d_0, \ldots, f_{L-1} = e\pi(L-1)+d_{L-1}$. Defining $D = M^0 d_0 + \cdots + M^{L-1} d_{L-1}$ we then have $F = M^0 f_0 + \cdots M^{L-1} f_{L-1}$. Using the standard technique we can deduce that a ciphertext E contains the content hidden in F, i.e., E has plaintext $\sum_{j=0}^{L-1} \pi(j) M^j$. This means that E contains a valid vote.

This argument can be optimized such that we do not need the intermediate commitment c but use the techniques from the known shuffle argument directly to show that E contains a valid priority vote. We have already seen plenty arguments of valid votes in the previous chapter so we will leave the idea at this sketchy state here without going into the details.

Chapter 6

Non-Malleable Commitment

6.1 Introduction

For some applications of commitments, the properties hiding and binding do not suffice. An example of an additional property is *non-malleability*, a concept introduced by Dolev, Dwork and Naor [37]. We considered in the introduction the problem of fair contract bidding. In particular, we noted that for the Pedersen commitment Bob can consistently underbid Alice, even though Bob at the time of commitment does not know exactly what Alice is bidding.

Several NM commitment schemes have been suggested to prevent such an attack. While the first [37] was highly interactive, the scheme by Di Crescenzo, Ishai and Ostrovsky [35] is the only previous scheme that is non-interactive and can be based on the minimal assumption of existence of one-way functions. This scheme, and later more efficient ones based on special assumptions [36, 41] are all in the common reference string (CRS) model, where it is assumed that players all have access to a string that is guaranteed to be selected with a prescribed distribution. A special case of this is a uniformly random CRS; we call this the uniform reference string (URS) model. Barak [5] has suggested an interactive non-malleable commitment scheme in the plain model without a CRS. It is unknown if non-interactive NM commitment with minimal assumptions is possible without a CRS, but since part of our goal is to look at the relation to universally composable commitments where a CRS is necessary (see below for details) we only consider the CRS model here.

The constructions from [35, 36, 41] were all proven secure according to a definition where the adversary sees *one* commitment from an honest player and then tries to make his own (related) commitment. However, if we consider the motivating example, it is clearly more natural to require non-malleability, even if the adversary gets any polynomial number of commitments as input (the adversary's goal might be to underbid everyone else, for instance). We call this notion of security *reusability*, referring to the fact that in the CRS model, it means that the same CRS can be reused for several commitments from any number of players.

We show that reusability is a strictly stronger notion, for both unconditionally hiding and unconditionally binding schemes. This may be slightly surprising since it was argued in [37] that the corresponding notions for NM *encryption* are equivalent. Unfortunately, the security proofs of [35, 36] break down in the more general setting, and so it might seem that to get reusability, one would have to either use several rounds of interaction [37] or use stronger, non-minimal, assumptions [21].

This chapter builds on the work in [27] where we offered a general technique for constructing reusable, non-interactive NM commitments in the CRS model. The main new technical idea is a way to use any digital signature scheme (even one with rather weak security properties) as a basis for NM commitments. They can therefore be based on any one-way function, but can also be instantiated more efficiently, for instance based on the strong RSA assumption. The version based on general one-way functions also extends to the URS model.

Independently [45] introduced the notion of simulation sound commitments and presented a construction based on the assumption that DSA is secure against chosen message attack. MacKenzie and Yang introduce a weaker notion, tag-based simulation soundness and tag-based non-malleability in [61], and show that the two notions are closely related. They too use signatures in their constructions. While we did not formulate the notion of simulation soundness in [27] the notion is indirectly used, our techniques resemble those of [45] and [61].

Universally composable (UC) commitment is a notion introduced by Canetti in [17], with the first such scheme being suggested by Canetti and Fischlin in [18]. An efficient interactive scheme based on a specialized assumption was suggested by Damgård and Nielsen in [32], while a scheme based on any trapdoor one-way permutation can be found in [21]. In a UC scheme, making a commitment is "equivalent" to giving in private the committed value to a trusted third party, who will then later reveal it on request from the committer. This is a very strong security notion: it implies (reusable) non-malleability, and security against concurrent composition and adaptive adversaries. In fact, there provably does not exist a 2-party UC commitment scheme in the "bare" model where no extra resources are given a priori. However, 2-party schemes *are* possible in the CRS model, and all the schemes mentioned above work in this scenario.

It is easy to see that *non-interactive* UC commitment implies key exchange [21], but previously no consequences of UC commitments in general were known. In this paper, we show that any 2-party UC commitment scheme secure against passive adversaries in the CRS model (interactive or not) implies key exchange, and if it is secure in the URS model then we get oblivious transfer. Key exchange and oblivious transfer are generally regarded as a stronger primitive than one-way functions. For instance, Impagliazzo and Rudich [52] show a relativized separation between them. So, our results can be seen as an indication that UC commitment is a strictly stronger primitive than NM commitment – even if we require the NM scheme to be non-interactive and reusable.

Our last result is an application of our efficient NM commitment scheme to improve the efficient UC commitment scheme from [32], where the size of the CRS must grow linearly with the number of players involved. We show that by combining the two, we obtain an equally efficient UC scheme where the size of the CRS can be independent of the number of players. To prove this we rely on the underlying properties of the commitment scheme that were also used in the proof of NM.

6.2 Definitions

6.2.1 Simulation Soundness

Consider a Pedersen commitment, c. Now, if we see a couple of equivocations of this commitment (m, r) and (m', r'), then we have $g^m h^r = g^{m'} h^{r'} \bmod p$. This gives us $g = h^{\frac{r'-r}{m-m'}} \bmod p$, i.e., we have learned the equivocation key. But this means that we can create commitments ourselves that we can equivocate.

[45, 61] note this problem and suggest simulation sound commitments where access to equivocated commitments does not allow one to equivocate other commitments. The definition can be varied in a couple of ways, we can strengthen it to say that any commitment that has been equivocated at most one time is still binding, this is the definition from [45]. Here we adopt the simpler definition of body-based simulation sound commitments from [61]. Unfortunately, our construction does not satisfy this definition for randomness revealing commitments, so we also define a weaker notion that we call static simulation soundness. This latter notion covers the case where the adversary must choose the commitment to equivocate before getting access to an equivocation oracle.

To define simulation soundness we consider the following two oracles.

Com(pk, ek)**:** Store $(c, e) \leftarrow \widehat{\text{com}}_{pk}(ek)$. Return c.

Dec(pk, ek, c, m)**:** Check if Com has stored a pair (c, e). In that case, return $d \leftarrow \text{equiv}_{pk,ek}(c, e, m)$.

Definition 6.1 (Simulation soundness) *A commitment scheme is simulation sound if for all adversaries $\mathcal{A}$ we have* $\Pr[(pk, ek) \leftarrow \widehat{K}(); (c, d_1, d_2) \leftarrow \mathcal{A}^{\text{Com}(pk,ek),\text{Dec}(pk,ek,\cdot,\cdot)}(pk, z); m_1 \leftarrow \text{dec}_{pk}(c, d_1); m_2 \leftarrow \text{dec}_{pk}(c, d_2) : \perp \neq m_1 \neq m_2 \neq \perp] \approx 0$*, where we demand that c has not been produced by Com(pk, ek).*

Definition 6.2 (Static simulation soundness) *A commitment scheme is static simulation sound if for all adversaries $\mathcal{A}$ we have* $\Pr[(pk, ek) \leftarrow \widehat{K}(); (c, d_1, d_2) \leftarrow \mathcal{A}^{\text{Com}(pk,ek)}(pk); (d_1, d_2) \leftarrow \mathcal{A}^{\text{Dec}(pk,ek,\cdot,\cdot)}(); m_1 \leftarrow \text{dec}_{pk}(c, d_1); m_2 \leftarrow \text{dec}_{pk}(c, d_2) : \perp \neq m_1 \neq m_2 \neq \perp] \approx 0$*, where we require that c was not produced by* Com.

6.2.2 Non-malleable Commitment

Non-malleability is a security notion concerned with man-in-the-middle attacks. With respect to commitments, the intuition is that the execution of some commitment protocols should not affect the execution of other commitment protocols. We capture this in a notion of non-malleability where the adversary does not get an advantage from having access to the execution of commitment protocols compared with the case where the adversary has no such access. In the latter case, we simply let the adversary specify the messages rather than first forming commitments and then opening them later on.

We consider two games. In the first game, we generate a tuple of messages according to some distribution. An adversary receives commitments to these messages and outputs a tuple of commitments himself. After receiving openings of the original commitments, the adversary then tries to open his own commitments in a way such that the contents are related to the original messages. It wins if indeed the messages are related.

In the second game, we generate a tuple of messages according to the same distribution as in the first game. However, this time we do not give the adversary the commitments to the messages. The adversary in the second game must try to output related messages without knowing anything about the original messages.

We wish that for any adversary playing the first game we can find one that fares (almost) as well in the second game. In this case, we will consider the commitment scheme non-malleable. In our case, the fact that we use the CRS model allows us to give some help to the adversary in the second game (without which we could not prove security). More specifically, in the second game we produce the public key for the commitment scheme with an algorithm that also provides some extra information that the adversary can use to its advantage (in this dissertation the extra information will enable it to equivocate commitments).

Let us describe the two games more accurately. In both games, there is a message generator, M. The message generator receives the public key, pk, for the commitment scheme as input and it also gets some auxiliary input z. M returns a value s and a vector of messages $\vec{m}$.

In the first game, we model the adversary $\mathcal{A}$ as a probabilistic polynomial time interactive Turing machine. It learns pk and z. In its firsts invocation $\mathcal{A}$ receives a tuple of commitments $\vec{c}$ to the messages in $\vec{m}$. It responds with $\vec{c}'$, a tuple of elements from $\mathcal{C}_{pk}$. We do not allow $\mathcal{A}$ to directly copy any of the commitments in $\vec{c}$ into $\vec{c}'$. Later $\mathcal{A}$ is activated again, this time receiving a tuple $\vec{d}$ of decommitments to the commitments in $\vec{c}$. It must now try to produce a tuple of decommitments $\vec{d}'$ to its own commitments.

In the second game, we run a *modified key generator* $\widehat{K}$ to generate the public key pk. This key must be indistinguishable from a real key. In addition to the key pk, the modified key generator also produces some extra information s_{pk} about the key.

The adversary in the second game $\mathcal{B}$ is only invoked once. Like $\mathcal{A}$, it learns pk and z. It also gets s_{pk} as input. However, it will not receive any information about the tuple of messages $\vec{m}$ except for the number of messages in the tuple, $t = |\vec{m}|$. It returns a tuple $\vec{m}'$ of messages from $\mathcal{M}_{pk} \cup \{\perp\}$.

We compare the outcome of the two games by running a polynomial time distinguisher D. This distinguisher gets as input $s, \vec{m}, \vec{m}'$, where in the first game $\vec{m}'$ is the resulting vector of messages when running the decommitment algorithm on the tuples $\vec{c}', \vec{d}'$. Note, s contains information from M possibly including the public key pk. The distinguisher returns a single bit, where we interpret 1 as being a success. We demand that the probability of D outputting 1 cannot be increased by changing a message in $\vec{m}'$ to $\perp$. This way the adversary $\mathcal{A}$ cannot get an advantage by deliberately refusing to open its commitments.

Let us write down the probabilities of success in the two games. For the first game we define

$$\begin{aligned}\mathrm{Succ}_{\mathcal{A},M,D}(k,z) \;=\; \Pr[&pk \leftarrow K(); (s,\vec{m}) \leftarrow M(pk,z);\\ &(\vec{c},\vec{d}) \leftarrow \mathrm{com}_{pk}(\vec{m}); \vec{c}' \leftarrow \mathcal{A}(pk,\vec{c},z);\\ &\vec{d}' \leftarrow \mathcal{A}(\vec{d}); \vec{m}' \leftarrow \mathrm{dec}_{pk}(\vec{c},\vec{d}) :\\ &D(s,\vec{m},\vec{m}') = 1],\end{aligned}$$

where we demand that neither of the commitments in $\vec{c}$ is contained in $\vec{c}'$.

In the second game the success probability is given by

$$\begin{aligned}\widehat{\mathrm{Succ}}_{\mathcal{B},M,D}(k,z) \;=\; \Pr[&(pk,s_{pk}) \leftarrow \widehat{K}(); (s,\vec{m}) \leftarrow M(pk,z);\\ &\vec{m}' \leftarrow \mathcal{B}(pk,s_{pk},t,z) : D(s,\vec{m},\vec{m}') = 1],\end{aligned}$$

where $t = |\vec{m}|$, and where we interpret commitments and decommitments of vectors in the natural way.

Definition 6.3 (Non-malleable commitment) *We say a commitment scheme is* non-malleable *if it has a modified key generator $\widehat{K}$ such that for all $\mathcal{A}, M$ there exists a $\mathcal{B}$, such that for all D we have*

$$\mathrm{Succ}_{\mathcal{A},M,D}(k,z) - \widehat{\mathrm{Succ}}_{\mathcal{B},M,D}(k,z) < negl(k)$$

for all z with lengths bounded by some polynomial in k.

We say a commitment scheme is ϵ-non-malleable *if for every $\mathcal{A}$ and ϵ, there exists $\mathcal{B}$ running in time polynomial in k and ϵ^{-1}, such that the above difference is at most $\epsilon + negl(k)$.*

We will later construct ϵ-non-malleable commitment schemes, where ϵ^{-1} may be any positive polynomial in k.

Remark. Our definition does not allow the adversaries to receive any "history", i.e., side information about the messages they receive commitment to. Like [35] and [36] we do not know how to achieve such security. However, in all cases where messages consistent with the side information can be sampled efficiently, our security proofs remain valid.

6.2.3 Comparison with Other Definitions of Non-malleability

As mentioned before non-malleability is a concept introduced in [37]. Their goal is to avoid man-in-the-middle attacks. Therefore, they define a protocol as being non-malleable if the adversary seeing a commitment cannot commit to a related value. [41] call this non-malleability with respect to commitment.

Unfortunately, this definition does not make much sense when considering unconditionally hiding commitments since such a commitment does not define any content by itself. Following the definition of [35, 36], we choose to consider the content of a commitment as what it is opened to. Such a definition is called non-malleability with respect to opening in [41]. For unconditionally binding commitments, non-malleability with respect to commitment is a stronger notion than non-malleability with respect to opening.

Our definition is stronger than the one in [35, 36]. Any commitment scheme that is non-malleable according to our definition is also non-malleable according to their definition. The main reason for our modification, reusability, was mentioned in the introduction: rather than just mauling one commitment the adversary may very well be attempting to maul many commitments at the same time. We show later that our definition is strictly stronger, there are schemes secure according to the [35, 36] definitions that are not secure according to our definition.

Increasing the number of commitments the adversary may see and may produce is not the only modification we have made. We give the message generator access to the public key for the commitment scheme. In our opinion, this is reasonable since in real life messages may indeed depend on this key. It does turn out that we pay a price for this though, since it forces us to give $\mathcal{B}$ access to some side information about the public key. In [35, 36, 61] this was not needed since they could simply let $\mathcal{B}$ choose a completely new public key and still work within the same message space since it was independent of the key.

We have also changed the notation. [35, 36, 61] speak of a distribution D instead of a messages generator M, and a relation approximator R instead of a distinguisher D. This change is purely cosmetic, but our notation seems to be more in line with other cryptographic literature.

Other definitions [41, 5] deal with interactive commitments. Barak [5] deals with a general method of setting up a URS interactively in a way such that a

subsequent execution of a non-malleable commitment scheme based on a URS will lead to a non-malleable interactive commitment. One variation of our scheme is set in the URS model and can therefore be used with Barak's compiler.

We note that the most secure version of commitment is universally composable commitments [18, 21, 32]. The scheme presented in [21] is both non-interactive and universally composable. However, it is based on the assumption that trap-door permutations exist. We provide some evidence in Section 6.5 that UC commitment must be based on stronger assumptions than NM commitment.

6.2.4 Comparison with Non-reusable NM

In this section we will be informal and call a commitment scheme (t, u)-NM if it is non-malleable (or ϵ-non-malleable for small ϵ's) in the case where the message generator produces a message vector of length t and the adversary produces a message of length u.

We will argue that $(1, 1)$-NM implies neither $(t > 1, 1)$-NM nor $(1, u > 1)$-NM. We give counterexamples both in the unconditionally hiding case and the unconditionally binding case.

Consider the unconditionally hiding commitment scheme from [36], which was proven $(1, 1)$-NM. This scheme is actually $(1, u)$-NM for any $u > 1$. However, for $(t, 1)$-NM the security proof fails for $t > 1$. It is not known whether the scheme is even $(2, 1)$-NM.

We can construct a variant of the scheme that is provably not $(2, 1)$-NM. We simply select two keys pk_1, pk_2 for the scheme, and a commitment to a message m now consists of the pair $(\mathrm{com}_{pk_1}(m), \mathrm{com}_{pk_2}(m))$. By running the proof from [36] "twice in parallel", we can prove that this commitment scheme is unconditionally hiding and $(1, 1)$-NM. However, consider a message generator M that simply outputs a message vector (m, m). Given commitments (c_1, c_2) and (c'_1, c'_2) to this message vector we may form the commitment (c_1, c'_2). After seeing openings to the original commitments, it is easy to open this commitment to the message m. In the first game, the adversary may therefore have success probability 1 in trying to commit to the same message. In the second game, however, we do not get any help in producing the message, and we can only try to guess m. The scheme is therefore not $(2, 1)$-NM.

All $(1, 1)$-NM commitment schemes we know of in the literature are in fact $(1, u)$-NM. However, we can also construct an example showing that this need not be the case. Again, we pick two public keys pk_1, pk_2 for the commitment scheme in [36]. A commitment to message m is formed as $(\mathrm{com}_{pk_1}(r), \mathrm{com}_{pk_2}(m \oplus r))$, where r is chosen at random. Again, it can be shown that this scheme is an unconditionally hiding $(1, 1)$-NM commitment scheme. Consider, however, an adversary in the first game getting such a commitment (c_1, c_2). He form the commitments $(\mathrm{com}_{pk_1}(0), c_2)$ and $(c_1, \mathrm{com}_{pk_2}(0))$. Later when receiving an opening of (c_1, c_2), he can open his commitments to $m_1 = r, m_2 = r \oplus m$. The

exclusive-OR of these messages is m. On the other hand, any adversary in the second game has to guess m to form a pair of messages m_1, m_2 with $m = m_1 \oplus m_2$.

To show that for unconditionally binding commitment schemes $(1,1)$-NM does not imply $(t > 1, 1)$-NM we use a semantically secure encryption scheme with errorless decryption. We construct a $(1,1)$-NM commitment scheme the following way. As public key, we generate $2k$ public keys $pk_{1,0}, pk_{1,1}, \ldots, pk_{k,0}, pk_{k,1}$. When having to commit to a message m the sender selects a k-bit public key vk for a one-time signature scheme. For each $i = 1, \ldots, k$ he encrypts m using pk_{i,vk_i}, where vk_i is the i'th bit of vk. The commitment now consists of the k ciphertexts, the public verification key for the signature scheme, and a signature on all of it.

To see that this scheme is $(1,1)$-NM for any u, consider an adversary $\mathcal{A}$ that upon seeing a commitment to some message m generates a commitment to a related message m'. Certainly, he cannot use the same public verification key since he does not know how to make signatures with this key. Therefore, he must produce at least one ciphertext with a previously unused public key.

We may now use $\mathcal{A}$ to break the semantic security of the cryptosystem. Given k ciphertexts under keys $pk_1, \ldots, pk_k$ of a message m (this is still semantically secure) we may select at random public keys $pk'_1, \ldots, pk'_k$, where we know the corresponding secret keys. We then generate a key vk for the signature scheme and arrange the keys $pk_1, \ldots, pk_k, pk'_1, \ldots, pk'_k$ in a pattern so that $pk_{i,vk_i} = pk_i$, while $pk_{i,1-vk_i} = pk'_i$. This way we can transform the ciphertexts into what looks like a commitment. Giving the adversary the commitment and the public key just constructed we let him form a commitment. Since his commitment uses one of the keys we formed ourselves, we may decrypt his commitment. If this commitment has any relation to the original ciphertexts, this means that we have broken the semantic security of the cryptosystem. Therefore, the commitment scheme must be non-malleable.

The commitment scheme is not $(k,1)$-NM though. If the message generator outputs a message vector with k identical messages, then with overwhelming probability we will know encryptions of the message under all $2k$ keys, and therefore we may easily pick a new one-time signature key ourselves and form a commitment to the message. An adversary without access to commitments does not have the same easy time creating a commitment to the message in the message vector from the message generator.

Finally, we may argue as in [37] that $(1,1)$-NM does not imply $(1,2)$-NM for unconditionally binding commitment schemes. Here we simply take a non-malleable cryptosystem with errorless encryption, pick two public keys pk_1 and pk_2, and form a commitment to m as $(E_{pk_1}(r), E_{pk_2}(m \oplus r))$. It can be proved that this commitment scheme is $(1,1)$-NM. However, by the same argument as in the unconditionally hiding case it is not $(1,2)$-NM.

6.3 A Framework for Constructing Non-malleable Commitment

Instead of aiming directly for non-malleability, we try to construct a static simulation sound commitment scheme. Theorem 6.1 below states that this is sufficient for obtaining ϵ-non-malleability.

Theorem 6.1 *A static simulation sound commitment scheme is ϵ-non-malleable for any ϵ^{-1} polynomial in k.*

Proof. We let the modified key generator $\widehat{K}$ be the same as the modified key generator used for equivocation. In other words, along with the public key pk it also provides as side information the equivocation key ek. Static simulation soundness says that the adversary $\mathcal{A}$ cannot generate a commitment that it can open in two different ways, even when being able to see equivocations of other commitments after it has generated its own commitment. We use this fact when constructing $\mathcal{B}$ and later proving that $\mathcal{B}$ fares as well as $\mathcal{A}$.

First, let us look at the experiment with the real commitment key and the adversary $\mathcal{A}$. We modify this experiment by letting the public key be generated by $\widehat{K}$ and forming equivocable commitments instead of real commitments. We then use the equivocation information to open the commitments to the messages chosen by M. By definition of equivocability, these two experiments are indistinguishable. Our task is therefore reduced to find an algorithm $\mathcal{B}$ that can make $\widehat{\text{Succ}}_{\mathcal{B},M,D}$ at least as high as the success probability in the modified experiment minus ϵ.

We now describe the algorithm $\mathcal{B}$. It runs a copy of $\mathcal{A}$ and starts by giving a vector of t equivocable commitments to $\mathcal{A}$. As a response, $\mathcal{A}$ produces a vector of commitments. As mentioned. the intuition is that $\mathcal{A}$ cannot open the commitments he produced in more than one possible way. Our goal is therefore to find out the contents of as many of $\mathcal{A}$'s commitments as possible, and we will then submit this information to D.

To this end $\mathcal{B}$ runs $M(pk, z)$ $4k\text{time}_M(k)\text{time}_{\mathcal{A}}(k)/\epsilon^2$ times. It picks out the first $2k\text{time}_{\mathcal{A}}(k)/\epsilon$ of the message vectors that have length t. We note that in case there is more than $\epsilon/(2 \cdot \text{time}_M(k))$ chance that M will output a vector of length t there is also overwhelming chance that $\mathcal{B}$ samples enough vectors of length t. The probability of being in a situation where $\mathcal{B}$ cannot sample enough vectors of length t is therefore less than $\epsilon/2 + negl(k)$ since $\text{time}_M(k)$ is an upper bound on t. In the following, we only investigate the experiment conditioned on having sampled enough message vectors.

For each message tuple $\vec{m}$ sampled from M, $\mathcal{B}$ now equivocates the commitments it gave to (its copy of) $\mathcal{A}$, such that $\vec{m}$ is opened. For each of its own commitments $\mathcal{A}$ may or may not in each run produce an opening to a message

$m' \neq \perp$. For each commitment, $\mathcal{B}$ takes the first opening different from $\perp$. Finally, $\mathcal{B}$ hands the message vector $\vec{m}^*$ found to the distinguisher (putting $\perp$ in unopened positions).

This ends the description of $\mathcal{B}$. We observe that $\mathcal{A}$ simulated by $\mathcal{B}$ sees exactly the same distribution as $\mathcal{A}$ does in the modified experiment, where we condition both experiments on M outputting a message vector with a length t (and where t is such that the probability of getting length t is at least $\epsilon/2\text{time}_M(k)$). More precisely, the two games behave in exactly the same way up to the point where $\mathcal{A}$ is about to receive the decommitment of $\vec{m}$. At this point, we can either play the modified experiment, i.e., give $\mathcal{A}$ the "right" $\vec{d}$ and have him produce $\vec{m}'$. Or we can execute $\mathcal{B}$'s algorithm producing $\vec{m}^*$.

Now fix a "snapshot" of all variables known to $\mathcal{A}$ just after it has formed its commitments. Consider any single commitment $\mathcal{A}$ made. If the probability (taken over the choice of $\vec{m}$) that $\mathcal{A}$ will open it is smaller than $\epsilon/(2 \cdot \text{time}_A(k))$, we say the commitment is *bad*, otherwise it is *good*. Since $\text{time}_A(k)$ is an upper bound on the number of commitments $\mathcal{A}$ can produce, the probability that $\vec{m}'$ contains non-$\perp$ values for any of the bad commitments is at most $\epsilon/2$. On the other hand, since $\mathcal{B}$ uses a total of $2k\text{time}_{\mathcal{A}}(k)/\epsilon$ samples of $\vec{m}$-values, the probability that $\vec{m}^*$ contains non-$\perp$ values for all good commitments is overwhelming (it may contain more non-$\perp$ values, but this does not matter since the probability of the distinguisher outputting success cannot be decreased by replacing $\perp$'s with messages).

Summarizing what we have so far, except with probability $\epsilon + negl(k)$, $\mathcal{B}$ manages to sample enough $\vec{m}$-values, $\mathcal{A}$ does not open any of the bad commitments in the modified real life game, and $\mathcal{B}$ learns a way to open all good commitments. Hence, the only way in which $\mathcal{B}$ could do worse than $\mathcal{A}$ is if, even assuming all the above, D outputs 1 with significantly larger probability when seeing $\vec{m}'$, than when seeing $\vec{m}^*$. In particular this means that for at least one good commitment $\vec{m}'$ contains m', and $\vec{m}^*$ contains m^* such that $m' \neq m^*$ and $m', m^* \neq \perp$. This contradicts the static simulation soundness property.

Overall, $\mathcal{B}$ does worse than $\mathcal{A}$ in the real experiment only in case of events that occur with total probability at most $\epsilon + negl(k)$. □

6.3.1 A Non-malleable Commitment Scheme

We make use of two signature schemes. One of them is a signature scheme that is secure against known message attack, where the distribution of messages is the verification keys of the other signature scheme. The other signature scheme is a one-time signature scheme secure against known message attack, where the distribution of messages consists of initial messages a for knowledge of a signature under the first signature scheme.

We present the general structure of a commitment scheme in Figure 6.1.

This commitment scheme is binding. If we could open a commitment in two

Key Generation: Generate keys (vk, sk) for a standard signature scheme secure against known message attack.

If needed generate also a common reference string σ for the Σ-protocol for proving knowledge of a signature.

Let h be a collision-free function.

The public key is $pk = (vk, \sigma, h)$.

Commitment: To commit to a message m we first choose keys (vk_{OT}, sk_{OT}) for the one-time signature scheme. Then we simulate an argument of knowledge of a signature on vk_{OT} with $h(m)$ being the challenge. I.e., we set $(a, z) \leftarrow S(\sigma, (vk_{OT}, vk), h(m))$. Finally, we make a one-time signature s_{OT} on a.

The commitment is $c = (vk_{OT}, a, s_{OT})$, while the decommitment information is (m, z).

Decommitment: We verify the signature s_{OT} on a. Next we verify that the argument $(\sigma, a, h(m), z)$ is acceptable. If both checks succeed we output m, otherwise we output $\perp$.

Figure 6.1: Non-malleable Commitment Scheme

different ways, then we could by the special soundness of the Σ-protocol find a signature on $h(m)$, which is infeasible since we do not know the signature key.

The commitment scheme is also hiding. The only possible link to information about m is a. Since the Σ-protocol is special honest verifier zero-knowledge this a is indistinguishable from the corresponding value occurring in a real conversation – but in real conversations, a is independent of m.

Remark. The only part of the commitment scheme that can contain any information about m is the initial message a of the simulated proof. Therefore, if the proof system is statistically or perfectly special honest verifier zero-knowledge then the commitment is unconditionally hiding.

Theorem 6.2 *The commitment scheme in Figure 6.1 is equivocable.*

Proof. We let the modified key generator $\widehat{K}$ be one that runs K but as side information outputs the signature key sk associated with the verification key. This means that we can sign any message.

With this information, we can of course sign any vk_{OT}. This means, no matter the vk_{OT} we can, in a real zero-knowledge argument, answer the challenge $h(m)$ in the Σ-protocol. Therefore, we simply modify the commitment procedure to pick a according to the Σ-protocol and to return also the equivocation information s, a signature on vk_{OT}.

When having to produce the decommitment information we then open as usual, except we compute the answer z to the challenge $h(m)$ with our knowledge of the signature on vk_{OT}.

To see that this gives us the algorithms needed for the commitment scheme note first that the public keys generated by K and $\widehat{K}$ are identically distributed.

Imagine now that we could distinguish real commitments and equivocated commitments. By a hybrid argument, we can reduce this to distinguishing two scenarios that only differ in one commitment being equivocated instead of being formed and opened according to the commitment scheme.

But this means that we can distinguish simulated and real arguments using the Σ-protocol, since this is the only difference between equivocable and real commitments. We thus arrive at a contradiction with the special honest verifier zero-knowledge property of the Σ-protocol. □

Theorem 6.3 *The commitment scheme in Figure 6.1 is simulation sound.*

Proof. Consider a commitment (vk_{OT}, a, s_{OT}) produced by the adversary. If the adversary can open this commitment to two different messages m_1 and m_2, then it can answer the two challenges $h(m_1)$ and $h(m_2)$. By the collision-freeness of the hash-function, $h(m_1)$ and $h(m_2)$ are different. The special soundness property of the Σ-protocol therefore gives us a signature on vk_{OT}. If vk_{OT} has not been output by Com, then we have produced a signature on a new vk_{OT}. But Com only forms signatures on correctly generated vk_{OT}'s it has selected itself. Therefore, we have formed a new signature under a known message attack, which by our assumption is infeasible.

We are left to consider vk_{OT}'s that have been produced by Com. However, then $\mathcal{A}$ must produce a one-time signature on a. If $\mathcal{A}$ is to produce a non-trivial one-time signature, then it must have seen at least two signatures already. If Dec has been invoked at most once on (vk_{OT}, a) we are therefore safe from double openings. □

Corollary 6.1 *The commitment scheme in Figure 6.1 is ϵ-non-malleable.*

Strong simulation soundness. A slight modification of the scheme in Figure 6.1 yields stronger notions of simulation soundness. Namely, instead of using the strong one-time signature to sign a when making the commitment, we wait until the opening phase and sign the entire opening. We do not need the one-time signature to be strong in this case, on the other hand we do need it to be secure against a one-time chosen message attack. Now, the adversary cannot equivocate a commitment even if it has been equivocated once by the decommitment oracle. This construction therefore satisfies the stronger variation of simulation soundness suggested in [45]. If we let the initial signature scheme be secure against general adaptive chosen message attack we can even equivocate a commitment as many times as we want without the adversary being able to open it to a new message.

6.3.2 Non-malleable Commitment with Randomness Opening

Unfortunately, if we strengthen the requirement on the commitment scheme to demand that the opening reveals all the randomness used then Theorem 6.3 does not hold any more. One of the problems is that if we reveal the randomness used to generate the one-time signature key, then the adversary can sign anything under this key. It can therefore query Com for a commitment, ask Dec for a couple of equivocations and from the special soundness property of the Σ-protocol it now knows a signature on vk_{OT}. With knowledge of a signature on vk_{OT} as well as the corresponding signing key sk_{OT} it is easy to generate commitments that can be opened in many ways. A similar attack suggested to us by MacKenzie and Yang [60] shows that Lemma 5 as originally stated in [27] is false, the construction in that paper is not simulation sound.

Not all is lost though. In the following, we suggest a similar commitment scheme that does include the entire randomness in the opening. We place a restriction on the Σ-protocol used in Figure 6.1. Consider an argument $(a, h(m), z)$ generated by a real prover who knows the witness w. By the special honest verifier zero-knowledge property, this argument is indistinguishable from an argument made by the simulator. We require that it is also possible to simulate the randomness used by the simulator. I.e., the prover can generate the argument $(a, h(m), z)$ and generate randomness r, such that the simulator running with r and challenge $h(m)$ would end up producing the argument $(a, h(m), z)$. The tuples $(a, h(m), z, r)$ generated by either the prover or the simulator must be indistinguishable. For the Σ-protocols we can think of this property does hold, in particular it holds for the Σ-protocols used in this chapter.[1]

Theorem 6.4 *Instantiating the commitment scheme in Figure 6.1 with a Σ-protocol that has simulatable randomness for the simulator yields a static simulation sound commitment scheme.*

Proof. Let us look at a commitment $c = (vk_{OT}, a, s_{OT})$ produced by $\mathcal{A}$, which is different from any commitment produced by Com. We first argue that vk_{OT} must be different from the vk_{OT}'s of the equivocable commitments the adversary has seen. Com only generates signatures on initial messages a that are chosen with a distribution that is indistinguishable from the distribution of simulated initial messages a. Therefore, essentially we have access to a known one-time signature attack through Com. Any reuse of vk_{OT} would therefore imply a violation of the strong security against existential forgery that the signature scheme is supposed to have.

[1] It is easy to come up with probable counterexamples though. Consider for instance a simulator that uses its randomizer as a seed to a pseudo-random number generator and then uses pseudo-random numbers to generate the simulated argument.

Suppose now that $\mathcal{A}$ could open c to two different messages m_1 and m_2. By the collision-freeness property, $h(m_1)$ and $h(m_2)$ are different. The special soundness property of the Σ-protocol therefore gives us a signature on vk_{OT}. In Com we have generated signatures on different public keys for the strong one-time signature scheme, however, by construction of Com these have all been correctly generated. In other words, we have forged a signature on vk_{OT} under a known message attack. □

Corollary 6.2 *The randomness revealing commitment scheme obtained by instantiating Figure 6.1 with a Σ-protocol that has simulatable randomness for the simulator is ϵ-non-malleable.*

Comparison with the scheme in [27]. The construction of a non-malleable commitment in [27] bears much resemblance with the randomness revealing commitment just suggested, and in a similar manner it is proven static simulation sound. Instead of using a strong one-time signature in [27] we commit to an authentication key and simulate an argument of knowledge of a signature on that commitment. Using the authentication key we then form a message authentication code on the initial message a. Once the commitment is opened, anybody can verify the message authentication code. On the other hand, before opening the commitment the hiding property of the commitment scheme makes it impossible for an adversary to reuse the initial commitment since he does not know the authentication key inside it and thus cannot form a message authentication code on some simulated initial message a.

6.4 Construction of Non-Malleable Commitment Schemes

6.4.1 An Implementation Based on any One-Way Function

We let h be the identity-function. Since it is injective, it is also collision-free.

Signatures can be based on one-way functions. For instance, we may modify the Merkle one-time signature scheme the following way. We form a balanced binary tree of height k. Our public key consists of a key for a one-time signature on the root of the tree. A message to be signed is placed at a previously unused leaf, and the signature consists of one-time signatures and public keys on the path up to the root. At all internal nodes, we sign the public one-time signature keys of the children. Finally, for any internal node we have used we store the public keys chosen for the two children and always use these keys when passing by a node again. This signature scheme is existential forgery secure against adaptive chosen message attack, more than enough for our purpose. To avoid storing the

state we can use pseudorandom functions to compute the keys in the nodes of the tree.

A Σ-protocol exists for this signature scheme, simply because the known zero-knowledge protocols for NP-complete problems have the right form: since deciding validity of a signature is in NP, we can transform our problem to, say, a Hamiltonian path problem and use Blum's zero-knowledge interactive proof system for this language. As a building block, we need bit-commitment, which we can get from any one-way function [63]. Although this commitment scheme is originally interactive, it can be made non-interactive in the CRS model, following [35]. In its basic form, this is a 3-move protocol with soundness error 1/2, and we repeat this in parallel k times. It is easy to see (and well-known) that this will satisfy our conditions for the Σ-protocol.

The commitment scheme in [63] has commitments that are indistinguishable from a random string. In other words, we can obliviously choose something which looks like a commitment but which we cannot open to anything. In our case, this is interesting because it means that the Σ-protocol has simulatable randomness for the simulator. For any commitment in the Σ-protocol above that is not opened, we can simply claim that it is a string chosen at random by the simulator. With the implementation described here, we therefore get a randomness revealing ϵ-non-malleable commitment.

Remark. Most cryptographic tools based on simpler primitives are black-box constructions. It is therefore worth noting that our construction is not a black-box construction. The reason for this is that we use general reduction techniques to form an NP problem and then create a Σ-protocol for this. This reduction depends on the one-way function that we use.

6.4.2 Unconditional Hiding, Unconditional Binding and Uniform Random String

It is not known whether unconditionally hiding commitment schemes can be constructed from one-way functions. Marc Fischlin has recently shown that there is no proof of one-way functions implying unconditionally hiding commitments that relativizes [40]. However, assuming the minimal possible, namely that we have an unconditionally hiding bit-commitment scheme, we note that it can be transformed into an unconditionally hiding non-malleable and equivocable bit-commitment scheme. This is done by using the unconditionally hiding commitment scheme to make all the commitments in the Σ-protocol in the previous section. This way, even a computationally unbounded adversary seeing a, has no idea about how the sender is capable of opening the commitments, and thus no idea about the bits of m.

Since we can make unconditionally binding commitment schemes from one-

way functions, an obvious question is whether we can use our construction to get an unconditionally binding non-malleable commitment scheme. The answer to this question is yes.[2] The idea is to modify the commitment scheme we constructed before so the public key now also includes an unconditionally binding commitment to a bit $b = 0$. We may from one-way functions construct a Σ-protocol for proving that the commitment contains $b = 1$ with challenge m in addition to the proof of knowledge of a signature. Obviously, this is a false statement, so it is impossible for even a computationally unbounded sender to find a witness for this. Therefore, by the special soundness of the Σ-protocol he cannot open the commitment to reveal two different messages. However, since the commitment to b is hiding, there is no problem in simulating a proof for $b = 1$, nobody will notice that anything is wrong. For non-malleability we let the modified key-generator output a public key with a commitment to $b = 1$. Now we may equivocate commitments and therefore the simulation proof where we use rewinding to make the adversary open his commitments goes through.

The CRS used in the one-way function based commitment scheme is not uniformly random in its basic form, but we can modify the scheme so it can use the URS model instead. By backtracking through the papers constructing the tools we use we note that all of them can be built from a one-way function and uniformly random bits. The only exception is the signature scheme, where we do not know whether the verification key can be chosen uniformly at random. Nevertheless, we can get by with interpreting some of the uniformly random bits as an unconditionally binding commitment to a verification key. We can modify the Σ-protocol to the case where we prove that we know an opening of the bits to a public verification key and can sign vk_{OT} with this key. In reality we do not have such a key, however, since the commitment scheme is hiding the security proof goes through.

6.4.3 An Implementation Based on the Strong RSA Assumption

Based on the RSA assumption we can construct a strong one-time signature in the following way. We let q be a large prime and we let n be an RSA modulus. We use a universal one-way hash-function h_{OT} that maps its inputs to $\mathbb{Z}_q$. We select at random elements g, h in $\mathbb{Z}_n$ and set $G = g^q \bmod n, H = h^q \bmod n$. The public key is (n, q, G, H). A signature on message a is the element $Z = g^{h_{OT}(a)}h \bmod n$. To verify a signature check that $Z^q = G^{h_{OT}(a)}H \bmod n$.

The signature can be seen as an honest verifier zero-knowledge argument of knowledge of a q-root of G. To argue security against known message attack consider an adversary that can find a signature Z' on a message m' after having

[2]Of course, the non-malleability cannot hold against a computationally unbounded adversary. Only the binding property holds against a computationally unbounded adversary.

seen a known message attack giving him signature (m, Z). We can use this adversary to break the RSA-assumption in the following way. We let G be the challenge that we wish to find a q-root of. We select a message a at random from the distribution and choose Z at random. We set $H = Z^q G^{-h_{OT}(a)} \bmod n$. Now we have $Z^q = G^{h_{OT}(a)} H \bmod n$ and $(Z')^q = G^{h_{OT}(a')} H$ giving us $(Z/Z')^q = G^{h_{OT}(a') - h_{OT}(a)} \bmod n$. We have $\gcd(h_{OT}(a') - h_{OT}(a), q) = 1$ since q is prime and larger than both $h_{OT}(a)$ and $h_{OT}(a')$ and since it is infeasible for the adversary to find a collision with a. Pick α, β so $\alpha(h_{OT}(a) - h_{OT}(a')) + \beta q = 1$. We have $(G^\beta (Z/Z')^\alpha)^q = G \bmod n$, i.e., we have broken the RSA assumption.

We also need to suggest a signature scheme secure against known message attack. Here we let the signature scheme be the following. We select an RSA-modulus n and random elements x, g. We also select a hash-function h_{sign} that maps its inputs down to ℓ-bit messages. A signature on vk_{OT} then consists of (y, e) where e is chosen as a random $\ell + 1$-bit prime and $y^e = x g^{h_{sign}(vk_{OT})} \bmod n$. The signature can be verified by checking that e is an odd $\ell + 1$ bit number and $y^e = x g^{h_{sign}(vk_{OT})} \bmod n$. Using techniques from [40] it is straightforward to see that the strong RSA assumption implies that the signature scheme is secure against existential forgery under known message attack.

We are not going to hide e. What we need then is an honest verifier zero-knowledge argument of knowledge of a signature on vk_{OT} using exponent e. We can do this by picking α at random and setting $a = \alpha^e \bmod n$. On challenge $h_{sign}(vk_{OT})$ we answer with $z = y^{h_{sign}(vk_{OT})} \alpha \bmod n$. To simulate the argument with challenge $h(m)$ we pick z at random and set $a = z^e (x g^{h_{sign}(vk_{OT})})^{-h(m)} \bmod n$.

To save some computations we can use the same n for both the strong one-time signature and standard signature secure against known message attacks.

If we pick the hash-functions as injective functions then we get a non-malleable commitment scheme for k-bit messages with security based on the strong RSA assumption. In comparison with the somewhat similar scheme in [27], we do not have to rely on an additional assumption on the maximal distance between consecutive primes.

If we pick the hash-functions as cryptographic hashes that hash down to, say, 160 bits, then we have a very efficient non-malleable commitment scheme. It takes five short exponentiations to form a commitment and similarly 5 short exponentiations to decommit.

6.5 UC Commitment implies Key Exchange or Oblivious Transfer

We briefly recall how the UC framework [17] is put together. The framework allows us to say that a protocol π securely implements an ideal functionality $\mathcal{F}$.

In the case of UC commitment, $\mathcal{F}$ will be a trusted party that receives in private the committed value m, and later reveals it to the receiver on request from the sender. The real-life protocol π is attacked by an adversary $\mathcal{A}$ who schedules communication and adaptively corrupts players and controls their actions, while an attack in the ideal setting (where $\mathcal{F}$ is present) is limited to corrupting players and submit inputs on their behalf to $\mathcal{F}$, and block/observe the outputs from $\mathcal{F}$. Security of π now means that for every efficient real-life adversary $\mathcal{A}$, there is an efficient ideal model adversary $\mathcal{S}$ who can obtain "the same" as $\mathcal{A}$. To define what this means, an environment machine $\mathcal{Z}$ is used. $\mathcal{Z}$ may communicate with the adversary ($\mathcal{A}$ or $\mathcal{S}$) at any point during the attack, and it may specify inputs for the honest players and see the results they obtain. Security more precisely means that it is infeasible for $\mathcal{Z}$ to tell if it is talking to $\mathcal{A}$ in a real-life attack, or to $\mathcal{S}$ in an ideal model attack. So to prove UC security, one must construct, given $\mathcal{A}$, a suitable $\mathcal{S}$. It is important to note that, in order to ensure robustness against concurrent composition and adaptive security, the model explicitly forbids rewinding of $\mathcal{Z}$, so $\mathcal{S}$ is forced to go through the entire game in the same time sequence as in real life.

Theorem 6.5 *If there exists a UC commitment scheme in the CRS model secure against non-adaptive passive adversaries, then there exists a key exchange protocol secure against passive attacks. Furthermore, if there exists a UC commitment scheme in the URS model secure against non-adaptive passive adversaries then there exists an oblivious transfer protocol secure against passive adversaries.*

Sketch of proof. Assume we have a UC commitment scheme for committer C and recipient R, and let π be the protocol used when C commits to, say, a bit b. Let $\mathcal{A}$ be the adversary that first corrupts C, it then sends the CRS to $\mathcal{Z}$, and now (through its communication with $\mathcal{Z}$) lets $\mathcal{Z}$ decide the actions of C. Let $\mathcal{Z}$ be the environment that expects the behavior we just specified for the adversary (this means that $\mathcal{Z}$ gets to play C's part of the protocol). $\mathcal{Z}$ follows the honest protocol for C, in order to commit to some bit b. Once this is over, it asks to have the commitment opened, again $\mathcal{Z}$ plays honestly C's part of the opening, and it receives as a result of this the bit that R gets as output (which should normally be equal to b). Finally, $\mathcal{Z}$ tries to distinguish whether it is in the real-life model or the ideal process by outputting a bit.

UC security guarantees that there exists a good ideal model adversary $\mathcal{S}$ for this $\mathcal{A}$. Of course, $\mathcal{S}$ must send a CRS to $\mathcal{Z}$ and then play in an indistinguishable way R's part of π against $\mathcal{Z}$ – otherwise $\mathcal{Z}$ could immediately distinguish. Moreover, after completing π, $\mathcal{S}$ must be able to compute the bit b that $\mathcal{Z}$ "committed to", since this bit must be given to $\mathcal{F}$ before opening can take place in the ideal model. If this bit is not correct, R at opening time receives a bit different from b in the ideal model. This would allow $\mathcal{Z}$ to distinguish.

This observation immediately implies a secure key exchange protocol for parties A, B: B starts an instance of $\mathcal{S}$ and sends the CRS it produces to A, who

then chooses a random bit b and runs the commitment protocol acting as the sender. B lets $\mathcal{S}$ play the receivers part, and can now extract b by the observation above. An eavesdropper is clearly in a situation that is no better than that of an honest receiver in the non-adaptive UC commitment scheme, and so he cannot distinguish $b = 0$ from $b = 1$.

If we are in the URS[3] model, we can build an OT protocol for sender A with input bits b_0, b_1 and receiver B with selection bit c. The goal is for B to learn b_c and nothing else, whereas A should learn nothing new. The OT protocol goes a follows

1. B computes and sends to A strings CRS_0, CRS_1, where he runs $\mathcal{S}$ to get CRS_c and chooses CRS_{1-c} at random.

2. A executes the commitment protocol π twice, playing the role of C using CRS_0, respectively CRS_1 as reference string and b_0 respectively b_1 as the bit to commit to. B follows the protocol for R in the instance that uses CRS_{1-c}, and lets $\mathcal{S}$ conduct the protocol in the other case, i.e., he simply relays messages between A and $\mathcal{S}$.

3. When $\mathcal{S}$ outputs a bit (to give to $\mathcal{F}$), B uses this as his output.

From A's point of view, the two instances of π are indistinguishable, since otherwise $\mathcal{S}$ would not satisfy the conditions in the UC definition, as we discussed. Hence, A learns nothing new. On the other hand, we also argued that $\mathcal{S}$ must output the bit committed to, so B does indeed learn b_c. With respect to b_{1-c}, B is in exactly the same position as party R when receiving a commitment. Since UC commitments are hiding, B does not learn b_{1-c}. □

Consequences of adaptive UC commitment. If we have UC commitment secure against passive adaptive adversaries in the URS model, then by a similar argument to the one above we get oblivious transfer secure against passive adaptive adversaries. This in turn means that any well-formed ideal functionality can be securely realized against adaptive passive adversaries in the URS model [21].

Furthermore, if we have a UC commitment secure against active adaptive adversaries, then we can use it to build UC zero-knowledge proof of knowledge as in [18]. As shown in [21] UC zero-knowledge proof of knowledge implies UC combined commitment and zero-knowledge proof of knowledge. The construction in [21] that combines the commit and prove functionality with UC oblivious transfer secure against passive adaptive adversaries can then be used to realize any well-formed ideal functionality in the URS model against adaptive and active adversaries.

[3]We need to ensure that B can generate the CRS without getting any information from the key generation that allows it to break the OT protocol. Since it is easy to generate a URS at random, we certainly are guaranteed this in the URS model.

6.6 Application to UC Commitment

6.6.1 Damgård-Nielsen UC Commitment

Let us briefly sketch why the scheme of [32] get a CRS that grows linearly with the number of players.

The UC commitment scheme is based on a *mixed commitment scheme*. This is a commitment scheme where we first generate a master key N that defines the message space $\mathcal{M}_N$ of the commitment scheme. Knowing N we may now generate a public key K for the commitment scheme. This key will belong to a group $\mathcal{K}_N$. If chosen at random, this key with overwhelming probability will be an extraction key X-key. However, we may also select the key as an equivocation key E-key. X-keys and E-keys are indistinguishable. Knowing a trapdoor t_N associated to the master key, we may extract the contents of commitments formed under X-keys. On the other hand, if we use an E-key K to commit, with some associated trapdoor information t_K we may equivocate the commitment to anything.

The UC commitment scheme is interactive. The sender and receiver run a two round coin-flip protocol to determine a public key K for the mixed commitment scheme. For this purpose, they have an equivocable commitment scheme. The sender commits to $K_1 \in_R \mathcal{K}_N$, the receiver sends back $K_2 \in_R \mathcal{K}_N$, and the sender in the third round opens his initial commitment so both the sender and receiver now knows the key $K = K_1 + K_2$. In the third round, the sender also sends a commitment to the message m under key K. To open the commitment the sender just has to open the mixed commitment to m.

In the UC commitment scheme the ideal process adversary $\mathcal{S}$ runs a simulated real life execution where it uses $\mathcal{A}$ as a black box to produce communication with $\mathcal{Z}$. It faces two problems: When a message is submitted to $\mathcal{F}$ it must simulate this without knowing the actual message. And when $\mathcal{A}$ sends a UC commitment on behalf of a corrupted party, then it must figure out which message to give to $\mathcal{F}$. These problems can be solved if we commit to the message using an E-key whenever sending a UC commitment on behalf of an honest party, and force $\mathcal{A}$ to use an X-key whenever it makes a UC commitment. We enable $\mathcal{S}$ to put itself in this situation by giving it equivocation information for the initial commitment scheme used in the coin-flip protocols. This way it may bias the coin-flip protocols and thus always end up with the right type of key K.

The problem in the commitment scheme above is that in [32] each player must have an individual key for the initial commitment scheme. Otherwise, $\mathcal{A}$ might be able to use the fact that $\mathcal{S}$ is equivocating commitments to equivocate commitments on his own. He could possibly even select E-keys for his own commitments and then ruin the entire simulation since $\mathcal{S}$ could no longer extract messages.

This is exactly the kind of problem that the commitment schemes in this paper can solve: they allow us to create a situation where for a fixed size public

key, a simulator can make any number of commitments and equivocate them, while the adversary cannot do so. This is the essential property that implies non-malleability and it is also useful here. So instead of letting each participant have his own commitment key on the CRS, we let a public key for a unconditionally hiding and randomness revealing commitment scheme with the structure in Figure 6.1 be on the CRS.

6.6.2 Improving the Damgård-Nielsen UC Commitment Scheme

In the following, we describe the ideal functionality $\mathcal{F}$, the UC commitment scheme and the ideal process adversary $\mathcal{S}$.

The Ideal Functionality

We label the dummy parties in the ideal process $\widetilde{P}_1, \ldots, \widetilde{P}_n$ to distinguish them from the parties $P_1, \ldots, P_n$ in the real-life model.

Generating the key: Generate a public key (pk, N) for the commitment scheme. Send (sk, t_N) to $\mathcal{S}$.

Committing: On $(\text{commit}, sid, cid, P_i, P_j, m)$ from $\widetilde{P}_i$ send $(\text{receipt}, sid, cid, P_i, P_j)$ to $\mathcal{S}$ and $\widetilde{P}_j$. Ignore all subsequent $(\text{commit}, sid, cid, \ldots)$ messages.

Opening: On $(\text{open}, sid, cid, P_i, P_j)$ from $\widetilde{P}_i$ where $(\text{commit}, sid, cid, P_i, P_j, m)$ has been recorded, send $(\text{verify}, sid, cid, P_i, P_j, m)$ to $\mathcal{S}$ and $\widetilde{P}_j$.

The Commitment Scheme[4]

We call the public key and equivocation key for the unconditionally hiding and randomness revealing commitment scheme with structure as in Figure 6.1 pk and sk. Like above we use N, t_N, K, t_K for the keys and trapdoors in the mixed commitment scheme. The CRS for the commitment scheme is (pk, N).

Commitment: Player P_i makes a commitment to a message m and sends it to P_j by doing the following:

Initial Commitment: P_i selects $K_1 \in \mathcal{K}_N$ randomly and sets $(c, d) \leftarrow com_{pk}(sid, cid, P_i, P_j, m, K_1)$. He sends $(\text{init}, sid, cid, P_i, P_j, c)$ to P_j.

[4]This protocol is modified slightly compared with [32], since in their proof they allowed $\mathcal{S}$ to block commitments from being submitted to $\mathcal{F}$, something which is not possible under the standard UC model [17]. The idea and the proof of security is the same as in [32] though.

Commitment Response: P_j on incoming message $(\text{init}, sid, cid, P_i, P_j, c)$ selects $K_2 \in \mathcal{K}_N$ at random. He sends $(\text{response}, sid, cid, P_i, P_j, K_2)$ to P_i and ignores subsequent $(\text{init}, sid, cid, \ldots)$ messages.

Final Commitment: P_i on incoming message $(\text{response}, sid, cid, P_i, P_j, c)$ from P_j checks that he has sent an initial commitment and not sent a final commitment in this run of the protocol. In that case he sets $K = K_1 + K_2$ and $(C, D) \leftarrow com_K(m)$. He sends $(\text{final}, sid, cid, P_i, P_j, K, C)$ to P_j.

Receival: P_j on $(\text{final}, sid, cid, P_i, P_j, K, C)$ from P_i checks that he has earlier sent out a $(\text{response}, sid, cid, \ldots)$ message to P_i. He outputs $(\text{receipt}, sid, cid, P_i, P_j)$. He ignores subsequent $(\text{final}, sid, cid, \ldots)$ messages from P_i.

Opening: To open the commitment P_i sends $(\text{open}, sid, cid, P_i, P_j, d, D)$ to P_j.

Verification: P_j on $(\text{open}, sid, cid, P_i, P_j, d, D)$ from P_i checks that he has sent out a receipt $(\text{receipt}, sid, cid, \ldots)$. He sets $K_1 = K - K_2$ and checks that $(sid, cid, P_i, P_j, m, K_1) = \text{dec}_{pk}(c, d)$ and $m = \text{dec}_K(C, D)$. In that case, he outputs $(\text{verify}, sid, cid, P_i, P_j, m)$.

The Ideal Process Adversary

$\mathcal{S}$ runs a copy of $\mathcal{A}$ trying to simulate everything $\mathcal{A}$ would see in a real-life execution of the commitment protocol. Messages between $\mathcal{A}$ and $\mathcal{Z}$ are simply forwarded.

Whenever commitments are made using $\mathcal{F}$, $\mathcal{S}$ will not know the content but knowing the equivocation keys it can select the commitments for simulated honest parties so that they are equivocable. This way it can make the simulation for honest parties work out by making appropriate equivocations.

If $\mathcal{A}$ decides to corrupt a simulated party P_i, then $\mathcal{S}$ corrupts the corresponding ideal-process party $\tilde{P}_i$ and learns all messages it has committed to. It can then perform suitable equivocations and give $\mathcal{A}$ those equivocations. Here it is important that both the initial commitment scheme and the mixed-commitment scheme from [32] have decommitment information d, D that contain all randomness r, R used in forming the commitments, so $\mathcal{A}$ can see all the randomness it would when corrupting a party in the real-life model. In case $\mathcal{A}$ corrupts a party after sending the initial commitment but before sending the final commitment, we equivocate to a randomly chosen K_1.

6.6.3 Security Proof of the UC Commitment Scheme

Theorem 6.6 *The commitment scheme described above securely realizes the commitment functionality.*

Proof. We show that $\mathcal{Z}$ cannot distinguish the real-life protocol and the ideal process model. For this purpose we look at the following distributions of $\mathcal{Z}$'s (binary) output:

- $REAL$ is the real-life execution.
- HYB_1 is a modified real-life experiment: Instead of the usual CRS, we select the CRS with equivocable keys. We then equivocate both initial and final commitments. In particular, this means that for honest parties about to send the final commitment we equivocate the initial commitment so K becomes an E-key.
- HYB_2 is a modified ideal process. When $\mathcal{A}$ makes a commitment where he is involved in both the initial phase and the final phase of committing we do not extract the message m. Instead, we submit 0 to $\mathcal{F}$ if $\mathcal{A}$ later opens this commitment to contain m, then we patch $\mathcal{F}$ by inserting m in place of 0.[5]
- $IDEAL$ is the ideal process.

$REAL$ and HYB_1 are indistinguishable. To see this let us modify the real-life model step by step into the hybrid model. First, we select the CRS with equivocable key pk, ek and trapdoor t_N associated with N. Second we form the initial commitments in an equivocable manner and equivocate when we have to open. Whenever an honest party has to make the final commitment, we let it select K_1 at random at this point and equivocate the initial commitment to K_1. By definition of equivocability, this change cannot be noticed by $\mathcal{Z}$. However, now we may instead simply choose K as a random E-key and set $K_1 = K - K_2$. Since E-keys are indistinguishable from random keys, this change cannot be noticed. Finally, this enables us to equivocate when the party's commitment has to be opened without $\mathcal{Z}$ being able to notice it.

HYB_1 and HYB_2 are the same experiment phrased in two different ways.

HYB_2 and $IDEAL$ are indistinguishable. The difference between them consists in the opening from the adversary that is patched. There are three possibilities for the message that is patched:

1. The final commitment uses X-key K. Then $\mathcal{S}$ extracts the correct message m.
2. The final commitment uses E-key K made by $\mathcal{S}$. The initial (equivocated) commitment has been opened to contain some m. Due to the binding property of the commitment scheme, $\mathcal{A}$ cannot open the commitment otherwise so the patching is correct. This binding property holds even though

[5] Of course modifying $\mathcal{F}$ like this is illegal in the ideal process but those restrictions do not apply in a hybrid model.

we equivocate some commitments, because otherwise we could distinguish whether we were in the $REAL$ experiment or the HYB_2 experiment.

3. The final commitment does not use an X-key K and K has not been selected by $\mathcal{S}$.

   Let us look at the initial commitment that $\mathcal{A}$ uses. If the one-time signature key vk_{OT} has not been generated by $\mathcal{S}$ then $\mathcal{A}$ cannot know a signature on vk_{OT}. This means that the commitment is binding, so when $\mathcal{A}$ gets the randomly chosen challenge K_2 there is overwhelming probability that the resulting K is indeed an X-key.

   Another possibility is that vk_{OT} has been generated by $\mathcal{S}$ and used in an initial commitment, however, this commitment has not yet been opened. In that case $\mathcal{A}$ does not know how to make one-time signatures, so it must recycle the entire initial commitment made by $\mathcal{S}$. By the indistinguishability from the $REAL$ experiment, $\mathcal{A}$ does not know how to open the commitment. If $\mathcal{S}$ opens the commitment, then $\mathcal{A}$ must open it the same way. Again, we therefore have with overwhelming probability that K is an X-key. This argument also shows that if $\mathcal{A}$ corrupts the party making the commitment, then it is bound by the opening that $\mathcal{S}$ hands it.

   Left is the case where $\mathcal{A}$ reuses vk_{OT} from a commitment made by $\mathcal{S}$, and this commitment has been opened to something. If $\mathcal{A}$ makes an exact copy of the commitment from before, it is again stuck with opening it the same way as $\mathcal{S}$. However, we could imagine that $\mathcal{A}$ makes a different initial commitment (vk_{OT}, a', s'_{OT}). However, if this has noticeable probability of happening we could have guessed the vk_{OT} that it was going to do this with in advance. If we go back to the real experiment, we could make a real initial commitment to (sid, P_i, P_j, m, K_1) without knowing a signature on vk_{OT}. When $\mathcal{A}$ creates a different commitment using one-time signature key vk_{OT}, then this commitment is binding. No matter which K_2 we give as challenge $\mathcal{A}$ always opens the commitment the same way. By the indistinguishability of the $REAL$ and the HYB_2 experiment this must also hold when we equivocate the initial commitment. Therefore, with overwhelming probability we get an X-key.

   Overall, we have negligible probability of ending up in this third situation. □

6.7 Open Problems

It is easy to form commitments in the plain model without any CRS. For instance by creating a public key for an ElGamal cryptosystem and then forming an ElGamal encryption of the message we want to commit to. This commitment is

computationally hiding under the DDH assumption and unconditionally binding. However, this commitment scheme is also malleable. It is an interesting question whether it is possible to create non-malleable commitments in the plain model.

A non-malleable public key cryptosystem automatically gives us a commitment scheme that is non-malleable also when we allow the adversary to get some history about the messages we committed to. However, when encrypting a message the ciphertext is larger than the plaintext.[6] It is an open problem to find a commitment scheme that has commitments that are shorter than the message we commit to and at the same time is non-malleable against history-aware adversaries. The black-box rewind and extract method we used to prove non-malleability in this chapter does not work against such adversaries, because the history may be some one-way function of the message and in that case we cannot sample messages to equivocate the commitments to.

It is worth noting that the problem of history-aware adversaries is not the fact that a commitment shorter than the message cannot be unconditionally binding. We can create unconditionally binding commitments that are non-malleable against history-aware adversaries. Consider for instance the following modification of the Cramer-Shoup cryptosystem [24]. The Cramer-Shoup cryptosystem has a public key consisting of elements $(p, q, g_1, g_2, h, c, d)$. To encrypt a message $m \in G_q$ with randomness $r \in \mathbb{Z}_q$ we compute $u_1 = g_1^r \bmod p, u_2 = g_2^r \bmod p, v = h^r m \bmod p$ and set $\alpha = (cd^{\text{hash}(u_1,u_2,v)})^r \bmod p$. The ciphertext is (u_1, u_2, v, α). This cryptosystem is secure against adaptive chosen ciphertext attacks and therefore it is a non-malleable commitment scheme secure against history-aware adversaries. We first modify the scheme in the following way, we pick at random x and set $G_1 = g_1^x \bmod p, G_2 = g_2^x \bmod p, H = h^x \bmod p, C = c^x \bmod p, D = d^x \bmod p$. To encrypt we set $u_1 = g_1^r G_1^R \bmod p, u_2 = g_2^r G_2^R, v = h^r H^R \bmod p$ and compute $\alpha = (cd^{\text{hash}(u_1,u_2,v)})^r (CD^{\text{hash}(u_1,u_2,v)})^R \bmod p$. It is still basically the same cryptosystem, so it is secure against adaptive chosen ciphertext attack and therefore non-malleable. Consider now the following further modification. We compute $H = h^X \bmod p$, where $x \neq X$. Now we have an unconditionally hiding commitment scheme. At the same time the commitment scheme must still be secure against history-aware adversaries, because if it was not, then we could distinguish between H computed as $h^x \bmod p$ or $h^X \bmod p$ and that would violate the DDH assumption.

[6] Or more correctly, the ciphertext is larger than the entropy of the plaintext.

Chapter 7

Group Signature

7.1 Introduction

Group signatures. Recall from the introduction that a group signature allows members to anonymously sign messages on behalf of the group, except the group manager can break the anonymity and identify a signer. Let us give this loose description a little more flesh.

We have three types of parties: members, non-members and a group manager. The group manager creates a public verification key. Individual members and the group manager cooperate to generate secret keys that members can use to sign messages. We have some functional requirements and some security requirements that we place on group signatures.

Functional Requirements. Some algorithms that can be associated with a group signature scheme are the following.

KeyGen: This algorithm is used to set up the system. It produces $(vk, gmsk) \leftarrow$ KeyGen() as output, where vk is a public verification key, while $gmsk$ is the group managers secret verification key. If the group of members is fixed, we may assume that the algorithm also outputs a vector $\vec{sk}$ of secret keys to be used by the members.

Join: If the group of members is dynamic, we use this protocol to let non-members join the group. A prospective member joins the group by running the interactive protocol Join with the group manager. We assume the communication takes place over a secure channel, though this requirement can sometimes be skipped. As a result, she receives a secret key sk_i.

Sign: To sign a message m the member runs $\sigma \leftarrow \text{Sign}(sk_i, m)$.

Verify: To verify a signature σ on message m we compute $\text{Verify}(vk, m, \sigma)$.

Open: Given a signature σ on m it can be opened by the group manager by computing $\text{Open}(gmsk, m, \sigma)$, which outputs the identity of the member who created the signature.

Revoke: There may be one or more methods to revoke a member's membership.

We do of course require that the verification algorithm accept correctly formed signatures by members in good standing.

Security requirements.

Unforgeability: Without a member's signature key, it is impossible to sign any message.

Anonymity: Without the group manager's secret key, it is impossible to learn who signed a particular message.

Unlinkability: It is impossible to see whether two signatures have been created by the same member.

No-framing: The group manager cannot falsely accuse a member of having made a signature.

Traceability and Coalition-resistance: Even if a coalition of members cooperates to create a signature, at least one of the coalition members will be identified by the group manager.

Bellare, Micciancio and Warinschi [10] simplified all these security requirements into two: full-anonymity and full-traceability. We present the definitions in Section 7.2.

Revocation of membership. One tricky question pertains to revocation of membership. Since the signatures are anonymous, they do not contain a certificate that can be placed in a revocation list. Furthermore, depending on the context we may wish to do different types of revocation.

Consider the intelligence agency example from the introduction. If a smart card containing a member's secret key is stolen then we do of course want to revoke the certificate. However, all signatures made before the theft (possibly enforced through time-stamping) should still be considered valid. They can remain anonymous and unrevoked.

Consider on the other hand a spook that turns out to be a double agent. Here we want to call back all signatures made in the past by this agent. Furthermore, depending on the situation we may or may not want to reveal the identity of the signer with respect to all these revoked signatures.

Opening. Another question is who should be allowed to open a signature and break the anonymity. In some cases, it is the same authority who accepted the member into the group. However, we may also imagine instead that it is a law enforcement agency that should be able to trace the abuser in case of law violations.

In our scheme, a simple modification makes it possible to separate the issuing of certificates and the opening of signatures completely. This means that the issuer is not able to open or link any signatures, on the other hand the opening authority is unable to enroll new members. This property can be achieved in a flexible manner such that the same public key of the opening authority can be used by different issuers.

State of the art. The current state of the art group signature scheme is due to Ateniese, Camenisch, Joye and Tsudik [1]. While being reasonably efficient, this scheme does not support certificate revocation.

An extension by Ateniese, Song and Tsudik [3] implements the full revocation mentioned before, i.e., all bad signatures by the revoked member are revealed. Unfortunately, this scheme is rather inefficient. Camenisch and Lysyanskaya [16] and Tsudik and Xu [75] propose schemes with dynamic revocation. This means that after a certificate has been revoked the member cannot any longer make signatures. Both schemes are less efficient than [1]. [75] is more efficient than [16], but relies on a trusted third party to generate some of the data, and need to update the key both when members join and leave the group. [16] can easily be modified to only updating the verification key when memberships are revoked.

Security assumptions. All the schemes mentioned here include in their assumptions the strong RSA assumption and the random oracle model. Ateniese and de Medeiros [2] suggest a scheme that does not rely on knowledge of the factorization of the modulus, but this scheme is much less efficient than [1]. [10] suggest a scheme based on any trapdoor permutation and without the random oracle model. This scheme is only a proof of concept; it is very inefficient.

Anonymity. The security requirement of anonymity can be interpreted in different ways. In [10] full-anonymity implies that even if a member's secret key is exposed it is still not possible to recognize which signatures are made by this member. The schemes in [1] and [16] have this strong anonymity property. Other papers, however, are less demanding, they just require anonymity as long as a member's secret key is not exposed. [22, 2, 75] are examples of such schemes. One positive effect of not requiring anonymity is that potentially it makes it possible for the member to claim a signature he made, i.e., prove that he signed a particular signature, without having to store specific data such as randomness,

etc., used to generate the signature. This latter property is called claiming in [55].

Our contributions. We present a new practical group signature scheme. We prove that it satisfies a strong security definition very similar to [10]. Security is proved in the random oracle model under the strong RSA assumption and a couple of DDH assumptions.

Our scheme is considerably faster than the state of the art scheme in [1]. Moreover, in our scheme the Join protocol only takes two rounds. The prospective member sends a join request to the group manager. The group manager sends a certificate back to the member.

The scheme supports dynamically joining new members to the group without changing the public key. Furthermore, it is possible to revoke a secret key such that it can no longer be used to sign messages. Revocation of a membership does require the public key to be modified. However, the modification is of constant size and allows group members in good standing to update their secret keys easily. To accomplish this goal we use methods similar to those of [16] and [75]. Their schemes are not as efficient as our scheme.

We present a modification of our scheme that with only a small loss of efficiency also allows us to make a full revocation, i.e., reveal all signatures signed with a revoked key. This scheme does not satisfy the [10] definition of security though. The problem is that given a private signature key it *is* possible to determine which signatures belong to the member in question.

As a separate theoretical contribution, we show that the existence of one-way functions and NIZK arguments can be used to construct a group signature scheme. Again, we obtain a scheme that does not satisfy the [10] definition because a member's secret key does make it possible to identify signatures made by this member. We propose how to define security of group signature schemes when compromise of members' secret keys does matter.

We prove that the [10] definition implies IND-CCA2 secure public key bit-encryption. The existence of one-way functions and NIZK arguments does to our knowledge not entail the existence of public key encryption. Therefore, it seems that to satisfy [10] one must use stronger security assumptions than what is needed for just making a group signature scheme.

7.2 Definitions

[10] propose two properties, full-traceability and full-anonymity, that capture the security requirements mentioned in the introduction.

Full-traceability. The short description of full-traceability is that without a member's secret key it must be infeasible to create a valid signature that frames

this member. This must hold even if the group manager's secret key and an arbitrary number of the members' secret keys are exposed.

Formally, we say that the group signature scheme has full-traceability if the expectation of the following experiment is negligible.

$\mathbf{Exp}_{\mathcal{A}}^{\text{f-trace}}(k)$:

$(vk, gmsk, \vec{sk}) \leftarrow \text{KeyGen}(k)$
$(m, \sigma) \leftarrow \mathcal{A}^{\text{Sign}(sk_\cdot,\cdot),\text{Corrupt}(\cdot)}(vk, gmsk)$
If $\text{Verify}(vk, m, \sigma) = 1$, $i = \text{Open}(gmsk, m, \sigma) \in [k]$, i was not queried $\text{Corrupt}(\cdot)$ and (i, m) was not queried to $\text{Sign}(sk_\cdot, \cdot)$ then return 1
If $\text{Verify}(vk, m, \sigma) = 1$ and $i = \text{Open}(gmsk, m, \sigma) \notin [k]$ then return 1
Else return 0

Here $\text{Corrupt}(\cdot)$ is an oracle that on query $i \in [k]$ returns sk_i.

[10] argue that full-traceability implies what is meant by the more informal notions of unforgeability, no-framing, traceability and coalition resistance.

Full-anonymity. We want to avoid that signatures can be linked to group members or other signatures. For this purpose, we define full-anonymity as the notion that an adversary cannot distinguish signatures from two different members. This must hold even when we give the secret keys to the adversary. In other words, even if a member's key is exposed, then it is still not possible for the adversary to see whether this member signed some messages in the past, neither is it possible to see if any future messages are signed by this member.

$\mathbf{Exp}_{\mathcal{A}}^{\text{f-anon}}(b, k)$:

$(vk, gmsk, \vec{sk}) \leftarrow \text{KeyGen}(k)$
$(i_0, i_1, m) \leftarrow \mathcal{A}^{\text{Open}(gmsk,\cdot,\cdot)}(vk, \vec{sk}); \sigma \leftarrow \text{Sign}(sk_{i_b}, m)$
$d \leftarrow \mathcal{A}^{\text{Open}(gmsk,\cdot,\cdot)}(\sigma)$
If $\mathcal{A}$ did not query m, σ return d, else return 0

We say the group signature scheme has full-anonymity if $\Pr[\mathbf{Exp}_{\mathcal{A}}^{\text{f-anon}}(1, k) = 1]$ - $\Pr[\mathbf{Exp}_{\mathcal{A}}^{\text{f-anon}}(0, k) = 1]$ is negligible.

[10] argue that full-anonymity entails what is meant by the more informal notions of anonymity and unlinkability.

Anonymity. As we mentioned in the introduction the [10] model is strict in its anonymity requirements. It demands that even if a member's secret key is exposed it must still be impossible to tell which signatures are made by the member in question. This is a good definition of security in a threat model where parties may be corrupted adaptively but can erase data.

In other threat models, this may be aiming too high. Consider for instance a static adversary, then the key is exposed before any messages are signed or it is never exposed. Or consider an adaptive adversary where parties cannot erase data, in this case full-anonymity does not buy us more security. We therefore define a weaker type of anonymity that is satisfied if both the group manager's secret key and the member's secret key are not exposed.

$\mathbf{Exp}^{\text{anon}}_{\mathcal{A}}(b,k)$:

$(vk, gmsk, \vec{sk}) \leftarrow \text{KeyGen}(k)$
$(i_0, i_1, m) \leftarrow \mathcal{A}^{\text{Open}(gmsk,\cdot,\cdot),\text{Sign}(sk_\cdot,\cdot),\text{Corrupt}(\cdot)}(vk); \sigma \leftarrow \text{Sign}(sk_{i_b}, m)$
$d \leftarrow \mathcal{A}^{\text{Open}(gmsk,\cdot,\cdot),\text{Sign}(sk_\cdot,\cdot)}(\sigma)$[1]
If $\mathcal{A}$ did not query m, σ to Open and did not query i_0, i_1 to Corrupt$(\cdot)$ then return d, else return 0

We say the group signature scheme is anonymous if $\Pr[\mathbf{Exp}^{\text{anon}}_{\mathcal{A}}(1,k) = 1]$ - $\Pr[\mathbf{Exp}^{\text{anon}}_{\mathcal{A}}(0,k) = 1]$ is negligible.

As in [10] we can argue that anonymity implies the informal notions of anonymity and unlinkability mentioned in Section 7.1.

7.3 The Basic Group Signature Scheme

CL-signatures. Recall the signature scheme from Section 2.4 where we choose n as a safe-prime product of length ℓ_n, and have $a, g, h \in_R \text{QR}_n$ as part of the public key. Camenisch and Lysyanskaya [16] suggest a similar signature scheme where the signature is on the form (y, e, r), but r can be large, and both m and r are allowed to be negative. It is clear that if $y^e = ag^m h^r \bmod n$, then we also have $(yh)^e = ag^m h^{r+e} \bmod n$, so this signature scheme is not strong. It is secure against chosen message attack though.

The CL-signature scheme makes it possible to sign a committed message. One party computes the commitment $g^m h^{r'} \bmod n$, where $r' \in_R \mathbb{Z}_n$ such that m is statistically hidden. This party also proves knowledge of m, r'. The signer now picks e as a random ℓ_e-bit prime, and picks $r'' \in \mathbb{Z}_e$. He then computes y so $y^e = ag^m h^{r'+r''} \bmod n$ and returns (y, e, r''). Now the party has a signature on m without the signer having any knowledge about which message was signed. Camenisch and Lysyanskaya suggest parameters $\ell_n = 1024, |r| = 1346$, but we can get by with $|r| = 1024$ and still have statistical hiding of m.

[1] We do not allow $\mathcal{A}$ to corrupt member's in the second phase. This is simply because we WLOG may assume that it corrupts all other members than i_0 and i_1 before getting the challenge signature.

The ideas underlying our group signature scheme. We base our group signature scheme on the Fiat-Shamir heuristic. The signer will prove knowledge of some secret information to sign a message. We proceed to describe the problem to which the signer knows a solution.

In our group signature scheme the group manager will compute a CL-signature on some element x_i and give this signature to member i. In other words, the member gets a signature (y_i, E_i, r_i) such that $y_i^{E_i} = ag^{x_i}h^{r_i} \bmod n$. Since nobody can compute CL-signatures on their own, any signer who can argue that she knows a CL-signature on something must be part of the group.

To remain anonymous the member must make this argument in zero-knowledge, and therefore transmit her identity secretly. We rely on public key encryption to achieve that. The group manager holds the secret key of the cryptosystem. This means that he may open up a group signature and see who the member is. Note that we do not know in advance, which signatures the group manager will open, so we do need the encryption to be semantically secure against adaptive chosen ciphertext attack.

It is inherent in the model that the group manager can always issue a phony membership and sign any message he wants. We want to avoid that the group manager frames members by claiming that they signed something that they did not sign. To accomplish this we let the element x_i that the member gets a signature on be an exponent in $G^{x_i} \bmod P$ and ensure that the group manager does not know x_i. The member includes in her signature an argument that she knows the discrete logarithm of $G^{x_i} \bmod P$. The group manager does not know x_i and can therefore not make such an argument.

The protocol itself is described in Figure 7.1. We remark that better and more efficient group signatures exist for the static case with no join or revoke. We present this particular scheme because it is easy to extend to have advanced properties like dynamic join and revocation.

Parameters. We use ℓ_s as a bit-length such that for any integer a when we pick r as a $|a| + \ell_s$-bit random number then $a + r$ and r are statistically indistinguishable. ℓ_c is the length of the output of the hash-function. ℓ_e is a number large enough that we can assign all members different numbers and make the E_i's prime.

It must be the case that $\ell_c, \ell_e < \ell_Q < \ell_c + \ell_e + \ell_s < \ell_n/2$.

A suggestion for parameters is $\ell_n = \ell_P = 1024, \ell_Q = \ell_c = 160, \ell_e = \ell_s = 40$.

Security.

Theorem 7.1 *The basic group signature scheme has full-traceability and full-anonymity.*

Proof. The theorem follows from Lemma 7.3 and Lemma 7.4. □

Basic Group Signature Scheme

KeyGen(k)**:** Choose an ℓ_n-bit RSA modulus $n = pq$ as a product of two safe primes $p = 2p' + 1, q = 2q' + 1$. Select at random $a, g, h \in \mathrm{QR}_n$.

Select at random ℓ_Q-bit and ℓ_P-bit primes Q, P such that $Q|P-1$. Let F be an element of order Q in $\mathbb{Z}_P^*$. Choose at random $X_G, X_H \in \mathbb{Z}_Q$ and set $G = F^{X_G} \bmod P, H = F^{X_H} \bmod P$.

Select at random $x_1, \ldots, x_q \in \mathbb{Z}_Q$ and select also at random $r_1, \ldots, r_k \in \mathbb{Z}_n$.

Choose different random ℓ_e-bit numbers $e_1, \ldots, e_k$ such that $E_1 = 2^{\ell_e+\ell_c+\ell_s} + e_1, \ldots, E_k = 2^{\ell_e+\ell_c+\ell_s} + e_k$ are primes. Compute $y_1, \ldots, y_k$ such that $y_1^{E_1} = ag^{x_1}h^{r_1} \bmod n, \ldots, y_k^{E_k} = ag^{x_k}h^{r_k} \bmod n$.

Public key: $vk = (n, a, g, h, Q, P, F, G, H)$.

Group managers private key: $gmsk = (vk, X_G, G^{x_1} \bmod P, \ldots, G^{x_k} \bmod P)$.

Member i's private key: $sk_i = (vk, x_i, y_i, e_i, r_i)$.

Sign(sk_i, m)**:** Select at random $r \in \{0,1\}^{\ell_n/2}, s \in \{0,1\}^{\ell_n+\ell_c+\ell_s}, d_x \in \{0,1\}^{\ell_c+\ell_Q+\ell_s}, d_e \in \{0,1\}^{\ell_c+\ell_e+\ell_s}, R, S \in \mathbb{Z}_Q$.

Set $u = h^r y_i \bmod n, v = h^{s-rd_e} g^{d_x} y_i^{-d_e} \bmod n$.[a]

Let furthermore $U_1 = F^R \bmod P, U_2 = G^R G^{x_i} \bmod P, U_3 = H^R H^{e_i} \bmod P$ and $V_1 = F^S \bmod P, V_2 = G^S G^{d_x} \bmod P, V_3 = H^S H^{d_e} \bmod P$.

Compute a challenge $c = \mathrm{hash}(m, u, v, U_1, U_2, U_3, V_1, V_2, V_3)$.

Set $z_e = ce_i + d_e, z_E = c2^{\ell_e+\ell_c+\ell_s} + z_e, z_r = cr_i + z_E r + s, z_x = cx_i + d_x$ and $Z_R = cR + S \bmod Q$.

Signature: $\sigma = (c, u, U_1, U_2, U_3, z_r, z_x, z_e, Z_R)$.

Verify(vk, m, σ)**:** Check that all elements are in the correct intervals and respectively belong to $\mathbb{Z}_n$ and $\mathbb{Z}_P^*$. Let $z_E = c2^{\ell_e+\ell_c+\ell_s} + z_e$.

Compute $v = a^c g^{z_x} h^{z_r} u^{-z_E} \bmod n$ and $V_1 = F^{Z_R} U_1^{-c} \bmod P, V_2 = G^{Z_R+z_x} U_2^{-c} \bmod P, V_3 = H^{Z_R+z_e} U_3^{-c} \bmod P$.

Verify that $c = \mathrm{hash}(m, u, v, U_1, U_2, U_3, V_1, V_2, V_3)$.

Open$(gmsk, m, \sigma)$**:** Verify that the signature is valid.

Using X_G decrypt $(U_1^{\frac{P-1}{Q}} \bmod P, U_2^{\frac{P-1}{Q}} \bmod P)$ to get $G^{\frac{P-1}{Q}x_i} \bmod P$ and return i.

[a] Goldreich and Rosen [48] show that to someone not knowing the factorization of n, the element $h^r \bmod n$ is indistinguishable from a random element in QR_n. This means u hides y_i.

Figure 7.1: The Basic Group Signature Scheme.

Lemma 7.1 *Let $n = pq$, where $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes. We assume the strong RSA assumption holds for $\mathbb{Z}_n^*$. Let $s, t \in \mathbb{Z}_n^*$ and $d > 1$ be produced by an adversary $\mathcal{A}$ such that $s^d = t^d \bmod n$. If d is odd we have $s = t$ and if d is even we have $s = \pm t$.*

Proof. Assume to start with that d is odd. Then we have $(s/t)^d = 1 \bmod n$. If $p'q'|d$ then we can break the strong RSA assumption, since for any h we have $h = h^{2d+1} \bmod n$. If $p'|d$ and $q' \not| d$ then $\gcd(n, 2^{2d} - 1) = p$ and we have factored n. Likewise we get a non-trivial factorization of n if $p' \not| d$ and $q'|d$. Finally, if $p' \not| d$ and $q' \not| d$ then we are looking at a normal RSA-exponentiation and therefore $s = t$.

If d is even, then we can use the theorem to take care of the odd part. Assume therefore WLOG that $d = 2^e$. Since $\gcd(2, p'q') = 1$ we get $(s/t)^{2^e} = 1 \bmod n$ implies $(s/t)^2 = 1 \bmod n$. Then $(s/t)^2 - 1 = ((s/d) - 1)((s/d) + 1) \bmod n$ gives us $s = \pm t$ or a non-trivial factorization of n. □

Lemma 7.2 *Let $n = pq$, where $p = 2p' + 1$ and $q = 2q' + 1$. We will assume the strong RSA assumption holds. Pick $g_1, \ldots, g_l$ at random from QR_n. Let an adversary produce $u \in \mathbb{Z}_n^*$ and $x, x_1, \ldots, x_l$ such that $u^x = g_1^{x_1} \cdots g_l^{x_l} \bmod n$. Then with overwhelming probability $x|x_1, \ldots, x|x_l$.*

Proof. We will transform a successful adversary into one that finds a non-trivial root of a random $h \in \mathrm{QR}_n$. Choose $\alpha_1, \ldots, \alpha_l$ at random from $\mathbb{Z}_{n^2}$ and set $g_1 = h^{\alpha_1} \bmod n, \ldots, g_l = h^{\alpha_l} \bmod n$. These are statistically indistinguishable from randomly chosen $g_1, \ldots, g_l$. We run the adversary on these elements.

Assume WLOG that $x \not| x_l$. Let r be a prime such that $r|x$ and $r \not| x_l$. Since α_l from the point of view of the adversary is indistinguishable from $\alpha_l - p'q'$, we have with at least 50% probability that $r \not| \sum_{i=1}^{l} \alpha_i x_i$. From now on, we assume this is the case.

Let $d = \gcd(x, \sum_{i=1}^{l} \alpha_i x_i)$. Let $y = \frac{x}{d}$ and $\sigma = \frac{(\sum_{i=1}^{l} \alpha_i x_i)}{d}$. By Lemma 7.1 have $u^y = \pm h^\sigma \bmod n$. Since $h \in \mathrm{QR}_n$ then $u^y = -h^\sigma \bmod n$ would imply that y is odd and $(-u)^y = h^\sigma \bmod n$. WLOG we therefore assume $u^y = h^\sigma \bmod n$.

Choose γ, δ such that $\gamma y + \delta\sigma = 1$. We then have $h = h^{\gamma y + \delta\sigma} = (h^\gamma u^\delta)^y \bmod n$. Since $r|y$, this gives us a non-trivial root of h. □

Lemma 7.3 *The basic group signature scheme has full-traceability.*

Proof. First note that in any signature (U_1, U_2) define a unique x such that $G^{\frac{P-1}{Q}x} = (U_2 U_1^{-X_G})^{\frac{P-1}{Q}} \bmod P$, where X_G is such that $G = F^{X_G} \bmod P$. A group signature contains an argument of knowledge of this x and an argument of knowledge of a signature on x.

We will first argue in Claim 7.1 that we can use the group manager's secret key to make perfect simulations of signatures without knowing the x_i's. This

means that the signing oracle does not reveal anything about the x_i belonging to a member.

Next, we will argue in Claim 7.2 that if an adversary successfully produces a signature on a new message m not queried before, and this signature has (U_1, U_2) encrypting x_i, then this implies knowledge of x_i. This means that we can use $\mathcal{A}$ in an algorithm to break the DDH problem in $< F >$.

Finally, we will argue in Claim 6.2 that any valid signature must contain (U_1, U_2) pointing to one of the x_i's.

Claim 7.1 *Given $G^{x_i} \bmod P, e_i, y_i$ we can make a perfect simulation of a group signature for member i.*

Proof. Pick r at random and set $u = h^r y_i \bmod n$. Choose R at random and set $U_1 = F^R \bmod P, U_2 = G^R G^{x_i} \bmod P, U_3 = H^R H^{e_i} \bmod P$.

Choose c at random and choose Z_R, z_x, z_e, z_r at random. Set $v = a^c g^{z_x} h^{z_r} u^{-z_e} \bmod n$ and $V_1 = F^{Z_R} U_1^{-c} \bmod P, V_2 = G^{Z_R + z_x} U_2^{-c} \bmod P, V_3 = H^{Z_R + z_e} \bmod P$.

Define the random oracle to output c on query $(m, u, v, U_1, U_2, U_3, V_1, V_2, V_3)$.

Claim 7.2 *If the x_i of an uncorrupted member is inside the group signature, then the group signature contains an argument of knowledge of x_i.*

Suppose $(U_2 U_1^{-X_G})^{\frac{P-1}{Q}} = G^{\frac{P-1}{Q} x_i} \bmod P$. In a valid signature we have, among other things, $V_1 = F^{Z_R} U_1^{-c} \bmod P$ and $V_2 = G^{Z_R + z_x} U_2^{-c} \bmod P$.

The c matches that of the hash-function evaluated in, among other things, U_1, U_2, V_1, V_2. Modeling the hash-function as a random oracle the adversary can only have negligible chance of guessing this, so we can assume that the adversary actually at some point made a query containing U_1, U_2, V_1, V_2. As an answer to this query it received a random challenge c and was able to successfully produce Z_R and z_x. To have noticeable chance of success it should also have had noticeable chance of answering another challenge c', i.e., produce satisfactory Z'_R, z'_x such that $V_1 = F^{Z'_R} U_1^{-c'} \bmod P$ and $V_2 = G^{Z'_R + z'_x} U_2^{-c} \bmod P$.

We deduce that $U_1^{c-c'} = F^{Z_R - Z'_R} \bmod P$ and $U_2^{c-c'} = G^{Z_R - Z'_R} G^{z_x - z'_x} \bmod P$. This implies that $((U_2 U_1^{-X_G})^{\frac{P-1}{Q}})^{c-c'} = G^{\frac{P-1}{Q}(z_x - z'_x)} \bmod P$. So, if (U_1, U_2) point to x_i, then we must have $x_i = \frac{z_x - z'_x}{c - c'} \bmod Q$. Thus, we have successfully computed the discrete logarithm x_i of $G^{x_i} \bmod P$.

Claim 7.3 *A valid group signature contains one of the G^{x_i}'s inside (U_1, U_2).*

Proof. Define x as the element from $\mathbb{Z}_Q$ such that $(U_2 U_1^{-X_G})^{\frac{P-1}{Q}} = G^{\frac{P-1}{Q} x} \bmod P$. We will show that the strong RSA assumption implies that a valid group signature with x not being one of the x_i's implies that the adversary is capable of an existential forgery attack on the CL-signature.

First, we have to make sure that the key generation algorithm does not reveal anything about the factorization of n, otherwise we cannot use the strong RSA assumption in the proof.

We choose $a_{\text{base}}, g_{\text{base}}, h_{\text{base}}$ at random from QR_n, and choose $E_1, \ldots, E_k$ as random primes with the correct distribution. We may now compute $a = a_{\text{base}}^{\prod_{j=1}^{k} e_j} \bmod n, g = g_{\text{base}}^{\prod_{j=1}^{k} e_j} \bmod n, h = h_{\text{base}}^{\prod_{j=1}^{k} e_j} \bmod n$. This means that we do not need the factorization of n to produce the signatures (y_i, E_i, r_i) on x_i that we give to the members.

Consider now $\mathcal{A}$ trying to form a valid signature that does not point to one of the members. From correct answers to two different challenges c, c' we have $v = a^c g^{z_x} h^{z_r} u^{-z_E} \bmod n$ and $v = a^{c'} g^{z'_x} h^{z'_r} u^{-z'_E} \bmod n$. This implies that $u^{z_E - z'_E} = a_{\text{base}}^{(c-c')\prod_{j=1}^{k} E_j} g_{\text{base}}^{(z_x - z'_x)\prod_{j=1}^{k} E_j} h_{\text{base}}^{(z_r - z'_r)\prod_{j=1}^{k} E_j} \bmod n$. By Lemma 7.2, this implies that $z_E - z'_E | (c-c') \prod_{j=1}^{k} E_j$. Considering the sizes of $z_E - z'_E, E_1, \ldots, E_k, c - c'$ that are respectively $\ell_e + 2\ell_c + \ell_s, \ell_e + \ell_c + \ell_s, \ell_c$ we get that $z_E - z'_E = \pm E_i(c - c')$ for some E_i. Furthermore, considering the sizes of $E_i, c - c', z_x - z'_x$ we get $c - c' | z_x - z'_x$ and we can define x as $\frac{z_x - z'_x}{c - c'}$. Finally, we may define $r = \frac{z_r - z'_r}{c - c'}$. Removing the $c - c'$ factor in the exponent we get $(\pm u^{\pm 1})^{E_i} = u^{\pm E_i} = a g^x h^r \bmod n$, which is a CL-signature on x.

In a real execution we do not set up the protocol with known E_i-roots of a, g, h. Rather we just reveal a set of signatures. What we have shown above is that the adversary will generate a CL-signature on x. This means that we have an existential forgery on the CL-signature scheme unless $x = x_i$ for some i.

At the same time we have already seen in Claim 7.2 that a signature contains an argument of knowledge of x such that $(U_2 U_1^{-X_G})^{\frac{P-1}{Q}} = G^{\frac{P-1}{Q} x} \bmod P$, where $x = \frac{z_x - z'_x}{c - c'} \bmod Q$. Since $\frac{z_x - z'_x}{c - c'} = x_i$ we therefore deduce that a valid group signature does indeed point out a member $i \in [k]$. □

It is worth noting that Claim 7.1 and Claim 7.2 hold even if we know the factorization of n. This matters when we consider scenarios where the group manager generates n and therefore may generate it maliciously and with knowledge of a lot of extra information about it.

Lemma 7.4 *The basic group signature scheme has full-anonymity.*

Proof. The intuitive reason that we have full-anonymity is that since everything is encrypted and proved in zero-knowledge, a signature does not reveal anything about the signer's secret key. The obstacle is that the adversary can ask the group manager to open messages. Essentially this gives a chosen ciphertext attack on the cryptosystem used to hide the identities. We will argue that a signature actually is a CCA2 secure encryption of the identity of the signer. To do this we follow the standard method of [65, 24, 73] to argue CCA2 security, namely attaching an argument of (U_1, U_2) and (U_1, U_3) decrypting to the same.

Consider an adversary that recycles $(m, u, v, U_1, U_2, U_3, V_1, V_2, V_3)$ from a simulated signature. Let us start with the expectation of $\mathbf{Exp}_{\mathcal{A}}^{\text{f-anon}}(1, k)$. We may set up a, g, h as in Claim 7.3, such that we do not reveal the factorization of n. By Claim 7.1, we may simulate the signature when generating the challenge signature and still have the same expected outcome.

Since we are simulating the argument, we may form (U_1, U_3) as a ciphertext encrypting $H^{e_{i_0}} \bmod P$ instead of $H^{e_{i_1}} \bmod P$. If $\mathcal{A}$ can notice the difference, then it means that it can break the semantic security of the ElGamal encryption.

The next step we will take is to switch the key we are using to decrypt. We might have set up the signature scheme with knowledge of X_H instead of X_G, and use decryption of (U_1, U_3) to find out who signed the message. Since a argument that (U_1, U_2) and (U_1, U_3) point to the same member is included in the signatures $\mathcal{A}$ queries to $\text{Open}(gmsk, \cdot, \cdot)$ this oracle answers the same on all queries where $(m, u, v, U_1, U_2, U_3, V_1, V_2, V_3)$ is not the same as in the challenge signature. In other words, $\mathcal{A}$ does not learn from the openings whether it is X_G or X_H that we use to decrypt.

The only thing we have to guard against is that $\mathcal{A}$ reuses $(m, u, v, U_1, U_2, U_3, V_1, V_2, V_3)$ from the challenge signature. Call the answers used in the simulated signatures for Z_R, z_x, z_e, z_r, and let the answers produced by $\mathcal{A}$ be Z'_R, z'_x, z'_e, z'_r.

Since $U_1^c V_1 = F^{Z_R} \bmod P$ we must have that $Z'_R = Z_R$. $U_2^c V_2 G^{-Z_R} = G^{z_x} = G^{z'_x} \bmod P$ implies that $z_x = z'_x \bmod Q$. $U_3^c V_3 H^{-Z_R} = H^{z_r} = H^{z'_r} \bmod P$ implies that $z_e = z'_e \bmod Q$. Since z_e and z'_e are rather small this shows that $z_e = z'_e$.

$v = a^c g^{z_x} h^{z_r} u^{-z_E} = a^c g^{z'_x} h^{z'_r} u^{-z_E} \bmod n$ shows that $1 = g^{z'_x - z_x} h^{z'_r - z_r} \bmod n$. Unless $z'_x - z_x = 0$ this gives a discrete logarithm of g base h. $1 = h^{z_r - z'_r} \bmod n$ implies $p'q' | z_r - z'_r$. Therefore $z_r = z'_r$ because otherwise we could break the strong RSA assumption since for any element z we have $z = z^{1+z_r - z'_r} \bmod n$.

Overall, we therefore see that the adversary cannot recycle $(m, u, v, U_1, U_2, U_3, V_1, V_2, V_3)$ without recycling the entire challenge signature. Therefore, the adversary cannot tell the difference between using X_G and X_H to decrypt.

We now have a hybrid signature with simulated argument and $G^{x_{i_1}} \bmod P$ in (U_1, U_2) and $H^{e_{i_0}} \bmod P$ in (U_1, U_3) and we use X_H to reveal the identity of parties. We now consider the experiment where we use $U_2 = G^R G^{x_{i_0}} \bmod P$. By the semantic security of ElGamal encryption, this is not something that $\mathcal{A}$ will notice.

Finally, we notice that the experiment has now been modified so much that it is simply $\mathbf{Exp}_{\mathcal{A}}^{\text{f-anon}}(0, k)$ with simulated argument and simulated signatures. □

7.4 Join and Revoke

Flexible Join. It may be unpractical to set up the signature scheme with all members in advance. Often groups are more dynamic and we may have members joining after the public keys have been generated. The recent schemes [1, 16, 75] support members joining at arbitrary points in time. The schemes [16, 75] require that the public key be updated when a new member joins. However, they can easily be modified to the more attractive solution where the public key does not need to be updated when a member joins.

Our scheme supports members joining throughout the protocol. The idea is that the member generates $G^{x_i} \bmod P$ herself, so only she knows the discrete logarithm. Jointly the group manager and the member generate $ag^{x_i}h^{r_i} \bmod n$, where r_i is so large that x_i is statistically hidden. Then she gives it to the group manager who generates (y_i, e_i) and gives them to the member. Here we use that the CL-signature scheme is secure against adaptive chosen message attack such that members cannot forge signatures and thereby falsely join themselves.

Revocation. On occasions, it may be necessary to revoke a member's secret key. Since signatures are anonymous, the standard approach of using certificate revocation lists cannot be used. Following [16] we suggest using roots of some element w to implement revocation. A signature contains an argument of knowledge of a pair (w_i, E_i) such that $w = w_i^{E_i} \bmod n$. If we want to revoke a membership we update the public key to contain $w \leftarrow w_i$. Now this member may no longer prove knowledge of a root of w and thus she cannot sign messages any more.[2]

When changing the public key we need to communicate to the remaining members how they should update their secret keys. In our scheme, we do this by publishing e_i corresponding to the revoked member. Members in good standing may use this to obtain a root of the new w through a simple computation. This means that the change in the public key is of constant size, and old members may update their secret keys by downloading only a constant amount of public information.

The protocol is described in Figure 7.2.

Performance. The signer can of course precompute $G^{x_i} \bmod P, H^{e_i} \bmod P$ and reuse them to many signatures. We are then left with 12 exponentiations to compute a signature, but most of the exponents are small. Only r is of length $\ell_n/2$ and s of length $\ell_n + \ell_c + \ell_s$. To verify a signature we use 10 exponentiations. Again, most of them are small, only z_s is of size $\ell_n + \ell_c + \ell_s$. With the parameters

[2] A member with a revoked key can still sign messages under the old verification key and claim that they were signed when this key was valid. Whether such an attack makes sense depends on the application of the group signature scheme and is beyond the scope of the dissertation. One obvious solution is of course to add a time-stamp.

Join and Revoke

KeyGen(): Run KeyGen(0) of the basic scheme. Choose also at random $w \in \mathrm{QR}_n$ and include it in the public key vk. Set $gmsk = (vk, p, q, X_G)$ where $n = pq$.

Join: The member selects at random $x_i \leftarrow \mathbb{Z}_Q$ and computes $G^{x_i} \bmod P$. She also forms a commitment to x_i, $g^{x_i}h^{r_i'} \bmod n$ with $r_i' \in_r \mathbb{Z}_n$ and makes a non-interactive zero-knowledge argument of knowledge of x_i', r_i' fitting the above. She sends $G^{x_i} \bmod P$, $g^{x_i}h^{r_i'}$ and the argument to the group manager.

The group manager selects an ℓ_e-bit number e_i so $E_i = 2^{\ell_e+\ell_c+\ell_s} + e_i$ is prime. He computes $w_i = w^{E_i^{-1}} \bmod n$. He selects at random $r_i'' \in_R \mathbb{Z}_{E_i}$ and sets $y_i = (ag^{x_i}h^{r_i'+r_i''})^{E_i^{-1}} \bmod n$. He sends w_i, y_i, e_i, r_i'' back to the new member.

The member sets $r_i = r_i' + r_i''$. Her secret key is $sk_i = (vk, w_i, x_i, r_i, y_i, e_i)$.

Sign(vk, sk_i, m)**:** Select at random $r \in \{0,1\}^{\ell_n/2}, s \in \{0,1\}^{\ell_n+\ell_e+\ell_s}, d_x \in \{0,1\}^{\ell_c+\ell_Q+\ell_s}, d_e \in \{0,1\}^{\ell_c+\ell_e+\ell_s}, R, S \in \mathbb{Z}_Q$.

Set $u = h^r y_i w_i \bmod n, v = h^{s-rd_e} g^{d_x} (y_i w_i)^{-d_e} \bmod n$.

Let furthermore $U_1 = F^R \bmod P, U_2 = G^R G^{x_i} \bmod P, U_3 = H^R H^{e_i} \bmod P$ and $V_1 = F^S \bmod P, V_2 = G^S G^{d_x} \bmod P, V_3 = H^S H^{d_e} \bmod P$.

Compute a challenge $c = \mathrm{hash}(m, u, v, U_1, U_2, U_3, V_1, V_2, V_3)$.

Set $z_e = ce_i + d_e, z_E = c2^{\ell_e+\ell_c+\ell_s}, z_r = cr_i + z_E r + s, z_x = cx_i + d_x$ and $Z_R = cR + S \bmod Q$.

Signature: $\sigma = (c, u, U_1, U_2, U_3, z_r, z_x, z_e, Z_R)$.

Verify(vk, m, σ)**:** Check that all elements are in the correct intervals and respectively belong to $\mathbb{Z}_n^*$ and $\mathbb{Z}_P^*$.

Compute $v = (aw)^c g^{z_x} h^{z_r} u^{-z_E} \bmod n$ and $V_1 = F^{Z_R} U_1^{-c} \bmod P, V_2 = G^{Z_R+z_x} U_2^{-c} \bmod P, V_3 = H^{Z_R+z_e} U_3^{-c} \bmod P$.

Verify that $c = \mathrm{hash}(m, u, v, U_1, U_2, U_3, V_1, V_2, V_3)$.

OpenProof$(gmsk, i, m, \sigma)$**:** Notice that in a valid signature from member i there are among other things (U_1, U_2) containing $G^{x_i} \bmod P$. Since the group manager knows $G^{x_i} \bmod P$ with a digital signature from member i on it, it is easy for the group manager to prove in zero-knowledge that member i signed the message. This means that the group manager can open individual messages in a verifiable manner without making further compromises to a member's data.

Revoke$(gmsk, i)$**:** Publish e_i. Replace in vk the element w with w_i.

Any member in good standing may update her secret key sk_j as follows. She selects α, β such that $\alpha E_i + \beta E_j = 1$. Then she computes the new $w_j \leftarrow w_i^{\beta E_i} w_j^{\alpha E_j} \bmod n$.

Figure 7.2: Protocol for Dynamic Join and Revoke.

suggested before the longest exponent is 1224 bits and we work with 1024 bit moduli.

Let us compare this with the [1] group signature scheme. In this scheme, the member gives $B_i = a^{x_i} a_0 \bmod n$ to the group manager, and the member's secret is x_i. The group manager, however, knows the factorization of n and thus he has an advantage when trying to compute the discrete logarithm. To prevent that,

the modulus n must be very large, say 2048 bits. Moreover, the scheme requires raising elements to very large powers, typically of length higher than 4000 bits. Therefore, our scheme gains more than an order of magnitude in efficiency in comparison with [1].

Security. As in the proof of Claim 7.3, we could have generated the elements w, a, g, h independently of the factorization of n such that we could still make signatures. Therefore, the adversary does not gain anything extra from the ability to adaptively join members and revoke members. In other words, the group signature satisfies suitable extensions of full-traceability and full-anonymity that take joining and revocation into account. Furthermore, without the group managers secret key the adversary cannot make a signature on an x_i that does not belong to one of the members, so it cannot frame the group manager.

Separating issuer and opener. As we discussed in the introduction there may be cases where we want to separate the process of granting membership and being able to break the anonymity. A simple modification allows this.

The idea is that n is generated by the group manager who can produce the needed CL signatures that we use in our scheme. On the other hand, we let the opener generate G, H. The opener registers $G^{x_i} \bmod P$ and $H^{e_i} \bmod P$ with the opener and gets a signature on it. He shows the opener's signature on $G^{x_i} \bmod P$ and $H^{e_i} \bmod P$ to the member and creates the CL-signature on x_i as well as computes the E_i-root of w, which he gives to the member.

Now, if the member that wants to sign a message picks r and s large enough, then in the group QR_n everything is statistically hidden and in $\mathbb{Z}_P^*$ everything is encrypted. Therefore, the issuer cannot see who signs a particular message. The issuer should prove that $g, w_i \in < h >$, otherwise the side classes might leak information.

On the other hand, the opener can verify signatures and decrypt the ciphertexts. It is therefore easy to open messages. He cannot issue membership certificates because he does not have the ability to produce CL-signatures.

It is interesting to note that it is possible for many issuers to use the same opener and the same public key of that opener. Also interesting is that the ElGamal style encryption used for the opener can easily be made into a threshold encryption scheme.

7.5 Full Revocation

Revocation revisited. The current method of revocation does not allow us to revoke signatures valid under an old key. It would be highly unpractical to demand that all members re-sign messages when the public key is updated. Instead, we would prefer a solution parallel to that of certificate revocation lists

that allow us to publish information that marks signatures signed by the now distrusted member. Nevertheless, of course we still want to preserve the privacy of all other members so we cannot simply reveal the group manager's secret key.

We propose an addition that solves this problem. The idea is to pick a random element $s_i \in \mathbb{Z}_Q$ when the member joins. The member can now form $F^R \bmod P$ and $F^{Rs_i} \bmod P$ and include them in a group signature. According to the DDH assumption, this will just look like two random elements. However, if the group manager releases s_i, then all signatures suddenly become clearly marked as belonging to said member.

We do need to force the member to use s_i, otherwise the member could create group signatures that could not be full-revoked. Therefore, we include a random element $f \in \mathrm{QR}_n$ in the public key and give the member a CL-signature on the form (y_i, E_i, r_i), where $y_i^{E_i} = af^{s_i}g^{x_i}h^{r_i} \bmod n$. The member will form U_4, V_4 as $U_4 = F^{Rs_i} \bmod P$ and $V_4 = F^{d_s} \bmod P$, when making the signature and argues correctness of this together with an argument that s_i is included in the CL-signature that she knows.

The protocol is described in Figure 7.3.

Security. A member's secret key contains s_i. Therefore, if the secret key is exposed it is easy to link the member with the signatures she has made. We can therefore not hope to have full anonymity but must settle for anonymity.

In theory, it is possible to construct a signature scheme that supports full revocation and full-anonymity. One idea could be that the group manager selects elements A_i, B_i with $B_i = A_i^{X_i} \bmod P$ and signs these elements. Then the member must produce in addition to the standard signature a pair $(A_i^R \bmod P, B_i^{RX_i} \bmod P)$ and prove in zero-knowledge that it has been properly formed. Once the group manager wants to make a full revocation, he publishes X_i. However, the member's secret key does not include X_i so exposure of this key does not reveal which messages she has signed. This method is not very efficient though. It is an open problem to come up with an efficient group signature scheme that has full-anonymity and supports full revocation.

On the flip side we note that it may be seen as a positive thing that the members key reveals which messages he signed. In [55]'s notion of traceable signatures it is a requirement that the member should be able to claim his signature. When the member's secret key links him to her signatures then this can be done easily without her having to store old randomness used in specific signatures that she might later want to claim.

7.6 Separating Full-Anonymity and Anonymity

Full-anonymity implies IND-CCA2 public key bit-encryption. To appreciate the strength of the [10] definition of security of a group signature scheme,

Group Signature with Full Revocation

KeyGen(): As in the basic scheme except now we also include a random element f from QR_n in the public key, as well as $w \in_R \mathrm{QR}_n$.

Join: The Join protocol remains the same except now the member chooses a random element $s_i \in \mathbb{Z}_Q$ and gets $y_i = (af^{s_i}g^{x_i}h^{r'_i+r''_i})^{E_i^{-1}} \bmod n$.

Sign(vk, sk_i, m)**:** Proceed as in the Join and Revoke scheme and select at random $d_s \in \{0,1\}^{\ell_c+\ell_Q+\ell_s}$.

Set $u = h^r y_i w_i \bmod n, v = h^{s-rd_e} f^{d_s} g^{d_x} (y_i w_i)^{-d_e} \bmod n$.

Let furthermore $U_1 = F^R \bmod P, U_2 = G^R G^{x_i} \bmod P, U_3 = H^R H^{e_i} \bmod P, U_4 = F^{Rs_i} \bmod P$ and $V_1 = F^S \bmod P, V_2 = G^S G^{d_x} \bmod P, V_3 = H^S H^{d_e} \bmod P, V_4 = F^{Rd_s} \bmod P$.

Compute a challenge $c = \mathrm{hash}(m, u, v, U_1, U_2, U_3, U_4, V_1, V_2, V_3, V_4)$.

Set $z_e = ce_i + d_e, z_E = c2^{\ell_e+\ell_c+\ell_s} + z_e, z_r = cr_i + z_E r + s, z_x = cx_i + d_x, z_s = cs_i + d_s$ and $Z_R = cR + S \bmod Q$.

Signature: $\sigma = (c, u, U_1, U_2, U_3, U_4, z_r, z_x, z_e, z_s, Z_R)$.

Verify(vk, m, σ)**:** Check that all elements are in the correct intervals and respectively belong to $\mathbb{Z}_n^*$ and $\mathbb{Z}_P^*$.

Compute $v = (aw)^c f^{z_s} g^{z_x} h^{z_r} u^{-z_E} \bmod n$ and $V_1 = F^{Z_R} U_1^{-c} \bmod P, V_2 = G^{Z_R+z_x} U_2^{-c} \bmod P, V_3 = H^{Z_R+z_e} U_3^{-c} \bmod P, V_4 = U_1^{z_s} U_4^{-c} \bmod P$.

Verify that $c = \mathrm{hash}(m, u, v, U_1, U_2, U_3, U_4, V_1, V_2, V_3, V_4)$.

Open$(gmsk, m, \sigma)$**:** The opening protocol remains the same.

Revoke$(gmsk, i)$**:** The revocation protocol remains the same.

FullRevoke$(gmsk, i)$**:** Look up s_i and publish it on the certificate revocation list. Execute Revoke$(gmsk, i)$.

Since s_i is now public anybody may check in old signatures whether $U_4^{\frac{P-1}{Q}} = U_1^{\frac{P-1}{Q}s_i} \bmod P$ and therefore whether the signatures have been formed by the fully revoked member.

Figure 7.3: Group Signature with Full Revocation.

let us see that full-anonymity implies CCA2 secure public key bit-encryption.

Theorem 7.2 *If a group signature scheme satisfying full-anonymity exists, then an* IND-CCA2 *public key cryptosystem for encrypting bits exists.*

Sketch of proof. We set the group signature scheme up with just two identities i_0, i_1. The secret keys corresponding to these two members is published, they are the public key of the cryptosystem. To encrypt a bit b, we use sk_{i_b} to sign the message $m = 0$. The group manager's key $gmsk$ corresponds to the secret key of the cryptosystem. With $gmsk$, it is possible to open the signature to see whether it was signed with sk_{i_0} or sk_{i_1}. This means that the bit b can be recovered.

Let us now restrict ourselves to adversaries that output $(i_0, i_1, 0)$ in the first phase. Then the definition of full-anonymity corresponds exactly to the definition of IND-CCA2 security. □

[2] speculate whether it is possible to construct a group signature scheme based only on one-way functions. Following [52] we believe it is not possible to construct public key encryption from one-way functions, and therefore not possible to construct a group signature scheme from one-way functions that satisfies the security definition of [10].

A group signature scheme based on one-way functions and NIZK arguments. We will present a group signature scheme with full-traceability and anonymity based on the assumptions that one-way functions exist and that non-interactive zero-knowledge arguments exist.

Recall that one-way functions imply the existence of pseudorandom functions, signature schemes secure against existential forgery under adaptive chosen message attack and statistically binding commitment to any string. As shown in [35] the statistically binding commitment scheme based on one-way functions from [63] can be made non-interactive.

The central idea in the group signature scheme we are going to present is that a member can demonstrate membership by producing a signature on the message under an authorized verification key that is part of the public key.

There is the question how the group manager will be able to tell the members apart. We let the public key contain a commitment to a seed s_i for a pseudorandom function. By evaluating this function on a randomly chosen element r, the member allows the group manager to identify him. On the other hand, nobody else can tell the pseudorandom string apart from a random string, and therefore cannot tell who signed the message. To force the member to use a correct seed we require him to produce a NIZK argument of correctness.

The protocol. The protocol is described in Figure 7.4. For simplicity and to facilitate comparison with the [10] definition we do not include the join protocol in this description.

Security.

Group Signature Based on One-Way Functions and NIZK Arguments

KeyGen: Generate (vk_{GM}, sk_{GM}) as the public keys for a one-time signature scheme. Let pk be a public key for an unconditionally binding commitment scheme. Select at random a seed s_{GM} and let c_{GM} be a commitment to s_{GM} using randomness r_{GM}.

For each member generate signature keys (vk_i, sk_i). Select also a seed s_i for the pseudorandom function. Let c_i be a commitment to s_i using randomness r_i.

The public key is $vk = (vk, pk, c_{GM}, vk_{GM}, c_1, vk_1, \ldots, c_k, vk_k)$. The group manager's secret key is $gmsk = (s_1, \ldots, s_k)$.

Member i's secret key is $sk_i = (vk, vk_i, sk_i, s_i, r_i)$.

Sign: The member wishing to sign message m does the following.

She generates a strong one-time signature key pair (vk_{OT}, sk_{OT}). She then forms $\sigma_{OT} = \mathrm{Sign}(sk_i, vk_{OT})$. She picks at random point r. She sets $e = PRF_{s_i}(r) \oplus \sigma_{OT}$. Let p be a NIZK argument that e is indeed the pseudorandom function evaluated in r under a seed s_i in one of the commitments, XORed with a valid signature on vk_{OT} under key vk_i, or it is the pseudorandom function evaluated in r under seed s_{GM} XORed with vk_{OT} signed under key vk_{GM}. Having done all this she computes a one-time signature $\sigma_{all} = \mathrm{Sign}(sk_{OT}, m, vk_{OT}, r, e, p)$.

The signature consists of $\sigma = (vk_{OT}, r, e, p, \sigma_{all})$.

Verify: The verifier wishing to verify a signature $(vk_{OT}, r, e, p, \sigma_{all})$ on message m checks the signature σ_{all} and that the argument p is valid.

Open: The group manager, wishing to open a signature $(vk_{OT}, r, e, p, \sigma_{all})$ on message m first verifies the signature. Then he goes through the list of seeds s_i that he has given members computing $\mathrm{PRF}_{s_j}(r)$. When he reaches $j = i$ he can extract a signature σ_{OT} on vk_{OT}. Then he knows that i is the member and he may demonstrate this by revealing σ_{OT} together with the key vk_i.

Figure 7.4: Group signature based on any one-way function and any general NIZK argument

Theorem 7.3 *The group signature scheme described in Figure 7.4 has full-traceability and anonymity.*

Sketch of proof.
Full-traceability. The zero-knowledge argument implies that either a valid

signature points to one of the members or e contains a signature under vk_{GM}. But if the latter is the case then we would be able to forge signatures. Therefore, e must be such that it points to one of the members.

Suppose now that the adversary is able to generate a valid signature pointing to member i, without corrupting party i or querying (i, m) to the signing oracle. We will use the adversary to forge a signature under vk_i or vk_{GM}. First, note that since we are using a strong one-time signature scheme the adversary cannot reuse a public key vk_{OT} that it has received from $Sign(sk_\cdot, \cdot)$. By the NIZK argument p we have $e = PRF_{s_i}(r) \oplus \sigma_{OT}$, where σ_{OT} is a signature under vk_i or vk_{GM}. However, with the knowledge of s_i and s, we may then extract this signature and therefore we have made a forgery.

Anonymity. Imagine the two experiments defining anonymity where we use respectively sk_{i_0} or sk_{i_1} to generate the signature are distinguishable. In that case, the two experiments where we first guess i_0, i_1 at random, then run the two experiments but only output d if we guessed correctly and the adversary did not query m, σ to Open or i_0, i_1 to Corrupt, are also distinguishable. Now, set these two experiments up such that instead of using s_i and vk_i in forming the e's when making signatures under the Sign oracle we instead simulate the arguments. By the zero-knowledge property of the NIZK arguments, this means that we still have two distinguishable experiments. Imagine further, that we form c_{i_0} and c_{i_1} as commitments to 0 but still use s_{i_0} and s_{i_1} when making signatures. By the hiding property of the commitment scheme, the two experiments are again indistinguishable. Now, modify further the experiments such that we use $e = PRF_s(r) \oplus \text{Sign}(sk_{GM}, vk_{OT})$ whenever Sign outputs a signature. Since PRF is a pseudorandom function, the two experiments are still indistinguishable. Finally, instead of simulating arguments make instead arguments using vk and s. Now, the two experiments are both distinguishable and identical, a clear contradiction. □

We do not know of any construction of public key encryption from one-way functions and non-interactive zero-knowledge arguments. Theorems 7.2 and 7.3 therefore indicate that a group signature scheme having full-anonymity may require stronger assumptions than what is needed to obtain anonymity.

The scheme in Figure 7.4 can easily be extended to a traceable signature scheme [55]. Theorems 7.2 and 7.3 can then be seen as indications that group signatures require stronger assumptions than traceable signature schemes.

7.7 Conclusion

We have contributed in two directions. On the practical side we have suggested a new group signature scheme which is efficient and can be extended to support

revocation and full revocation. Of a more theoretical nature, we have noted that full-anonymity may require stronger assumptions than what is needed to achieve anonymity.

This leaves an interesting open problem of suggesting a group signature scheme that supports full revocation but at the same time has full-anonymity.

Another obvious problem is to suggest a good security model for group signatures that includes revocation. We have shown that in the static model, our group signature realizes security model of [10], but for the dynamic case, our security proofs are more of the hand-waving nature.

Bibliography

[1] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure group signature scheme. In *proceedings of CRYPTO '00, LNCS series, volume 1880*, pages 255–270, 2000.

[2] G. Ateniese and B. de Medeiros. Efficient group signatures without trapdoors. In *proceedings of ASIACRYPT '03, LNCS series, volume 2894*, pages 246–268, 2003. Revised paper available at `http://eprint.iacr.org/2002/173`.

[3] G. Ateniese, D. X. Song, and G. Tsudik. Quasi-efficient revocation in group signatures. In *proceedings of Financial Cryptography '02*, pages 183–197, 2002.

[4] B. Barak. How to go beyond the black-box simulation barrier. In *proceedings of FOCS '01*, pages 106–115, 2001.

[5] B. Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *proceedings of FOCS '02*, pages 345–355, 2002.

[6] B. Barak and R. Pass. On the possibility of one-message weak zero-knowledge. In *proceedings of TCC '04, LNCS series, volume 2951*, pages 121–132, 2004.

[7] O. Baudron, P.-A. Fouque, D. Pointcheval, G. Poupard, and J. Stern. Practical multi-candidate election scheme. In *proceedings of PODC '01*, pages 274–283, 2001.

[8] M. Bellare. A note on negligible functions. *Journal of Cryptology*, 15(4):271–284, 2002.

[9] M. Bellare, A. Boldyreva, and A. Palacio. An uninstantiable random-oracle-model scheme for a hybrid encryption problem. In *proceedings of EUROCRYPT '04, LNCS series, volume 3027*, pages 171–188, 2004. Full paper available at `http://eprint.iacr.org/2003/077`.

[10] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *proceedings of EUROCRYPT '03, LNCS series, volume 2656*, pages 614–629, 2003.

[11] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS '93*, pages 62–73, 1993.

[12] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *proceedings of STOC '88*, pages 103–112, 1988.

[13] F. Boudot. Efficient proofs that a committed number lies in an interval. In *proceedings of EUROCRYPT '00, LNCS series, volume 1807*, pages 431–444, 2002.

[14] J. Camenisch and J. Groth. Group signatures: Better efficiency and new theoretical aspects. In *proceedings of SCN '04, LNCS series*, 2004.

[15] J. Camenisch and J. Groth. Group signatures: Revisiting definitions, assumptions and revocation, 2004. Manuscript.

[16] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *proceedings of CRYPTO '02, LNCS series 2442, volume*, pages 61–76, 2002.

[17] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *proceedings of FOCS '01*, pages 136–145, 2001. Full paper available at `http://eprint.iacr.org/2000/067`.

[18] R. Canetti and M. Fischlin. Universally composable commitments. In *proceedings of CRYPTO '01, LNCS series, volume 2139*, pages 19–40, 2001. Full paper available at `http://eprint.iacr.org/2001/055`.

[19] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *proceedings of STOC '98*, pages 209–218, 1998.

[20] R. Canetti, O. Goldreich, and S. Halevi. On the random-oracle methodology as applied to length-restricted signature schemes, 2004.

[21] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *proceedings of STOC '02*, pages 494–503, 2002. Full paper available at `http://eprint.iacr.org/2002/140`.

[22] D. Chaum and E. van Heyst. Group signatures. In *proceedings of EUROCRYPT '91, LNCS series, volume 547*, pages 257–265, 1991.

[23] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally eficient multi-authority election scheme. In *proceedings of EUROCRYPT '97, LNCS series, volume 1233*, pages 103–118, 1997.

[24] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. In *proceedings of CRYPTO '98, LNCS series, volume 1462*, pages 13–25, 1998. Full paper available at `http://eprint.iacr.org/2001/108`.

[25] I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *proceedings of EUROCRYPT '00, LNCS series, volume 1807*, pages 418–430, 2000.

[26] I. Damgård and E. Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *proceedings of ASIACRYPT '02, LNCS series, volume 2501*, pages 125–142, 2002.

[27] I. Damgård and J. Groth. Non-interactive and reusable non-malleable commitment schemes. In *proceedings of STOC '03*, pages 426–437, 2003.

[28] I. Damgård, J. Groth, and G. Salomonsen. The theory and implementation of an electronic voting system. In D. Gritzalis, editor, *Secure Electronic Voting*, pages 77–100. Kluwer Academic Publishers, 2003.

[29] I. Damgård and M. J. Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *proceedings of PKC '01, LNCS series, volume 1992*, 2001.

[30] I. Damgård and M. J. Jurik. A length-flexible threshold cryptosystem with applications. In *proceedings of ACISP '03, LNCS series, volume 2727*, pages 350–364, 2003.

[31] I. Damgård, M. J. Jurik, and J. B. Nielsen. A generalization of paillier's public-key system with applications to electronic voting. Manuscript, 2003. `http://www.brics.dk/~ivan/GenPaillierfinaljour.ps`.

[32] I. Damgård and J. B. Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In *proceedings of CRYPTO '02, LNCS series, volume 2442*, pages 581–596, 2002. Full paper available at `http://www.brics.dk/RS/01/41/index.html`.

[33] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero knowledge. In *proceedings of CRYPTO '01, LNCS series, volume 2139*, pages 566–598, 2002.

[34] A. De Santis and G. Persiano. Zero-knowledge proofs of knowledge without interaction. In *proceedings of FOCS '92*, pages 427–436, 1992.

[35] G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. Non-interactive and non-malleable commitment. In *proceedings of STOC '98)*, pages 141–150, 1998.

[36] G. Di Crescenzo, J. Katz, R. Ostrovsky, and A. Smith. Efficient and non-interactive non-malleable commitment. In *proceedings of EUROCRYPT '01, LNCS series, volume 2045*, pages 40–59, 2001.

[37] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. *SIAM J. of Computing*, 30(2):391–437, 2000. Earlier version at STOC '91.

[38] U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero knowledge proofs based on a single random string. In *proceedings of FOCS '90*, pages 308–317, 1990.

[39] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *proceedings of CRYPTO '86, LNCS series, volume 263*, pages 186–194, 1986.

[40] M. Fischlin. On the impossibility of constructing non-interactive statistically-secret protocols from any trapdoor one-way function. In *proceedings of CT-RSA '02, LNCS series, volume 2271*, pages 79–95, 2002.

[41] M. Fischlin and R. Fischlin. Efficient non-malleable commitment schemes. In *proceedings of CRYPTO '00, LNCS series, volume 1880*, pages 413–431, 2000.

[42] J. Furukawa. Efficient, verifiable shuffle decryption and its requirement of unlinkability. In *proceedings of PKC '04, LNCS series, volume 2947*, pages 319–332, 2004.

[43] J. Furukawa, H. Miyauchi, K. Mori, S. Obana, and K. Sako. An implementation of a universally verifiable electronic voting scheme based on shuffling. In *proceedings of Financial Cryptography '02, LNCS series, volume 2357*, pages 16–30, 2002.

[44] J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. In *proceedings of CRYPTO '01, LNCS series, volume 2139*, pages 368–387, 2001.

[45] J. A. Garay, P. D. MacKenzie, and K. Yang. Strengthening zero-knowledge protocols using signatures. In *proceedings of EUROCRYPT '03, LNCS series, volume 2656*, pages 177–194, 2003. Full paper available at `htpp://eprint.iacr.org/2003/037`.

[46] O. Goldreich and H. Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal of Computation*, 25(1):169–192, 1996.

[47] O. Goldreich and Y. Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994.

[48] O. Goldreich and V. Rosen. On the security of modular exponentiation with application to the construction of pseudorandom generators. *Journal of Cryptology*, 16(2):71–93, 2003.

[49] S. Goldwasser and Y. T. Kalai. On the (in)security of the fiat-shamir paradigm. In *proceedings of FOCS '03*, pages 102–, 2003. Full paper available at `http://eprint.iacr.org/2003/034`.

[50] J. Groth. A verifiable secret shuffle of homomorphic encryptions. In *proceedings of PKC '03, LNCS series, volume 2567*, pages 145–160, 2003.

[51] J. Groth. Evaluating security of voting schemes in the universal composability framework. In *proceedings of ACNS '04, LNCS series, volume 3089*, pages 46–60, 2004.

[52] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *proceedings of STOC '89*, pages 44–61, 1989.

[53] N. Ishida, S. Matsuo, and W. Ogata. Divisible voting scheme. In *ISC '03, LNCS series, volume 2851*, pages 137–150, 2003.

[54] S. Jarecki and A. Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In *proceedings of EUROCRYPT '00, LNCS series, volume 1807*, pages 221–242, 2000.

[55] A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. Cryptology ePrint Archive, Report 2004/007, 2004. `http://eprint.iacr.org/`.

[56] J. Kilian and E. Petrank. An efficient noninteractive zero-knowledge proof system for np with general assumptions. *Journal of Cryptology*, 11(1):1–27, 1998.

[57] Y. Lindell. Parallel coin-tossing and constant round secure two-party computation. In *proceedings of CRYPTO '01, LNCS series, volume 2139*, pages 408–432, 2001. Full paper available at `http://eprint.iacr.org/2001/107`.

[58] H. Lipmaa. On diophantine complexity and statistical zero-knowledge arguments. In *proceedings of ASIACRYPT '03, LNCS series, volume 2894*, pages 398–415, 2003.

[59] H. Lipmaa, N. Asokan, and V. Niemi. Secure vickrey auctions without threshold trust. In *proceedings of Financial Cryptography '02, LNCS series, volume 2357*, pages 87–101, 2002.

[60] P. MacKenzie. Personal communication, 2003.

[61] P. D. MacKenzie and K. Yang. On simulation-sound trapdoor commitments. In *proceedings of EUROCRYPT '04, LNCS series, volume 3027*, pages 382–400, 2004. Full paper available at `http://eprint.iacr.org/2003/252`.

[62] A. J. Menezes, P. C. van Oorschoot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[63] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[64] M. Naor. On cryptographic assumptions and challenges. In *proceedings of CRYPTO '03, LNCS series, volume 2729*, pages 96–109, 2003.

[65] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *proceedings of STOC '90*, pages 427–437, 1990.

[66] A. C. Neff. A verifiable secret shuffle and its application to e-voting. In *CCS '01*, pages 116–125, 2001. Full paper available at `http://www.votehere.net/vhti/documentation/egshuf.pdf`.

[67] J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *proceedings of CRYPTO '02, LNCS series, volume 2442*, pages 111–126, 2002.

[68] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In *proceedings of EUROCRYPT '98, LNCS series, volume 1403*, pages 308–318, 1998.

[69] P. Paillier. Public-key cryptosystems based on composite residuosity classes. In *proceedings of EUROCRYPT '99, LNCS series, volume 1592*, pages 223–239, 1999.

[70] P. Paillier. Composite-residuosity based cryptography - an overview. *Cryptobytes*, 5(1):20–26, 2002.

[71] R. Pass. On deniability in the common reference string and random oracle model. In *proceedings of CRYPTO '03, LNCS series, volume 2729*, pages 316–337, 2003.

[72] M. Rabin and J. Shallit. Randomized algorithms in number theory. *Commun. Pure and Appl. Math*, 39, suppl.:S240–S256, 1986.

[73] A. Sahai. Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security. In *proceedings of FOCS '01*, pages 543–553, 2001.

[74] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, 15(2):75–96, 2002.

[75] G. Tsudik and S. Xu. Accumulating composites and improved group signing. In *proceedings of ASIACRYPT '03, LNCS series, volume 2894*, pages 269–286, 2003.

[76] Y. Zheng and J. Seberry. Immunizing public key cryptosystems against chosen ciphertext attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):715–724, 1993.

[77] H. Zhu. A formal proof of zhu's signature scheme. Cryptology ePrint Archive, Report 2003/155, 2003. http://eprint.iacr.org/.

Recent BRICS Dissertation Series Publications

DS-04-3 **Jens Groth.** ***Honest Verifier Zero-knowledge Arguments Applied*****. October 2004. PhD thesis. xii+119 pp.**

DS-04-2 **Alex Rune Berg.** ***Rigidity of Frameworks and Connectivity of Graphs*****. July 2004. PhD thesis. xii+173 pp.**

DS-04-1 **Bartosz Klin.** ***An Abstract Coalgebraic Approach to Process Equivalence for Well-Behaved Operational Semantics*****. May 2004. PhD thesis. x+152 pp.**

DS-03-14 **Daniele Varacca.** ***Probability, Nondeterminism and Concurrency: Two Denotational Models for Probabilistic Computation*****. November 2003. PhD thesis. xii+163 pp.**

DS-03-13 **Mikkel Nygaard.** ***Domain Theory for Concurrency*****. November 2003. PhD thesis. xiii+161 pp.**

DS-03-12 **Paulo B. Oliva.** ***Proof Mining in Subsystems of Analysis*****. September 2003. PhD thesis. xii+198 pp.**

DS-03-11 **Maciej Koprowski.** ***Cryptographic Protocols Based on Root Extracting*****. August 2003. PhD thesis. xii+138 pp.**

DS-03-10 **Serge Fehr.** ***Secure Multi-Player Protocols: Fundamentals, Generality, and Efficiency*****. August 2003. PhD thesis. xii+125 pp.**

DS-03-9 **Mads J. Jurik.** ***Extensions to the Paillier Cryptosystem with Applications to Cryptological Protocols*****. August 2003. PhD thesis. xii+117 pp.**

DS-03-8 **Jesper Buus Nielsen.** ***On Protocol Security in the Cryptographic Model*****. August 2003. PhD thesis. xiv+341 pp.**

DS-03-7 **Mario José Cáccamo.** ***A Formal Calculus for Categories*****. June 2003. PhD thesis. xiv+151.**

DS-03-6 **Rasmus K. Ursem.** ***Models for Evolutionary Algorithms and Their Applications in System Identification and Control Optimization*****. June 2003. PhD thesis. xiv+183 pp.**

DS-03-5 **Giuseppe Milicia.** ***Applying Formal Methods to Programming Language Design and Implementation*****. June 2003. PhD thesis. xvi+211.**

DS-03-4 **Federico Crazzolara.** ***Language, Semantics, and Methods for Security Protocols*****. May 2003. PhD thesis. xii+160.**

DS-03-3 **Jiří Srba.** ***Decidability and Complexity Issues for Infinite-State Processes*****. 2003. PhD thesis. xii+171 pp.**