Nada är en gemensam institution mellan
Kungliga Tekniska högskolan och Stockholms universitet.

Two Topics in Cryptography: Lattice Problems and the Security of Protocols

MÅRTEN TROLIN

Avhandling som med tillstånd av Kungliga Tekniska högskolan
framlägges till offentlig granskning för avläggande av teknologie licentiatexamen
fredagen den 11 mars 2005 kl 10.00
i sal E2, Lindstedtsvägen 3,
Kungliga Tekniska högskolan, Stockholm.

TRITA-NA-0504
ISSN 0348-2952
ISRN KTH/NA/R–05/04–SE
ISBN 91-7283-965-1
© Mårten Trolin, mars 2005

TRITA-NA-0504
ISSN 0348-2952
ISRN KTH/NA/R–05/04–SE
ISBN 91-7283-965-1

KTH Numerisk analys och datalogi
SE-100 44 Stockholm
SWEDEN

© Mårten Trolin, mars 2005

Tryck: Universitetsservice US AB

Abstract

In this thesis we present new results in two areas – cryptographic protocols and lattice problems.

- We present a new protocol for electronic cash which is designed to function on hardware with limited computing power. The scheme has provable security properties and low computational requirements, but it still gives a fair amount of privacy. Another feature of the system is that there is no master secret that could be used for counterfeiting money if stolen.
- We introduce the notion of hierarchical group signatures. This is a proper generalization of group signatures, which allows multiple group managers organized in a tree with the signers as leaves. For a signer that is a leaf of the subtree of a group manager, the group manager learns which of its children that (perhaps indirectly) manages the signer. We provide definitions for the new notion and construct a scheme that is provably secure given the existence of a family of trapdoor permutations. We also present a construction which is relatively practical, and prove its security in the random oracle model under the strong RSA assumption and the DDH assumption.
- We show a weakness in the specification for offline capable EMV payment cards. The weakness, which applies to cards without RSA capability, enables an attacker to duplicate a card and make transactions that cannot be tied to the original card.
- We give a method for approximating any n-dimensional lattice with a lattice Λ whose factor group $\mathbb{Z}^n/\Lambda$ has $n-1$ cycles of equal length with arbitrary precision. We also show that a direct consequence of this is that the Shortest Vector Problem and the Closest Vector Problem cannot be easier for this type of lattices than for general lattices.

Sammanfattning

Vi presenterar nya resultat inom två områden inom kryptografi – kryptografiska protokoll och gitterproblem.

- Vi beskriver ett nytt protokoll för elektroniska pengar. Protokollet är avsett för enheter med liten beräkningskraft, är bevisbart säkert men ger ändå användare ett visst mått av anonymitet. Dessutom har banken i protokollet ingen hemlighet som kan stjälas och användas för att tillverka pengar.
- Vi introducerar begreppet hierarkiska gruppsignaturer, vilket är en generalisering av gruppsignaturer. I ett system för hierariska gruppsignaturer finns det flera gruppchefer organiserade i ett träd med gruppmedlemmarna som löv. Givet en signatur från ett löv kan en gruppchef som är rot i ett delträd där lövet ingår avgöra vilket av hans barn som signaturen (eventuellt indirekt) tillhör. Vi ger definitioner för hierarkiska gruppsignaturer och beskriver en konstruktion som är säker om det existerar en familj lönndörrspermutationer. Vi ger också en relativt praktisk konstruktion som är bevisbart säker i slumporakelmodellen under starka RSA-antagandet och DDH-antagandet.
- Vi beskriver en svaghet i specifikationen för EMV-kort avsedda för offline-betalningar. Svagheten gäller kort utan RSA-funktionalitet och ger en angripare möjlighet att genomföra transaktioner som inte kan kopplas till hans kort.
- Vi ger en metod för att med godtycklig noggrannhet approximera ett n-dimensionellt gitter med ett gitter Λ sådant att kvotgruppen $\mathbb{Z}^n/\Lambda$ har $n-1$ cykler av lika längd. Vi visar vidare att en direkt konsekvens av detta är att kortastevektor-problemet och närmstavektor-problemet inte kan vara lättare för dessa gitter än för allmänna gitter.

Acknowledgments

Now that the scientific part of the thesis is ready, the hardest part remains, namely to remember all the people who helped me finish the thesis. Many of my colleagues have helped me proof-read manuscripts, have listened to and commented on seminars, or have taken time to discuss topics related or unrelated to my research. Although it is impossible to mention everyone by name, I will do my best.

First of all I would like to thank my advisor Johan Håstad for all the help and excellent ideas. Without this support, my work had not been possible. I would also like to thank Mikael Goldmann for good advice and useful tips. I have had many interesting and rewarding discussions with my fellow students. These discussions have often given me new insights and new perspective on my work. Especially I would like to thank Gustav Hast for valuable discussions on the protocol for electronic cash as well on some of the ideas behind the group signature scheme. I would also like to mention Lars Engebretsen, whose comments gave me a new understanding of the cycle structure of a lattice. Of course Douglas Wikström, with whom I co-authored the paper that is the basis for the chapter on group signatures, deserves a special acknowledgment.

Part of the thesis was written while I was visiting University of Latvia in Riga. I am very grateful to Janis Barzdins and Rusins Freivalds, who made this visit possible.

Finally I would like to thank my beloved Anna and our son Leo. Without Anna's support and encouragement I would not have been able to finish this work, and Leo has shown to me that there is more to life than theoretical computer science.

Contents

Chapter 1

Introduction

1.1 Confidentiality and Authenticity

When the word "cryptography" is mentioned, what first comes to mind is probably sending secret messages. This is justified, as hiding information from eavesdroppers, *confidentiality*, is the traditional reason to use cryptography. An analogy is to send a message in a sealed envelope (or maybe in a locked safe, although it is debatable how realistic such an analogy is). Sometimes we are not primarily interested in hiding information, but rather in ensuring that information isn't modified or counterfeited, *authenticity*. By this we mean that the receiver can be convinced that sender is who he claims to be, and that the message has not been altered on the way. The analogy here is to sign a paper with the message on it. Since signatures are assumed to be hard to forge, a signature identifies the sender.

In an environment where messages are mainly sent electronically, we need methods to achieve confidentiality and authenticity by digital means, and this is one major part of what cryptographic research is about. The traditional approach is to set up a key k and define a function E to encrypt and a function D to decrypt so that $D_k(E_k(m)) = m$ for any legal message m. We will call m the *plaintext* and the encryption $E_k(m)$ the *ciphertext*. Since we want the system to be secure, we want it to be infeasible to compute any useful information about the plaintext from the ciphertext, provided that the key k is unknown. We even want it to be infeasible if certain side information is known, such as a subset of legal messages from which m is drawn, or encryptions of other messages under the same key.

Consider the functions necessary to ensure that a message isn't counterfeited or modified. The usual approach is to define a function S to create a *message authentication code* (MAC) and a function V to verify that a MAC is valid. The function S takes as input a message m and a key k and returns a MAC. The function V takes a key, a message and a MAC, and returns 1 if the MAC is valid and 0 otherwise. It must hold that $V_k(m, S_k(m)) = 1$, and it should be infeasible to compute a message m and a MAC s such that $V_k(m, s)$ without knowledge of

k. Also here the attacker may have access to side information such as MACs on messages of his choice.

1.2 Public Key Cryptography

Asymmetric Encryption Schemes

In the above definition, the same key is used for encryption and decryption. For a long time, this was the only known way to perform cryptography. In the middle of the 1970s, a major breakthrough was made when methods to perform *asymmetric* cryptography were discovered. Asymmetric systems use two keys, the *public key*, pk and the *private key* (sometimes called *secret key*), sk. The public key is used to encrypt, and the private key to decrypt so that $D_{sk}\left(E_{pk}(m)\right) = m$. The public key can be published, since it is used only for encryption, but the private key must be kept secret.

Let us now compare this with symmetric cryptosystems to see what the differences may mean in practice. Assume ten people work at the same company, and that they want to be able to send encrypted messages to each other. First consider a symmetric cryptosystem. One solution is to have a single common key that everything is encrypted with, but there are several drawbacks with this approach. Someone who gets hold of the key (for example by bribing one of the employees) is able to read all messages sent. Also any employee can read any message, even it wasn't meant for him. If an employee quits, a new key has to be set up and distributed in a secure manner. A second solution is to have one key between every pair of employees. Then only the intended recipient can read his messages, and if one employees sells (or accidently discloses) his keys, only the messages sent or received by that employee can be read. However, the number of keys necessary for such a system is high. Our ten employees need a total of 45 keys. Although this number may not seem very high, we must take into account that agreeing on a symmetric key is a cumbersome task. It is not advisable to the keys electronically, since they can be eavesdropped, and if a key is sent by mail, there is always the risk that someone opens the envelope and gets the key. The only safe way is to meet in person. Now consider a company with 1000 employees. Then a total of $499,500$ keys are necessary! It is obvious that symmetric cryptosystems have certain drawbacks.

Now let us consider using asymmetric cryptography to solve the problem. Each of the ten employees generates a key pair consisting of a private and a public key. The public keys are published, say in the company phone book. If Alice wants to send a message to Bob, she looks up Bob in the phone book, encrypts using his public key and sends the message. Bob uses his private key to decrypt, and no-one else can read the message. If the company hires new employees, each of them generates a key pair. No keys have to be exchanged under secure conditions.

Digital Signatures

Also authenticity can be achieved by asymmetric means. When a MAC is used, the same key is used for computing the MAC and and verifying it. Therefore only the intended recipient can check the validity of the message. Furthermore, ability to verify implies ability to compute a MAC, making it hard to use a signature as proof in case of a dispute. Therefore, in many situations, it is desirable to have a scheme in which it is possible to verify without being able to sign. Using asymmetric techniques we can construct a scheme where the *signing* is performed using the private key sk and the *verification* with the public key pk. Now it must hold that $V_{pk}\left(m, S_{sk}(m)\right) = 1$. This is also what we expect from real-world signing schemes – anyone can look at a signature and check whether it has been written by the putative sender (by comparing it with other signatures written by the same person), but no-one but the sender else should be able to produce such a signature.

A digital signature is in one sense more secure than a physical signature on paper. When a paper with the message written on it is signed, it is hard to ensure that the message is not altered afterwards. A forger may add new text to a signed document or combine pages from two or more signed documents into a new document. A secure digital signature scheme withstands attacks of this type, since the signature is tied to the message and becomes invalid if the message is modified.

1.3 Building Cryptographic Protocols

Two of the most important building blocks for cryptographic functions are *one-way functions*, i.e., functions that are easy to compute but hard to invert, *trapdoor functions*, i.e., functions that are one-way functions with the additional property that there is a secret which makes the function easy to invert. Take, for example, multiplication. It is easy to multiply two numbers, but no method is known that factors a numbers into its prime factors in reasonable time. It should be noted that the existence of one-way and trapdoor functions is a classical open problem, and a proof of their existence would be a major breakthrough. However, there are functions that have been subject to intensive research for more than thirty years, and no evidence contradicting the hypothesis that they are trapdoor functions have been found. It is therefore reasonable to assume that they are indeed trapdoor functions. From functions that are assumed to be trapdoor functions, it is possible to build cryptographic primitives, e.g., encryption and signature schemes.

To achieve more complex tasks, such as setting up a secure channel between parties who have not previously met or creating digital coins, we need to describe how to combine primitives to get the functionality we need. The result is called a protocol, and the protocol describes how the participants should act. A protocol can be seen as a set of algorithms, one for each participant.

A protocol may be interactive or non-interactive. An interactive protocol is used when the parties can send messages to each other in an interleaved manner. An example may be a user logging on to a web-site. In a non-interactive protocol the sender creates the message on his own, and only then sends it to the receiver. Encrypting and signing emails are a typical examples of non-interactive protocols.

1.4 Efficient vs. Practical Protocols

Naturally we want our protocols to be as efficient as possible. However, in different contexts effiency may have different meanings. The common definition of an efficient algorithm is that the execution time is bounded by a polynomial in the size of the input. For example, the grade school algorithm for multiplication is polynomial time, since the number of steps needed is less than $2n^2$, where n is the number of digits of each factor.

An example of an algorithm that is not polynomial is factoring by exhaustive search. To factor an n-bit number m we may need to check each number up to $\sqrt{m}$, that is, $2^{n/2}$ different numbers. Even if we assume that we can check divisibility in a single step, we still need an exponential number of steps before we are guaranteed to have a result.

It is clear that this definition of efficient algorithms does not cover everything we need from an algorithm to be usable in practice. If we design an algorithm that runs in n^{30} steps, it would still be considered efficient according to the above definition. However, the algorithm would be impossible to use in practice except for extremely small inputs.

In this thesis we focus on protocols that are not only efficient in the above meaning, but that are practical. Therefore the protocols must be specified in such detail that it is possible to analyze their running time precisely and not only show that it is bounded by some polynomial. Also, being practical is not a strict definition. In some cases, we want a protocol that can be executed on devices with little computing power such as smart-cards or mobile phones. In other cases it is enough if the protocol runs reasonably fast on a personal computer, and in still other cases the protocol will run on a server with large storage capabilities.

1.5 Security of Cryptographic Primitives and Protocols

Obviously we want the cryptographic primitives we use to be secure. However, we need to define precisely what we mean by security of a primitive. Let us consider an encryption scheme. One definition of security is that the scheme is secure if an attacker who sees a ciphertext cannot recover the plaintext. However, in some scenarios this is not enough, since the attacker may have access to additional information. Maybe the attacker knows that the plaintext is either “yes” or “no”, and maybe the attacker has seen encryptions of other plaintexts. Maybe the attacker even has seen encryptions of “yes” and “no”. A good cryptosystem should remain secure even under these circumstances. For example, to remain secure even if the attacker knows encryptions of “yes” and “no”, the encryption must be probabilistic, so that when the same plaintext gives different ciphertexts when encrypted several times.

Designing protocols that are as secure as the primitives used is not trivial. It may very well be the case that a protocol turns out to be insecure although all components used are secure. Also in the case of protocols, the term "secure" must be properly defined. Take, for example, a scheme for electronic cash involving customers, merchants and a bank. Naturally a customer should not be able to counterfeit money, but what happens if a customer and a merhcant collaborates to produce counterfeit money? Or maybe when two customers together try to create a coin that appears to be valid to the merchant but which is rejected by the bank? Obviously there are many subtle details when deciding what kind of security we want from a protocol. Therefore it is important to make a clear definition of security and to prove that the protocol fulfils those definition under some plausible assumptions.

1.6 Anonymity

Assume the cash you withdraw had your name on it. What would that mean? In most cases it wouldn't mean anything. No-one would be interested in knowing that it was you who bought that pack of chewing gum. You might feel a little bit uncomfortable if you knew that a curious trainee working in the pharmacy can keep track of what medicine you use. If the government can figure out your political viewpoint by monitoring what newspapers you purchase and what events you buy tickets to, you have reason to be really worried.

We often take anonymity for granted. If you purchase a newspaper with cash, it is not possible to trace the purchase back to you by looking at the coins you paid with. If you buy a couple of tokens for the metro, it is not possible to see if two trips were paid by tokens purchased at the same time. The simple reason neither coins nor metro tokens are traceable is that they don't have a serial number. The reason they don't have a serial number is that their low value don't make them an interesting target for counterfeiter – the cost of producing a fake coin or metro token probably exceeds its value.

Now you may argue that these transactions are not at all anonymous – if you go and buy the newspaper in person, anyone can see what you bought. However, the important point here is that it requires considerable resources to track a person that way, and it is impossible to do in an automated way on a large scale.

When the physical coins and metro tokens are replaced with electronic counterparts, the scenario is changed. The cost of copying an electronic coin, which is nothing but a sequence of zeros and ones, is next to nothing. Therefore even low-value coins need some kind of serial number to detect duplicates, and that potentially makes them traceable. One of the challenges when designing protocols for transactions that people assume to be anonymous is to make them anonymous also when performed electronically.

Before we can design anonymous protocols, we must decide what we mean by anonymity. One definition of anonymity is that a transaction cannot be connected to the identity of any involved party. This definition, however, is weaker than the anonymity of real-world transactions, because it does not say anything about connecting transactions. Assume, for example, that you use your electronic coins first to buy a train ticket that is mailed to your home and then to buy a political newsletter. If the coins are anonymous only in the above sense, the identity of the buyer of the newsletter may still be revealed if the two purchases can be connected. Clearly the latter kind of anonymity is preferable to the former.

If a protocol involves several parties, in the case of electronic coins a customer, a merchant and the bank, we may settle for anonymity only towards the merchant to make the protocol more efficient. In other words, the merchant cannot link two purchases, but once the coin reaches the bank, the bank can see who withdrew the coin. Another concept is *revokable anonymity*. Here some trusted third party (who could, for example, be a judge) can extract the identity from a coin, but otherwise the coin is anonymous towards both the bank and the vendor.

Although anonymity is desirable from the user's point of view, protocols that ensure anonymity tend to be less efficient than non-anonymous protocols. Also from a legal point of view anonymity might be problematic. If electronic coins are achieved through black-mailing or other illegal activities, anonymity works in favor of the criminal.

In an anonymous scheme for electronic coins the bank cannot monitor the flow of coins. It will detect irregularities only after a long period of time (if ever). This may be one reason why the schemes for electronic cash that are in use are non-anonymous.

1.7 Payment Systems

When making purchases, the most common ways to pay for the goods is either by using cash or by using a payment card or check. Cash has the property that it is anonymous and that it is possible to verify that it is valid by just looking at it and without calling the bank. This *offline* property of cash is important, and very desirable. It reduces communication costs, it makes the scheme more robust since it doesn't require the bank to be available, and it is fast. The merchant can deposit the cash with his bank, use it as change, buy goods, pay salary etc. Unfortunately cash also has the not so nice property that it can be stolen. A payment card or check, on the other hand, is not itself a proof that the customer has the money to pay. The issuer must be contacted to verify that the customer has the necessary funds, but once the transaction is completed, it cannot be stolen like cash. Since the merchant's name is part of the payment, no-one else can get credited for the transaction.

Digital payment systems try to mimic these properties. Systems for digital cash try to keep the anonymity of the customer, possibly with a trusted party that can revoke the anonymity. However, since a digital coin is just a bit-string, it can be copied and spent twice. The most common way to deal with this is to design the system so that the identity of the owner is revealed if the same coin is spent twice. Another solution is to make the system online, but then part of the motivation to use coins is lost.

Systems for digital cash often require that the merchant deposits the cash with the bank after the transaction rather than reuse it. However, digital cash may also have the useful property that in cannot be stolen while at the merchant, since the merchant's name is part of the transaction.

If digital cash does not completely correspond to cash in the real world, payment card transactions are easier to make purely electronic. In many cases this simply means that the physical signature on the receipt is replaced by a digital signature by the cardholder. Here, however, we can ask for more and make payment card transactions anonymous towards the merchant.

The goal then is to design a system such that two transactions cannot be linked by the merchants. The system will still be non-anonymous towards the issuer, since it must be able to charge the correct account. A trivial way to achieve anonymity towards the merchant is to give each cardholder not just one card number, but several one-time numbers. The bank keeps a list of which number belongs to which cardholder, and the cardholder makes sure each number is only used once. Provided that the card numbers are generated randomly, such a system would be anonymous towards the merchants.

1.8 Group Signatures

In this section we discuss a more general approach to the problem of creating anonymous credit cards. We use the concept of *group signatures*. In a group signature scheme, there are *group members* and a *group manager*. Group members can sign documents on behalf of the group, but the only information that someone other than the group manager gets is that *someone* in the group signed the document. The group manager, however, is able to determine the identity of a signer. As the alert reader already has seen, this is exactly what we need to make payment cards anonymous. The group members are the cardholders, and the issuer is the group manager. When making a payment, the cardholder produces a group signature on the transaction. The merchant verifies that the signature is produced by someone in the group of cardholders, but does not get any additional information. When the transaction is passed on to the card issuer, the issuer, who acts as group manager, extracts the identity of the cardholder to debit the correct account.

The scheme described above with group signatures works for payment cards when there is just one issuer, and every merchant sends all transactions directly to that issuer. In reality this is not the case. There is not just one but several issuers cooperating within a network. Rather than sending the transaction directly to the issuer, the merchant sends it to the network, which routes it to the issuer. The obvious way to solve the problem is to set up a group signature scheme for each issuer. With this solution we lose some anonymity, since the merchant learns the name of the issuer, and in some cases this can give quite a lot of information. Therefore we would like a variant of group signatures where there are group managers that only get partial information about the identity of the signer. More specifically, in the case of payment cards, we need a scheme such that the signature is anonymous to the merchant, the network can see which issuer the card belongs to, and the issuer sees the identity of the cardholder. Naturally this can be generalized so that there are several intermediate group managers that get more and more detailed information about the identity. In this thesis we describe such an extension of group signatures. Because of the hierarchical way information about the identity is revealed, we call the scheme *hierarchical group signatures*.

1.9 EMV Payment Cards

Still the majority of payment cards are equipped with a magnetic stripe where the cardholder data is encoded. Although a convenient and cheap solution, it has its security problems. The magnetic stripe can be copied and modified, making it a good target for counterfeit and fraud. The transactions made with a magnetic stripe are not digitally signed, making it possible to modify the transaction data after the transaction took place.

One alternative to the magnetic stripe is *smart-cards*. A smart-card is a tiny computer placed on a plastic card. As with any computer, it can store and process data. It can also have some parts of its memory protected from direct access. This is a very useful property to prevent copying and modification of cards.

Since the amount of money lost on fraud by the payment networks is growing, there is an on-going program to switch to smart-cards. The switch is currently in progress, with some issuers already issuing smart-cards, and some still using the magnetic stripe.

With smart-cards, the security is increased considerably. A smart-card cannot be copied or modified the same way a magnetic stripe can. It can hold secret data used only internally by the card. Smart-cards can sign transactions, thus ensuring they are sent to the payment network unmodified. Some smart-cards also contain a private key for authentication purposes. Since the private key is accessible only to the internal smart-card software, such a card cannot be duplicated. Cardholder data on a smart-card may be digitally signed by the issuer, preventing it from being modified as data on the magnetic stripe can.

With the magnetic stripe a cardholder can pay wherever his brand of card is accepted. He doesn't have to worry about who manufactured the terminal or which bank will process the payment, since all magnetic stripe cards and all terminals work according to the same standards. For the switch to smart-cards to be successful, the same interoperability is necessary also for smart-cards. Therefore an international, publicly available standard called EMV has been developed.

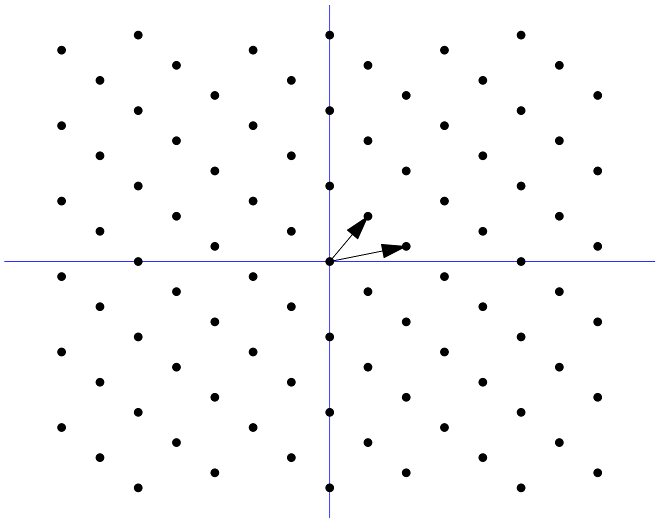


Figure 1.1: A two-dimensional lattice

In this thesis we point out a vulnerability in some EMV cards. Although the EMV standard builds on primitives in which no vulnerabilities are known, we show that certain EMV card configurations are insecure. The vulnerability would allow an attacker to use an EMV card to perform an unlimited number of offline transactions. EMV does allow for offline transactions, but there is a limit on the maximum number of consecutive offline transactions stored on the card. In Chapter 4 we show how to perform the attack, and also, where it is possible, how to configure a card to protect against the vulnerability.

1.10 Cryptography and Lattices

As we have seen, we need an underlying hard problem to design cryptosystems. One family of such problems are *lattice* problems. A lattice is defined as the set $\{\lambda_1\mathbf{b_1} + \lambda_2\mathbf{b_2} + \cdots + \lambda_n\mathbf{b_n}\}$ where λ_i are integers and $\mathbf{b_i} \in \mathbb{R}^n$. Put differently, a lattice is defined by n basis vectors in $\mathbb{R}^n$. The lattice consists of points in $\mathbb{R}^n$ (sometimes called *lattice vectors*) generated by adding combinations of the basis vectors with integral coefficients.[1] In Figure 1.1 a basis for a two-dimensional lattice is shown together with the lattice points generated by the lattice.

The two most studied lattice problems are the shortest vector problem (SVP) and the closest vector problem (CVP). In SVP, the task is to compute the shortest non-zero vector in a lattice given a basis for the lattice. (The zero vector obviously is the shortest vector for any lattice.) In Figure 1.2 the shortest vector in the two-dimensional lattice is marked, and here we see that in general the shortest vector is not one of the basis vectors, and that the shortest vector is never unique, since if $\mathbf{v}$

[1]This definition only gives *full-dimensional* lattices.

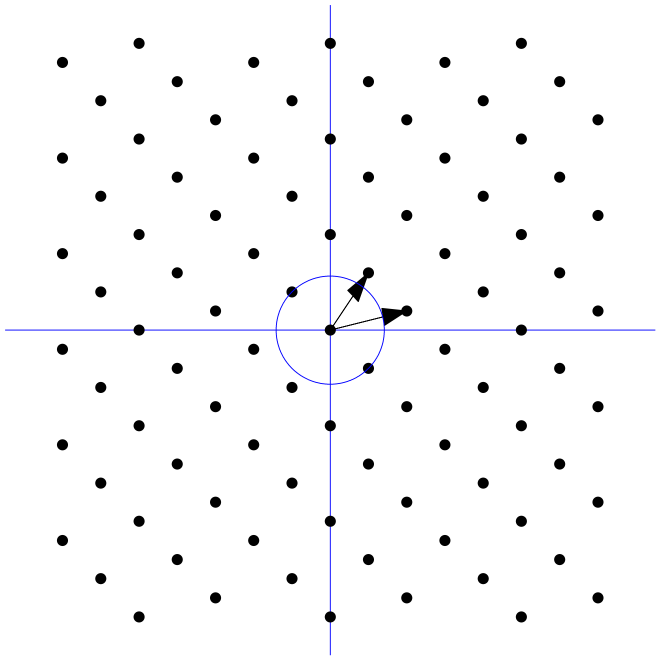


Figure 1.2: The shortest vector in the lattice

is a lattice vector, then so is $-\mathbf{v}$. Computing a shortest vector in a two-dimensional lattice is not difficult, but in lattices of higher dimension the general consensus is that no algorithm which efficiently solves SVP exists.

Now, if we can't expect to find the shortest vector in reasonable time, it is natural to ask if we can find a vector which may not be the shortest, but which isn't too much longer than the shortest. It turns out that the answer to this question depends on what one means by "not too much longer". It is known that finding a lattice vector that is up to a factor k longer than the shortest is essentially as hard as finding the shortest vector for any constant k. On the other hand there is an efficient algorithm that is known to always give a vector that is at most $2^{n/2}$ times as long as the shortest vector, and that in practice often produces even better results. It is still unknown precisely where the border lies between what can be computed efficiently and what cannot.

Lattice problems have cryptographic applications. It is known that the cryptosystem NTRU would be insecure if short vectors could be found in a certain type of lattices. Since the NTRU lattices are of very high dimension, it is believed to be infeasible to find such short vectors. However, the NTRU lattices have a certain structure that could potentially make them weaker. In this thesis we study this structure and show that SVP isn't easier in this type of lattices. Our approach is to show that given an arbitrary lattice Λ_1, it is possible to compute a lattice Λ_2 which has the special structure and lies very close to Λ_1. This is shown in Figure 1.3. Now we can conclude that if SVP were easy in Λ_2, then it would be easy in

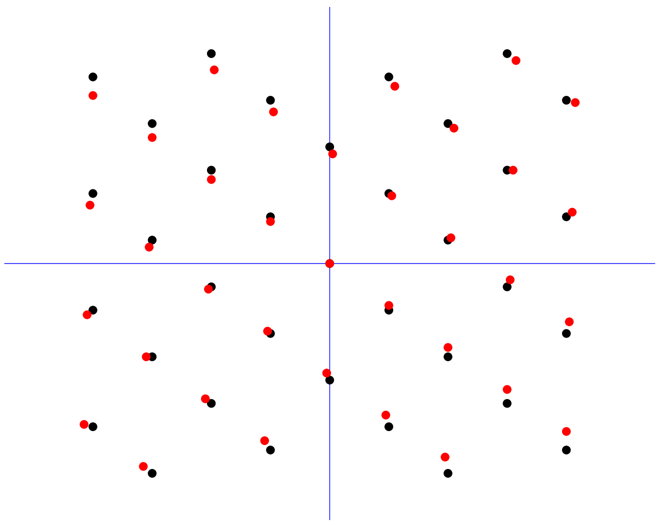


Figure 1.3: Approximating a lattice with another lattice

Λ_1 as well, since a solution to SVP in Λ_2 can be translated into a solution in Λ_1 as well. Therefore the special structure of Λ_2 does not help when solving SVP.

1.11 Thesis Overview

The thesis is organized as follows. In Chapter 2 (also [68]) we describe a protocol for electronic cash that is designed specifically to be as efficient as possible. In Chapter 3 (also [70]) the protocol for hierchical signatures with proofs of security can be found. Chapter 3 is joint work with Douglas Wikström. In Chapter 4 the weakness of certain EMV payment cards is analyzed. In Chapter 5 (also [69]) we give the full details of the lattice result.

Chapter 2

An Efficient Protocol for Electronic Cash

2.1 Introduction

Today, a large and growing part of payments are made by electronic means, but there is still much room for improvement. Credit cards may be suitable for large amounts, but for small amounts the cost of using a credit card is too high. Also, a credit card is closely tied to the person's identity, and we may want a system where the merchant learns less or even nothing about the identity of the customer. These are the issues to be addressed by electronic cash. The purpose of electronic cash is to give an alternate option for payment which provides some anonymity to the customer, and possibly avoids the need for contacting the bank for every transaction.

We present a system for electronic cash that is based on symmetric primitives. The advantage of this is that we get a system where the coins are small and where the cryptographic functions performed by the customer requires little processing power.

Previous Work

A system for electronic cash is usually designed for a situation where the coin is withdrawn from the bank by the customer, transferred from the customer to the merchant (as means of payment) and later deposited by the merchant at the bank. Sometimes it is desirable to have a system where the coins can be transferred between customers in several steps before they are deposited at the bank.

The different security issues that need to be addressed include forgeability (creating a coin that the merchant accepts without performing the withdrawal protocol with the bank first), double spending (making a copy of the coin and spending it twice), and revealing of identity (ability for the bank and the merchant to see who withdrew a coin used in a purchase).

Previously published systems for electronic cash include the system presented by Chaum, Fiat and Naor [19], which addresses the issues of anonymity and detection of double spenders. Later systems [11, 12, 31, 55, 63] have made improvements. In [31], it is shown how to make the communication between the merchant and the customer more efficient. In [55], a proposal for how to make the coins divisible is introduced. In [12], the possibility of later revoking the anonymity of the coins is added, which may be desirable for legal reasons. Sander and Ta-Shma [63] present a system where the bank does not have a secret key. Our system is based on the ideas of that system. The similarities and differences between our system and the system introduced by Sander and Ta-Shma is discussed in more detail in section 2.2.

All these systems use asymmetric encryption or non-interactive zero-knowledge proofs ([8]) to achieve security. The use of asymmetric techniques such as RSA appears to imply that a coin must include numbers of size at least 768 bits, and probably at least 1024 bits. Since a coin often consists of several such numbers, storing the coins on a smart card where the storage is limited is problematic. With non-interactive zero-knowledge proofs, especially when based on general methods, the coins get even larger.

We could of course use a handheld computer to store the coins. This would however make the system less convenient to use and thus less likely to be accepted. The cost of such devices would reduce the likelyhood that the system is widely accepted. Therefore we want the emphasize the small coin size of the presented system.

Privacy, Coin Sizes and Efficiency

As mentioned before, in the systems presented so far, anonymity is a major concern. They ensure that neither the merchant nor the bank can identify the owner of a coin. This is achieved either by the use of blind signatures or non-interactive zero-knowledge proofs of knowledge. Both methods generate coins that are "large" (meaning having a size such that a number n of the same size is hard to factor, or that it is hard to find the discrete logarithm modulo n).

We propose a system that is significantly more efficient than the previously published systems, and still provides full anonymity towards the merchant. The system only uses symmetric encryption and computation of hash functions, thus eliminating the need for costly operations like exponentiation. The emphasis is on efficiency – the same functionality can easily be accomplished using public key cryptography, but this would yield much larger coins.

To get some perspective we can compare the size of the coins in our system with previously proposed systems. The system proposed by Ferguson in [31] needs five RSA-sized number per coin, giving each coin a size of $5 \cdot 1024 = 5120$ bits, or 640 bytes. The system presented in [72] has reduced this to three numbers of 1024 bits, giving each coin a size of 384 bytes. The systems of [19] and [55] need even more space for the coins. In comparison, in our system, only 25 bytes need to be stored at the customer.

Our Solution

In this chapter we describe a simple and efficient system for electronic cash with provable security properties. The system relies on symmetric encryption technologies rather than asymmetric. This enhances performance, and the system still gives a fair level of anonymity. Another advantage is that the bank does not have a master secret that can be stolen and used for counterfeiting money. We present two variants of the system, one that is completely offline and one that is online. In the online variant, central databases are used to store information that otherwise would be stored on the customer's smart card.

Previously published systems focus on anonymity, both from the merchant and from the bank. They make it impossible for both the merchant and the bank to trace a payment. This untraceability may be desirable in certain cases, and it is certainly in the customer's interest. It is however far from certain that a bank would want (or accept) that kind of anonymity. There are also law enforcement aspects – if money is used in blackmailing, we would like to be able to trace the money.

The system presented here is semi-anonymous. The merchant cannot trace a payment. In fact, it cannot see whether or not two payments have been made by the same customer. The bank, however, can see the identity of the customer when the merchant deposits his money, just as it would with, for example, a credit card. By sacrificing anonymity against the bank, we win a lot in coin size.

The technique used to avoid the need for a signature on every coin is the use of hash trees as proposed by Merkle [49] and used by Sander and Ta-Shma [63]. The idea is that the bank keeps the coins it has issued in hash trees, where each father is the hash value of the concatenated values its sons. Creating hash trees is a one-way process – given the leaves it is easy to compute the root value, but given only the root value it is infeasible to construct a matching tree (except for the trivial tree consisting of only the root).

The roots of the trees are made public, and any coin which has a path leading to a published root is regarded as valid. Since the paths are not secret, it is possible to publish these paths, removing the need for the paths to be stored on the smart card. Also, the correctness of these paths is defined by the fact that they lead to a certified root. This means that the databases containing the paths do not need to be in any way secure or authenticated. The merchant can himself verify the outcome by comparing the root with the certified roots he has received from the bank.

How does this system differ from an ordinary credit card system? When paying with a credit card, the customer have to reveal his identity to the merchant, whereas in the proposed system the customer remains anonymous to the merchant. In the online version the purchase must be verified against a database, but unlike a credit card system, this database does not need to be authenticated. Also, the communication does not have to be secure. A large merchant may even keep a copy of the database locally, to speed up the processing. The offline version has the obvious advantage that there is no need for an online connection to an external database.

An implementation of the scheme is underway [75].

2.2 Overview of the System

The system uses hash trees to keep track of which coins should be considered valid. In the online variant, the hash trees are distributed to local databases via not necessarily secure lines. In the offline variant, only the bank keeps a copy of the hash trees and the customers keep track of the path of every coin. Only the roots need to be transmitted on an authenticated line.

All participants have agreed on a parameter k, which needs to be even. H is a hash function. $R_a(x)$ is a pseudorandom function with the key a. In practice we can think of H as SHA-1. As pseudorandom function we can use a symmetric cryptosystem (like AES) with a as the key.

The Participants

There are three participants – the customer, the merchant and the bank. The customer is assumed to have a smart card or similar device with some, but limited, storage and computational capabilities. The customer's identity is id. The customer's smart card is assumed to have a secret, which we call a. The card also contains a secret key used to identify the customer with some signature scheme. The bank has the corresponding public key.

The bank does not have a master secret key. The bank only needs to be able to, in an authenticated way, publish roots of the trees of hash values. The bank also has a symmetric cryptosystem, whose encryption we call E, and whose decryption is called D.

The merchant has no secret. The merchant has to get the root of the hash tree the bank has published in a secure way, and it has to have access to a database which contains the hash tree, although this access does not have to be authenticated.

The Protocol

The protocol consists of three steps – withdrawal, payment and deposition. In the withdrawal phase, the customer receives coins from the bank and the bank charges the customer's bank account. In the payment phase, the customer transfers coins to the merchant. In the deposition phase, the merchant deposits the money with the bank, and the bank credits the amount to the merchant's account.

$$\begin{array}{lcl}
\text{Customer} & & \text{Bank} \\
 & & s \in_R T \\
 & & c := E(s, id) \\
 & \xleftarrow{c} & \\
z_i := H(R_a(c, i)) & & \\
 & \xrightarrow{\langle z_i \rangle} & \\
 & & \text{Allocates a position in the hash tree and sends} \\
 & & \text{the path } p \in \{0,1\}^n\text{. Later } H(c, z_1, z_2, \ldots, z_k) \\
 & & \text{is inserted into the hash tree.} \\
 & \xleftarrow{p} & \\
p & &
\end{array}$$

Figure 2.1: The withdrawal protocol

Withdrawal

When withdrawing money from his account, a secure channel is set up between the customer and the bank. The customer first identifies himself to the bank in some way (possibly using his private key). A withdrawal of a coin then proceeds as follows

1. The bank generates a serial number, s, and sends $c = E(s, id)$ to the customer.

2. The customer comutes $z_i = H(R_a(c, i))$ for $i = 1, \ldots, k$, and sends these signed to the bank.

3. The banks allocates a position in the next hash tree, and sends the path to this position (as a $\{0, 1\}$-string) to the customer. Later, when the bank actually builds the tree, the customer's coin is inserted as a leaf in the allocated position.

This is described in Figure 2.1. We call the pair (c, a) a *coin*.

After the protocol has finished, the customer has the coin represented by c, and knows k values that hash to values in the hash tree. Also, this information is not known to anyone but the customer.

Note that the use of a pseudo-random function is only to save space. The same security would be achieved if the customer in step 2 generated k values, say $b_1, b_2, \ldots, b_k$ and sent $H(b_i)$ to the bank. In such a setup, the customer would need to store the values k $b_1, b_2, \ldots, b_k$, whereas he in the current setup does not need to store any extra information apart from c.

An alternative to having c as an encryption of (s, id) would be to have c as a unique random number, and to have the bank store the pair (c, id) in a private database. The current setup avoids the need for an extra database with sensitive information. Such a setup would, on the other hand, remove the need for a secret key for the bank.

Payment

In the payment phase, the customer sends the public part of the coin c together with the challenges $\langle z_i \rangle$ and the path through the hash tree to the merchant. In the online variant, the merchant turns to the database to get the necessary path. In the offline variant, the user has the path and sends it to the merchant. All the merchant needs to check is that one end of the path contains the hash value of c and the challenges and that the other end contains the root it has received on a secure channel. Together with the path, the merchant receives the z_i's.

The procedure is as follows. From the start the customer has a coin (c, a).

1. The customer sends c, $\langle z_i \rangle$ and the path to the merchant.
2. The merchant verifies c and $\langle z_i \rangle$ with the database. The merchant picks numbers $b_1, \ldots, b_k$ at random from $\{0, 1\}$ under the constraint that $\sum b_i = k/2$.
3. The customer computes $y_i = R_a(c, i)$ for each i such that $b_i = 1$ and sends the result to the merchant.
4. The merchant accepts if $H(y_i) = z_i$ for every y_i. If this is the case, the merchant stores these values.

Deposition

The merchant deposits the coin at the bank by sending the y_i's together with c. The bank decrypts c and marks the coin as used in its database. Should the coin already be marked as used, it checks which y_i's were used in the previous transaction. If the same y_i's were used in that transaction, it is assumed that the merchant is trying to deposit the same coin twice, and the merchant's account is only credited once. If the y_i's are different, the customer has tried to double spend his coin. It is then easy for the bank to retrieve the identity of the double spender, since this information is stored in the coin.

Maintaining the Hash Tree

The idea of the hash tree is the idea presented by [63]. After a certain period of time t, say one minute, the bank creates a hash tree where each leaf consists of a coin c issued during that period and the corresponding challenges z_i. The bank forms the tree and publishes the root. It then gives the path to the customers that have withdrawn coins during that last period.

Customer		Merchant		Database
(c, a, p)				
	$\xrightarrow{c,p,\langle z_i \rangle}$	c, p		
			$\xrightarrow{p}$	
				Retrieves the hash values $\langle p_j \rangle$ along the path p
			$\xleftarrow{\langle p_j \rangle}$	
		Verifies that the path p is valid and leads to a certified root. Picks $\mathbf{b} = (b_1, b_2, \ldots, b_k) \in_R \{0,1\}^k$ so that $\sum b_i = k/2$		
	$\xleftarrow{\mathbf{b}}$			
$y_i := R_a(c, i)$ for $b_i = 1$				
	$\xrightarrow{\langle y_i \rangle}$			
		$H(y_i) \overset{?}{=} z_i$		

Figure 2.2: The payment protocol in the online variant

The next time period t, the bank creates a new tree in the same way, and publish the root. It then joins these two trees and publish the root of that joined tree. The merchants now only need to hold that new root, and they need to know how to derive that the previous roots are secure from the new root. For this they only need to know the path, and since this is no information that needs to kept authenticated, it could reside in the (not authenticated) databases containing hash trees for coins.

After joining the first two trees, the bank starts building the next tree. After time t, this tree is ready, and the bank publishes the root. At this moment, there are two live roots for the merchants to trust.

Thus, after time $2t$ we join two trees into one. In the same way, after time $4t$ we join two $2t$-trees into one, after $8t$ we join two $4t$-trees and so on. We see that after time $2^k t$ there is exactly one root (since we have just joined the two subtrees into one tree). After time $\left(2^k - 1\right) t$, we have k different trees, with k roots. However, even if we set $t = 10$ seconds and want to have a system that will work for 100 years, there will never be more than 30 roots at any time.

Comparison with Sander and Ta-Shma

The system presented by Sander and Ta-Shma [63] was the first system for electronic cash to use hash trees for identification of coins. The maintenance of the hash trees in our system is the same as proposed in their system.

The system of Sander and Ta-Shma is fully anonymous, and the identity of the customer is reveiled only if he spends a coin twice. This is achieved by the use of non-interactive zero-knowledge proofs. We, on the other hand, use pseudorandom functions to hide the identity of the customer, and the anonymity is only anonymous towards the merchant and not towards the bank.

In the system of Sander and Ta-Shma, there is no need for the bank to store the hash trees after publishing the roots, since the customer stores the path himself. This avoids having external interaction during the payment. On the other hand every coin becomes larger, since the complete path needs to be stored.

2.3 Analysis of the System

For the analysis we want to prove two things. First we need to prove that for honest participants, the system gives a fair result. Then we need to prove that the system is secure. The first part follows directly from the construction of the system and that $D(E(s, id)) = (s, id)$ and that H and R_a are deterministic.

Proof of Security

The setting for the system is that the customer trusts the bank for anonymity and for providing fair coins. Both the merchant and the customer trust the roots of the hash trees. The bank does not need to trust anyone. The customer does not need to trust the merchant and the merchant does not need to trust the customer. No one needs to trust the hash trees provided by the third party database providers.

By neglible probability we mean a probability lower than $1/p(n)$ for any polynomial p and large enough security parameter n. The relevant security parameters are the number of challenges, the length of the customer's private key, the length of the bank's private key and the output length of the hash function.

We start by giving definitions of the tools we need. Since these definitions are standard definitions, we only give informal descriptions.

Definition 2.3.1 (Collision resistant hash function).
A keyed hash function $H_a : A \to \{0,1\}^n$ with key a is called collision resistant *if a polynomial time (in $|a|$) probabilistic Turing machine has negligible probability of finding $x, y \in A$, $x \neq y$ such that $H(x) = H(y)$.*

Definition 2.3.2 (Pseudorandom functions).
A function $R_a : A \to B$, where a is a parameter, is called pseudorandom *if it indistinguishable from a uniformly distributed functions $f : A \to B$ to a Turing machine that runs in time polynomial in $|a|$.*

Definition 2.3.3 (Semantic security).
We call a symmetric cryptosystem (E, D) semantically secure *if no polynomial time probabilistic Turing machine* T *can distinguish between* $E(m_0)$ *and* $E(m_1)$ *with non-negligible probability, even if* T *is allowed to pick* m_0 *and* m_1 *itself.*

Definition 2.3.4 (Detect identity). *We say that two merchants can* detect identity *if they can play the following game and be successful with a probability non-negligibly higher than* $1/2$.

Flip a coin and set the bit t *to the result. If* $t = 0$*, let* $d_1 = (c_1, a)$ *and* $d_2 = (c_2, a)$ *be two coins withdrawn by* the same *customer using the withdrawal protocol. If* $t = 1$*, let* $d_1 = (c_1, a_1)$ *and* $d_2 = (c_2, a_2)$ *be two coins withdrawn by* different *customers using the withdrawal protocol. Let* d_1 *and* d_2 *be used by the respective owners in the payment protocol with two (possibly equal) merchants. Given the information in the payment protocol and the function* R_a *as a black box, the merchants (who are allowed to cooperate) pick* $t' \in \{0, 1\}$*. They are successful if* $t = t'$*.*

It is possible to give a more general definition of anonymity. We can say that for any values of n, $m \leq n$ and positive $q_1, \ldots, q_m$ such that $\sum q_i = n$ it is infeasible to decide whether n purchases were made by m distinct customers, where the i'th customer made q_i purchases. We omit the formal definition.

We prove the following on the security of the system, assuming that the hash function H is collision resistant, the bank's symmetric encryption system E is semantically secure and the family of functions R_a is pseudorandom.

1. The customer cannot forge coins except for with negligible probability.
2. The customer cannot double spend a coin without detection, except for with negligible probability of success.
3. No two merchants can detect identity (as described in definition 2.3.4).

We go through these theorems one by one.

Theorem 2.3.5 (Nonforgeability (1)). *It is infeasible for a customer who has withdrawn* l *coins to perform the payment protocol* $l + 1$ *times without being detected as a double spender.*

Proof. Suppose we have a user U^* which with non-negligible probability can provide hash trees so that he can answer the merchant's questions in the payment protocol $l + 1$ times after withdrawing l coins, where the probability is taken over the input to U^* (the information U^* gets during the interactions, from the hash trees and by monitoring other customers interact with merchants). We want to use U^* to either find a collision of H or compute $R_a(x)$ without the knowledge of a.

We start by creating hash trees $\mathbf{T}_i$ with roots $\mathbf{R}$ while interacting with U^* and other simulated customers. During this interaction we allow U^* to withdraw l coins. After creating the interaction we give $\mathbf{T}_i$ and $\mathbf{R}$ to U^*. U^* is allowed to exchange $\mathbf{T}_i$ with his own $\mathbf{T}'_i$ after the interaction.

We now interact with U^* (playing the part of the merchant) $l+1$ times using the payment protocol. According to the assumption, U^* answers correctly each of these times with non-negligible probability. Should U^* not answer correctly, we restart. Since we have assumed that U^* will not be caught as a double spender, we know that U^* cannot provide us with the same c twice. Also, if U^* gives us a value c which has never been issued by the bank, U^* has had to modify a hashtree $\mathbf{T}$ into $\mathbf{T}'$ so that $H(c, \langle z_i \rangle)$ becomes a leaf of $\mathbf{T}'$. Since the root is unchanged, we will find a collision of H by comparing the two trees. We omit the details.

This leaves us with the possibility that U^* has given us a c and a p which already exists in the tree, but which belongs to another customer. He then has to answer the queries correctly without the knowledge of a. We show that the ability of answering queries without knowing a (that is, in effect stealing the coin) implies the ability to compute $R_a(x)$ for any x without knowledge of a.

If U^* can use the coin c without prior knowledge of a he must solve the following problem. Given c and z_i, find y_i such that $H(y_i) = z_i$. If we have such an oracle we can run it on random input c and $z_i = H(y_i)$, where either $y_i = R_a(c, i)$. The oracle then outputs y_i' that satisfies the requirements. If $y_i \neq y_i'$, we have found a collision in the hash function. If $y_i = y_i'$ U^* has succeeded to compute a value of R_a which it has not seen before. □

Theorem 2.3.6 (Double spending detection (2)). *If a customer executes the payment protocol twice with the same coin (c, a), the bank detects that the customer has double-spent the coin with probability $1 - \binom{k}{k/2}^{-1}$.*

Proof. Assume a customer is performing the payment protocol twice with the same coin c. Then, at deposition, the bank detects that the coin has been double spent, and also finds the identity of the customer guilty of double spending.

The chance of "getting away" with double spending is the same as the chance of getting exactly the same challenges twice. The chance of this happening is $\binom{k}{k/2}^{-1}$ if the challenges are chosen at random. □

We can note that we can remove the probability $\binom{k}{k/2}^{-1}$ by not choosing the challenges at random. This is discussed in 2.4.

Theorem 2.3.7 (Anonymity (3)). *No two merchants can detect identity (as described in definition 2.3.4).*

Proof. Assume two merchants can play the game and succeed with probability non-negligibly higher than 1/2. We use this as an oracle O, which interacts using the payment protocol with two customers spending two coins (c^1, a^1) and (c^2, a^2). O outputs 1 with probability p if the two payments were made by the same customer and outputs 1 with probability q if the payments were made by different customers, where $p - q$ is positive and non-negligible. More precisely, the input to O is $c^1, \langle y_i^1 \rangle, c^2, \langle y_i^2 \rangle$ where $y_i^j = R_{a^j}(c^j, i)$. The output bit is 1 with probability p if there exists an a such that $y_i^1 = R_a(c^1, i)$ and $y_i^2 = R_a(c^2, i)$ for every challenge y_i. Otherwise the output bit is 1 with probability q.

We want to use the oracle to violate the pseudorandomness of R_a. Assume we have a family of functions $\mathfrak{F}$ given as a black box and we want to decide whether it is pseudorandom or random. We want to create an algorithm A (which uses O as an oracle) that with non-negligible probability outputs 1 if $\mathfrak{F}$ is pseudorandom and 0 otherwise.

Consider the following four distributions of coins, where a and b are two different keys for the pseudorandom function, and B is a random function.

1. Both coins are withdrawn using R_a.
2. Both coins are withdrawn using B.
3. The first coin is withdrawn using R_a and the second coin using B.
4. The first coin is withdrawn using R_a and the second coin using R_b.

Let p_i be the probability the oracle O outputs 1 on indata from distribution i. From the assumption that O can detect identity it follows that $p_1 - p_4$ is non-negligible. This implies that one of $p_1 - p_2$, $p_2 - p_3$ or $p_3 - p_4$ is non-negligible. We show how to implement A in each of these cases

- If $p_1 \neq p_2$ we create both coins using $\mathfrak{F}$ and use this as input to O.
- If $p_2 \neq p_3$ we create one coin using $\mathfrak{F}$ and the other using a random function and execute O with this as input.
- If $p_3 \neq p_4$ we create one coin using R_a and the other using $\mathfrak{F}$ and use this as input to O.

Since for at least one of these inequalities the difference is non-negligible, our algorithm is able to distinguish the family of pseudorandom functions from random functions, which was assumed to be infeasible. This is a contradiction which proves the theorem. □

2.4 Some Practical Details

Choosing the Challenges and Stealing of Coins

As we have seen, the way the challenges are chosen is very important for the detection of double spendings. The simplest would be to choose the challenges at random. The probability of the challenges being identical is then very low. One could, however, imagine legal problems if the bank tries to prove in court that two merchants have collaborated to perform fraud against the bank by both depositing the same coin. The merchants can of course argue that they happened to accidently pick the same random challenge (maybe due to bad (pseudo-) random number generators).

With this setup, a merchant A could steal coins from merchant B. If A steals the hard-disk containing the only copy of B's coins, the theft would never be discovered by the bank. Several practical precautions are of course possible, but we will see that with a slight modification of the protocol, stealing coins is impossible.

One way of choosing challenges would be to have the challenge consist of a hash value of, say, the merchant's identity, the time, the amount and possibly other parameters. The problem with this setup is that since k cannot be very large, it would be possible for the customer to find a collision. He can then choose to spend his coin where he has found the collision, and hence get exactly the same challenge. The bank would not be able to decide whether it is the merchants or the customer that is guilty of the fraud.

Since the only property of randomness we used is that we have low probability of the same challenge, we could instead allocate a certain interval of challenges to each merchant (or rather to each terminal accepting payments). We would then have a counter in every terminal to ensure that no set of challenges is used twice. If a terminal runs out of queries, it would notify the bank which would assign a new interval. If we assume that we want each interval to consist of 10000 queries, and we want 10^9 such intervals, $k = 50$ would be enough.

A feature of this setup is that a merchant A cannot steal a coin from another merchant B, since this stolen coin has a challenge that does not correspond to a valid challenge of A, and the bank detects that the coin has been stolen. The same is true if a merchant eavesdrops a purchase or a deposition of a coin.

The Coin Databases

Another important part of the payment system is the coin databases. Since every coin issued is to be stored in these databases, they need quite large storage capabilities. We can note that the information in these databases is in no way sensitive. It does not have to be authenticated or secure, and with the cost of storage medium being low (and only getting lower), this is not such big a problem as it may seem. On the other hand, smart card memory is expensive, so it is a good thing to save smart card memory in favor of hard disk space.

For each coin only the hash value needs to be stored in the database. To further reduce storage need we can design the system so that the coins are issued in a distributed manner. For every entity with a certain number of customers (e.g., every city), the coins would be combined into a hash tree. These local hash trees would be joined into national hash trees, which in turn would be joined on an international level.

With this setup most databases can be designed to hold only information on coins issued in the same city or country, and during the last period of time, say a month. There would probably be a few complete databases that would be contacted in case the coin could not be found in the local database, and these would need to contain every coin that has been issued. If we assume that a person spends 100 coins per day, one million customers would spend approximately 3×10^9 coins per month. Since every coin needs 160 bit in the database (assuming we use SHA-1 as hash function) the total storage need would be 60 GB, which is far from infeasible, especially as the security requirements of the database are low. The central databases would need to hold more information.

Values of Coins

The system as described here assumes all coins have the same value. It is easy to adopt the system to handle coins of different values. When the bank issues the coin, it can simply add the value of the coin to the leaf in the database. The value does not have to be added explicitly – it suffices to have the hash value stored in the database include the value of the coin.

The Size of a Coin

We now calculate the size of a coin (c, a). The secret a is common to all coins, so the size of it does not have to be counted. The variable c consists of the encryption of a serial number s and the identity id. If we want the system to scale for world-wide use, we need to reserve, say, 64 bits for the identity. We can then use 64 bits for the serial number, and still be able to fit the coin in 128 bits. We only need the serial number to be unique per customer. Using a symmetrical cryptosystem with a key-length of 128 bits, we get an output of 128 bits.

In the online variant, we need to store the path in the hash tree. If we assume that the system globally has 10^{10} users and every user uses 100 coins per day for 100 years, no path needs more than 60 bits. This gives a total coin size of less than 200 bits, or 25 bytes, which is a major improvement compared to the asymmetric cash systems described in the introduction.

In the offline variant, the path needs to be stored by the customer. This is more than can be stored on a smart card, which means we need some means of secondary storage.

Assuming a tranfer rate of 9600 bps between the card and the terminal, the most costly phase of the payment phase, transfer of the responses to the challenges, requires less than half a second, assuming that 25 challenges have to be answered.

2.5 Conclusions

We have presented a system for electronic cash that is both practical and provably secure. The privacy properties are such that banks are likely to accept the system, and the system still protects the customer's identity against the merchants. The coins are small enough to fit on smart cards.

Chapter 3

Hierarchical Group Signatures

3.1 Introduction

Consider the notion of group signatures introduced by Chaum and van Heyst [21]. A group member can compute a signature that to an outsider reveals nothing about the signer's identity except that he is a member of the group. On the other hand the group manager can always reveal the identity of the signer.

An application for group signatures is anonymous credit cards. The cardholder wishes to preserve his privacy when he pays a merchant for goods, i.e., he is interested in unlinkability of payments. The bank must obviously be able to extract the identity of a cardholder from a payment or at least an identifier for an account, to be able to debit the account. To avoid fraud, the bank, the merchant, and the cardholder all require that a cardholder cannot pay for goods without holding a valid card. To solve the problem using group signatures we let the bank be the group manager and the cardholders be signers. A cardholder signs a transaction and hands it to the merchant. The merchant then hands the signed transaction to the bank, which debits the cardholder and credits the merchant. Since signatures are unlinkable, the merchant learns nothing about the cardholder's identity. The bank on the other hand can always extract the cardholder's identity from a valid signature and debit the correct account.

The above scenario is somewhat simplified since normally there are many banks that issue cards of the same brand and which are processed through the same payment network. The payment network normally works as an administrator and routes transactions to several independent banks. Thus, the merchant hands a payment to the payment network which hands the payment to the issuing bank. We could apply group signatures here as well by making the payment network act as the group manager. The network would then send the extracted identity to the issuing bank. Another option is to set up several independent group signatures schemes, one for each issuer. In the first approach, the payment network learns the identity of the customer, and in the second approach the merchant learns which bank issued the customer's card. A better solution would reveal nothing except what is absolutely necessary to each party. The merchant needs to be convinced that the credit card is valid, the payment network must be able to route the payment to the correct card issuer and the issuer must be able to determine the identity of the cardholder.

A solution that comes to mind is to use ordinary group signatures with the modification that the customer encrypts his identity with his bank's public key. Then we have the problem of showing to the merchant that this encryption contains valid information. However, the customer cannot reveal the public key of the bank to the merchant, making such a proof far from trivial.

In this chapter we introduce and investigate the notion of *hierarchical group signatures*. These can be employed to solve the above problem. When using a hierarchical group signature scheme there is not one single group manager. Instead there are several group managers organized in a tree, i.e., each group manager either manages a group of signers or a group of group managers. In the original notion the group manager can always identify the signer of a message, but nobody else can distinguish between signatures by different signers. The corresponding property for hierarchical group signatures is more complicated. If a manager directly manages a group of signers, it can identify all the signers that it manages, but the signatures of all other signers are indistinguishable to it. This corresponds directly to the original notion. If a manager manages a group of managers, it cannot identify the signer, but it can identify the manager directly below it which (perhaps indirectly) manages the signer. Thus, a manager that does not manage signers directly get only partial information on the identity of the signer.

When we use hierarchical group signatures to construct anonymous credit cards for the more realistic setting we let the payment network be the root manager that manages a set of group managers, i.e., the issuing banks, and we let the cardholders be signers. The credit card application also demonstrates what kind of responsibility model is likely to be used with a hierarchical group signature scheme. With a valid signature on a transaction, the merchant has a valid demand on the payment network. If the payment network has a signature that can be shown to belong to a certain bank, the network has a valid demand on that bank. Thus, it is in the network's interest to open the signatures it receives from merchants, and it is in the issuing banks' interest to open the signatures they receive from the network.

Previous Work

The concept of group signatures was first introduced by Chaum and van Heyst [21] in 1991. This and the group signature schemes that followed [22, 14] all had the property that the complexity of the scheme grows with the number of participants. In [17] Camenisch and Stadler presented a system where the key does not grow with the number of participants. This system, however, relies on a non-standard number-theoretic assumption. The assumption was actually found to be incorrect and modified in [4]. An efficient system whose security rests on the strong RSA assumption and the Diffie-Hellman decision assumption was presented by Camenisch and Michels in 1998 [15]. This system was improved in [3].

In [7] Bellare et al. presented a scheme for group signatures based on general methods. Our scheme based on general assumptions can be seen as a generalization of their scheme.

In [4] the concepts of *multi-group signatures* and *subgroup signatures* are described, and in [42] a system for hierarchical multi-groups is given. It may be worthwhile to consider the differences between these concepts and hierarchical signatures introduced here. Multi-group signature schemes allow a signer who is a member of two groups to produce a signature that shows membership of either both groups or just one of them. In hierarchical multi-groups a signer who is a member of a supergroup with subgroups can produce a signature that reveals membership either of the supergroup or of a subgroup of his choice. However, the opening procedure is not hierarchical, e.g., there are no group managers for the subgroups.

Subgroup signatures make it possible for an arbitrary number i of signers to produce a joint signature which can be verified to stem from i distinct group members. None of these extensions contain the hierarchical property.

The connection between group signatures and anonymous payment systems is quite natural and has been studied before. In [45] a system for electronic cash based on group signatures is given by Lysyanskaya and Ramzan.

Group signatures, and especially hierarchical group signatures, should not be confused with zero-knowledge sets as described in [50]. Zero-knowledge sets enables a prover to commit to a set S. Given x he can then prove $x \in S$ or $x \notin S$ (whichever is true) without disclosing anything else about S. For zero-knowledge sets the prover has the necessary information to produce a proof of membership for any element in the set. With group signatures on the other hand the set of members may be public, and the signer proves that it belongs to this set.

Notation

We write $[a, b]$ to denote the set $\{x \in \mathbb{Z} \mid a \leq x \leq b\}$. We say that an element is chosen "randomly" instead of the more cumbersome "independently and uniformly at random". If T is a tree we denote by $\mathcal{L}(T)$ its set of leaves. We let ϕ denote Euler's ϕ function. By $r \in_R S$ we mean that r is chosen randomly in S. Throughout the chapter, κ denotes the security parameter. A function $f : \mathbb{N} \to [0, 1]$ is said to be negligible if for each $c > 0$ there exists a $\kappa_0 \in \mathbb{N}$ such that $f(\kappa) < \kappa^{-c}$ for $\kappa_0 < \kappa \in \mathbb{N}$. We say that a function $f : \mathbb{N} \to [0, 1]$ is non-negligible whenever it is not negligible. When we say that a number is k-bit, we implicitly mean that it has a leading one (i.e., that it is in the interval $[2^{k-1}, 2^k - 1]$).

We sometimes do not explicitly state how a group given as input to an algorithm is described, e.g., we write $\mathsf{CHPg}(G_q)$ to denote that the algorithm CHPg is given a description of a group G_q of prime order q as input. Whenever we do that, G_q is assumed to be the unique subgroup of order q of $\mathbb{Z}_p^*$ for a prime $p = 2q + 1$, so the obvious description of G_q is p. In Section 3.5 we consider the issue of existence of such primes. We use QR_N to denote the subgroup of squares in $\mathbb{Z}_N^*$, i.e., the quadratic residues. We write $\emptyset$ to denote both the empty set and the empty string.

We say that a distribution ensemble $\mathcal{D} = \{D_\kappa\}$ is efficiently sampleable if there exists a polynomial time Turing machine $T_{\mathcal{D}}$ that on input 1^κ outputs a random sample distributed according to D_κ.

All adversaries in this chapter are modeled as polynomial time Turing machines with *non-uniform* auxiliary advice string. We denote the set of such adversaries by PPT^*.

A public-key cryptosystem is said to be *CCA2-secure* if it is infeasible for an attacker to determine which one of two messages of his choice that a given cryptotext is the encryption of, even if the attacker has access to a decryption oracle both before the choice is made and after the cryptotext is received [60]. The following formalizes this property.

Let $\mathcal{CS} = (\mathsf{Kg}, E, D)$ be a public key cryptosystem. Consider the following experiment.

Experiment 1 (CCA2, $\mathbf{Exp}^{\mathsf{cca2}-b}_{\mathcal{CS},A}(\kappa)$).

$$\begin{aligned}
&(pk, sk) \leftarrow \mathsf{Kg}(1^\kappa)\\
&(m_0, m_1, \mathrm{state}) \leftarrow A^{D_{sk}(\cdot)}(\mathsf{choose}, pk)\\
&c \leftarrow \mathsf{Enc}_{sk}(m_b)\\
&d \leftarrow A^{D_{sk}(\cdot)}(\mathsf{guess}, \mathrm{state}, pk)
\end{aligned}$$

The experiment returns 0 if the encryption oracle was queried on c, and d otherwise. The advantage of an adversary is defined as

$$\mathbf{Adv}^{\mathsf{cca2}}_{\mathcal{CS},A}(\kappa) = |\Pr[\mathbf{Exp}^{\mathsf{cca2}-0}_{\mathcal{CS},A}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{cca2}-1}_{\mathcal{CS},A}(\kappa) = 1]| \enspace .$$

Definition 3.1.1 (CCA2-secure). *The cryptosystem $\mathcal{CS}$ is said to be* CCA2-secure *if $\mathbf{Adv}^{\mathsf{cca2}}_{\mathcal{CS},A}(\kappa)$ is negligible for any $A \in \mathrm{PPT}^*$.*

A signature scheme is said to be *CMA-secure* if it infeasible for an attacker to output a message-signature pair even if given a signing oracle [36]. Formally CMA-security is defined using the following experiment, where $\mathcal{SS} = (\mathsf{Kg}, \mathsf{Sig}, \mathsf{Vf})$ is a signature scheme

Experiment 2 (CMA, $\mathbf{Exp}^{\mathsf{cma}}_{\mathcal{SS},A}(\kappa)$).

$$\begin{aligned}
&(pk, sk) \leftarrow \mathsf{Kg}(1^\kappa)\\
&(m, s) \leftarrow A^{\mathsf{Sig}_{sk}(\cdot)}(\mathsf{guess}, pk)
\end{aligned}$$

If $\mathsf{Vf}_{pk}(m, s) = 1$ and A's oracle was never queried on m return 1, else return 0. The advantage of an adversary A is defined as $\mathbf{Adv}^{\mathsf{cma}}_{\mathcal{SS},A}(\kappa) = \Pr[\mathbf{Exp}^{\mathsf{cma}}_{\mathcal{SS},A}(\kappa) = 1]$.

Definition 3.1.2 (CMA-secure). *The signature scheme $\mathcal{SS}$ is said to be* CMA-secure *if $\mathbf{Adv}^{\mathsf{cma}}_{\mathcal{SS},A}(\kappa)$ is negligible for any $A \in \mathrm{PPT}^*$.*

Two ensembles $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are *statistically close* if $\sum_\alpha |\Pr[X_n = \alpha] - \Pr[Y_n = \alpha]|$ is negligible.

Definition 3.1.3 (Trapdoor Permutation Family).
A trapdoor permutation family *is a tuple of probabilistic polynomial time Turing machines $\mathcal{F} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Invert})$ such that:*

1. $\mathsf{Gen}(1^\kappa)$ *outputs a pair (f, f^{-1}) such that f is a permutation of* $\{0,1\}^\kappa$.
2. $\mathsf{Eval}(1^\kappa, f, x)$ *is a deterministic algorithm which on input f, where $(f, f^{-1}) \in \mathsf{Gen}(1^\kappa)$, and $x \in \{0,1\}^\kappa$ outputs $y = f(x)$.*
3. $\mathsf{Invert}(1^\kappa, f^{-1}, y)$ *is a deterministic algorithm which on input f^{-1}, where $(f, f^{-1}) \in \mathsf{Gen}(1^\kappa)$, and $y \in \{0,1\}^\kappa$ outputs some $x = f^{-1}(y)$.*
4. *For all κ, $(f, f^{-1}) \in \mathsf{Gen}(1^\kappa)$, and $x \in \{0,1\}^\kappa$ we have $f^{-1}f(x) = x$.*
5. *For all adversaries $A \in \mathrm{PPT}^*$, the following is negligible*

$$\Pr[(f, f^{-1}) \leftarrow \mathsf{Gen}(1^\kappa), \quad x \leftarrow \{0,1\}^\kappa, \quad A(f, f(x)) = f^{-1}(y)] \ .$$

Definition 3.1.4 (Hard-Core Bit). *Let $\mathcal{B} = \{B_\kappa : \{0,1\}^\kappa \to \{0,1\}\}$ be a collection of functions such that there exists a polynomial time Turing machine that outputs $B_\kappa(x)$ on input $(1^\kappa, x)$, where $x \in \{0,1\}^\kappa$. Let $\mathcal{F} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Invert})$ be a trapdoor permutation family. $\mathcal{B}$ is a* hard-core bit *for $\mathcal{F}$ if the following is negligible for all adversaries $A \in \mathrm{PPT}^*$*

$$\left|\Pr[(f, f^{-1}) \leftarrow \mathsf{Gen}(1^\kappa), \quad x \leftarrow \{0,1\}^\kappa, \quad A(f, f(x)) = B(x)] - \frac{1}{2}\right| \ .$$

Outline of Chapter

In Section 3.2 we formalize the notion of hierarchical group signatures and give definitions of security. We also briefly discuss why it is not trivial to transform a non-hierarchical group signature scheme into a hierarchical scheme. In Section 3.3 we introduce the concept of cross-indistinguishability, which we use in both the general construction and the explicit construction. Our construction under general assumptions is presented in Section 3.4 and in Section 3.5 we give the explicit construction. The zero-knowledge proofs used in Section 3.5 can be found in Section 3.6. Finally in Sections 3.7 and 3.8 we discuss possible modifications and extensions of the current scheme.

Contributions

We introduce and formalize the notion of *hierarchical group signatures*. We give a construction that is provably secure under the existence of a trapdoor permutation family. As part of our investigations we introduce a new property of cryptosystems, which we call cross-indistinguishability. This property may be of independent interest.

Then we consider how a practical hierarchical group signature scheme can be constructed under specific complexity assumptions. We show that by a careful selection of primitives one can construct a relatively practical hierarchical group signature scheme that is provably secure under the DDH assumption and the strong RSA assumption in the random oracle model. For reasonable security parameters a few hundred exponentiations are required to produce a signature.

As part of the construction we show how to prove efficiently in zero-knowledge that a committed value is a signature of an encrypted message. This technique may be useful for other applications.

3.2 Hierarchical Group Signatures

In this section we discuss the notion of hierarchical group signatures. We begin by describing the parties of a hierarchical group signature system. Then we proceed by giving formal definitions.

Parties

There are two types of parties: signers denoted S_α for α in some index set $\mathcal{I}$, and group managers denoted M_α for indices α described below. The parties form a tree T, where the signers are leaves and the group managers are inner nodes. The indices of the group managers are formed as follows. If a group manager manages a set of signers $\{S_\alpha \mid \alpha \in \beta \subset \mathcal{I}\}$ we denote it by M_β. This corresponds to M_β having S_α for $\alpha \in \beta$ as children. If a group manager M_γ manages a set of group managers $\{M_{\beta_1}, \ldots, M_{\beta_l}\}$ we denote it by M_γ where $\gamma = \{\beta_1, \ldots, \beta_l\}$. This corresponds to M_γ having M_{β_i} for $i = 1, \ldots, l$ as children. Let M_ρ denote the root group manager. We assume that the root group manager is at depth 0 and that all leaves in the tree are at the same depth. When there is no risk of confusion we write α instead of M_α or S_α.

Note that standard group signatures correspond to having a single group manager $M_{[1,l]}$ that manages all signers $S_1, \ldots, S_l$.

Definition of Security

The first thorough investigation of the fundamentals of group signatures was carried out by Bellare et al. [7]. They give a definition of a group signature scheme, but more importantly they argue that two properties of group signatures, full anonymity and full traceability, imply any reasonable security requirements one can expect from a group signature scheme.

We follow their definitional approach closely and develop definitions that are proper generalizations of the original.

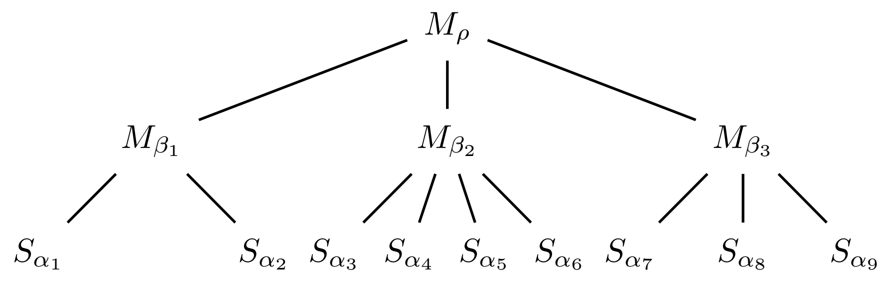


Figure 3.1: A tree of group managers and signers, where $\rho = \{\beta_1, \ldots, \beta_3\}$, $\beta_1 = \{\alpha_1, \alpha_2\}$, $\beta_2 = \{\alpha_3, \alpha_4, \alpha_5, \alpha_6\}$, and $\beta_3 = \{\alpha_7, \alpha_8, \alpha_9\}$.

The idea is that the managers and signers are organized in a tree T, and we wish to associate with each node (or leaf) α a public value $hpk(\alpha)$ and a private value $hsk(\alpha)$.

Definition 3.2.1 (Hierarchical Group Signature). *A* hierarchical group signature scheme $\mathcal{HGS} = (\mathsf{HKg}, \mathsf{HSig}, \mathsf{HVf}, \mathsf{HOpen})$ *consists of four polynomial-time algorithms*

1. *The randomized* key generation algorithm HKg *takes as input* $(1^\kappa, T)$*, where* T *is a tree of size polynomially bounded in* κ *with all leaves at the same depth, and outputs a pair of maps* $hpk, hsk : T \to \{0,1\}^*$.

2. *The randomized* signature algorithm HSig *takes as input a message* m*, a tree* T*, a public map* hpk*, and a secret signing key* $hsk(\alpha)$*, and returns a signature of* m.

3. *The deterministic* signature verification algorithm HVf *takes as input a tree* T*, a public map* hpk*, a message* m *and a candidate signature* σ *of* m *and returns either* 1 *or* 0.

4. *The deterministic* opening algorithm HOpen *takes as input a tree* T*, a public map* hpk*, a secret opening key* $hsk(\beta)$*, a message* m*, and a candidate signature* σ*. It outputs an index* $\alpha \in \beta$ *or* $\perp$.

In the definition of HSig above, it is assumed that it is possible to verify in polynomial time given the public tree gpk, a secret key $gsk(\alpha)$ and an index α', if $\alpha = \alpha'$. This is the case for the construction in [7]. We assume that hpk and hsk map any input that is not a node of T to $\perp$ and that $\mathsf{HOpen}(\cdot, \cdot, \perp, \cdot, \cdot) = \perp$.

We need to define what we mean when we say that a hierarchical group signature scheme is secure. Here we generalize the definitions of [7]. We begin with anonymity. Assume a message has been signed by either $\alpha^{(0)}$ or $\alpha^{(1)}$. Then any group manager on the path leading from $\alpha^{(0)}$ or $\alpha^{(1)}$ to the first group manager who is an ancestor of both $\alpha^{(0)}$ and $\alpha^{(1)}$, can determine who the signer is. In Figure 3.2 those group managers are marked with black. In the definition of anonymity we capture the property that an adversary that is not allowed to corrupt any of these group managers cannot determine whether $\alpha^{(0)}$ or $\alpha^{(1)}$ signed the message, even if the adversary itself is given the private keys of all signers and is allowed to select $\alpha^{(0)}$, $\alpha^{(1)}$ and the message himself.

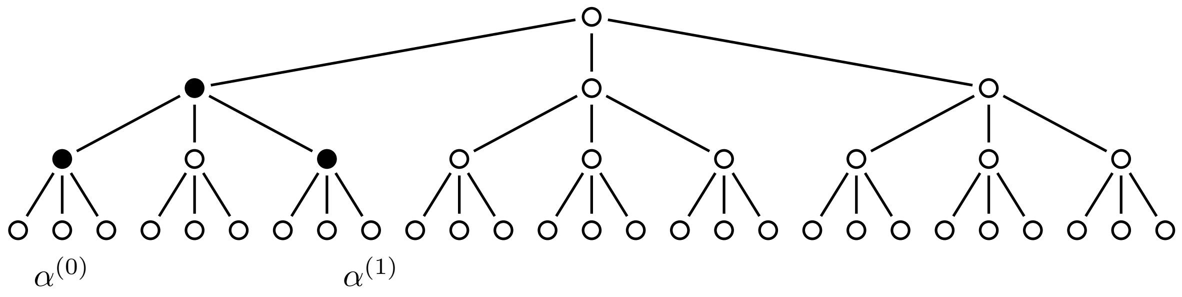


Figure 3.2: Nodes in black represent group managers able to distinguish between $\alpha^{(0)}$ and $\alpha^{(1)}$.

We define Experiment 3 to formalize these ideas. Throughout the experiment the adversary has access to an $\mathsf{HOpen}(T, hpk, hsk(\cdot), \cdot, \cdot)$ oracle. At the start of the experiment the adversary is given the public keys of all parties and the private keys of all signers. Then it can adaptively ask for the private keys of the group managers. At some point it outputs the indices $\alpha^{(0)}$ and $\alpha^{(1)}$ of two leaves and a message m. The $\mathsf{HSig}(\cdot, T, hpk, hsk(\cdot))$ oracle computes the signature of m using the private key $hsk(\alpha^{(b)})$ and hands it to the adversary. The adversary finally outputs a guess d of the value of b. If the scheme is anonymous the probability that $b = d$ should be negligibly close to $1/2$ when b is a randomly chosen bit. The labels $\mathsf{corrupt}$, choose and guess below distinguish between the phases of the experiment.

Experiment 3 (Hierarchical Anonymity, $\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa, T)$).

```
$(hpk, hsk) \leftarrow \mathsf{HKg}(1^\kappa, T)$;   $s_{\mathrm{state}} \leftarrow (hpk, hsk(\mathcal{L}(T)))$;   $\mathcal{C} \leftarrow \emptyset$;   $\alpha \leftarrow \emptyset$;
*While* $(\alpha \neq \perp)$ *do*
    $(s_{\mathrm{state}}, \alpha) \leftarrow A^{\mathsf{HOpen}(T,hpk,hsk(\cdot),\cdot,\cdot)}(\mathsf{corrupt}, s_{\mathrm{state}}, hsk(\alpha))$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{\alpha\}$
*Done*
$(s_{\mathrm{state}}, \alpha^{(0)}, \alpha^{(1)}, m) \leftarrow A^{\mathsf{HOpen}(T,hpk,hsk(\cdot),\cdot,\cdot)}(\mathsf{choose}, s_{\mathrm{state}})$
$\sigma \leftarrow \mathsf{HSig}(T, hpk, hsk(\alpha^{(b)}), m)$
$d \leftarrow A^{\mathsf{HOpen}(T,hpk,hsk(\cdot),\cdot,\cdot)}(\mathsf{guess}, s_{\mathrm{state}}, \sigma)$
```

Let B be the set of nodes on paths from $\alpha^{(0)}$ and $\alpha^{(1)}$ up to their first common ancestor α_t excluding $\alpha^{(0)}$ and $\alpha^{(1)}$ but including α_t, i.e., the set of nodes $\alpha_l^{(0)}$, $\alpha_l^{(1)}$, $l = t, \ldots, \delta - 1$, such that

$$\alpha^{(0)} \in \alpha^{(0)}_{\delta-1} \in \alpha^{(0)}_{\delta-2} \in \ldots \in \alpha^{(0)}_{t+1} \in \alpha_t \ni \alpha^{(1)}_{t+1} \ni \ldots \ni \alpha^{(1)}_{\delta-2} \ni \alpha^{(1)}_{\delta-1} \ni \alpha^{(1)} .$$

If $B \cap \mathcal{C} \neq \emptyset$ or if A asked its $\mathsf{HOpen}(T, hpk, hsk(\cdot), \cdot, \cdot)$ *oracle a question* $(\alpha_l^{(0)}, m, \sigma)$ *or* $(\alpha_l^{(1)}, m, \sigma)$ *return 0. Otherwise return d.*

Consider the above experiment with a depth one tree T with root ρ. In that case we may assume that $hsk(\rho)$ is never handed to the adversary, since the adversary fails in that case anyway. Similarly the $\mathsf{HOpen}(T, hpk, hsk(\cdot), \cdot, \cdot)$ oracle reduces to the Open oracle in [7]. Thus, our experiment reduces to the experiment for full anonymity given in [7] where the adversary gets the secret keys of all signers, but only the public key of the group manager.

Next we consider how the notion of full traceability can be defined in our setting. Full traceability as defined in [7] is similar to security against chosen message attacks (CMA-security) as defined by Goldwasser, Micali and Rivest [36] for signatures. The only essential difference is that the group manager must always be able to open a signature and identify the signer. In our setting this amounts to the following. Given a signature deemed valid by the HVf algorithm, the root should always be able to identify the child directly below of which the signer is a descendent. The child should have the same ability for the subtree of which it is a root and so on until the child itself is a signer.

Again we define an experiment consisting of two phases. To start with the adversary is given the secret keys of all group managers. Then the adversary adaptively chooses a set of signers to corrupt. Then in a second phase the adversary guesses a message and signature pair. If the guess amounts to a signature deemed valid by HVf and the signer cannot be traced, or if the signature is traced to a non-corrupted signer, the adversary has succeeded and the experiment outputs 1. Otherwise it outputs 0. Thus, the distribution of the experiment should be negligibly close to 0 for all adversaries if the scheme is secure.

Experiment 4 (Hierarchical Traceability, $\mathbf{Exp}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa, T)$).

```
$(hpk, hsk) \leftarrow \mathsf{HKg}(1^\kappa, T)$;   $s_{\mathrm{state}} \leftarrow (hpk, hsk(T \backslash \mathcal{L}(T))$;   $\mathcal{C} \leftarrow \emptyset$;   $\alpha \leftarrow \emptyset$;
*While*   $(\alpha \neq \bot)$   *do*
    $(s_{\mathrm{state}}, \alpha) \leftarrow A^{\mathsf{HSig}(\cdot, T, hpk, hsk(\cdot))}(\mathsf{choose}, s_{\mathrm{state}}, hsk(\alpha))$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{\alpha\}$
*Done*
$(m, \sigma) \leftarrow A^{\mathsf{HSig}(\cdot, T, hpk, hsk(\cdot))}(\mathsf{guess}, s_{\mathrm{state}})$
```

If $\mathsf{HVf}(T, hpk, m, \sigma) = 0$ *return 0. Define* $\alpha_0 = \rho$ *and* $\alpha_l = \mathsf{HOpen}(T, hpk, hsk(\alpha_{l-1}), m, \sigma)$ *for* $l = 1, \ldots, \delta$*. If* $\alpha_l = \bot$ *for some* $0 < l \leq \delta$ *return 1. If* $\alpha_\delta \notin \mathcal{C}$ *and the* $\mathsf{HSig}(\cdot, T, hpk, hsk(\cdot))$ *oracle did not get a question* (m, α_δ) *return 1. Otherwise return 0.*

Consider the experiment above with a depth one tree. This corresponds to giving the adversary the secret key of the group manager, and letting it adaptively choose additional signing keys. Furthermore, the $\mathsf{HSig}(\cdot, T, hpk, hsk(\cdot))$ oracle reduces to the GSig oracle in [7]. This is precisely the setting of [7].

The advantage of the adversary is defined in the natural way by

$$\mathbf{Adv}^{\mathsf{anon}}_{\mathcal{HGS},A}(\kappa,T) = |\Pr[\mathbf{Exp}^{\mathsf{anon}-0}_{\mathcal{HGS},A}(\kappa,T)=1] - \Pr[\mathbf{Exp}^{\mathsf{anon}-1}_{\mathcal{HGS},A}(\kappa,T)=1]|$$

and

$$\mathbf{Adv}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa,T) = \mathbf{Exp}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa,T) \ .$$

Definition 3.2.2 (Security of Hierarchical Group Signatures). *A hierarchical group signature scheme $\mathcal{HGS} = (\mathsf{HKg}, \mathsf{HSig}, \mathsf{HVf}, \mathsf{HOpen})$ is* secure *if for all trees T of polynomial size in κ with all leaves at the same depth, and all $A \in \mathrm{PPT}^*$, $\mathbf{Adv}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa,T) + \mathbf{Adv}^{\mathsf{anon}}_{\mathcal{HGS},A}(\kappa,T)$ is negligible.*

Remark 3.2.3. *The reason the adversary is not given access to both the* HOpen *and the* HSig *oracle in the experiments is that it is given sufficient private keys to simulate the missing oracle by itself.*

Remark 3.2.4. *An ordinary signature scheme $\mathcal{SS} = (\mathsf{Kg}, \mathsf{Sig}, \mathsf{Vf})$, with key generator* Kg*, signature algorithm* Sig*, and verification algorithm* Vf*, can be viewed as a hierarchical group signature scheme* $(\mathsf{Kg}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{HOpen})$ *of depth 0 by defining* $\mathsf{HOpen}(\sigma) = \perp$*. Then $\mathbf{Adv}^{\mathsf{anon}}_{\mathcal{HGS},A}(\kappa,T) = 0$ and Definition 4 reduces to the definition of security against chosen message attacks as defined by Goldwasser, Micali, and Rivest [36].*

Alternative Definitions

Above we define a hierarchical group signature scheme such that the group managers are organized in a tree where all leaves are at the same depth. Furthermore, a group manager can by looking at a signature decide whether the signer belongs to it or not without any interaction with other group managers. Several other variants are possible. Below we discuss some of these variants informally.

Trees with leaves on different depths could be considered. Any such tree can clearly be replaced by a tree with all leaves at the same depth by inserting dummy group managers in between signers and their immediate parents until all signers are at the same depth.

We could let group managers sign on behalf of its group. If this is needed a signer that correspond to the group manager is added. Depending on if the parent of the group manager should be able to distinguish between a signature of the group manger itself and its children or not, the signer is added to the group manager's parent or itself.

We could consider a forest of trees, i.e. there could be several roots. Such a scheme can be simulated in our definition by first joining the trees into a single tree by adding a root and then disposing of the private root key.

The group managers could be organized in a directed acyclic graph (DAG), e.g. two trees could share a common subtree. This would give alternative paths to some signers. There may be situations where this is advantageous, but the semantics of such a scheme are complex and involves many subtle issues, e.g. should all group managers (indirect and direct) of a signer get information on its identity, or should the signer decide on a path from a root and only reveal information to group managers along this path? Although we believe that the techniques we use for our construction would be useful also for this type of scheme we do not investigate such schemes further.

On Constructing Hierarchical Group Signatures

All known group signatures are based on the idea that the signer encrypts a secret of some sort using the group manager's public key, and then proves that the resulting cryptotext is on this special form. The security of the cryptosystem used implies anonymity, since no adversary can distinguish cryptotexts of two distinct messages if they are encrypted using the *same* public key.

Suppose we wish to generalize this approach to construct a hierarchical group signature scheme. In the hierarchical setting protecting the identity of the signer implies protecting the identity of the group managers along the path of to the signer. On the other hand these group managers (and nobody else) must be able to extract partial knowledge on the identity on the identity of the signer. Thus, it seems that hierarchical group signatures must somehow contain embedded cryptotexts. To ensure anonymity, signatures with embedded cryptotexts corresponding to distinct public keys must be indistinguishable, since otherwise the cryptotexts embedded in a signature would reveal information on the identity of the signer. This type of indistinguishability does not follow from the indistinguishability of a cryptosystem. We say that a cryptosystem that has this property is cross-indistinguishable. This property is investigated in detail in Section 3.3 below.

On the other hand, to ensure traceability, the signer must prove that a signature contains the identity of the signer encrypted with public keys corresponding to the path to the signer. In principle this is not a problem, since there is a non-interactive zero-knowledge proof system for any language in **NP**, but the details must be resolved. It is far from obvious how to construct a practical proof system.

3.3 Cross-Indistinguishability

It turns out that the cryptosystem we use must not only be indistinguishable (semantically secure), but it must also have an incomparable security property which we call *cross-indistinguishability*. We formalize cross-indistinguishability in Definition 3.3.2 below, but first we recall the definition of indistinguishability of a cryptosystem $\mathcal{CS} = (\mathsf{Kg}, E, D)$, with key generator Kg, encryption algorithm E, and decryption algorithm D, as defined by Goldwasser and Micali [34].

Experiment 5 (Indistinguishability, $\mathbf{Exp}_{\mathcal{CS},A}^{\mathsf{ind}-b}(\kappa)$ (cf. [34])).

$$\begin{array}{rcl} (pk, sk) & \leftarrow & \mathsf{Kg}(1^\kappa) \\ (m_0, m_1, s_{state}) & \leftarrow & A(pk) \\ d & \leftarrow & A(E_{pk}(m_b), s_{state}) \end{array}$$

We let $\mathbf{Adv}_{\mathcal{CS},A}^{\mathsf{ind}}(\kappa) = |\Pr[\mathbf{Exp}_{\mathcal{CS},A}^{\mathsf{ind}-0}(\kappa) = 1] - \Pr[\mathbf{Exp}_{\mathcal{CS},A}^{\mathsf{ind}-1}(\kappa) = 1]|$.

Definition 3.3.1. *Let $\mathcal{CS}$ be a cryptosystem. We say that $\mathcal{CS}$ is* indistinguishable *if for all $A \in \mathrm{PPT}^*$, $\mathbf{Adv}_{\mathcal{CS},A}^{\mathsf{ind}}(\kappa)$ is negligible.*

Informally, cross-indistinguishability boils down to the property that the adversary cannot distinguish cryptotexts encrypted with distinct public keys.

Experiment 6 (Cross-Indistinguishability, $\mathbf{Exp}_{\mathcal{CS},A}^{\mathsf{cross}-b}(\kappa)$).

$$\begin{array}{rcl} (pk_0, sk_0) & \leftarrow & \mathsf{Kg}(1^\kappa), \quad (pk_1, sk_1) \leftarrow \mathsf{Kg}(1^\kappa) \\ (m, s_{state}) & \leftarrow & A(pk_0, pk_1) \\ d & \leftarrow & A(E_{pk_b}(m), s_{state}) \end{array}$$

Note that compared to the standard definition of indistinguishability of cryptotexts, the roles played by public keys and messages are reversed. One could consider a variant definition which captures both types of indistinguishabilities, but we think it is more natural to think of cross-indistinguishability as an additional property. We let $\mathbf{Adv}_{\mathcal{CS},A}^{\mathsf{cross}}(\kappa) = |\Pr[\mathbf{Exp}_{\mathcal{CS},A}^{\mathsf{cross}-0}(\kappa) = 1] - \Pr[\mathbf{Exp}_{\mathcal{CS},A}^{\mathsf{cross}-1}(\kappa) = 1]|$.

Definition 3.3.2. *Let $\mathcal{CS}$ be a cryptosystem. We say that $\mathcal{CS}$ is* cross-indistinguishable *if for all $A \in \mathrm{PPT}^*$, $\mathbf{Adv}_{\mathcal{CS},A}^{\mathsf{cross}}(\kappa)$ is negligible.*

The property of cross-indistinguishability is clearly useless if the cryptosystem is not indistinguishable, since it allows the encryption function to be the identity map. Thus, cross-indistinguishability does not imply indistinguishability. To see that the other implication does not hold, note that if $\mathcal{CS}$ is an indistinguishable cryptosystem, then so is the cryptosystem where the encryption and decryption functions $c = E_{pk}(m)$ and $D_{sk}(c) = m$ are replaced by $(c, c') = E'_{pk}(m) = (E_{pk}(m), pk)$ and $D'_{sk}(c, c') = D_{sk}(c) = m$ respectively.

The lemma below characterizes the set of cryptosystems which are both indistinguishable and cross-indistinguishable. Denote by $\mathbf{Exp}_{\mathcal{CS},A}^{\mathsf{ind}-\mathcal{D}_{\mathrm{ind}}}(\kappa)$ Experiment 5, but with the input $E_{pk_b}(m)$ replaced by an element distributed according to a distribution D_κ, where $\mathcal{D}_{\mathrm{ind}} = \{D_\kappa\}$, and correspondingly for $\mathbf{Exp}_{\mathcal{CS},A}^{\mathsf{cross}-\mathcal{D}_{\mathrm{ind}}}(\kappa)$. We use $T_{\mathcal{D}}$ to denote the Turing machine that on input 1^κ returns a sample distributed according to D_κ.

Experiment 7 ($\mathcal{D}_{\mathrm{ind}}$-Indistinguishability, $\mathbf{Exp}_{\mathcal{CS},A}^{\mathsf{ind}-\mathcal{D}_{\mathrm{ind}}}(\kappa)$).

$$\begin{array}{rcl} (pk, sk) & \leftarrow & \mathsf{Kg}(1^\kappa) \\ (m_0, m_1, s_{state}) & \leftarrow & A(pk) \\ d & \leftarrow & (T_{\mathcal{D}}(1^\kappa), s_{state}) \end{array}$$

Experiment 8 ($\mathcal{D}_{\text{ind}}$-Cross-Indistinguishability, $\mathbf{Exp}_{\mathcal{CS},A}^{\mathsf{cross}-\mathcal{D}_{\text{ind}}}(\kappa)$).

$$\begin{aligned}(pk_0, sk_0) &\leftarrow \mathsf{Kg}(1^\kappa), \quad (pk_1, sk_1) \leftarrow \mathsf{Kg}(1^\kappa)\\ (m, s_{state}) &\leftarrow A(pk_0, pk_1)\\ d &\leftarrow A(T_{\mathcal{D}}(1^\kappa), s_{state})\end{aligned}$$

Lemma 3.3.3. *Let $\mathcal{CS}$ be a cryptosystem which is both indistinguishable and cross-indistinguishable. Then there exists an efficiently sampleable distribution $\mathcal{D}_{\text{ind}}$ such that for all $A \in \mathrm{PPT}^*$*

$$|\Pr[\mathbf{Exp}_{\mathcal{CS},A}^{\mathsf{ind}-b}(\kappa) = 1] - \Pr[\mathbf{Exp}_{\mathcal{CS},A}^{\mathsf{ind}-\mathcal{D}_{\text{ind}}(\kappa)}(\kappa) = 1]| \ ,$$

is negligible for $b \in \{0,1\}$. The reverse implication holds as well.

Proof. Suppose that a distribution $\mathcal{D}_{\text{ind}}$ as in the lemma exists. The indistinguishability of $\mathcal{CS}$ then follows by a trivial hybrid argument. Suppose that $\mathcal{CS}$ is not cross-indistinguishable. Then there exists an adversary $A \in \mathrm{PPT}^*$ such that

$$|\Pr[\mathbf{Exp}_{\mathcal{CS},A}^{\mathsf{cross}-0}(\kappa) = 1] - \Pr[\mathbf{Exp}_{\mathcal{CS},A}^{\mathsf{cross}-1}(\kappa) = 1]|$$

is non-negligible which by a trivial hybrid argument implies that

$$|\Pr[\mathbf{Exp}_{\mathcal{CS},A}^{\mathsf{cross}-b}(\kappa) = 1] - \Pr[\mathbf{Exp}_{\mathcal{CS},A}^{\mathsf{cross}-\mathcal{D}_{\text{ind}}}(\kappa) = 1]|$$

is non-negligible for a fixed $b \in \{0,1\}$, which we without loss assume to be 0. Let A' be the adversary in Experiment 5 defined as follows. On input pk it sets $pk_0 = pk$ generates $(pk_1, sk_1) = \mathsf{Kg}(1^\kappa)$ and hands (pk_0, pk_1) to A, which returns (m, s_{state}). Then A' returns (m, m, s_{state}). When handed (c, s_{state}) from the experiment, where c is either is $E_{pk_0}(m)$ or a sample from D_κ, it returns the output of $A(c, s_{\text{state}})$. By construction $\mathbf{Exp}_{\mathcal{CS},A'}^{\mathsf{ind}-0}(\kappa)$ is identically distributed to $\mathbf{Exp}_{\mathcal{CS},A}^{\mathsf{cross}-0}(\kappa)$, and $\mathbf{Exp}_{\mathcal{CS},A'}^{\mathsf{ind}-\mathcal{D}_{\text{ind}}}(\kappa)$ is identically distributed to $\mathbf{Exp}_{\mathcal{CS},A}^{\mathsf{cross}-\mathcal{D}_{\text{ind}}}(\kappa)$. This is a contradiction, since it implies that

$$|\Pr[\mathbf{Exp}_{\mathcal{CS},A'}^{\mathsf{ind}-0}(\kappa) = 1] - \Pr[\mathbf{Exp}_{\mathcal{CS},A'}^{\mathsf{ind}-\mathcal{D}_{\text{ind}}}(\kappa) = 1]|$$

is non-negligible.

Suppose next that $\mathcal{CS}$ is indistinguishable and cross-indistinguishable. We define our prospective distribution $\mathcal{D}_{\text{ind}}$ as follows. To generate a sample from $\mathcal{D}_{\text{ind}}$, generate a key pair $(pk', sk') = \mathsf{Kg}(1^\kappa)$ and output an encryption $E_{pk'}(m')$, where $m' = 0^\kappa$. This implies that $\mathcal{D}_{\text{ind}}$ is efficiently sampleable. Assume that

$$|\Pr[\mathbf{Exp}_{\mathcal{CS},A}^{\mathsf{ind}-b}(\kappa) = 1] - \Pr[\mathbf{Exp}_{\mathcal{CS},A}^{\mathsf{ind}-\mathcal{D}_{\text{ind}}}(\kappa) = 1]|$$

is non-negligible for $b = 0$ (then it is also non-negligible for $b = 1$, since $\mathcal{CS}$ is indistinguishable). Let A'_0 be the adversary in Experiment 6 that does the following. On input (pk_0, pk_1) it hands pk_0 to A which returns (m_0, m_1). Then A'_0 returns

m_0, and is given $E_{pk_b}(m_0)$ for a randomly chosen $b \in \{0,1\}$ by the experiment. It hands $E_{pk_b}(m_0)$ to A and returns the output of A. A'_1 is identical to A'_0 except that it hands m' to the experiment instead of m_0. From the construction follows that $\mathbf{Exp}^{\mathsf{ind}-0}_{\mathcal{CS},A}(\kappa)$ and $\mathbf{Exp}^{\mathsf{ind}-\mathcal{D}_{\mathrm{ind}}}_{\mathcal{CS},A}(\kappa)$ are identically distributed to $\mathbf{Exp}^{\mathsf{cross}-0}_{\mathcal{CS},A'_0}(\kappa)$ and $\mathbf{Exp}^{\mathsf{cross}-1}_{\mathcal{CS},A'_1}(\kappa)$ respectively. Thus

$$|\Pr[\mathbf{Exp}^{\mathsf{cross}-0}_{\mathcal{CS},A'_0}(\kappa)=1]-\Pr[\mathbf{Exp}^{\mathsf{cross}-1}_{\mathcal{CS},A'_1}(\kappa)=1]|$$

is non-negligible. From the cross-indistinguishability of $\mathcal{CS}$ we have that

$$|\Pr[\mathbf{Exp}^{\mathsf{cross}-0}_{\mathcal{CS},A'_b}(\kappa)=1]-\Pr[\mathbf{Exp}^{\mathsf{cross}-1}_{\mathcal{CS},A'_b}(\kappa)=1]|$$

is negligible for $b \in \{0,1\}$. A hybrid argument implies that

$$|\Pr[\mathbf{Exp}^{\mathsf{cross}-b}_{\mathcal{CS},A'_0}(\kappa)=1]-\Pr[\mathbf{Exp}^{\mathsf{cross}-b}_{\mathcal{CS},A'_1}(\kappa)=1]|$$

is non-negligible for some $b \in \{0,1\}$. Without loss we assume $b=0$. Denote by A'' the adversary in Experiment 5 defined as follows. Given input pk it hands (pk) to A. When A returns (m_0,m_1), it outputs (m_0,m'), and receives either $E_{pk}(m_0)$ or $E_{pk}(m')$, which it forwards to A. Finally, it returns the output of A. Since, $\mathbf{Exp}^{\mathsf{ind}-0}_{\mathcal{CS},A''}(\kappa)$ is identically distributed to $\mathbf{Exp}^{\mathsf{cross}-0}_{\mathcal{CS},A'_0}(\kappa)$ and $\mathbf{Exp}^{\mathsf{ind}-1}_{\mathcal{CS},A''}(\kappa)$ is identically distributed to $\mathbf{Exp}^{\mathsf{cross}-0}_{\mathcal{CS},A'_1}(\kappa)$, this contradicts the indistinguishability of $\mathcal{CS}$. □

Note that $\mathcal{D}_{\mathrm{ind}}$ depends on $\mathcal{CS}$ but is independent of all stochastic variables in the experiment. Below we show that the probabilistic cryptosystem of Goldwasser and Micali [34] is cross-indistinguishable.

Remark 3.3.4. *Several standard probabilistic cryptosystems can be made cross-indistinguishable by minor modifications. E.g. it is not hard to see that the ElGamal [28] cryptosystem is cross-indistinguishable if the group in which it is employed is fixed for each value of the security parameter.*

3.4 A Construction under General Assumptions

In this section we show how hierarchical group signatures can be constructed under general assumptions. Our focus is on feasibility and conceptual simplicity. We prove the following theorem.

Theorem 3.4.1. *If there exists a family of trapdoor permutations, then there exists a secure hierarchical group signature scheme.*

To prove the theorem we construct a hierarchical group signature scheme by augmenting the group signature scheme of [7] with additional cryptotexts and a non-interactive zero-knowledge proof.

Assumptions and Primitives Used

Before we give our construction we review some constructions and results on which our construction is based.

Group Signature Scheme

The first building block we need is a group signature scheme secure under the assumption that trapdoor permutations exists. As shown by Bellare et al. such a scheme exists.

Theorem 3.4.2 (cf. [7]). *If there exists a family of trapdoor permutations, then there exists a secure group signature scheme $\mathcal{GS} = (\mathsf{GKg}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open})$.*

Public Key Cryptosystem

The probabilistic cryptosystem of Goldwasser and Micali [34] is indistinguishable, but we are not aware of any proof of cross-indistinguishability. We prove that their construction is also cross-indistinguishable, but first we recall their construction.

Their construction is based on the existence of non-approximable trapdoor predicates. This concept can be captured in modern terminology as follows. A family of trapdoor permutations is a triple of polynomial time algorithms $\mathcal{F} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Invert})$. The instance generator $\mathsf{Gen}(1^\kappa)$ outputs a description f of a permutation of $\{0,1\}^\kappa$ and a trapdoor f^{-1}. The evaluation algorithm $\mathsf{Eval}(1^\kappa, f, x)$ evaluates the permutation on input $x \in \{0,1\}^\kappa$. The the corresponding inversion algorithm $\mathsf{Invert}(1^\kappa, f^{-1}, y)$ evaluates the inverse permutation on input $y \in \{0,1\}^\kappa$. We abuse notation and write $f(x)$ and $f^{-1}(y)$ for the evaluation of the permutation and inverse permutation as described above. The last requirement on the family of trapdoor permutations is that it must be infeasible for any $A \in \mathrm{PPT}^*$ given f and $y = f(x)$, where $x \in \{0,1\}^\kappa$, to compute $x = f^{-1}(y)$. A hard-core bit for $\mathcal{F}$ is a family of functions $\mathcal{B} = \{B_\kappa : \{0,1\}^\kappa \rightarrow \{0,1\}\}$ such that it is infeasible to compute $B_k(x)$, given only f and $f(x)$ for a random $x \in \{0,1\}^\kappa$. Goldreich and Levin [33] show how to construct a family of trapdoor permutations $\mathcal{F}$ with a hard-core bit $\mathcal{B}$ from any family of trapdoor permutations.

The cryptosystem $\mathcal{GM} = (\mathsf{GMKg}, E, D)$ of Goldwasser and Micali [34] using $\mathcal{F}$ and $\mathcal{B}$ can be defined as follows (using modern terminology). The key generator $\mathsf{GMKg}(1^\kappa)$ simply outputs $(pk, sk) = (f, f^{-1}) = \mathsf{Gen}(1^\kappa)$. To compute a cryptotext $E_{pk}(m)$ of a bit $m \in \{0,1\}$, choose $r \in \{0,1\}^\kappa$, and output $(f(r), \mathcal{B}(r) \oplus m)$. To decrypt a cryptotext (c, c'), compute $D_{sk}(c, c') = \mathcal{B}(f^{-1}(c)) \oplus c'$. To encrypt a bit-string the encryption function is invoked with a fresh randomly chosen r for each bit in the natural way. Goldwasser and Micali essentially show the following theorem.

Theorem 3.4.3. *If $\mathcal{F}$ is a trapdoor permutation family with hard-core bit $\mathcal{B}$, then $\mathcal{GM}$ is indistinguishable.*

We show that the $\mathcal{GM}$ cryptosystem is also cross-indistinguishable.

Lemma 3.4.4. *If $\mathcal{F}$ is a trapdoor permutation family with hard-core bit $\mathcal{B}$, then $\mathcal{GM}$ is cross-indistinguishable.*

Proof. Suppose that $\mathcal{GM}$ is not cross-indistinguishable. Let $U_{\kappa+1}$ be the uniform and independent distribution over $\{0,1\}^{\kappa+1}$. Then for some adversary $A \in \mathrm{PPT}^*$,

$$|\Pr[\mathbf{Exp}^{\mathsf{ind}-b}_{\mathcal{CS},A}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{ind}-U_{\kappa+1}}_{\mathcal{CS},A}(\kappa) = 1]|$$

is non-negligible for a fixed $b \in \{0,1\}$. Without loss we assume $b = 0$. Since $\mathcal{GM}$ is a bitwise cryptosystem, we may without loss assume that $m_0 = 0$ and $m_1 = 1$. Let $m \in \{0,1\}$ be randomly distributed, then a cryptotext $E_{pk}(m) = (f(r), \mathcal{B}(r) \oplus m)$ is distributed according to $U_{\kappa+1}$, since f is a permutation and $\mathcal{B}(r) \oplus m$ is uniformly and independently distributed. A trivial average argument implies that $|\Pr[\mathbf{Exp}^{\mathsf{ind}-b}_{\mathcal{CS},A}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{ind}-1}_{\mathcal{CS},A}(\kappa) = 1]|$ is non-negligible which is a contradiction. □

Non-Interactive Zero-Knowledge Proofs

Non-interactive zero-knowledge proofs (NIZK) were introduced by Blum, Feldman, and Micali [8]. Several works have since refined and extended the notion in various ways. Following [7] we employ the definition of adaptive zero-knowledge for NIZK introduced by Feige, Lapidot, and Shamir [30] and we use the notion of simulation soundness introduced by Sahai [62]. The notion of simulation soundness is strengthened by De Santis et al. [64]. In contrast to [7], the NIZK we use must be adaptive zero-knowledge for polynomially many statements, and not only for a single statement. The requirement on simulation soundness is in fact unchanged compared with [7], i.e. single statement simulation soundness suffices.

Definition 3.4.5 (NIPS). *A triple $(p(\kappa), P, V)$ is an* efficient adaptive non-interactive proof system *(NIPS) for a language $L \in \mathrm{NP}$ with witness relation R if $p(\kappa)$ is a polynomial and P and V are probabilistic polynomial time machines such that*

1. *Completeness. $(x, w) \in R$ and $\xi \in \{0,1\}^{p(\kappa)}$ implies $V(x, P(x, w, \xi), \xi) = 1$.*
2. *Soundness. For all computable functions A, $\Pr_{\xi \in \{0,1\}^{p(\kappa)}}[A(\xi) = (x, \pi) \wedge x \notin L \wedge V(x, \pi, \xi) = 1]$ is negligible in κ.*

We suppress p in our notation of a NIPS and simply write (P, V).

Loosely speaking a non-interactive zero-knowledge proof system is a NIPS, which is also zero-knowledge, but there are several flavors of zero-knowledge. We need a NIZK which is adaptive zero-knowledge (for a single statement) in the sense of Feige, Lapidot, and Shamir [30].

Experiment 9 (Adaptive Indistinguishability, $\mathbf{Exp}^{\mathsf{adind}-b}_{(P,V,S),A}(\kappa)$ (cf. [30])).

```
    $\xi \leftarrow \{0,1\}^{f(\kappa)}$                      *if* $b = 0$
    $(\xi, s_{\text{simstate}}) \leftarrow S(1^\kappa)$        *if* $b = 1$
$s_{\text{state}} = \xi, \quad t \leftarrow \emptyset$
*While* $(t \neq \bot)$ *do*
    $(s_{\text{state}}, t, w) \leftarrow \begin{cases} A(\mathsf{choose}, P(t, w, \xi)) & \text{if } (t,w) \in R \text{ and } b = 0 \\ A(\mathsf{choose}, S(t, \xi, s_{\text{simstate}})) & \text{if } (t,w) \in R \text{ and } b = 1 \\ A(\mathsf{choose}, \bot) & \text{otherwise} \end{cases}$
*Done*
$d \leftarrow A(s_{\text{state}})$
```

The advantage in the experiment is defined

$$\mathbf{Adv}^{\mathsf{adind}}_{(P,V,S),A}(\kappa) = |\Pr[\mathbf{Exp}^{\mathsf{adind}-0}_{(P,V,S),A}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{adind}-1}_{(P,V,S),A}(\kappa) = 1]|$$

and the notion of adaptive zero-knowledge is given below.

Definition 3.4.6 (Adaptive Zero-Knowledge (cf. [30])).
A NIPS (P, V) is adaptive zero-knowledge (NIZK) if there exists a polynomial time Turing machine S such that $\mathbf{Adv}^{\mathsf{adind}}_{(P,V,S),A}(\kappa)$ is negligible for all $A \in \mathrm{PPT}^*$.

In cryptographic proofs one often performs hypothetic experiments where the adversary is run with simulated NIZKs. If the experiment simulates NIZKs to the adversary, the adversary could potentially gain the power to compute valid proofs of false statements. For a simulation sound NIZK this is not possible.

Experiment 10 (Simulation Soundness, $\mathbf{Exp}^{\mathsf{sims}}_{(P,V,S),A}(\kappa)$ (cf. [64])).

$$\begin{aligned} &(\xi, s_{\text{simstate}}) \leftarrow S(1^\kappa) \\ &(t, \pi) = A^{S(\cdot, \xi, s_{\text{simstate}})}(\xi) \end{aligned}$$

Let Q be the set of proofs returned by the $S(\cdot, \xi, s_{\text{simstate}})$ oracle. Return 1 if $\pi \notin Q$, $t \notin L$, and $V(t, \pi, \xi) = 1$, and 0 otherwise.

Definition 3.4.7 (Simulation Soundness (cf. [62, 64])). *A NIZK (P, V) with polynomial time simulator S for a language L is unbounded simulation sound if $\mathbf{Adv}^{\mathsf{sims}}_{(P,V,S),A}(\kappa) = \mathbf{Exp}^{\mathsf{sims}}_{(P,V,S),A}(\kappa)$ is negligible for all $A \in \mathrm{PPT}^*$.*

De Santis et al. [64] extend the results in [30] and [62] and prove the following result.

Theorem 3.4.8. *If there exists a family of trapdoor permutations, then there exists a simulation sound NIZK for any language in* NP.

In the the rest of this chapter we abbreviate "efficient non-interactive adaptive zero-knowledge unbounded simulation sound proof" by NIZK.

Our Construction

We now describe our hierarchical group signature scheme $\mathcal{HGS} = (\mathsf{HKg}, \mathsf{HSig}, \mathsf{HVf}, \mathsf{HOpen})$. We let $\mathcal{F} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Invert})$ denote a family of trapdoor permutations with a hard-core bit $\mathcal{B}$, and assume that a Goldwasser-Micali cryptosystem $\mathcal{GM}$ has been constructed from this. We denote by $\mathcal{GS} = (\mathsf{GKg}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open})$ the group signature scheme of Bellare et al. also constructed from $\mathcal{F}$. However, we view this as a hierarchical group signature scheme of depth 1 and use the corresponding notation, i.e. the public and secret keys of the group manager M_ρ are denoted by $hpk(\rho)$ and $hsk(\rho)$ (not by gpk and $gmsk$ etc. as in [7]). Below we also use $\mathcal{F}$ to construct a NIZK for a language L_{HGS}.

First keys to the $\mathcal{GS}$ group signature scheme are generated, where the signers correspond to the signers in the hierarchical group signature scheme we are constructing. However, the root group manager is not given its usual secret opening key $gsk(\rho)$. Instead, each group manager (including the root) is given a key pair (pk_β, sk_β) of the $\mathcal{GM}$ cryptosystem. When a signer S_α signs a message m it first forms a group signature σ of the message m. Suppose that the signer corresponds to the path $\alpha_0, \ldots, \alpha_\delta$ in the tree, i.e. $\alpha_0 = \rho$ and $\alpha_\delta = \alpha$. Then the signer forms a chain of cryptotexts $C = (E_{pk_{\alpha_0}}(pk_{\alpha_1}), \ldots, E_{pk_{\alpha_{\delta-1}}}(pk_{\alpha_\delta}))$. Finally, it forms a NIZK π that the chain of cryptotexts C is formed in this way, and that the encrypted path corresponds to the identity of the signer hidden in the group signature σ. The hierarchical group signature consists of the triple (σ, C, π). Verification of a signature corresponds to verifying the NIZK. Opening a signature using the secret opening key of a group manager at depth l corresponds to decrypting the lth cryptotext.

Algorithm 1 (Key Generation, $\mathsf{HKg}(1^\kappa, T)$). *The key generation is defined as follows.*

1. *Generate a random string $\xi \in \{0,1\}^*$ sufficiently long for a NIZK based on $\mathcal{F}$ of the language L_{HGS} defined below.*

2. *For each node α in T, compute $(pk_\alpha, sk_\alpha) = \mathsf{GMKg}(1^\kappa)$.*

3. *Let I be the bijection mapping each list $(pk_{\alpha_0}, \ldots, pk_{\alpha_\delta})$ such that $\alpha_0, \ldots, \alpha_\delta$ is a path in T, where $\alpha_0 = \rho$ and $\alpha_\delta \in \mathcal{L}(T)$ to α_δ. Define I to map anything else to $\perp$. Denote by $T_{\mathcal{GS}}$ the tree with root ρ and leaves $\mathcal{L}(T)$.*

4. *Run $(gpk, gsk) = \mathsf{GKg}(1^\kappa, T_{\mathcal{GS}})$, and set $(hpk(\alpha), hsk(\alpha)) = ((pk_\alpha, gpk(\alpha)), (sk_\alpha, gsk(\alpha)))$ for $\alpha \in \mathcal{L}(T)$.*

5. *Set $(hpk(\rho), hsk(\rho)) = ((\xi, pk_\rho, gpk(\rho)), sk_\rho)$ and set $(hpk(\beta), hsk(\beta)) = (pk_\beta, sk_\beta)$ for $\beta \notin \mathcal{L}(T)$, $\beta \neq \rho$ (note that $hsk(\rho)$ does not contain $gsk(\rho)$).*

6. *Output (hpk, hsk).*

The result of running the above algorithm is illustrated in Figure 3.3.

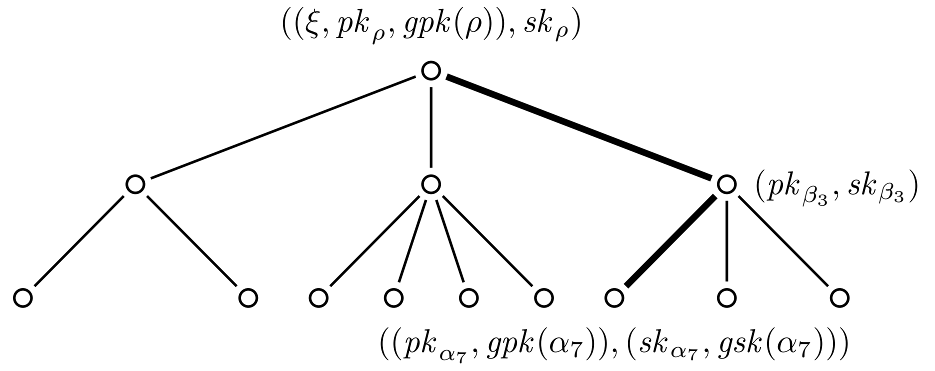

$((\xi, pk_\rho, gpk(\rho)), sk_\rho)$
$(pk_{\beta_3}, sk_{\beta_3})$
$((pk_{\alpha_7}, gpk(\alpha_7)), (sk_{\alpha_7}, gsk(\alpha_7)))$

Figure 3.3: The figure illustrates the public and secret keys along a path (the thick edges) in the tree of keys corresponding to Figure 3.1. Each node contains a pair of public and secret keys.

Algorithm 2 (Signing, $\mathsf{HSig}(m, T, hpk, hsk(\alpha))$). *Let $\alpha_0, \ldots, \alpha_\delta$ be the path to the signer S_α, i.e. $\rho = \alpha_0$ and $\alpha_\delta = \alpha$.*

1. *Compute*
$$\sigma = \mathsf{GSig}(m, T_{\mathcal{GS}}, gpk, gsk(\alpha))$$
and
$$C = (C_0, \ldots, C_{\delta-1}) = (E_{pk_{\alpha_0}}(pk_{\alpha_1}), \ldots, E_{pk_{\alpha_{\delta-1}}}(pk_{\alpha_\delta})) \ .$$

2. *Compute a NIZK π of the language L_{HGS}*
$$\left\{ (T, hpk, m, \sigma, C) \;\middle|\; \begin{array}{l} \exists pk_0, \ldots, pk_{\delta-1} : \quad \alpha_0 = \rho \ , \\ C_l = E_{pk_l}(pk_{l+1}) \quad \text{for } l = 0, \ldots, \delta - 1 \ , \\ I(pk_0, \ldots, pk_{\delta-1}) = \alpha \ , \quad \text{and} \\ \sigma = \mathsf{GSig}(m, T_{\mathcal{GS}}, gpk, gsk(\alpha)) \end{array} \right\} .$$

3. *Output (σ, C, π).*

Remark 3.4.9. *Above the complete tree of public keys hpk is given to the signing algorithm, despite that only the public keys $hpk(\alpha_0), \ldots, hpk(\alpha_\delta)$ along the path are needed. This is for notational convenience.*

Algorithm 3 (Verification, $\mathsf{HVf}(T, hpk, m, (\sigma, C, \pi))$). *On input a signature $(\sigma,$ $C,$ $\pi)$ invoke the NIZK verifier V on input $((T, hpk, m, \sigma, C,), \pi)$ and return the result.*

Algorithm 4 (Opening, $\mathsf{HOpen}(T, gpk, gsk(\beta), m, (\sigma, C, \pi))$). *If $\mathsf{HVf}(T, hpk, m, (\sigma, C, \pi)) = 0$, then return $\perp$. Otherwise compute $pk_\alpha = D_{sk_\beta}(C_l)$. If $\alpha \in \beta$ return α and otherwise $\perp$.*

Consider the construction $\mathcal{HGS}$ above, where $\mathcal{GM}$ is replaced by any indistinguishable cryptosystem $\mathcal{CS}$. Is the result secure? The answer is no. The problem is that the security of the cryptosystem $\mathcal{CS}$ does not imply that a cryptotext does not reveal the public key used for encryption. An adversary could possibly identify which key was used for encryption simply by looking at a cryptotext, and thereby extract partial information on the identity of the signer. Fortunately, we have shown that $\mathcal{GM}$ is cross-indistinguishable which solves the problem.

Remark 3.4.10. *In Section 3.7 we describe an alternative construction that seems better suited if we try to eliminate the trusted key generator, but which is harder to analyze.*

Remark 3.4.11. *It is an interesting question whether we can instantiate the Goldwasser-Micali scheme using RSA in our setting. The problem is that for a given security parameter the RSA permutations are defined for different moduli. This can be solved as follows. We modify the Goldwasser-Micali encryption algorithm such that it repeatedly chooses r until $f(r) < 2^\kappa$. This implies that $f(r) < N$ for all κ-bit moduli N. The probability that r has this property is at least* $1/4$. *Given that we put a polynomial bound on the number of tried r, the encryption process fails with negligible probability. The security of the modified scheme follows from the security of the original since the original scheme uses an r with $f(r) < 2^\kappa$ with probability at least* $1/4$.

Security Analysis

We prove the following lemma on the security of our construction, from which Theorem 3.4.1 follows immediately.

Lemma 3.4.12. *If $\mathcal{F}$ is a family of trapdoor permutations, then $\mathcal{HGS}$ is secure.*

The proof of hierarchical anonymity is similar in structure to the proof of full anonymity for the one level case given in [7], e.g. we need only single-statement simulation soundness. In the proof of hierarchical traceability we cannot proceed as in the proof of full traceability [7], since we cannot simulate answers of the signature oracle without invoking the simulator of the NIZK a polynomial number of times. This is why we must assume that the NIZK is adaptive zero-knowledge.

Proof. We prove the hierarchical anonymity and the hierarchical traceability of $\mathcal{HGS}$ separately.

PROOF OF HIERARCHICAL ANONYMITY. Suppose to the contrary that the adversary A breaks hierarchical anonymity. Then we have $\mathbf{Adv}^{\mathsf{anon}}_{\mathcal{HGS},A}(\kappa, T) \geq 1/\kappa^c$ for some polynomial size tree T, constant $c > 0$ and κ in an infinite index set $\mathcal{N}$. We construct a machine A' that runs A as a blackbox and breaks the hierarchical anonymity (i.e. full anonymity [7]) of $\mathcal{GS}$.

Definition of A'.

The adversary A' simulates the hierarchical anonymity experiment, Experiment 3, with $\mathcal{HGS}$ to A. It also plays the role of adversary in Experiment 3 with $\mathcal{GS}$.

The key generation is simulated as follows. First the NIZK simulator S is invoked to compute a reference string with a trapdoor $(\xi, s_{\text{simstate}})$. The string ξ is used instead of a random string. Recall that $T_{\mathcal{GS}}$ denotes the tree having ρ (the root of T) as root, and children $\mathcal{L}(T)$. A' first waits until it receives gpk and $(gsk(\alpha))_{\alpha \in \mathcal{L}(T)}$. Then it simulates the remaining part of the key generation honestly except that it uses these values, and it does not define $gsk(\rho)$ at all. Thus, the keys of all intermediate group managers are generated by A'.

In each iteration in the simulated experiment A may request $gsk(\alpha)$ for some group manager M_α. The only such request A' cannot answer honestly and correctly is a request for $gsk(\rho)$ which it answers by $\perp$, but this is not a problem since the experiment outputs 0 in this case anyway.

Queries to the $\mathsf{HOpen}(T, hpk, hsk(\cdot), \cdot, \cdot)$ oracle are simulated in the following way. Given a query on the form $(\beta, m, (\sigma, C, \pi))$, A' first checks that $\beta \in T$ and

$$\mathsf{HVf}(T, hpk, m, (\sigma, C, \pi)) = 1 \ .$$

If not it returns $\perp$. If so it asks its $\mathsf{Open}(T_{\mathcal{GS}}, gpk, gsk(\cdot), \cdot, \cdot)$ oracle the question (β, m, σ), which replies by $\alpha \in \mathcal{L}(T)$. Let $\alpha_0, \ldots, \alpha_\delta$ be its corresponding path, i.e. $\alpha_0 = \rho$ and $\alpha_\delta = \alpha$. Let β be on depth l. Then A' instructs the $\mathsf{HOpen}(T, hpk, hsk(\cdot), \cdot, \cdot)$ oracle to return α_{l+1} to A. Note that the answers computed in this way are not necessarily correct.

When A outputs $(\alpha^{(0)}, \alpha^{(1)}, m)$, A' outputs this as well. When A' is given a signature σ from its experiment, it computes δ samples $C = (C_0, \ldots, C_{\delta-1})$ distributed according to the distribution $\mathcal{D}_{\text{ind}}$ guaranteed to exist by Lemma 3.3.3. Then it invokes the simulator S on $((T, hpk, m, \sigma, C), s_{\text{simstate}})$ to form a proof π, and hands (σ, C, π) to A.

Eventually A outputs a bit d, which A' then returns as output.

Analysis of A'. We divide our analysis into three claims. Denote by $A^b_{\text{c,o,p}}$ the machine that on input κ simply simulates Experiment 3 with $\mathcal{HGS}$ to A and outputs the result. Then clearly $A^b_{\text{c,o,p}}(\kappa)$ is identically distributed to $\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa)$ for $b \in \{0, 1\}$. Denote by $A^b_{\text{c,o}}$ the machine which is identical to $A^b_{\text{c,o,p}}$ except for the following two changes. Firstly, instead of generating ξ as a random string, it invokes the NIZK simulator S, which returns $(\xi, s_{\text{simstate}})$. Secondly, to form the NIZK π, it invokes the NIZK simulator S on input $((T, hpk, m, \sigma, C), s_{\text{simstate}})$. Thus, except from the fact that the proof π is simulated, $A^b_{\text{c,o}}$ simulates Experiment 3 with $\mathcal{HGS}$ to A. We also define A^b_{c} to be identical to $A^b_{\text{c,o}}$ except that it simulates the $\mathsf{HOpen}(T, hpk, hsk(\cdot), \cdot, \cdot)$ oracle to A precisely as A' does. Finally, we define A^b to be identical to A^b_{c} except that the $C_0, \ldots, C_\delta$ in the challenge signature are generated precisely as A' does.

Thus, by construction A^b is identically distributed to $\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{GS},A'}(\kappa)$. This gives us a chain of distributions $A^b_{\text{c,o,p}}, A^b_{\text{c,o}}, A^b_{\text{c}}, A^b$ starting with $\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa)$ and ending with $\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{GS},A'}(\kappa)$. In the following claims we show that the distance between each pair of distributions is negligible.

Claim 2. *There is a negligible function $\epsilon_1(\kappa)$ in κ such that*

$$|\Pr[A^b_{\mathrm{c,o,p}}(\kappa) = 1] - \Pr[A^b_{\mathrm{c,o}}(\kappa) = 1]| < \epsilon_1(\kappa) \ .$$

Proof. The proof follows from the adaptive zero-knowledge of the NIZK (P, V, S).

Consider the adaptive zero-knowledge adversary A_{adzk} in Experiment 9 which we define as follows. It waits for ξ from the experiment. Then starts the simulation of $A_{\mathrm{c,o}}$ except that it uses the ξ received from the experiment. Then it continues the simulation of $A_{\mathrm{c,o}}$ until it is about to generate the NIZK π. Instead of generating the NIZK, it requests a NIZK π of the statement (T, hpk, m, σ, C) from its experiment. It must also hand the experiment a witness of this statement, but this is easy since the statement was generated honestly. Finally, it continues the simulation of $A_{\mathrm{c,o}}$ until it halts.

It follows that $A^b_{\mathrm{c,o,p}}(\kappa)$ and $A^b_{\mathrm{c,o}}(\kappa)$ are identically distributed to the outcome of the experiments $\mathbf{Exp}^{\mathsf{adind}-0}_{(P,V,S),A_{\mathrm{adzk}}}(\kappa)$ and $\mathbf{Exp}^{\mathsf{adind}-1}_{(P,V,S),A_{\mathrm{adzk}}}(\kappa)$ respectively. The reader should note that if π is a simulated proof, then the "proved" statement is always true. Thus, simulation soundness plays no role here. From the adaptive zero-knowledge of the NIZK we have that there exists a negligible function $\epsilon_1(\kappa)$ such that

$$|\mathbf{Exp}^{\mathsf{adind}-0}_{(P,V,S),A_{\mathrm{adzk}}}(\kappa) - \mathbf{Exp}^{\mathsf{adind}-1}_{(P,V,S),A_{\mathrm{adzk}}}(\kappa)| < \epsilon_1(\kappa) \ ,$$

and the claim follows. □

Claim 3. *There is a negligible function $\epsilon_2(\kappa)$ such that*

$$|\Pr[A^b_{\mathrm{c,o}}(\kappa) = 1] - \Pr[A^b_{\mathrm{c}}(\kappa) = 1]| < \epsilon_2(\kappa) \ .$$

Proof. The proof of this claim is similar to the proof in [62] and follows from the simulation soundness of the NIZK.

A query $(\beta, m, (\sigma, C, \pi))$ from A to the simulated $\mathsf{Open}(T, hpk, hsk(\cdot), \cdot, \cdot)$ is answered incorrectly precisely when π is a valid proof, i.e., $V((T, hpk, m, \sigma, C),$ $\pi, \xi) = 1$, but $(T, hpk, m, \sigma, C) \notin L_{\mathrm{HGS}}$. Denote by $E_{\mathrm{bq}}(A^b_{\mathrm{c}})$ the event that A asks such a query in the simulation by A^b_{c}, and correspondingly for $A^b_{\mathrm{c,o}}$.

We construct an adversary A_{sims} against simulation soundness, i.e. Experiment 10, as follows. It simulates A^b_{c} (or $A^b_{\mathrm{c,o}}$). Whenever A asks a query $(\beta, m, (\sigma, C, \pi))$, A_{sims} interrupts the simulation of A^b_{c} (or $A^b_{\mathrm{c,o}}$) and checks whether the query is such that $(T, hpk, m, \sigma, C) \in L_{\mathrm{HGS}}$. This is easily done using the keys to the cryptosystems and the group signature scheme. If $(T, hpk, m, \sigma, C) \notin L_{\mathrm{HGS}}$, then A_{sims} outputs $((T, hpk, m, \sigma, C),\ \pi)$. From the simulation soundness we conclude that there exists a negligible function $\epsilon(\kappa)$ such that

$$\Pr[E_{\mathrm{bq}}(A^b_{\mathrm{c}})] = \Pr[E_{\mathrm{bq}}(A^b_{\mathrm{c,o}})] = \mathbf{Exp}^{\mathsf{sims}}_{(P,V,S),A_{\mathrm{sims}}}(\kappa) < \epsilon(\kappa) \ .$$

By construction $(A^b_{\mathrm{c}}(\kappa) \mid \overline{E_{\mathrm{bq}}(A^b_{\mathrm{c}})})$ and $(A^b_{\mathrm{c,o}}(\kappa) \mid \overline{E_{\mathrm{bq}}(A^b_{\mathrm{c,o}})})$ are identically distributed. The claim follows. $\square$

Claim 4. *There is a negligible function $\epsilon_3(\kappa)$ in κ such that*

$$|\Pr[A^b_{\mathrm{c}}(\kappa) = 1] - \Pr[A^b(\kappa) = 1]| < \epsilon_3(\kappa) \ .$$

Proof. The claim follows from the indistinguishability and cross-indistinguishability of $\mathcal{GM}$ using Theorem 3.4.3, Lemma 3.4.4, and Lemma 3.3.3 by use of a standard hybrid argument.

We define a sequence of hybrid machines $A_{\mathrm{ind},l}$ for $l = 0, \ldots, \delta - 1$ as follows. $A_{\mathrm{ind},l}$ simulates A^b_{c} until it has computed $(C_0, \ldots, C_{\delta-1})$. Then it computes l samples $(C'_0, \ldots, C'_l)$ distributed according to the $\mathcal{D}_{\mathrm{ind}}$ distribution guaranteed by Lemma 3.3.3. Finally, it replaces $(C_0, \ldots, C_{\delta-1})$ in its simulation by $(C'_0, \ldots, C'_l, C_{l+1}, \ldots, C_{\delta-1})$, and continues the simulation of A^b_{c}. By construction, $A_{\mathrm{ind},-1}(\kappa)$ and $A_{\mathrm{ind},\delta-1}(\kappa)$ are identically distributed to $A^b_{\mathrm{c}}(\kappa)$ and $A^b(\kappa)$ respectively.

Suppose that the claim is false, i.e. there exists a constant c and an infinite index set $\mathcal{N}'$ such that

$$|\Pr[A_{\mathrm{ind},-1} = 1] - \Pr[A_{\mathrm{ind},\delta-1} = 1]| \geq \kappa^{-c}$$

for $\kappa \in \mathcal{N}'$. A hybrid argument implies that there exists a fixed $0 \leq l < \delta - 1$ such that

$$|\Pr[A_{\mathrm{ind},l-1} = 1] - \Pr[A_{\mathrm{ind},l} = 1]| \geq \kappa^{-c}/\delta \ .$$

Consider the adversary A_{ind} for the indistinguishability experiment (Experiment 5) run with $\mathcal{GM}$ defined as follows. It chooses $\beta^{(b)}_\delta$ randomly from $\mathcal{L}(T)$. Let $\beta_0, \ldots, \beta_\delta$ be the corresponding path, i.e. $\rho = \beta_0$ and $\beta_\delta = \beta^{(b)}_\delta$. Then it simulates $A_{\mathrm{ind},l-1}$ except that the keys $(pk_{\beta_l}, sk_{\beta_l})$ are not generated. Instead it defines pk_{β_l} to be the public key it received from Experiment 5. Then it hands (β_l, β_l) to its experiment. The experiment returns a sample C'_l.

If A requests the secret key sk_{β_l}, the simulation can not be continued and A_{ind} outputs 0. Similarly, if A outputs $(\alpha^{(0)}, \alpha^{(1)})$, where $\alpha^{(b)} \neq \beta^{(b)}$, then A_{ind} outputs 0. Since $\beta^{(b)}$ is randomly chosen, we have $\Pr[\alpha^{(b)} = \beta^{(b)}] = 1/|\mathcal{L}(T)|$.

If neither of the two events above occur, A_{ind} continues the simulation until the list of elements $(C'_0, \ldots, C'_{l-1}, C_l, \ldots, C_{\delta-1})$ has been computed. It interrupts the simulation and replaces C_l by the challenge C'_l it received from Experiment 5. Then it continues the simulation. We have that

$$\begin{aligned}
&|\Pr[\mathbf{Exp}^{\mathsf{ind}-b}_{\mathcal{GM},A_{\mathrm{ind}}}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{ind}-\mathcal{D}_{\mathrm{ind}}}_{\mathcal{GM},A_{\mathrm{ind}}}(\kappa) = 1]| \\
&\geq |\Pr[A_{\mathrm{ind},l-1} = 1 \wedge \alpha^{(b)} = \beta^{(b)}] - \Pr[A_{\mathrm{ind},l} = 1 \wedge \alpha^{(b)} = \beta^{(b)}]| \\
&= (1/|\mathcal{L}(T)|)|\Pr[A_{\mathrm{ind},l-1} = 1] - \Pr[A_{\mathrm{ind},l} = 1]| \geq 1/(|\mathcal{L}(T)|\delta\kappa^c) \ .
\end{aligned}$$

The inequality follows by construction, the equality follows by independence of the events $A_{\mathrm{ind},l} = 1$ and $\alpha^{(b)} = \beta^{(b)}$. In view of Theorem 3.4.3, Lemma 3.4.4, and Lemma 3.3.3 this contradicts either the indistinguishability or the cross-indistinguishability of $\mathcal{GM}$. Thus, the claim is true. □

Claim 5. *The hierarchical anonymity of $\mathcal{GS}$ is broken.*

Proof. From Claim 2, Claim 3, and Claim 4 follows that

$$|\Pr[A^b_{\mathrm{c,o,p}}(\kappa) = 1] - \Pr[A^b(\kappa) = 1]| < \epsilon_1(\kappa) + \epsilon_2(\kappa) + \epsilon_3(\kappa) \ ,$$

which gives

$$\begin{aligned} &|\Pr[\mathbf{Exp}^{\mathsf{anon}-0}_{\mathcal{GS},A'}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{anon}-1}_{\mathcal{GS},A'}(\kappa) = 1]| \\ &\geq |\Pr[\mathbf{Exp}^{\mathsf{anon}-0}_{\mathcal{HGS},A}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{anon}-1}_{\mathcal{HGS},A}(\kappa) = 1]| - \\ &\quad 2(\epsilon_1(\kappa) + \epsilon_2(\kappa) + \epsilon_3(\kappa)) \ . \end{aligned}$$

The assumption about A implies that the hierarchical anonymity is broken. □

Proof of Hierarchical Traceability. Suppose to the contrary that A breaks the hierarchical traceability of $\mathcal{HGS}$. Then $\mathbf{Adv}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa, T) \geq 1/\kappa^c$ for some polynomial size tree T, constant $c > 0$ and κ in an infinite index set $\mathcal{N}$. We construct a machine A' that runs A as a blackbox and breaks the hierarchical traceability (i.e. full traceability [7]) of $\mathcal{GS}$.

Definition of A'. The adversary A' simulates the hierarchical traceability experiment, Experiment 4, with $\mathcal{HGS}$ to A. It also plays the role of the adversary in Experiment 4 with $\mathcal{GS}$.

The key generation is simulated as follows. First the NIZK simulator is invoked to compute a reference string with a trapdoor $(\xi, s_{\mathrm{simstate}})$. The string ξ is used instead of a random string. Recall that $T_{\mathcal{GS}}$ denotes the tree having ρ (the root of T) as root, and children $\mathcal{L}(T)$. A' first waits until it receives the keys $(gpk(\rho), gsk(\rho))$ of the root, and the public keys of all signers $(gpk(\alpha))_{\alpha \in \mathcal{L}(T)}$ from its experiment. Then it simulates the key generation honestly except that it uses the received values, and it does not define $gsk(\alpha)$ for any $\alpha \in \mathcal{L}(T)$ at all. Thus, the keys of all intermediate group managers are generated by A'.

In each iteration in the experiment simulated to A, it may request $gsk(\alpha)$ for some signer S_α. When this happens A' requests $gsk(\alpha)$ from its experiment, and hands $(sk_\alpha, gsk(\alpha))$ to A.

When A queries its $\mathsf{HSig}(\cdot, T, hpk, hsk(\cdot))$ oracle on (m, α) the reply is computed as follows. First A' queries its $\mathsf{GSig}(\cdot, T_{\mathcal{GS}}, gpk, gsk(\cdot))$ oracle on (m, α). The answer is a $\mathcal{GS}$ signature σ. Then A' computes $C = (C_0, \ldots, C_{\delta-1})$ as defined by HSig. Finally, it invokes the NIZK simulator S on input $((T, hpk, m, \sigma, C), \xi, s_{\mathrm{simstate}})$ to get a simulated proof π, and hands (σ, C, π) to A.

At some point A outputs a message signature pair $(m,(\sigma,C,\pi))$. Then A' outputs (m,σ). This concludes the definition of A'.

Analysis of A'. We divide our analysis into several claims. Denote by $A_{\pi,\mathrm{p}}$ the machine that simulates Experiment 4 with $\mathcal{HGS}$ to A and outputs 1 if the experiment outputs 1 and the output $(m,(\sigma,C,\pi))$ of A satisfies $(T,hpk,m,\sigma,C) \in L_{\mathrm{HGS}}$.

Denote by A_π the machine that is identical to $A_{\pi,\mathrm{p}}$ except that it simulates the answers from the $\mathsf{HSig}(\cdot,T,hpk,hsk(\cdot))$ oracle to A precisely as A' does.

Claim 6. *There is a negligible function $\epsilon_1(\kappa)$ in κ such that*

$$\Pr[\mathbf{Exp}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa)=1] \leq \Pr[A_{\pi,p}(\kappa)=1]+\epsilon_1(\kappa) \ .$$

Proof. The claim follows from the soundness of the NIZK. Denote by $E_{\pi,p}$ the event that the output $(m,(\sigma,C,\pi))$ of A in the experiment satisfies $(T, hpk, m, \sigma, C) \in L_{\mathrm{HGS}}$. From the soundness of the NIZK follows that $\Pr[\overline{E_{\pi,\mathrm{p}}}]$ is negligible. By definition we have that $\Pr[A_{\pi,\mathrm{p}}(\kappa)=1] = \Pr[\mathbf{Exp}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa)=1 \wedge E_{\pi,\mathrm{p}}]$. The claim follows. □

Claim 7. *There is a negligible function $\epsilon_2(\kappa)$ in κ such that*

$$|\Pr[A_\pi(\kappa)=1]-\Pr[A_{\pi,p}(\kappa)=1]| \quad < \quad \epsilon_2(\kappa) \ .$$

Proof. The claim follows from the adaptive zero-knowledge of the NIZK. We construct an adversary A_{adzk} against the adaptive zero-knowledge, Experiment 9, as follows.

It simulates A_π except for the following two modifications. Firstly, it uses the random string ξ from the experiment instead of generating its own. Secondly, instead of invoking the simulator S on input $((T,hpk,m,\sigma,C), \xi, s_{\mathrm{simstate}})$ to produce a NIZK π it requests a NIZK of (T, hpk, m, σ, C) from its experiment. To do this it must also hand a witness to the experiment, but this is not a problem since it has generated all keys.

It follows that

$$\begin{aligned}&|\Pr[A_{\pi,\mathrm{p}}(\kappa)=1]-\Pr[A_\pi(\kappa)=1]|\\&=|\Pr[\mathbf{Exp}^{\mathsf{adzk}-0}_{(P,V,S),A_{\mathrm{adzk}}}(\kappa)=1]-\Pr[\mathbf{Exp}^{\mathsf{adzk}-1}_{(P,V,S),A_{\mathrm{adzk}}}(\kappa)=1]|<\epsilon_2(\kappa) \ ,\end{aligned}$$

for some negligible function $\epsilon_2(\kappa)$. □

Claim 8. $\Pr[A_\pi(\kappa)=1] \leq \Pr[\mathbf{Exp}^{\mathsf{trace}}_{\mathcal{GS},A'}(\kappa)=1]$.

Proof. All inputs to A in the simulation of A_π are identically distributed to the corresponding inputs in Experiment 4. The only difference in how the output of A_π and the experiment are defined is that A_π outputs 1 if the output $(m,(\sigma,C,\pi))$ of A satisfies $(T,hpk,m,\sigma,C) \in L_{\mathrm{HGS}}$, and the experiment outputs 1 if $\mathsf{GVf}(\emptyset,gpk,m,\sigma)=1$, but the former requirement implies the latter. Thus, the claim follows. □

Claim 9. *The hierarchical traceability of $\mathcal{GS}$ is broken.*

Proof. From Claim 6 and Claim 7 Claim 8 follows that

$$\Pr[\mathbf{Exp}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa) = 1] \leq \Pr[A_{\pi,\mathrm{p}}(\kappa) = 1] + \epsilon_1(\kappa) \leq$$
$$Pr[A_\pi(\kappa) = 1] + \epsilon_1(\kappa) + \epsilon_2(\kappa) \leq \Pr[\mathbf{Exp}^{\mathsf{trace}}_{\mathcal{GS},A'}(\kappa) = 1] + \epsilon_1(\kappa) + \epsilon_2(\kappa) \ .$$

The claim now follows from the assumption that $\mathcal{HGS}$ is broken by A. □

Conclusion of Proof. If $\mathcal{HGS}$ is not hierarchically anonymous, then by Claim 5 neither is $\mathcal{GS}$. If $\mathcal{HGS}$ is not hierarchically traceable, then by Claim 9 neither is $\mathcal{GS}$. This concludes the proof. □

3.5 A Construction under the DDH Assumption and the Strong RSA Assumption

In this section we show how to construct hierarchical group signatures under the DDH assumption and the strong RSA assumption. In contrast to the previous section our focus here is on practicality. We give an explicit construction where the details of all subprotocols are completely specified. Then we prove the security of our construction in the random oracle model. By now it is known that the random oracle hypothesis is not literally true [18]. However, all known counter-examples are contrived, and for practical protocols a security proof in the random oracle model is often considered to be enough.

Review of Some Notions and Primitives

Before we give our construction we need to review some notions and primitives on which our construction is based. Readers familiar with these primitives can safely browse this section, but should observe that our notation differs slightly from the standard at some points because of name collisions.

Cunningham Chains

Let q_0 and q_1 be primes such that $q_0 = 2q_1 + 1$. Then there is a unique subgroup $G_{q_1} \subset \mathbb{Z}^*_{q_0}$ of order q_1. Let g_1 be a generator of G_{q_1}. The discrete logarithm of an element $z \in G_{q_1}$ in the basis g_1 is defined as the (unique) $r \in \mathbb{Z}_{q_1}$ such that $z = g_1^r$. This is usually written $\log_{g_1} z = r$.

The primes q_0 and q_1 above are clearly of a special form. In fact q_1 is called a Sophie Germain prime. One can demand that q_1 is of the same form as q_0 and so on. This leads to the following definition.

Definition 3.5.1 (Cunningham Chain). *A sequence $q_0, \ldots, q_{k-1}$ of primes is called a* Cunningham Chain[1] *of length k if $q_i = 2q_{i+1} + 1$ for $i = 0, \ldots, k-2$.*

[1] *This is a chain of the second kind. A chain of the first kind satisfies $q_i = 2q_{i+1} - 1$.*

Throughout this chapter $q_0, \ldots, q_3$ are chosen to form a Cunningham Chain of length 4 and we denote by G_{q_i} the unique subgroup of order q_i of $\mathbb{Z}^*_{q_{i-1}}$. We also pick a generator g_i of G_{q_i}. Thus, the groups are set up such that $\mathbb{Z}^*_{ql} \approx G_{q_{l+1}} \times \{-1, 1\}$, $\langle g_{l+1} \rangle = G_{q_{l+1}} \approx \mathbb{Z}_{q_{l+1}}$.

Before we start using Cunningham chains for cryptography we are obliged to ask if they exist at all. Unfortunately, there exists no proof that there are infinitely many Cunningham chains of any length, not even of length 2 which correspond to the Sophie Germain primes. However, one can apply a heuristic argument and assume that a randomly chosen integer n is prime with "probability" $1/\log n$. If we also assume that the event that $(n-1)/2$ is prime is independent of the event that n is prime for every prime we should find a Cunningham chain of length k with probability close to $1/\log^k n$. We stress that one must be careful with this type of heuristic argument since there exist counter examples [46], but the argument seems reasonable in our setting and agrees with computational experiments. In practice it is not hard to find Cunningham chains of length 4 for primes of the size used in current cryptography (cf. [58], [59]). Young and Yung [74] have also published some heuristic tricks for finding length-3 Cunningham chains. We let $\mathsf{CunnGen}_k$ denote the algorithm that on input 1^κ outputs a κ-bit Cunningham chain of length k. Note that the existence of 2-Cunningham chains is implicitly assumed in many papers, e.g. [25].

Although we describe our scheme for Cunningham chains because they are well-known, the scheme also works for a sequence of primes q_0, q_1, q_2, q_3 such that $q_i = a_i q_{i+1} + 1$ for positive numbers a_i. Chains of this type have the advantage that they are easier to generate. The existence of such chains follows from the following formal assumption:

Assumption 1. *There exists a constant c and a probabilistic polynomial Turing machine that given a κ-bit number n as input with overwhelming probability outputs a $\log^c \kappa$-bit number a such that $an + 1$ is prime.*

We generalize the notation G_n to denote the cyclic subgroup of order n of $\mathbb{Z}^*_{an+1}$, where a is the smallest number such that $an + 1$ is prime.

Decision Diffie-Hellman Problem

The Decision Diffie-Hellman problem in a group G_n is defined as the problem of distinguishing the distributions $(g^\alpha, g^\beta, g^\gamma)$ and $(g^\alpha, g^\beta, g^{\alpha\beta})$, where $\alpha, \beta, \gamma \in \mathbb{Z}_n$ are randomly distributed.

Experiment 11 (Decision Diffie-Hellman, $\mathbf{Exp}^{\mathsf{ddh}-b}_{G_n,A}(\kappa)$).

$$g \leftarrow G_n, \quad \alpha, \beta, \gamma \leftarrow \mathbb{Z}_n, \quad (D_1, D_2, D_3) \leftarrow (g^\alpha, g^\beta, g^{(b\gamma + (1-b)\alpha\beta)})$$
$$d \leftarrow A(g, D_1, D_2, D_3)$$

The advantage of the adversary is

$$\mathbf{Adv}^{\mathsf{ddh}}_{G_n,A}(\kappa) = |\Pr[\mathbf{Exp}^{\mathsf{ddh}-0}_{G_n,A}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{ddh}-1}_{G_n,A}(\kappa) = 1]| \ .$$

Assumption 2 (Decision Diffie-Hellman Assumption over G_n). *For all $A \in$ PPT* the advantage $A = \{A_\kappa\}$, $\mathbf{Adv}^{\mathsf{ddh}}_{G_n,A}(\kappa)$ is negligible.*

The ElGamal cryptosystem

We review the ElGamal [28] cryptosystem employed in G_q. We write $\mathsf{ElgKg}(G_q, g)$ for the algorithm that generates a random private key $x \in \mathbb{Z}_q$, computes a public key (g, y), where $y = g^x$, and outputs $((g, y), x)$. Encryption of a message $m \in G_q$ using the public key (g, y) is given by $E_{(g,y)}(m, r) = (g^r, y^r m)$ for $r \in_R \mathbb{Z}_q$, and decryption of a cryptotext on the form $(u, v) = (g^r, y^r m)$ using the private key x is given by $D_x(u, v) = u^{-x}v = m$.

The RSA cryptosystem

The key generation for the RSA cryptosystem [61] consists of generating primes p and q of the same size and computing $N = pq$. The parameters e and d are chosen so that $\gcd(e, d) = 1 \pmod{\phi(n)}$, where $\phi(n) = (p - 1)(q - 1)$. The public key is (e, n) and the private key is d. If we choose p and q so that $p = 2p' + 1$ and $q = 2q' + 1$ where p', q' are primes, we ensure that the order of the group QR_N, the quadratic residues modulo N, is $p'q'$. Throughout the chapter we write members of QR_N using bold font (e.g., $\mathbf{g}, \mathbf{y}$).

The Strong RSA Assumption

The *Strong RSA Assumption* y the assumption says that it is hard to compute any non-trivial root in $\mathbb{Z}_N$ where N is an RSA modulus, even if allowed to select which root to compute. This differs from the standard RSA assumption, where the root to compute is predetermined. Like the standard RSA assumption the strong RSA assumption implies that factoring is hard.

Assumption 3 (Strong RSA Assumption). *For any adversary $A \in$ PPT* the following holds: $\forall c > 0$, $\exists \kappa_0$, such that for $\kappa > \kappa_0$ we have:*

$$\Pr[(P, Q) \leftarrow \mathsf{csRSA}(1^\kappa), \boldsymbol{\sigma} \leftarrow \mathbb{Z}^*_{PQ}, (\mathbf{m}, e) \leftarrow A(PQ, \boldsymbol{\sigma}), \mathbf{m}^e = \boldsymbol{\sigma}, e > 1] < \frac{1}{\kappa^c} \ .$$

The Chaum van Heijst Pfitzmann Hash Function

We write $\mathsf{CHPg}(G_q, \delta)$ for the algorithm that takes as input the representation of a group G_q and $\delta \in \mathbb{N}$ and outputs a list $(h_1, \ldots, h_\delta) \in G_q^\delta$ of randomly chosen elements. We employ the Chaum van Heijst Pfitzmann hash function [20] defined as $H^{\mathsf{CHP}} : \mathbb{Z}_q \to G_q$, $H^{\mathsf{CHP}} : (z_1, \ldots, z_\delta) \mapsto \prod_{l=1}^{\delta} h_l^{z_l}$. They prove that this map is one-way and collision free if the discrete logarithm problem is hard in G_q. We abuse notation and use H^{CHP} to denote both the map and its representation $(h_1, \ldots, h_\delta)$.

Lemma 3.5.2 (cf. [20]). *The hash function H^{CHP} is one-way and collision-free if the discrete logarithm problem is hard in G_q.*

The Shamir Hash Function

We write $H^{\mathsf{Sh}}_{(N,\mathbf{g})}(x)$ for the algorithm that computes $\mathbf{g}^x \bmod N$, where N is a composite number, $\mathbf{g} \in \mathrm{QR}_N$ and $x \in \mathbb{Z}$. The idea to use this as a hash function was proposed by Shamir. When $(N, \mathbf{g})$ are clear from the context we may leave them out. We let H^{Sh} denote both the map and the pair $(N, \mathbf{g})$. We have the following security result for H^{Sh}:

Lemma 3.5.3. *The hash function H^{Sh} is collision-free if the factoring problem is hard.*

Proof. Assume that H^{Sh} is not collision-free, and that $H^{\mathsf{Sh}}_{(N,\mathbf{g})}(x_1) = H^{\mathsf{Sh}}_{(N,\mathbf{g})}(x_2)$ where $x_1 \neq x_2$. Then $\mathbf{g}^{x_1 - x_2} = 1$, meaning that $x_1 - x_2$ is a multiple of the order of QR_N. This information is enough to factor N as shown in [53]. □

The Cramer-Shoup Cryptosystem

We review the Cramer-Shoup cryptosystem [24] over G_q employed with a collision free one-way function H. We denote their cryptosystem by $\mathcal{CS}^{\mathsf{cs}}_H = (\mathsf{CSKg}^{\mathsf{cs}}, E^{\mathsf{cs}}, D^{\mathsf{cs}})$.

The key generation algorithm $\mathsf{CSKg}^{\mathsf{cs}}(G_q, q)$ generates random $\bar{g}_1, \bar{g}_2 \in G_q$ and $\bar{x}_1, \bar{x}_2, \bar{y}_1, \bar{y}_2, \bar{z} \in \mathbb{Z}_q$, computes $\bar{c} = \bar{g}_1^{\bar{x}_1} \bar{g}_2^{\bar{x}_2}$, $\bar{d} = \bar{g}_1^{\bar{y}_1} \bar{g}_2^{\bar{y}_2}$, and $\bar{h} = \bar{g}_1^{\bar{z}}$ and outputs $(\bar{g}_1, \bar{g}_2, \bar{c}, \bar{d}, \bar{h}, \bar{x}_1, \bar{x}_2, \bar{y}_1, \bar{y}_2, \bar{z})$. Encryption of a message $m \in G_q$ using the public key $Y = (\bar{g}_1, \bar{g}_2, \bar{c}, \bar{d}, \bar{h})$ is given by

$$E^{\mathsf{cs}}_Y(m, r) = (u, \mu, v, \nu) = (\bar{g}_1^r, \bar{g}_2^r, \bar{h}^r m, \bar{c}^r \bar{d}^{rH(u,\mu,v)})$$

for a randomly chosen $r \in \mathbb{Z}_q$. Note that (u, v) is an ElGamal encryption of the message m, so decryption of a cryptotext (u, μ, v, ν) using the private key $X = (\bar{x}_1, \bar{x}_2, \bar{y}_1, \bar{y}_2, \bar{z})$ is given by $D^{\mathsf{cs}}_X(u, \mu, v, \nu) = D_{\bar{z}}(u, v) = m$ for valid cryptotexts. A cryptotext is considered valid if the predicate

$$T^{\mathsf{cs}}_X(u, \mu, v, \nu) = (u^{\bar{x}_1 + \bar{x}_2 H(u,\mu,v)} \mu^{\bar{y}_1 + \bar{y}_2 H(u,\mu,v)} = \nu)$$

is satisfied. We let $T^{\mathsf{cs}}_X(u, \mu, v, \nu) = 1$ if it is satisfied and 0 otherwise. An invalid cryptotext decrypts to $\perp$.

Cramer and Shoup [24] prove that their cryptosystem is CCA2-secure under standard assumptions.

Lemma 3.5.4 (cf. [24]). *The cryptosystem $\mathcal{CS}^{\mathsf{cs}}_H$ is CCA2-secure under the DDH assumption in G_q if H is collision free.*

Cramer-Shoup RSA Signatures

We review the signature scheme by Cramer and Shoup [25]. We denote their construction by $\mathcal{SS}^{\mathsf{cs}}_{H_1,H_2} = (\mathsf{SSKg}^{\mathsf{cs}}, \mathsf{Sig}^{\mathsf{cs}}, \mathsf{Vf}^{\mathsf{cs}})$, and review the algorithms it consists of below. Here H_1 and H_2 are hash functions.

We write csRSA for the algorithm that on input 1^κ generates two random $\kappa/2$-bit primes $P = 2P' + 1$ and $Q = 2Q' + 1$, where P' and Q' are also prime, and returns (P, Q). Thus, csRSA generates a κ-bit RSA modulus $N = PQ$.

Algorithm 5 (Key Generation, $\mathsf{SSKg}^{\mathsf{cs}}(1^\kappa)$).

1. *Run* $(P, Q) = \mathsf{csRSA}(1^\kappa)$ *and set* $P' = (P-1)/2$, $Q' = (Q-1)/2$ *and* $N = PQ$.
2. *Choose* $\mathbf{h}, \mathbf{z} \in \mathrm{QR}_N$ *randomly.*
3. *Choose a random* $(\kappa + 1)$*-bit prime* e' *such that* $e' \equiv 1 \pmod 4$.
4. *Output* $(N, e', \mathbf{h}, \mathbf{z}, P'Q')$.

Algorithm 6 (Signing, $\mathsf{Sig}^{\mathsf{cs}}(m, H_1, H_2, N, \mathbf{h}, \mathbf{z}, e', P'Q')$).

1. *Choose a random* $(\kappa + 1)$*-bit prime* e *such that* $e \equiv 3 \bmod 4$ *and a random* $\boldsymbol{\sigma}' \in \mathrm{QR}_N$.
2. *Compute* $\mathbf{z}' = (\boldsymbol{\sigma}')^{e'} \mathbf{h}^{-H_1(m)}$, $\boldsymbol{\sigma} = \left(\mathbf{z}\mathbf{h}^{H_2(\mathbf{z}')}\right)^{1/e}$ *and output* $(e, \boldsymbol{\sigma}, \boldsymbol{\sigma}')$.

Algorithm 7 (Verification, $\mathsf{Vf}^{\mathsf{cs}}(H_1, H_2, N, \mathbf{h}, \mathbf{z}, e', m, (e, \boldsymbol{\sigma}, \boldsymbol{\sigma}'))$.).

1. *Verify that* e *is an odd number of length between* $(\kappa + 1)$ *bits and* $\left(\frac{3}{2}\kappa - 4\right)$ *bits that is distinct from* e'.
2. *Compute* $\mathbf{z}' = (\boldsymbol{\sigma}')^{e'} \mathbf{h}^{-H_1(m)}$ *and verify that* $\mathbf{z} = \boldsymbol{\sigma}^e \mathbf{h}^{-H_2(\mathbf{z}')}$. *If so output* 1, *otherwise output* 0.

We have modified the scheme slightly by making e' always equal to 1 modulo 4 and the primes generated at signing, e, equal to 3 modulo 4. This makes it easier to prove later in zero-knowledge that $e \neq e'$. In the original scheme H_1 and H_2 are equal, but in our setting they will be different. This does not affect the security proof in any way.

Also in our description the exponent e is longer than the modulus, but in the original description e is shorter than P' and Q'. Below we argue why the security proof still holds.

The above signature scheme can proven secure under the *Strong RSA Assumption* defined below. Informally the assumption says that it is hard to compute any non-trivial root in $\mathbb{Z}_N$ where N is an RSA modulus, even if allowed to select which root to compute. This differs from the standard RSA assumption, where the root to compute is predetermined. Like the standard RSA assumption the strong RSA assumption implies that factoring is hard.

Lemma 3.5.5. *The signature scheme* $\mathcal{SS}^{\mathsf{cs}}_{H_1,H_2}$ *is CMA-secure under the strong RSA assumption if* H_1 *and* H_2 *are collision-free one-way functions.*

Proof. We assume familiarity with [25]. When the length of the exponent is between $\kappa + 1$ bits and $\frac{3}{2}\kappa - 4$ bits, the proof of [25] holds except for how a Type III forger is used to break the strong RSA assumption, where a Type III forger is defined as a forger that outputs a signature (using our notation) $(e, \boldsymbol{\sigma}, \boldsymbol{\sigma'})$ such that $e \neq e_i$ for all signatures e_i it has seen previously. The output from the Type III forger is used to form the equation $\boldsymbol{\sigma}^{e'} = \mathbf{z}^t$ where $\mathbf{z}$ is the number we wish to compute a root of and t is known. Then the fact that $\gcd(e, t, 2P'Q') = 1$ is used, which in the original setting holds because e is odd and smaller than P' and Q'. In our setting it may or may not hold. If the forger outputs an e such that it does hold, then the original proof goes throgh. If it does not hold, then $\gcd(e, 2P'Q')$ is either P', Q', or $P'Q'$ since e is odd. In all of these cases we have enough information to factor N. □

Proofs of Knowledge

We use a relatively complex proof of knowledge in our construction, but we postpone a careful description of the properties we need for Section 3.6. We define some notation used in the description of our construction.

Given a three-move public coin interactive proof (P, V) for a language L, we can use the Fiat-Shamir heuristic to construct a non-interactive variant in the random oracle model, by simply replacing the message sent by the verifier by the output of the random oracle. We write $\pi = P^{\mathsf{O}(m,\cdot)}(x, w)$ to denote the transcript of such a protocol execution, where $x \in L$ is the common input, w is a witness for x, and P interacts with the random oracle $\mathsf{O}(m, \cdot)$. We write $V^{\mathsf{O}(m,\cdot)}(x, \pi)$ for the verification.

Our Construction

We are now ready to describe our construction. The basic idea is similar to the construction under general assumptions. A signer encrypts a path from the root to its leaf using the public keys along the path. Then it proves that it knows a $\mathcal{SS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}, H^{\mathsf{Sh}}}$ signature on the list of public keys it used. Thus, a private key of a signer is simply a $\mathcal{SS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}, H^{\mathsf{Sh}}}$ signature on the public keys along its path. We denote our construction by $\mathcal{HGS} = (\mathsf{HKg}, \mathsf{HSig}, \mathsf{HVf}, \mathsf{HOpen})$, and define algorithms HKg, HSig, HVf, and HOpen below.

Key Generation

The key generation phase proceeds as follows. Each group manager is given an ElGamal key pair, and each signer is given an RSA signature of the public keys of the group managers on the path from the root to the signer.

Algorithm 8 (Key Generation, $\mathsf{HKg}(1^\kappa, T)$).

1. *Run $(q_0, \ldots, q_3) = \mathsf{CunnGen}_4(1^\kappa)$ to generate a Cunningham chain of length 4, and choose $g_i \in G_{q_i}$ randomly for $i = 1, 2, 3$.*

2. *Let δ be the depth of the tree T, and run*

$$H^{\mathsf{CHP}} = (h_1, \ldots, h_\delta) = \mathsf{CHPg}(G_{q_2}, \delta)$$

to generate a collision free one-way function.

3. *Run $(X, Y) = \mathsf{CSKg}^{\mathsf{cs}}(G_{q_3}, q_3)$ to generate keys for a Cramer-Shoup cryptosystem over G_{q_3}.*

4. *Run $(N, \mathbf{h}, \mathbf{z}, e, t) = \mathsf{SSKg}^{\mathsf{cs}}(1^\kappa)$ and randomly choose $\mathbf{g} \in \mathrm{QR}_N$ to generate keys for a Cramer-Shoup RSA signature scheme employed with the hash functions H^{CHP} and $H^{\mathsf{Sh}}_{(N,\mathbf{g})}$.*

5. *Choose a prime on the form $aN + 1$ and let G_N be the unique subgroup of order N of $\mathbb{Z}^*_{aN+1}$. Choose g_N, y_N randomly in G_N.*

6. *For each node β in T, generate keys*

$$(hpk(\beta), hsk(\beta)) = (y_\beta, x_\beta) = \mathsf{ElgKg}(G_{q_3}, g_3)$$

for an ElGamal cryptosystem over G_{q_3}.

7. *For each leaf α let $\alpha_0, \ldots, \alpha_\delta$ with $\alpha_0 = \rho$ and $\alpha_\delta = \alpha$ be the path from the root to α, compute*

$$(e_\alpha, \boldsymbol{\sigma}_\alpha, \boldsymbol{\sigma}'_\alpha) = \mathsf{Sig}^{\mathsf{cs}}((y_{\alpha_1}, \ldots, y_{\alpha_\delta}), H^{\mathsf{CHP}}, H^{\mathsf{Sh}}, N, \mathbf{h}, \mathbf{z}, e', t)$$

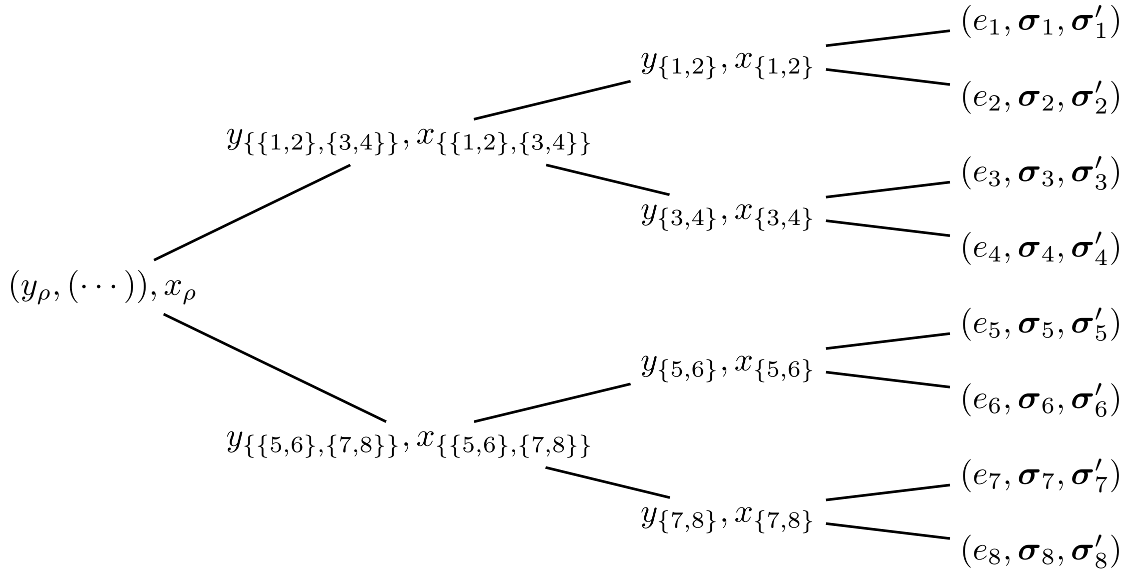
and redefine $hsk(\alpha) = (e_\alpha, \boldsymbol{\sigma}_\alpha, \boldsymbol{\sigma}'_\alpha)$.

8. *Let ρ be the root of T. Choose $\mathbf{y} \in \mathrm{QR}_N$ randomly. Choose $x_i \in \mathbb{Z}_{q_i}$ randomly and compute $y_i = g_i^{x_i}$ for $i = 1, 2, 3$. Set the three security parameters $\kappa_1, \kappa_2, \kappa_3 = \Theta(\kappa)$. Redefine the public key $hpk(\rho)$ of the root ρ to be $(hpk(\rho), q_0, H^{\mathsf{CHP}}, N, e, g_1, y_1, g_2, y_2, g_3, y_3, \mathbf{g}, \mathbf{y}, g_N, y_N, \kappa_1, \kappa_2, \kappa_3)$ and output hpk, hsk.*

Remark 3.5.6. *The three security parameters κ_1, κ_2 and κ_3 are used in the proof of knowledge. For details, see section 3.6.*

Computing, Verifying and Opening a Signature

Any leaf α can be associated with a path $\alpha_0, \ldots, \alpha_\delta$ where $\rho = \alpha_0$ and $\alpha_\delta = \alpha$ from the root to the leaf in the natural way. The signer encrypts its path using the public keys of the group managers along this path, i.e., the signer computes a list $(E_{y_{\alpha_0}}(y_{\alpha_1}), \ldots, E_{y_{\alpha_{\delta-1}}}(y_{\alpha_\delta}))$. Note the particular way in which the public keys are chained. It also commits to the $\mathcal{SS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}, H^{\mathsf{Sh}}}$ signature of the message $(y_{\alpha_1}, \ldots, y_{\alpha_\delta})$ it was given by the key generator. Then it computes a Schnorr-like



$(y_\rho, (\cdots)), x_\rho$
$y_{\{\{1,2\},\{3,4\}\}}, x_{\{\{1,2\},\{3,4\}\}}$
$y_{\{\{5,6\},\{7,8\}\}}, x_{\{\{5,6\},\{7,8\}\}}$
$y_{\{1,2\}}, x_{\{1,2\}}$
$y_{\{3,4\}}, x_{\{3,4\}}$
$y_{\{5,6\}}, x_{\{5,6\}}$
$y_{\{7,8\}}, x_{\{7,8\}}$
$(e_1, \boldsymbol{\sigma}_1, \boldsymbol{\sigma}'_1)$
$(e_2, \boldsymbol{\sigma}_2, \boldsymbol{\sigma}'_2)$
$(e_3, \boldsymbol{\sigma}_3, \boldsymbol{\sigma}'_3)$
$(e_4, \boldsymbol{\sigma}_4, \boldsymbol{\sigma}'_4)$
$(e_5, \boldsymbol{\sigma}_5, \boldsymbol{\sigma}'_5)$
$(e_6, \boldsymbol{\sigma}_6, \boldsymbol{\sigma}'_6)$
$(e_7, \boldsymbol{\sigma}_7, \boldsymbol{\sigma}'_7)$
$(e_8, \boldsymbol{\sigma}_8, \boldsymbol{\sigma}'_8)$

Figure 3.4: The output of HKg for a four-level tree. The common group parameters (key size, generators etc.) have been abbreviated by $(\cdots)$.

"proof of knowledge" in the random oracle model that the cryptotexts and commitments are indeed formed as described. The message to be signed is included in the input to the hash function (the random oracle) similarly to Schnorr signatures. Thus, the signer actually proves that it possesses an $\mathcal{SS}^{\mathsf{cs}}_{H^{\mathsf{CHP}},H^{\mathsf{Sh}}}$ signature of the list of public keys it uses to encrypt the public keys along its path. Hierarchical anonymity follows since only the holders of the secret keys corresponding to y_{α_l} can open any part of the signature. Hierarchical traceability follows since the signer cannot forge a $\mathcal{SS}^{\mathsf{cs}}_{H^{\mathsf{CHP}},H^{\mathsf{Sh}}}$ signature corresponding to a path different from its own.

Algorithm 9 (Signing, $\mathsf{HSig}(m, T, hpk, hsk(\alpha))$).
Let $\alpha_0, \ldots, \alpha_\delta$ with $\rho = \alpha_0$ and $\alpha_\delta = \alpha$ be the path to the signer S_α.

1. *Choose $r_0, \ldots, r_\delta \in \mathbb{Z}_{q_3}$ randomly and compute $(u_l, v_l) = E_{(y_{\alpha_l}, g_3)}(y_{\alpha_{l+1}}, r_l)$, for $l = 0, \ldots, \delta - 1$, and $C_\delta = E^{\mathsf{cs}}_Y(y_{\alpha_\delta}, r_\delta)$. This is the list of cryptotexts.*

2. *Choose $r, s, r', s', t \in [0, 2^{\kappa_3}N - 1]$ randomly and set $(\mathbf{u}, \mathbf{v}) = (\mathbf{g}^s\mathbf{y}^r, \mathbf{g}^r\boldsymbol{\sigma}_\alpha)$, $(\mathbf{u}', \mathbf{v}') = (\mathbf{g}^{s'}\mathbf{y}^{r'}, \mathbf{g}^{r'}\boldsymbol{\sigma}'_\alpha)$, and $\mathbf{C} = \mathbf{g}^{e_\alpha}\mathbf{y}^t$. This is a commitment of the signature $(e_\alpha, \boldsymbol{\sigma}_\alpha, \boldsymbol{\sigma}'_\alpha)$.*

3. *Denote by R_{HGS} the binary relation consisting of pairs (X, W), where*

$$\begin{aligned} X &= \big((u_l, v_l)_{l=0}^{\delta-1}, C_\delta, (\mathbf{u}, \mathbf{v}), (\mathbf{u}', \mathbf{v}'), \mathbf{C}\big) \in \\ &\qquad (G_{q_3} \times G_{q_3})^\delta \times G_{q_3}^4 \times \mathrm{QR}_N^2 \times \mathrm{QR}_N^2 \times \mathrm{QR}_N \\ W &= \big((\tau_0, \ldots, \tau_{\delta-1}), (\tau, \zeta, \tau', \zeta', \psi, \varepsilon)\big) \in \\ &\qquad \mathbb{Z}_{q_3}^\delta \times [0, 2^{\kappa_3}N - 1]^5 \times [2^\kappa, 2^{\kappa+1} - 1] \end{aligned}$$

such that

$$\begin{array}{l} \gamma_0 = y_{\alpha_0} \\ \big((u_l, v_l) = E_{(\gamma_l, g)}(\gamma_{l+1}, \tau_l)\big)_{l=0}^{\delta-1} \\ C_\delta = E_Y^{\mathsf{cs}}(\gamma_\delta, \tau_\delta) \end{array} \quad \textit{and} \quad \begin{array}{l} \mathbf{u} = \mathbf{g}^{\zeta}\mathbf{y}^{\tau}, \quad \mathbf{u}' = \mathbf{g}^{\zeta'}\mathbf{y}^{\tau}, \\ \mathbf{C} = \mathbf{g}^{\varepsilon}\mathbf{y}^{\psi} \\ \mathsf{Vf}^{\mathsf{cs}}(H^{\mathsf{CHP}}, H^{\mathsf{Sh}}, N, \mathbf{h}, \mathbf{z}, e', \\ \qquad (\gamma_1, \ldots, \gamma_\delta), \\ \qquad (\varepsilon, \mathbf{v}/\mathbf{y}^{\boldsymbol{\tau}}, \mathbf{v}'/\mathbf{y}^{\boldsymbol{\tau}'})) = 1 \end{array} \quad .$$

In Section 3.6 we construct an honest verifier public coin zero-knowledge proof of knowledge (P, V) for this relation. The signer computes a non-interactive proof $\pi = P^{\mathsf{O}(m,\cdot)}(X, W)$ in the random oracle model.

4. *Output the signature $\big((u_l, v_l)_{l=0}^{\delta-1}, C_\delta, (\mathbf{u}, \mathbf{v}), (\mathbf{u}', \mathbf{v}'), \mathbf{C}, \pi\big)$.*

Remark 3.5.7. *Note that we switch the order of the components in the ElGamal cryptosystem so that, for example, $D_{-x_\rho}(u_0, v_0) = y_{\alpha_1}$ in order to simplify the construction of the proof of knowledge.*

The construction of the proof of knowledge is involved and postponed until Section 3.6. The verification algorithm consists simply of verifying the proof of knowledge contained in a signature.

Algorithm 10 (Verification, $\mathsf{HVf}(T, hpk, m, \sigma)$).
On input a candidate signature $\sigma = (c, \pi) = \big(\big((u_l, v_l)_{l=0}^{\delta-1},\ C_\delta,\ (\mathbf{u}, \mathbf{v}),\ (\mathbf{u}', \mathbf{v}'),\ \mathbf{C}\big), \pi\big)$ return the result of $V^{\mathsf{O}(m,\cdot)}(c, \pi)$.

To open a signature a group manager on depth l simply decrypts the lth component of the chain of cryptotexts contained in the signature.

Algorithm 11 (Open, $\mathsf{HOpen}(T, hpk, hsk(\beta), m, \sigma)$).
On input $\sigma = \big((u_l, v_l)_{l=0}^{\delta-1}, C_\delta, (\mathbf{u}, \mathbf{v}), (\mathbf{u}', \mathbf{v}'), \mathbf{C}, \pi\big)$ proceed as follows. If $\mathsf{HVf}(T, hpk, m, \sigma) = \perp$, return $\perp$. Otherwise compute $y_\alpha = D_{-x_\beta}(u_l, v_l)$. If $\alpha \in \beta$ return α and otherwise $\perp$.

In our construction we require that cryptotexts encrypted with *distinct* public keys are indistinguishable, since otherwise the cryptotexts themselves leak information on the identity of the signer. This property, which we call cross-indistinguishability, is formalized and characterized in Section 3.3. Note that semantic security does not imply cross-indistinguishability. It is not hard to see that ElGamal is cross-indistinguishable if we fix a group for each security parameter such that all cryptotexts cryptotexts for a security parameter are formed over this group.

It may seem that we have picked arbitrary primitives and then deferred the problem of forming the proof of knowledge needed. This is *not* the case. The primitives are carefully selected and slightly modified to allow a reasonably practical proof of knowledge.

Security Analysis

We analyze the security of our construction in the random oracle model, and prove that its security follows from the DDH assumption and the strong RSA assumption.

Theorem 3.5.8. *The hierarchical signature scheme $\mathcal{HGS}$ is secure under the DDH assumption and the strong RSA assumption in the random oracle model. Furthermore, hierarchical traceability holds under the strong RSA assumption.*

Proof. The proof proceeds by contradiction. Assume that the signature scheme $\mathcal{SS}^{\mathsf{cs}}_{H^{\mathsf{CHP}},H^{\mathsf{Sh}}}$ is secure. Then we show that if there exists an adversary A which breaks $\mathcal{HGS}$, there exists another adversary which executes A as a blackbox and breaks either the Cunningham chain DDH assumption, the security of $\mathcal{CS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}}$ or the security of $\mathcal{SS}^{\mathsf{cs}}_{H^{\mathsf{CHP}},H^{\mathsf{Sh}}}$. In view of Corollary 3.5.5 and Lemma 3.5.4 this gives a contradiction.

Breaking $\mathcal{HGS}$ means either breaking the hierarchical anonymity, i.e., succeeding in Experiment 3, or hierarchical traceability, i.e., succeeding in Experiment 4. The adversary A' simulates to A that it participates in one of these experiments, i.e., it simulates the random oracle O, the HKg oracle, the HOpen oracle, and the HSig oracle. We consider the details of the simulation of each experiment below, starting with the case where A breaks hierarchical anonymity.

Hierarchical Anonymity. Suppose that the attacker A breaks hierarchical anonymity. Then we have $\mathbf{Adv}^{\mathsf{anon}}_{\mathcal{HGS},A}(\kappa, T) \geq 1/\kappa^c$ for some polynomial size tree T, constant $c > 0$ and sufficiently large κ.

Definition of A'. As an intermediate step we define a machine A' that runs A as a blackbox. A' is the basis of the adversaries we construct. A' takes as input a single bit b and outputs a single bit. However, A' itself also plays the role of the adversary in a DDH experiment, a CCA2 experiment, and a CMA experiment, i.e., it plays the adversary in Experiment 11 over G_{q_3}, Experiment 1 with $\mathcal{CS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}}$, and Experiment 2 with $\mathcal{SS}^{\mathsf{cs}}_{H^{\mathsf{CHP}},H^{\mathsf{Sh}}}$. We reach a contradiction by proving that A' is too successful in at least one of these experiments.

We now describe how A' simulates the oracles to A. The $\mathsf{HKg}(\cdot)$ oracle is simulated as follows. A' first waits until it receives (g_3, D_1, D_2, D_3) in the DDH experiment. Step 1 is simulated honestly, except that A' uses the value of g_3 received from its oracle instead of generating it randomly. Step 2 is simulated honestly. Then A' waits until it receives a $\mathcal{CS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}}$ public key Y in the CCA2 experiment. In Step 3 it takes Y to be the public key, and X is never defined. Then A' waits until it receives a $\mathcal{SS}^{\mathsf{cs}}_{H^{\mathsf{CHP}},H^{\mathsf{Sh}}}$ public key $(N, \mathbf{h}, \mathbf{z}, e')$ in the CMA experiment. In Step 4 it uses these values, and t is never defined. Step 5 is simulated honestly. Next A' chooses two leaves $\beta^{(0)}_{\delta}$ and $\beta^{(1)}_{\delta}$ randomly. Intuitively, this is the two leaves A' guess that A will later ask to be challenged on. Let $\beta^{(0)}_{\delta}$, ..., $\beta^{(0)}_{t}$ and $\beta^{(1)}_{\delta}$, ..., $\beta^{(1)}_{t}$ be the paths to their common ancestor $\beta^{(0)}_{t} = \beta^{(1)}_{t}$. Step 6 is then simulated honestly except that for $\beta^{(0)}_{l}, \beta^{(1)}_{l}$, for $l = t, \ldots, \delta$, the public keys are instead defined using

(D_1, D_2, D_3) as follows

$$y_{\beta_l^{(0)}} = D_1^{x_{\beta_l^{(0)}}} , \quad y_{\beta_l^{(1)}} = D_1^{x_{\beta_l^{(1)}}} .$$

A' simulates Step 7 by requesting the $\mathsf{Sig}^{\mathsf{cs}}(\cdot, (H, N, \mathbf{h}, \mathbf{z}, e'), t)$ oracle in the CMA experiment to sign the message $(y_{\alpha_1}, \ldots, y_{\alpha_\delta})$ for each α. Then A' uses the answer, a $\mathcal{SS}^{\mathsf{cs}}$ signature $(e_\alpha, \boldsymbol{\sigma}_\alpha, \boldsymbol{\sigma}'_\alpha)$, to define $hsk(\alpha)$ properly. Step 8 is simulated honestly.

In each iteration of Experiment 3 A may ask for the private key of any inner node α. If $\alpha = \beta_l^{(0)}$ or $\beta_l^{(1)}$ for some $l = t, \ldots, \delta$, then A' is unable to satisfy the request of A properly, since it does not know $\log_{g_3} y_\alpha$. Instead of continuing the simulation, A' outputs a random bit $d \in \{0, 1\}$ and halts. Otherwise it hands x_α to A.

The $\mathsf{HOpen}(T, hpk, hsk(\cdot), \cdot, \cdot)$ oracle is simulated as follows given a query (α, m, σ), where α is on depth l. If $\mathsf{HVf}(T, hpk, m, \sigma) = 0$, return $\perp$. Otherwise assume that the signature is on the form $\sigma = \big((u_l, v_l)_{l=0}^{\delta-1}, C_\delta, (\mathbf{u}, \mathbf{v}), (\mathbf{u}', \mathbf{v}'), \mathbf{C}, \pi\big)$. A' hands C_δ to its $D_X^{\mathsf{cs}}(\cdot)$ oracle in the CCA2 experiment. If the answer is not on the form y_{α_δ} for some $\alpha_\delta \in \mathcal{L}(T)$, then the $\mathsf{HOpen}(T, hpk, hsk(\cdot), \cdot, \cdot)$ oracle is instructed to return $\perp$. This is probably an incorrect reply, which reveals to A that it is simulated. Intuitively this can only happen if A somehow is able to manufacture a $\mathcal{SS}^{\mathsf{cs}}$ signature. Suppose now that y_{α_δ} is on the expected form. Then there is a path $\rho = \alpha_0, \ldots, \alpha_\delta$ corresponding to α_δ. If $\alpha = \alpha_l$ for some $0 \leq l \leq \delta$, then the $\mathsf{HOpen}(T, hpk, hsk(\cdot), \cdot, \cdot)$ oracle is instructed to return α_{l+1} to A. Otherwise it returns $\perp$. Also in this case the reply may be incorrect if A can manufacture a $\mathcal{SS}^{\mathsf{cs}}$ signature. In the analysis below we show that if incorrect replies are a non-negligible event we reach a contradiction.

At some point A outputs $(s_{\text{state}}, \alpha^{(0)}, \alpha^{(1)}, m)$. If $\alpha^{(0)} \neq \beta_\delta^{(0)}$ or $\alpha^{(1)} \neq \beta_\delta^{(1)}$, then A' guessed incorrectly the challenge indices chosen by A. It outputs a random bit $d \in \{0, 1\}$ and halts in this case. Thus, we assume from this point on that $\alpha^{(0)} = \beta_\delta^{(0)}$ and $\alpha^{(1)} = \beta_\delta^{(1)}$.

The single query m to the $\mathsf{HSig}(T, hpk, hsk(\alpha^{(b)}, \cdot)$ oracle is simulated as follows. To simplify the exposition we write α_l instead of $\alpha_l^{(b)}$ as in Experiment 3. A' chooses $r_l, r'_l \in \mathbb{Z}_{q_3}$ and $\tau, \zeta, \tau', \zeta', \psi \in [0, 2^{\kappa_3} N - 1]$ randomly and computes

$$\begin{aligned}
&(u_l, v_l) = (y_{\alpha_l}^{r_l} D_3^{x_{\alpha_l} r'_l}, y_{\alpha_{l+1}} g_3^{r_l} D_2^{r'_l}), \quad \text{for } l = 0, \ldots, \delta - 1 \ , \\
&C_\delta = E_Y^{\mathsf{cs}}(y_{\alpha_\delta}, r) \ , \quad \text{and} \\
&(\mathbf{u}, \mathbf{v}) = \left(\mathbf{g}^\zeta, \mathbf{g}^\tau\right) , \quad (\mathbf{u}', \mathbf{v}') = \left(\mathbf{g}^{\zeta'}, \mathbf{g}^{\tau'}\right) , \quad \mathbf{C} = \mathbf{g}^\psi \ .
\end{aligned}$$

To construct the proof π, A' simply invokes the simulator for the proof of knowledge.

If (D_1, D_2, D_3) is a DDH triple we have $(D_1^{x_{\alpha_l}}, D_2, D_3^{x_{\alpha_l}}) = (y_{\alpha_l}, g_3^f, y_{\alpha_l}^f)$ for some f. If on the other hand (D_1, D_2, D_3) is not a DDH triple, then we have that $(D_1^{x_{\alpha_l}}, D_2, D_3^{x_{\alpha_l}}) = (y_{\alpha_l}, g_3^f, g_3^{f'} y_{\alpha_l}^f)$ for some random $f' \in \mathbb{Z}_{q_3}$. It follows that if (D_1, D_2, D_3) is a DDH triple, the ElGamal cryptotexts are identically distributed to the corresponding parts of a signature returned by the $\mathsf{Sig}^{\mathsf{cs}}(T,\ hpk,\ hsk(\alpha^{(b)}),\ \cdot)$ oracle in Experiment 3, and otherwise they are encryptions of random elements. Note that when (D_1, D_2, D_3) is not a DDH triple there are no a priori guarantees for how the output of the simulator of the proof of knowledge is distributed.

A' simulates A until it halts with output $d \in \{0,1\}$. Then A' halts with output d. This concludes the definition of the adversary A'.

Analysis of the behavior of A'. We must now analyze the output of A' and reach a contradiction. We divide our analysis into a number of claims.

Denote by d'_b the output of A' on input b. Let E_{noask} denote the event that A never asks for $x_{\alpha_l^{(0)}}$ or $x_{\alpha_l^{(1)}}$. Let E_{guess} denote the event that $\alpha^{(0)} = \beta_\delta^{(0)}$, $\alpha^{(1)} = \beta_\delta^{(1)}$, i.e., that A' guesses the challenge leaves correctly. Let E_{ddh} denote the event that (D_1, D_2, D_3) is a DDH triple.

Claim 2. $\Pr[\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa, T) = 1 \mid E_{\mathrm{noask}} \wedge E_{\mathrm{guess}}] = \Pr[\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa, T) = 1]$.

Proof. The variables $\beta^{(0)}$ and $\beta^{(1)}$ are independent from $\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa, T)$. Thus,

$$\Pr[\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa, T) = 1 \mid E_{\mathrm{guess}}] = \Pr[\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa, T) = 1] \ .$$

Experiment 3 is defined such that it returns 0 if the event $\overline{E_{\mathrm{noask}}}$ occurs. This implies that

$$\Pr[\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa, T) = 1 \mid \overline{E_{\mathrm{noask}}}] = 0 \ .$$

Thus

$$\Pr[\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa, T) = 1 \mid E_{\mathrm{noask}}] = \Pr[\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa, T) = 1] \ .$$

The claim follows. □

Claim 3. *There exists a negligible function $\epsilon(\kappa)$, such that for $b \in \{0,1\}$*

$$\begin{aligned} &|\Pr[d'_b = 1 \mid E_{\mathrm{noask}} \wedge E_{\mathrm{guess}} \wedge E_{\mathrm{ddh}}] \\ &\quad - \Pr[\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa, T) = 1 \mid E_{\mathrm{noask}} \wedge E_{\mathrm{guess}}]| < \epsilon(\kappa) \ . \end{aligned}$$

Proof. The only part of how A' simulates the experiment to A that is not identically distributed to the corresponding part in Experiment 3 is the simulation of the $\mathsf{HOpen}(T, hpk, hsk(\cdot), \cdot, \cdot)$ oracle. Differently phrased, the only way that A can distinguish between being simulated by A' and actually run in Experiment 3 is by asking the $\mathsf{HOpen}(T, hpk, hsk(\cdot), \cdot, \cdot)$ oracle a question that it fails to answer correctly. We show that the oracle answers all questions correctly with overwhelming probability.

Let $p(\kappa)$ denote the running time of A. Then it follows that A asks the simulated $\mathsf{HOpen}(T,\ hpk,\ hsk(\cdot),\ \cdot,\ \cdot)$ oracle at most $p(\kappa)$ queries (α, m, σ) such that $\mathsf{HVf}(T, hpk, m, \sigma) = 1$. Denote by T_l the machine that simulates A' until $l - 1$ queries have been answered by the simulated $\mathsf{HOpen}(T, hpk, hsk(\cdot), \cdot, \cdot)$ oracle, and then halts outputting the lth query. We say that a query is *difficult* if it is answered incorrectly by A'.

We must show for $l = 0, \ldots, p(k)$ that T_l outputs a difficult query with negligible probability.

The statement is clearly true for T_0, since its output is empty. Suppose now that the statement is true for T_l for $l < s$, but false for T_s. Then the distribution of the output of T_s is indistinguishable from the distribution of the sth query A asks its $\mathsf{HOpen}(T, hpk, hsk(\cdot), \cdot, \cdot)$ oracle. This follows from the union bound, since the lth question is incorrectly answered with negligible probability for $l < s$. Thus, T_s outputs a difficult query (α, m, σ) with probability κ^{-c_1} for some constant c_1 (and large enough κ).

Intuitively, it is clear that we can extract a signature from T_s. Formally, we invoke Lemma 3.6.4 (in some sense a weak "Forking Lemma" [57]) and conclude that there exists a polynomial Turing machine T_s' that runs T_s as a black-box (simulating the random oracle) and outputs with probability $1/(k^{c_1}p(\kappa))^3$ a message $(\gamma_1, \ldots, \gamma_\delta)$ and a $\mathcal{SS}^{\mathsf{cs}}$ signature $(\varepsilon, \boldsymbol{\sigma}, \boldsymbol{\sigma}')$ such that

$$\mathsf{Vf}^{\mathsf{cs}}(H, N, \mathbf{h}, \mathbf{z}, e', (\gamma_1, \ldots, \gamma_\delta), (\varepsilon, \boldsymbol{\sigma}, \boldsymbol{\sigma}')) = 1 \ ,$$

with the added property that the query was difficult.

A query is difficult precisely when A' has not asked its $\mathsf{Sig}^{\mathsf{cs}}(\cdot, (H, N, \mathbf{h}, \mathbf{z}, e'), t)$ oracle the query $(\gamma_1, \ldots, \gamma_\delta)$. This implies that T_s' breaks the CMA-security of $\mathcal{SS}^{\mathsf{cs}}$, which is a contradiction.

Thus, we conclude that A asks a difficult query with negligible probability. □

Claim 4. *There exists a negligible function $\epsilon(\kappa)'$ such that*

$$\begin{aligned} &|\Pr[d_0' = 1 \mid E_{\text{guess}} \wedge E_{\text{noask}} \wedge \overline{E_{\text{ddh}}}] \\ &\quad - \Pr[d_1' = 1 \mid E_{\text{guess}} \wedge E_{\text{noask}} \wedge \overline{E_{\text{ddh}}}]| < \epsilon'(\kappa) \ . \end{aligned}$$

Proof. Suppose that the claim is false, i.e., that the left side of the equation is greater or equal to κ^{-c_2} for some constant c_2. The only part of the simulation of A' that differs between the two cases is how C_δ is formed. This implies that we can break the security of $\mathcal{CS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}}$ as follows.

Let A'' be the adversary that is identical to A' except for the following modifications. It receives no input, it simulates the DDH experiment by handing itself a random tuple $(g_3, D_1, D_2, D_3) \in G_{q_3}$, and it simulates the CMA experiment to itself honestly. The CCA2 experiment is not simulated, i.e., A'' plays the role of the adversary in a CCA2 experiment. When A' would construct C_δ as described above, A'' instead requests an encryption of $y_{\alpha_\delta^{(0)}}$ or $y_{\alpha_\delta^{(1)}}$ from the CCA2 experiment. The experiment hands back a $\mathcal{CS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}}$ cryptotext $C_\delta' = E_Y^{\mathsf{cs}}(y_{\alpha_\delta^{(b)}})$ for a randomly chosen $b \in \{0, 1\}$. Instead of generating C_δ by itself, A'' continues with $C_\delta = C_\delta'$. Let $d_b'' = \mathbf{Exp}^{\mathsf{cca}-b}_{\mathcal{CS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}}, A''}(\kappa)$. By construction d_b'' is identically distributed to the conditioned variable $(d'_b \mid \overline{E_{\mathrm{ddh}}})$. Thus

$$\begin{aligned}
&|\Pr[\mathbf{Exp}^{\mathsf{cca}-0}_{\mathcal{CS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}},A''}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{cca}-1}_{\mathcal{CS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}},A''}(\kappa) = 1]| \\
&= |\Pr[d''_0 = 1 \mid E_{\mathrm{guess}} \wedge E_{\mathrm{noask}}] + \Pr[d''_0 = 1 \mid \overline{E_{\mathrm{guess}}} \vee \overline{E_{\mathrm{noask}}}] \\
&\quad -(\Pr[d''_1 = 1 \mid E_{\mathrm{guess}} \wedge E_{\mathrm{noask}}] + \Pr[d''_1 = 1 \mid \overline{E_{\mathrm{guess}}} \vee \overline{E_{\mathrm{noask}}}])| \\
&= |\Pr[d''_0 = 1 \mid E_{\mathrm{guess}} \vee E_{\mathrm{noask}}] - \Pr[d''_1 = 1 \mid E_{\mathrm{guess}} \vee E_{\mathrm{noask}}]| \\
&= |\Pr[d'_0 = 1 \mid E_{\mathrm{guess}} \wedge E_{\mathrm{noask}} \wedge \overline{E_{\mathrm{ddh}}}] - \Pr[d'_1 = 1 \mid E_{\mathrm{guess}} \wedge E_{\mathrm{noask}} \wedge \overline{E_{\mathrm{ddh}}}]| \\
&\geq \kappa^{-c_2}
\end{aligned}$$

where the next to last equality follows from the fact that A' outputs a random bit when the event $\overline{E_{\mathrm{guess}}} \vee \overline{E_{\mathrm{noask}}}$ takes place. The resulting inequality contradicts Lemma 3.5.4 and Lemma 3.5.2. □

Claim 5. *There exists a* $b \in \{0,1\}$ *and a constant* $c_3 > 0$ *such that*

$$\kappa^{-c_3} \leq |\Pr[d'_b = 1 \mid E_{\mathrm{ddh}}] - \Pr[d'_b = 1 \mid \overline{E_{\mathrm{ddh}}}]| \ .$$

Proof. Write $p_{b_0,b_1} = \Pr[d'_{b_0} = 1 \mid E_{\mathrm{noask}} \wedge E_{\mathrm{guess}} \wedge E^{b_1}_{\mathrm{ddh}}]$, where we understand $E^{b_1}_{\mathrm{ddh}}$ to be $\overline{E_{\mathrm{ddh}}}$ or E_{ddh} depending on if $b_1 = 0$ or 1. Without loss we assume that $\epsilon(\kappa) = \epsilon'(\kappa)$. Then we have

$$\begin{aligned}
\kappa^{-c} &\leq |\Pr[\mathbf{Exp}^{\mathsf{anon}-0}_{\mathcal{HGS},A}(\kappa,T) = 1] - \Pr[\mathbf{Exp}^{\mathsf{anon}-1}_{\mathcal{HGS},A}(\kappa,T) = 1]| \\
&= |\Pr[\mathbf{Exp}^{\mathsf{anon}-0}_{\mathcal{HGS},A}(\kappa,T) = 1 \mid E_{\mathrm{noask}} \wedge E_{\mathrm{guess}}] \\
&\quad - \Pr[\mathbf{Exp}^{\mathsf{anon}-1}_{\mathcal{HGS},A}(\kappa,T) = 1 \mid E_{\mathrm{noask}} \wedge E_{\mathrm{guess}}]| \\
&\leq |\Pr[d'_0 = 1 \mid E_{\mathrm{noask}} \wedge E_{\mathrm{guess}} \wedge E_{\mathrm{ddh}}] \\
&\quad\quad - \Pr[d'_1 = 1 \mid E_{\mathrm{noask}} \wedge E_{\mathrm{guess}} \wedge E_{\mathrm{ddh}}]| + 2\epsilon(\kappa) \\
&\leq |p_{0,1} - p_{1,1}| + 2\epsilon(\kappa) \leq |p_{0,1} - p_{0,0}| + |p_{0,0} - p_{1,1}| + 2\epsilon(\kappa) \\
&\leq |p_{0,1} - p_{0,0}| + |p_{1,0} - p_{1,1}| + 3\epsilon(\kappa)
\end{aligned}$$

from which it follows that there exists a fixed $b_0 \in \{0,1\}$ and c_3 such that the claim holds. The first inequality holds by assumption, the equality follows from Claim 2, the second inequality follows from Claim 3, and the last inequality follows from Claim 4. □

Claim 6. *The DDH assumption is broken.*

Proof. Let A''' be identical to A' except from the following modifications. The input b is fixed to the value guaranteed to exist by Claim 5, so it takes no input. It simulates the CCA2 experiment and the CMA experiment to itself. It does not simulate the DDH experiment, i.e., A''' plays the role of the adversary in a DDH experiment. Then we have

$$|\mathbf{Exp}^{\mathsf{ddh}-0}_{G_{q_3},A'''}(\kappa) - \mathbf{Exp}^{\mathsf{ddh}-1}_{G_{q_3},A'''}(\kappa)| = |\Pr[d'_b = 1 \mid \overline{E_{\mathrm{ddh}}}] - \Pr[d'_b = 1 \mid E_{\mathrm{ddh}}]| \ .$$

Combined with Claim 5 this contradicts the DDH assumption over G_{q_3}. □

We now turn to the case when A breaks the hierarchical traceability and describe how A' is implemented in this case.

Hierarchical Traceability. Suppose that A breaks hierarchical traceability. Then

$$\mathbf{Adv}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa, T) \geq 1/k^c$$

for some polynomial size tree T, constant $c > 0$ and sufficiently large k. We construct a machine A' that runs A as a blackbox and breaks the security of $\mathcal{SS}^{\mathsf{cs}}$.

Definition of A'. A' takes no input, but it plays the role of the adversary in the CMA experiment.

A' simulates the HKg oracle to A as follows. Steps 1, 2 and 3 are simulated honestly. Then it waits until it receives a public key $(N, \mathbf{h}, \mathbf{x}, e')$ in the CMA experiment. In Step 4, $(N, \mathbf{h}, \mathbf{x}, e')$ is used instead of generating a $\mathcal{SS}^{\mathsf{cs}}$ key pair, i.e., the private key t is never defined. Step 5 and Step 6 are simulated honestly. Step 7 is not performed at all. Step 8 is simulated honestly.

In each iteration in the loop of the experiment simulated to A, A may request $hsk(\alpha)$. Given such a request A' requests from its own signature oracle $\mathsf{Sig}^{\mathsf{cs}}(\cdot, H, N, \mathbf{h}, \mathbf{z}, e', t)$ a signature $(e_\alpha, \boldsymbol{\sigma}_\alpha, \boldsymbol{\sigma}'_\alpha)$ of the message $(y_{\alpha_1}, \ldots, y_{\alpha_\delta})$, where $\rho = \alpha_0, \alpha_1, \ldots, \alpha_\delta = \alpha$ is the path corresponding to α.

Queries to the HSig oracle are simulated as follows. The first step is simulated honestly. In Step 2 $(\mathbf{u}, \mathbf{v})$, $(\mathbf{u}', \mathbf{v}')$ and $\mathbf{C}$ are replaced by $(\mathbf{g}^\zeta, \mathbf{g}^\tau)$, $(\mathbf{g}^{\zeta'}, \mathbf{g}^{\tau'})$ and $\boldsymbol{g}^\psi$ respectively. In Step 3, the simulator of the proof of knowledge is invoked to construct π. The resulting signature is identically distributed to the reply of the $\mathsf{Sig}^{\mathsf{cs}}$ oracle in Experiment 4.

At some point A halts with output (m, σ), where the signature is given by $\sigma = \big((u_l, v_l)_{l=0}^{\delta-1}, C_\delta, (\mathbf{u}, \mathbf{v}), (\mathbf{u}', \mathbf{v}'), \mathbf{C}, \pi\big)$. Then A' computes $\mathsf{HVf}(T, hpk, m, \sigma)$. If the result is 0, it outputs $\perp$.

Suppose that the running time of A' is bounded by $p(\kappa)$ for a polynomial κ. We invoke Lemma 3.6.4 (in some sense a weak "Forking Lemma" [57]) and conclude that there exists a polynomial Turing machine T' that runs A' as a black-box (simulating the random oracle) and outputs with probability $1/(k^c p(\kappa))^3$ a message $(\gamma_1, \ldots, \gamma_\delta)$ and a $\mathcal{SS}^{\mathsf{cs}}$ signature $(\varepsilon, \boldsymbol{\sigma}, \boldsymbol{\sigma}')$ such that

$$\mathsf{Vf}^{\mathsf{cs}}(H, N, \mathbf{h}, \mathbf{z}, e', (\gamma_1, \ldots, \gamma_\delta), (\varepsilon, \boldsymbol{\sigma}, \boldsymbol{\sigma}')) = 1 \ .$$

Claim 7. *The security of $\mathcal{SS}^{\mathsf{cs}}$ is broken.*

Proof. Consider the list $\gamma = (\gamma_1, \ldots, \gamma_\delta)$. A succeeds in Experiment 4 whenever γ does not correspond to a path in T, or γ corresponds to a path, but $\gamma_\delta \notin \mathcal{C}$. In the first case A' clearly succeeds as well since it never asks its $\mathsf{Sig}^{\mathsf{cs}}$ oracle to sign any message that is not a list of public keys corresponding to a path in T. In the second case A' succeeds since by construction, if $\gamma_\delta \notin \mathcal{C}$ it never asked its $\mathsf{Sig}^{\mathsf{cs}}$ oracle to sign the message γ. Thus, A' succeeds whenever A succeeds, and we have $\mathbf{Exp}^{\mathsf{trace}}_{\mathcal{SS}^{\mathsf{cs}}}(\kappa, \emptyset) \geq \mathbf{Exp}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa, T) \geq 1/\kappa^c$ which contradicts Lemma 3.5.5 and Lemma 3.5.2. □

Conclusion of Proof. If $\mathbf{Adv}^{\mathsf{anon}}_{\mathcal{HGS},A}(\kappa, T)$ is not negligible, Claim 6 gives a contradiction, and if $\mathbf{Adv}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa, T)$ is not negligible Claim 7 gives a contradiction. Thus, the theorem is true. □

As noted above, hierarchical anonymity as defined here is a proper generalization of full anonymity as defined in [7], and our scheme can be used as an ordinary (non-hierarchical) group signature scheme by setting the depth of the tree equal to two. Thus our scheme is fully anonymous.

The definition of full anonymity is stronger than previously considered anonymity definitions, since the adversary is allowed to use an Open oracle during the attack. In our case we can handle it since we use a CCA2-secure encryption scheme, and thus reach a contradiction if the adversary is able to form a signature on his own that we cannot answer.

It is shown in [7] that it is necessary to use a CCA2-secure cryptosystem to form a group signature scheme. Still we only use a CCA2-secure cryptosystem for the leaves. This apparent contradiction is resolved by noting that since the public keys y_α are distinct, and we may identify the leaves with their paths in the tree, any query to the $\mathsf{HOpen}(T, hpk, hsk(\cdot), \cdot, \cdot)$ oracle for intermediate levels of the tree can be answered using a single query to the decryption oracle for the cryptosystem used to encrypt leaves.

3.6 Construction of the Proof of Knowledge

We give honest verifier zero-knowledge public coin proofs of knowledge for a number of subprotocols which combined gives the proof of knowledge we need to apply the Fiat-Shamir heuristic to get a signature scheme in the random oracle model. Our protocols are based on a variety of proof techniques including, e.g., proofs of knowledge of exponents, double decker exponentiation, equality of exponents over distinct groups, interval proofs, and equality of exponents over an RSA modulus.

We divide the exposition into a number of subsections. First we describe proofs of knowledge over the groups G_{q_i}. Then we give a proof of equality of exponents over distinct groups. This is followed by the proofs over an RSA modulus $\mathbb{Z}_N$. Finally, the combined protocol is described.

The protocol can be seen as consisting of three steps. In the first step a value ν is shown to be a commitment to the hash of the encrypted public keys along the chain. In the second step it is shown that this value (which is in G_{q_3}) is equal (over $\mathbb{Z}$) to a value in $\mathbb{Z}_N$ hidden in a commitment $\mathbf{C}'$. Finally in the third step the Cramer-Shoup signature the prover has committed to is shown to be a valid signature of the value committed to in $\mathbf{C}'$. This is shown schematically in Figure 3.5.

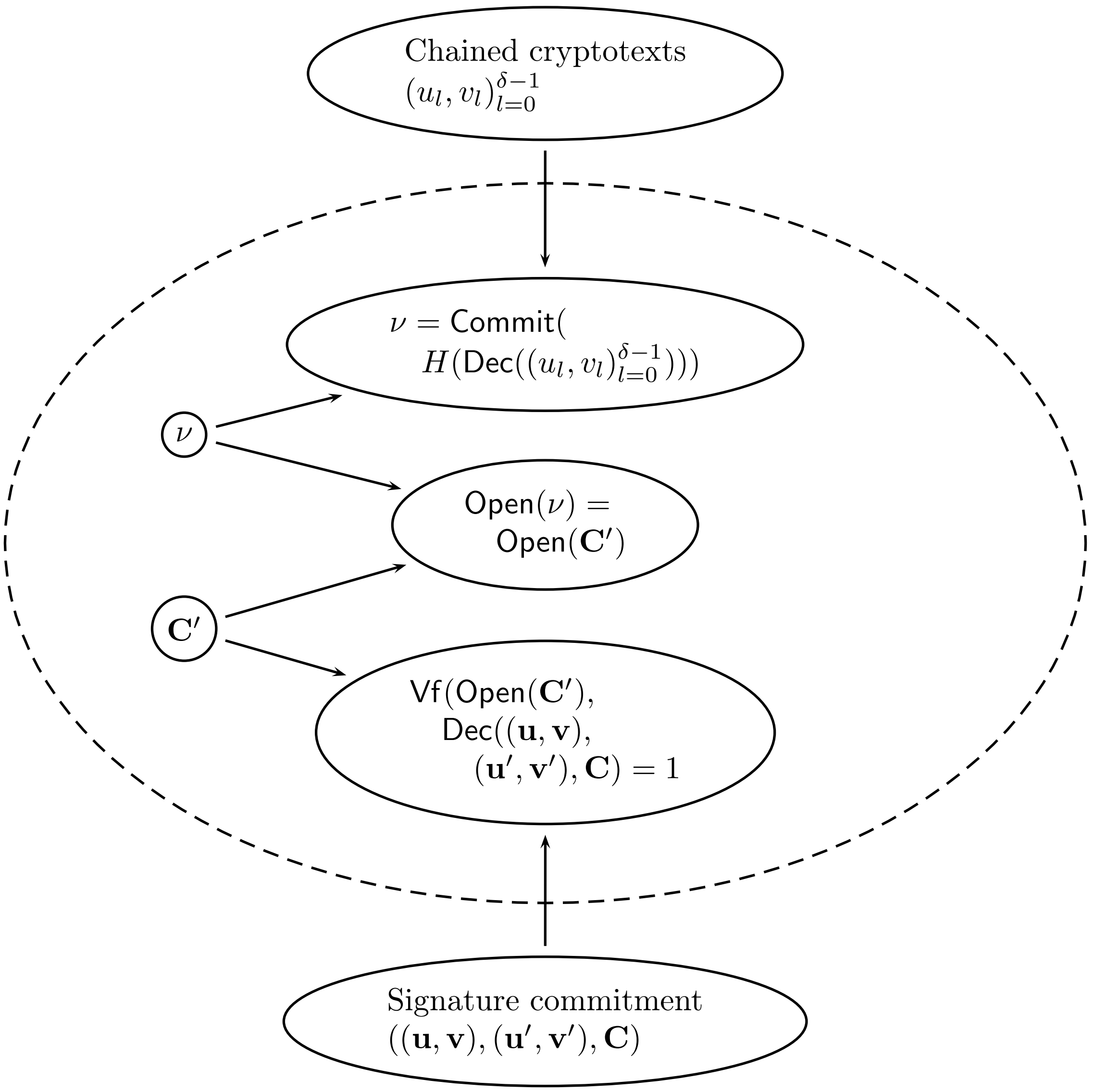

Chained cryptotexts
$(u_l, v_l)_{l=0}^{\delta-1}$
$\nu = \mathsf{Commit}($
$H(\mathsf{Dec}((u_l, v_l)_{l=0}^{\delta-1})))$
ν
$\mathsf{Open}(\nu) =$
$\mathsf{Open}(\mathbf{C}')$
$\mathbf{C}'$
$\mathsf{Vf}(\mathsf{Open}(\mathbf{C}'),$
$\mathsf{Dec}((\mathbf{u}, \mathbf{v}),$
$(\mathbf{u}', \mathbf{v}'), \mathbf{C}) = 1$
Signature commitment
$((\mathbf{u}, \mathbf{v}), (\mathbf{u}', \mathbf{v}'), \mathbf{C})$

Figure 3.5: Schematic overview of the proof of knowledge. The functions Commit and Open represent creation and opening of commitments.

Although we focus on efficiency, in some cases we have chosen to divide the protocol into subprotocols for clarity, thus sacrificing some efficiency. Since the by far most time-consuming part of the protocol is the proofs of exponential relations, where to our knowledge the best current method is based on cut-and-choose, saving a few exponentiations in other parts of the protocol yields little in terms of overall performance.

Before we start we recall some definitions.

Review of Some Notions

We assume familiarity with the notion of interactive proofs (IP) introduced by Goldwasser, Micali and Rackoff [35], and public-coin IPs called Arthur-Merlin games introduced by Babai [5]. The notion of zero-knowledge was introduced by Goldwasser, Micali and Rackoff [35]. Statistical special zero-knowledge and special soundness is defined below.

Let $\mathrm{view}_P^V(x)$ denote the view of the verifier V (including its random string) when interacting with the prover P on common input x. Note that the view of the verifier V in a three-move protocol can be written (r, x, α, c, d), where r is the random input of V, x is the common input, α is the first message sent by P, c is the random challenge from a set C sent by V, and d is final message sent by P.

Definition 3.6.1 (Statistical Special Honest Verifier Zero-Knowledge). *Let (P, V) be a three-move IP for a language L. We say that (P, V) is* statistical special honest verifier zero-knowledge *proof (HVZKP) if there exists a probabilistic polynomial time algorithm S such that the ensembles $\{S(x, c)\}_{x \in L, c \in C}$ and $\{\mathrm{view}_P^V(x)\}_{x \in L, c \in C}$ are statistically close as functions of $|x|$.*

The term *special* is used since the simulator S is not allowed to pick the challenge c itself, but must be able to compute a valid view when given c together with x as input.

Suppose the challenge $c = (c_1, \ldots, c_k)$ is randomly chosen from a product set $C_1 \times C_2 \times \cdots \times C_k$ for some constant k and that $|C_i| \geq 2^\kappa$ for $i = 1, \ldots, k$ where κ is the security parameter. Then the following slight generalization of special soundness makes sense. We get the standard definition of special-sound if $k = 1$.

Definition 3.6.2 (Special Soundness). *A three-move IP (P, V) for a binary relation R is $C_1 \times C_2 \times \cdots \times C_k$-*special-sound *if there exists a polynomial time algorithm that given two accepting outcomes of the view (r, x, α, c, d) and (r, x, α, c', d') with $c_i \neq c'_i$ for all $i = 1, \ldots, k$, outputs a witness w such that $(x, w) \in R$.*

We use a generalized definition of Σ-protocol along the lines suggested by Cramer, Damgård, and Schoenmakers [23].

Definition 3.6.3 (Σ-Protocol). *A $C_1 \times C_2 \times \cdots \times C_k$-$\Sigma$-*protocol *is a three-move interactive proof (P, V) that is both statistical special honest verifier zero-knowledge and $C_1 \times C_2 \times \cdots \times C_k$-special-sound for some product set.*

Observation 1. *Let (P_l, V_l) be a $C_1 \times C_2 \times \cdots \times C_k$-$\Sigma$-protocol for a language L_l for $l = 1, \ldots, k(\kappa)$, where $k(\kappa)$ is polynomial. Then the parallel composition (P, V) of the protocols where a* single *challenge in C is used for* all *protocols is a C-Σ-protocol.*

Observation 2. *Let (P_l, V_l) be a C_l-Σ-protocol for a language L_l for $l = 1, \ldots, k$, where k is constant. Then the parallel composition (P, V) of the protocols is a $C_1 \times C_2 \times \cdots \times C_k$-$\Sigma$-protocol.*

It essentially follows from the "Forking Lemma" of Pointcheval and Stern [57] that a Σ-protocol is a proof of knowledge in the sense of [6]. One must only generalize the lemma slightly to hold also for the generalized special soundness as we define it above. However, we only need the following lemma in the proof of Theorem 3.5.8.

Lemma 3.6.4. *Let (P, V) be a $C_1 \times C_2 \times \cdots \times C_k$-$\Sigma$-protocol for a language L. Let $L' \subset L$. If A^{O} is a probabilistic random oracle machine running in polynomial time T such that*

$$\Pr[A^{\mathsf{O}} = (m, \alpha, c, d) \wedge \alpha \in L' \wedge V^{\mathsf{O}(m,\cdot)}(\alpha, c, d) = 1] \geq p$$

where the probability is taken over the random choices of A^{O} and the random oracle, and $1/p$ is polynomial, then there exists a polynomial time machine A' running A as a blackbox (and simulating the random oracle), that outputs $(\alpha, w) \in R_L$ such that $\alpha \in L'$ with probability at least $(p/4T)^3 - \epsilon(\kappa)$, where $\epsilon(\kappa)$ is a negligible function.

Proof. Let $C = C_1 \times C_2 \times \cdots \times C_k$. First note that $V^{\mathsf{O}(m,\cdot)}(\alpha, c, d) = 1$ implies that $\mathsf{O}(m, \alpha) = c$. The machine A asks at most T queries to O. Thus, we may identify O with a list of random answers $(r_1, \ldots, r_T)$, where $r_i \in C$. Set $r = (r_0, \ldots, r_T)$ and write $A_r = (m_r, \alpha_r, c_r, d_r)$ to denote the output of A on internal randomness $r_0 \in \{0,1\}^T$ and using the oracle O defined by $(r_1, \ldots, r_T)$. Without loss we assume that A has queried O on (m_r, α_r) at some point.

We define A'_j as follows, where j is defined below. It chooses $r \in \{0,1\}^T \times C^T$ and $r' \in \{0,1\}^T \times C^T$ randomly under the restriction $(r_0, r_1, \ldots, r_{j-1}) = (r'_0, r'_1, \ldots, r'_{j-1})$ and executes A twice, the first time with r as random oracle and the second time with r' as random oracle. Then it invokes the extractor guaranteed by the Σ-protocol on (α_r, c_r, d_r) and $(\alpha_{r'}, c_{r'}, d_{r'})$ and outputs the result. Note that the extractor outputs a correct result only if $m_r = m_{r'}$, $\alpha_r = \alpha_{r'}$ and $c_r \neq c_{r'}$. We now show that this happens with non-negligible probability.

Denote by E_r the event $\alpha_r \in L' \wedge V^{\mathsf{O}(m_r,\cdot)}(\alpha_r, c_r, d_r) = 1$. Then we have $\Pr[E_r] = \sum_{l=1}^{T} \Pr[E_r \wedge c_r = r_l]$, which implies that there exists a j (used in the algorithm above) such that $\Pr[E_r \wedge c_r = r_j] \geq p/T$.

Define

$$F = \{(t_0, \ldots, t_{j-1}) \mid \Pr[E_r \wedge c_r = r_j \mid (r_0, \ldots, r_{j-1}) = (t_0, \ldots, t_{j-1})] \geq p/(2T)\}$$

and let $r^j = (r_0, r_1, \ldots, r_{j-1})$. Then $\Pr[r^j \in F] \geq p/(2T)$ by an average argument, and we have from independence that

$$\begin{aligned} & \Pr[(E_r \wedge c_r = r_j) \wedge (E_{r'} \wedge c_{r'} = r'_j)] \\ \geq\ & \Pr[(E_r \wedge c_r = r_j) \wedge (E_{r'} \wedge c_{r'} = r'_j) \mid r^j \in F] \Pr[r^j \in F] \\ =\ & \Pr[E_r \wedge c_r = r_j \mid r \in F] \Pr[E_{r'} \wedge c_{r'} = r'_j) \mid r^j \in F] \Pr[r^j \in F] \\ \geq\ & \left(\frac{p}{4T}\right)^3 \end{aligned}$$

Denote by E the event $(E_r \wedge c_r = r_j) \wedge (E_{r'} \wedge c_{r'} = r'_j)$. Denote by E_s the event that $\forall l \in [1,k] : r_{j,l} \neq r'_{j,l}$. This corresponds to fulfilling the hypothesis of the extractor guaranteed by the special soundness. Denote by E_u the event that all r_l and r'_l for $l = 1, \ldots, T$ are unique. The event E_u ensures that answer to the j'th query is used as challenge c, which in turn means that M must have decided on a m and α at that point. Given that the event E occurs this implies $(m_r, \alpha_r) = (m_{r'}, \alpha_{r'})$. We clearly have $\Pr[r_{j,l} = r'_{j,l}] = 1/|C_l|$, thus $\Pr[\overline{E_s}] \leq \sum_{l=1}^{k} 1/|C_l|$ by the union bound. We also have $\Pr[r_l = r'_l] = 1/|C|$ and the union bound gives $\Pr[\overline{E_u}] \leq T^2/|C|$. A final application of the union bound gives $\Pr[\overline{E_s} \vee \overline{E_u}] \leq \sum_{l=1}^{k} 1/|C_l| + T^2/|C|$.

We have

$$\begin{aligned} \Pr[E] &= \Pr[E \wedge E_s \wedge E_u] + \Pr\left[E \wedge \left(\overline{E_s} \vee \overline{E_u}\right)\right] \\ &\leq \Pr[E \wedge E_s \wedge E_u] + \sum_{l=1}^{k} 1/|C_l| + T^2/|C| \\ &\leq \Pr[E \wedge E_s \wedge E_u] + k/2^{\kappa} + T^2/2^{k\kappa} \ . \end{aligned}$$

It now follows that

$$\Pr[E \wedge E_s \wedge E_u] \geq \Pr[E] - \left(k/2^{\kappa} + T^2/2^{k\kappa}\right) \geq (p/4T)^3 - \left(k/2^{\kappa} + T^2/2^{k\kappa}\right)$$

which concludes the proof, since if the event $E \wedge E_s \wedge E_u$ occurs the extractor is guaranteed to succeed. □

Proofs of Knowledge in Groups of Known Prime Order

The goal of this section is to provide subprotocols that can be used to prove knowledge of $\gamma_1, \ldots, \gamma_\delta$ and $\tau_0, \ldots, \tau_{\delta-1}$ satisfying the relations that are defined exclusively over G_{q_1}, G_{q_2}, and G_{q_3} in Step 3 in Algorithm 9. Relations involving elements over $\mathbb{Z}_N$ are handled in Section 3.6 and Section 3.6. Most of the ideas we use in this section have appeared in various forms in the literature.

In some protocols we use the security parameters κ_1, κ_2 and κ_3. Where used the completeness depends on κ_1, the soundness depends on κ_2 and the amount of information disclosed depends on κ_3.

We begin our program by considering a problem related to that of proving that a list of cryptotexts is chained properly. To simplify the exposition the DDH assumption and strong RSA assumption are implicitly assumed in the formulation of the lemmas, and the bases in the common input are assumed to be chosen at random. In the application of the protocols this is the case.

Protocol 1 (Chained Cryptotexts).
Common Input: $y_0, g, y \in G_q$ *and* $\left((u_l, v_l), (\mu_l, \nu_l)\right)_{l=0}^{\delta-1} \in G_q^{4\delta}$
Private Input: $r_l, s_l, t_l \in \mathbb{Z}_q$ *for* $l = 0, \ldots, \delta - 1$ *and* $y_l \in G_q$ *for* $l = 1, \ldots, \delta$ *such that* $(u_l, v_l) = E_{(y_l, g)}(y_{l+1}, r_l) = (y_l^{r_l}, g^{r_l} y_{l+1})$ *and* $(\mu_l, \nu_l) = (g^{s_l} y^{t_l}, y^{s_l} y_{l+1})$.

1. *The prover chooses* $a_l \in \mathbb{Z}_q$ *randomly and computes*

$$A_{1,l} = g^{a_l}\mu_l^{r_l}, \quad A_{2,l} = y^{a_l}\nu_l^{r_l} \tag{3.1}$$

for $l = 0, \ldots, \delta - 1$.

2. *The prover chooses* $b_l, e_l, f_l, h_l, i_l, j_l \in \mathbb{Z}_q$ *randomly and computes*

$$B_0 = y_0^{e_0} \ , \tag{3.2}$$

for $l = 0, \ldots, \delta - 1$

$$B_{1,l} = g^{b_l}\mu_l^{e_l}, \quad B_{2,l} = y^{b_l}\nu_l^{e_l} \tag{3.3}$$

$$B_{3,l} = g^{e_l}y^{i_l}, \quad B_{4,l} = g^{i_l}y^{j_l} \ , \tag{3.4}$$

and for $l = 0, \ldots, \delta - 2$

$$B_{5,l} = g^{f_l}y^{h_l}, \quad B_{6,l} = y^{f_l} \tag{3.5}$$

and hands $B_0, \left(A_{1,l}, A_{2,l}, B_{1,l}, B_{2,l}, B_{3,l}, B_{4,l}\right)_{l=0}^{\delta-1}, \left(B_{5,l}, B_{6,l}\right)_{l=0}^{\delta-2}$ *to the verifier.*

3. *The verifier chooses* $c \in \mathbb{Z}_q$ *randomly and hands* c *to the prover.*

4. *The prover computes*

$$d_{1,l} = ca_l + b_l, \quad d_{2,l} = cr_l + e_l, \tag{3.6}$$

$$d_{3,l} = -cs_l + i_l, \quad d_{4,l} = -ct_l + j_l \tag{3.7}$$

for $l = 0, \ldots, \delta - 1$ *and for* $l = 0, \ldots, \delta - 2$

$$d_{5,l} = c\left(a_l + s_l r_l\right) + f_l, \quad d_{6,l} = ct_l r_l + h_l \tag{3.8}$$

and hands $(d_{1,l}, d_{2,l}, d_{3,l}, d_{4,l})_{l=0}^{\delta-1}, (d_{5,l}, d_{6,l})_{l=0}^{\delta-2}$ *to the verifier.*

5. *The verifier checks that*

$$u_0^c B_0 = y_0^{d_{2,0}} \ , \tag{3.9}$$

for $l = 0, \ldots, \delta - 1$ *that*

$$A_{1,l}^c B_{1,l} = g^{d_{1,l}}\mu_l^{d_{2,l}}, \quad A_{2,l}^c B_{2,l} = y^{d_{1,l}}\nu_l^{d_{2,l}} \tag{3.10}$$

$$(v_l/\nu_l)^c B_{3,l} = g^{d_{2,l}}y^{d_{3,l}}, \quad B_{4,l} = \mu_l^c g^{d_{3,l}}y^{d_{4,l}} \ . \tag{3.11}$$

and finally for $l = 0, \ldots, \delta - 2$ *that*

$$A_{1,l}^c B_{5,l} = g^{d_{5,l}}y^{d_{6,l}}, \quad (A_{2,l}/u_{l+1})^c B_{6,l} = y^{d_{5,l}} \ , \tag{3.12}$$

$$\begin{tikzcd}
(u_0, v_0) \arrow[d, leftrightarrow] & (u_1, v_1) \arrow[d, leftrightarrow] & \cdots & (u_{\delta-1}, v_{\delta-1}) \arrow[d, leftrightarrow] \\
(\mu_0, \nu_0) \arrow[ur, leftrightarrow] & (\mu_1, \nu_1) \arrow[ur, leftrightarrow] & \cdots \arrow[ur, leftrightarrow] & (\mu_{\delta-1}, \nu_{\delta-1})
\end{tikzcd}$$

Figure 3.6: Overview of Protocol 1.

Intuitively the proof works by first showing that (u_l, v_l) encrypt the key that is committed to in (μ_l, ν_l) and then by showing that key in the commitment (μ_l, ν_l) is the encryption key used for the encryption in (u_{l+1}, v_{l+1}). This concept is depicted in Figure 3.6.

The idea here is that the prover first computes two Pedersen commitments in the bases g, μ_l and y, ν_l respectively. Then $B_{1,l}$ and $B_{2,l}$ are used to show that the prover can open the commitments. With $B_{3,l}$ and $B_{4,l}$ the prover shows that v_l encrypts what is hidden in the commitment ν_l and that the exponent of g in μ_l is the same as the exponent of y in ν_l.With $B_{5,l}$ the prover shows that it can open $A_{1,l}$ also in the base g, y. With $B_{6,l}$ it finally shows that it can open $A_{2,l}/u_l$ as a power of y, which implies that $A_{2,l}$ and u_l contain y_l to the same power and hence that y_l is the key used in the encryption (u_l, v_l). The detailed proof follows.

Lemma 3.6.5. *Protocol 1 is a $\mathbb{Z}_{q_3}$-Σ-protocol.*

Proof. We prove special soundness first. Suppose we have a list $\big(A_{1,l},\ A_{2,l},\ B_{1,l},\ B_{2,l},\ B_{3,l},\ B_{4,l}\big)_{l=0}^{\delta-1}$, $\big(B_{5,l},\ B_{6,l}\big)_{l=0}^{\delta-2}$, c and $(d_{1,l},\ d_{2,l},\ d_{3,l},\ d_{4,l})_{l=0}^{\delta-1}$, $(d_{5,l},\ d_{6,l})_{l=0}^{\delta-2}$ that satisfy the Equations (3.1)-(3.5), and $c' \neq c$ and $(d'_{1,l},\ d'_{2,l},\ d'_{3,l},\ d'_{4,l})_{l=0}^{\delta-2}$, $(d'_{5,l},\ d'_{6,l})_{l=0}^{\delta-1}$ that satisfies the same equations. We solve the equation systems corresponding to Equations (3.6)-(3.8) to extract λ_l, α_l, ρ_l, ω_l, ζ_l, and τ_l such that

$$\begin{aligned}
&u_0 = y_0^{\rho_0}\\
&A_{1,l} = g^{\alpha_l}\mu_l^{\rho_l}, \quad A_{2,l} = y^{\alpha_l}\nu_l^{\rho_l},\\
&A_{1,l} = g^{\omega_l}y^{\lambda_l}, \quad A_{2,l}/u_{l+1} = y^{\omega_l}\\
&v_l/\nu_l = g^{\rho_l}y^{-\zeta_l}, \quad \mu_l = g^{\zeta_l}y^{\tau_l} \quad .
\end{aligned}$$

From this we can compute $\zeta_l^* = (\omega_l - \alpha_l)/\rho_l$ and $\tau_l^* = \lambda_l/\rho_l$ such that $\mu_l = g^{\zeta_l^*}y^{\tau_l^*}$ since

$$\left(g^{(w_l-\alpha_l)}y^{\lambda_l}\right)^{1/\rho_l} = \left(\frac{A_{1,l}}{g^{\alpha_l}}\right)^{1/\rho_l} = \mu_l \quad .$$

We have $\nu_l = y^{\zeta_l^*}\gamma_{l+1}$ for some γ_{l+1}, i.e., $(\mu_l, \nu_l) = (y^{\tau_l^*}g^{\zeta_l^*}, y^{\zeta_l^*}\gamma_{l+1})$. This implies that $u_{l+1} = A_{2,l}y^{-\omega_l} = y^{\alpha_l}\nu_l^{\rho_l}y^{-\omega_l} = y^{\alpha_l}y^{\zeta_l^*\rho_l}\gamma_{l+1}^{\rho_l} = y^{\alpha_l+\zeta_l^*\rho_l-\omega_l}\gamma_{l+1}^{\rho_l} = \gamma_{l+1}^{\rho_l}$. Define γ_{l+1}^* by $v_l = g^{\rho_l}\gamma_{l+1}^*$, i.e., $(u_l, v_l) = E_{(\gamma_l,g)}(\gamma_{l+1}^*, \rho_l)$. What remains is to argue that $\zeta_l^* = \zeta_l$, $\tau_l^* = \tau_l$, and $\gamma_{l+1}^* = \gamma_{l+1}$ to connect the "links in the chain". The first two equalities follow from $g^{\zeta_l} y^{\tau_l} = \mu_l = g^{\zeta_l^*} y^{\tau_l^*}$, since otherwise we could use the extractor to extract the discrete logarithm $(\zeta_l - \zeta_l^*)/(\tau_l - \tau_l^*) = \log_g y$, which is a contradiction. The last equality follows from $u_l = g^{\rho_l}$, $v_l/\nu_l = g^{\rho_l} y^{-\zeta_l}$, and $\mu_l = g^{\zeta_l} y^{\tau_l}$. Thus, the protocol is special-sound.

Simulation is straightforward. Choose $a_l \in \mathbb{Z}_q$ randomly and set $(A_{1,l}, A_{2,l}) = E_{(g,y)}(u_l, a_l)$. Then choose $(d_{1,l}, d_{2,l}, d_{3,l}, d_{4,l}, d_{5,l}, d_{6,l})$, where $d_{i,l} \in \mathbb{Z}_q$, and $c \in \mathbb{Z}_q$ randomly and define B_0, $\big(A_{1,l},\ A_{2,l},\ B_{1,l},\ B_{2,l},\ B_{3,l},\ B_{4,l},\ B_{5,l},\ B_{6,l}\big)$ by solving Equations (3.9)-(3.11). It is easy to see that the resulting simulation is perfectly distributed. Thus, the protocol is special honest verifier perfect zero-knowledge. □

Next we consider the problem of proving that the values $y_\alpha \in G_{q_3}$ and $g_2^{y_\alpha} \in G_{q_2}$ committed to in two commitments $(\mu, \nu) = (y_3^t g_3^s,\ y_3^s y_\alpha)$ and $(\mu', \nu') = (y_2^{t'} g_2^{s'},\ y_2^{s'} g_2^{y_\alpha})$ respectively satisfy an exponential relation. Stadler [66] studied a simpler problem, namely, given $E_{y_3}(m)$ and g_2^m, prove that an exponential relation holds between the cleartext and the exponent. Although we consider a more complex problem, our protocol is based on his ideas. Note that proving that our relation holds is equivalent to proving knowledge of $s, t \in \mathbb{Z}_{q_2}$ and $s', t' \in \mathbb{Z}_{q_3}$ such that $(\theta, \omega, \phi) = ((\mu')^{\nu^{-1}}, (\nu')^{\nu^{-1}}, \mu^{-1})$ is on the form $(y_2^{t'} g_2^{s'}, y_2^{s'} g_2^{y_3^s}, y_3^t g_3^s)$. For clarity we state this observation as a protocol.

Protocol 2 (Exponential Relation Between Committed Values).
Common Input: *$g_2, y_2, \mu', \nu' \in G_{q_2}$ and $g_3, y_3, \mu, \nu \in G_{q_3}$.*
Private Input: *$t', s' \in \mathbb{Z}_{q_2}$ such that $(\mu', \nu') = (y_2^{t'} g_2^{s'}, y_2^{s'} g_2^{y_\alpha})$ and $t, s \in \mathbb{Z}_{q_3}$ such that $(\mu, \nu) = (y_3^t g_3^s, y_3^s y_\alpha)$.*

1. *Invoke Protocol 3 on common input $g_2, y_2, \theta, \omega \in G_{q_2}$ and $g_3, y_3, \phi \in G_{q_3}$, where $(\theta, \omega, \phi) = \Big((\mu')^{\nu^{-1}}, (\nu')^{\nu^{-1}}, \mu^{-1}\Big)$, and private input $-t, -s \in \mathbb{Z}_{q_3}$ and $t'\nu^{-1}, s'\nu^{-1} \in \mathbb{Z}_{q_2}$. .*

We now give the double-decker exponentiation protocol called from within the protocol above. Here κ_2 is a security parameter that determines the soundness of the protocol.

Protocol 3 (Double-Decker Exponentiation).
Common Input: *$g_2, y_2, \theta, \omega \in G_{q_2}$ and $g_3, y_3, \phi \in G_{q_3}$.*
Private Input: *$t', s' \in \mathbb{Z}_{q_2}$ and $t, s \in \mathbb{Z}_{q_3}$ such that $(\theta, \omega, \phi) = (y_2^{t'} g_2^{s'},\ y_2^{s'} g_2^{y_3^s},\ y_3^t g_3^s)$.*

1. *The prover chooses $e_l, f_l \in \mathbb{Z}_{q_3}$ and $e_l', f_l' \in \mathbb{Z}_{q_2}$ randomly for $l = 1, \ldots, \kappa_2$, computes $F_{1,l} = y_2^{e_l'} g_2^{f_l'}$, $F_{2,l} = y_2^{f_l'} g_2^{y_3^{f_l}}$, and $A_l = y_3^{e_l} g_3^{f_l}$. Then it hands $(F_{1,l}, F_{2,l}, A_l)_{l=1}^{\kappa_2}$ to the verifier.*
2. *The verifier chooses $b = (b_1, \ldots, b_{\kappa_2}) \in \{0,1\}^{\kappa_2}$ randomly and hands b to the prover.*

3. *The prover computes* $d_{1,l} = e_l - b_l t$, $d_{2,l} = f_l - b_l s$, $d_{3,l} = f'_l - b_l y_3^{d_{2,l}} s'$, *and* $d_{4,l} = e'_l - b_l y_3^{d_{2,l}} t'$, *and hands* $(d_{1,l}, d_{2,l}, d_{3,l}, d_{4,l})_{l=1}^{\kappa_2}$ *to the verifier.*

4. *The verifier checks for* $l = 1, \ldots, \kappa_2$ *that*

$$\theta^{b_l y_3^{d_{2,l}}} y_2^{d_{4,l}} g_2^{d_{3,l}} = F_{1,l}, \quad y_2^{d_{3,l}} (\omega^{b_l} g_2^{(1-b_l)})^{y_3^{d_{2,l}}} = F_{2,l}, \tag{3.13}$$

$$\phi^{b_l} y_3^{d_{1,l}} g_3^{d_{2,l}} = A_l \ . \tag{3.14}$$

Since Protocol 2 only calls Protocol 3, we only need to consider Protocol 3:

Lemma 3.6.6. *Protocol 3 is a* $\{0,1\}^{\kappa_2}$*-*Σ*-protocol with soundness* $1 - 2^{\kappa_2}$.

Proof. We prove special soundness first. Suppose that we are given the outputs from two executions $(F_{1,l}, F_{2,l}, A_l)_{l=1}^{\kappa_2}$, b, $(d_{1,l}, d_{2,l})_{l=1}^{\kappa_2}$ and b', $(d'_{1,l}, d'_{2,l})_{l=1}^{\kappa_2}$ with $b \neq b'$ that satisfy Equations (3.13)-(3.14). Thus, for some l, $b_l \neq b'_l$.

Let (ε, τ) and $(\psi, \zeta) \in \mathbb{Z}_{q_3}$ be solutions to the equation systems

$$\left\{ \begin{array}{l} d_{1,l} = e_l - b_l t \\ d'_{1,l} = e_l - b'_l t \end{array} \right\} \quad \text{and} \quad \left\{ \begin{array}{l} d_{2,l} = f_l - b_l s \\ d'_{2,l} = f_l - b'_l s \end{array} \right\} ,$$

This implies that $\phi = y^\tau g^\zeta$.

Consider next the equation system

$$\left\{ \begin{array}{l} d_{3,l} = f'_l - b_l y_3^{d_{2,l}} s' \\ d'_{3,l} = f'_l - b'_l y_3^{d'_{2,l}} s' \end{array} \right\} .$$

Note that $b_l y_3^{d_{2,l}}$ is zero if $b_l = 0$ and non-zero otherwise. Thus, the system is solvable. Let (ψ', ζ') be a solution and assume without loss that $b'_l = 0$. Then we have

$$F_{2,l} = y_2^{d_{3,l}} \omega^{y_3^{d_{2,l}}} = y_2^{\psi' - y_3^{d_{2,l}} \zeta'} \omega^{y_3^{d_{2,l}}} = y_2^{\psi' - y_3^{\psi - \zeta} \zeta'} \omega^{y_3^{\psi - \zeta}}$$

$$F_{2,l} = y_2^{d'_{3,l}} g_2^{y_3^{d'_{2,l}}} = y_2^{\psi'} g_2^{y_3^{d'_{2,l}}} = y_2^{\psi'} g_2^{y_3^{\psi}}$$

Solving for ω gives $\omega = y_2^{\zeta'} g_2^{y_3^{\zeta}}$. Finally, let (ε', τ') be the solution to

$$\left\{ \begin{array}{l} d_{4,l} = e'_l - b_l y_3^{d_{2,l}} t' \\ d'_{4,l} = e'_l - b'_l y_3^{d_{2,l'}} t' \end{array} \right\} .$$

Then we have

$$F_{1,l} = \theta^{y_3^{d_{2,l}}} y_2^{d_{4,l}} g_2^{d_{3,l}} = \theta^{y_3^{d_{2,l}}} y_2^{\varepsilon' - y_3^{d_{2,l}} \tau'} g_2^{\psi' - y_3^{d_{2,l}} \zeta'}$$

$$F_{1,l} = y_2^{d'_{4,l}} g_2^{d'_{3,l}} = y_2^{\varepsilon'} g_2^{\psi'}$$

Solving for θ gives $\theta = y_2^{\tau'} g_2^{\zeta'}$. We conclude that the protocol is special-sound.

The simulator is defined as follows. For $l = 1, \ldots, \kappa_2$ choose $b_l \in \{0,1\}$ and $d_{1,l}, d_{2,l} \in \mathbb{Z}_{q_3}$ and $d_{3,l}, d_{4,l} \in \mathbb{Z}_{q_2}$ randomly and define $(F_{1,l}, F_{1,l}, A_l)$ by Equations (3.13)-(3.14). We conclude that the protocol is special honest verifier perfect zero-knowledge. □

Our next protocol shows that the cleartext of an ElGamal encryption is the value hidden in a commitment. Since the protocol is used in conjunction with Cramer-Shoup cryptotexts, we use a notation that is consistent with the Cramer-Shoup cryptosystem.

Protocol 4 (Equality of Committed and Encrypted Cleartexts).
COMMON INPUT: $g_3, y_3, u', v', \bar{g}_1, \bar{h}, u, v \in G_{q_3}$.
PRIVATE INPUT: *t', s', r such that $(u', v') = (g_3^{t'} y_3^{s'}, g_3^{s'} m)$ and $(u, v) = (\bar{g}_1^r, \bar{h}^r m)$.*

1. *The prover chooses $a, e, f \in \mathbb{Z}_{q_3}$ randomly, computes $A_1 = g_3^a y_3^e$, $A_2 = g_3^e \bar{h}^f$, $A_3 = \bar{g}_1^f$, and hands (A_1, A_2, A_3) to the verifier.*
2. *The verifier chooses $c \in \mathbb{Z}_{q_3}$ randomly and hands it to the verifier.*
3. *The prover computes $d_1 = ct' + a$, $d_2 = cs' + e$, $d_3 = -cr + f$ and hands (d_1, d_2, d_3) to the verifier.*
4. *The verifier checks that*

$$(u')^c A_1 = g_3^{d_1} y_3^{d_2}, \quad (v'/v)^c A_2 = g_3^{d_2} \bar{h}^{d_3}, \quad u^c A_3 = \bar{g}_1^{d_3} \ . \tag{3.15}$$

Lemma 3.6.7. *Protocol 4 is a $\mathbb{Z}_{q_3}$-Σ-protocol.*

Proof. Consider special soundness. Given (A_1, A_2, A_3), (c, d_1, d_2, d_3), and (c', d_1', d_2', d_3'), with $c \neq c'$, that satisfy the check above, we can solve the corresponding equation systems to find $\tau', \zeta', \tau \in \mathbb{Z}_{q_3}$ such that

$$(u', v'/v, u) \quad = \quad (g_3^{\tau'} y_3^{\zeta'}, g_3^{\zeta'} \bar{h}^{\tau}, \bar{g}_1^{\tau}) \ .$$

This implies that the cryptotext and commitment holds the same value $v/\bar{h}^{\tau}$ as prescribed. Thus, thus the protocol is special-sound.

The simulator chooses $c, d_1, d_2, d_3 \in \mathbb{Z}_{q_3}$ and defines A_1, A_2, A_3 by Equation (3.15). It is easy to see that the protocol is special honest verifier perfect zero-knowledge. □

Our next protocol shows that a Cramer-Shoup cryptotext is valid. We define it for an arbitrary hash function, although we will later instantiate it with a H^{CHP} hash function.

Protocol 5 (Validity of Cramer-Shoup Cryptotext).
COMMON INPUT: $H : G_{q_3}^3 \rightarrow \mathbb{Z}_{q_3}$, $\bar{g}_1, \bar{g}_2, u, \mu, v, \nu \in G_{q_3}$, *and* $\bar{c}, \bar{d} \in G_{q_3}$.
PRIVATE INPUT: $r \in \mathbb{Z}_{q_3}$ *such that* $(u, \mu, v, \nu) = (\bar{g}_1^r, \bar{g}_2^r, v, \bar{c}^r \bar{d}^{rH(u,\mu,v)})$.

1. *The prover randomly selects $a \in \mathbb{Z}_{q_3}$ and computes $B_1 = \bar{g}_1^a$, $B_2 = \bar{g}_2^a$, $B_3 = \left(\bar{c}\bar{d}^{H(u,\mu,v)}\right)^a$ and hands (B_1, B_2, B_3) to the verifier.*

2. *The verifier randomly selects $c \in \mathbb{Z}_{q_3}$ and hands it to the prover.*

3. *The prover computes $d = cr + a$ and hands d to the verifier.*

4. *The verifier checks that $u^c B_1 = \bar{g}_1^d$, $\mu^c B_2 = \bar{g}_2^d$ and $\nu^c B_3 = \left(\bar{c}\bar{d}^{H(u,\mu,v)}\right)^d$.*

Lemma 3.6.8. *Protocol 5 is a $\mathbb{Z}_{q_3}$-Σ-protocol.*

Proof. It can easily be verified that the protocol never fails on valid input.

Assuming the output of two executions B_1, B_2, B_3, c, d and B_1, B_2, B_3, c', d' for $c \neq c'$ both satisfying the verification of Step 4, we can compute $\rho = (d-d')/(c-c')$ such that $(u, \mu, \nu) = (\bar{g}_1^\rho, \bar{g}_2^\rho, \bar{c}^\rho \bar{d}^{\rho H(u,\mu,v)})$. Thus, the protocol is special-sound.

The simulator chooses $c, d \in \mathbb{Z}_{q_3}$ randomly and defines B_1, B_2, and B_3 by the equations in Step 4. It follows that the protocol is special honest verifier perfect zero-knowledge. □

The next protocol combines the protocols above and provides a solution to the goal of this section, i.e., proving the relations in Step 3 in Algorithm 9 involving only elements from G_{q_1}, G_{q_2}, and G_{q_3}.

Protocol 6 (Commitment to Hash of Chained Keys).
COMMON INPUT: $g_3, y_3, y_{\alpha_0} \in G_{q_3}$, $g_2, y_2 \in G_{q_2}$, $g_1, y_1 \in G_{q_1}$, $H^{\mathsf{CHP}} = (h_1, \ldots, h_\delta) \in G_{q_2}^\delta$, $(u_l, v_l)_{l=0}^{\delta-1} \in G_{q_3}^{2\delta}$, $(\mu'', \nu'') \in G_{q_1}^2$, $\bar{g}_1, \bar{g}_2, \bar{c}, \bar{d}, \bar{h} \in G_{q_3}$, $C_\delta = (\bar{u}, \bar{\mu}, \bar{v}, \bar{\nu}) \in G_{q_3}^4$.
PRIVATE INPUT: $r_0, \ldots, r_\delta \in \mathbb{Z}_{q_3}$, $r, y_{\alpha_1}, \ldots, y_{\alpha_\delta} \in G_{q_3}$ *satisfying the equations in Step 3 in Algorithm 9, and $s'', t'' \in \mathbb{Z}_{q_2}$ such that*

$$(\mu'', \nu'') = (y_1^{t''} g_1^{s''}, y_1^{s''} g_1^{H^{\mathsf{CHP}}(y_{\alpha_1}, \ldots, y_{\alpha_\delta})}) \enspace .$$

1. *The prover chooses $y_{\alpha_{\delta+1}} \in G_{q_3}$ and $r_\delta \in \mathbb{Z}_{q_3}$ randomly, computes $(u_\delta, v_\delta) = E_{y_{\alpha_\delta}}\left(y_{\alpha_{\delta+1}}, r_\delta\right)$, and hands (u_δ, v_δ) to the verifier.*

2. *The prover chooses $s_l, t_l \in \mathbb{Z}_{q_2}$, computes commitments*

$$(\mu_l, \nu_l) = \left(g_3^{s_l} y_3^{t_l}, y_3^{s_l} y_{\alpha_{l+1}}\right)$$

for $l = 0, \ldots, \delta - 1$, and hands $(\mu_l, \nu_l)_{l=0}^{\delta-1}$ to the verifier.

3. *The prover chooses $s_l', t_l' \in \mathbb{Z}_{q_3}$ randomly, computes commitments $(\mu_l', \nu_l') = (y_2^{t_l'} g_2^{s_l'}, y_2^{s_l'} h_{l+1}^{y_{\alpha_{l+1}}})$ for $l = 0, \ldots, \delta - 1$, and hands $(\mu_l', \nu_l')_{l=1}^{\delta}$ to the verifier.*

4. *The prover and verifier computes $(\mu', \nu') = \left(\prod_{l=0}^{\delta-1} \mu_l', \prod_{l=0}^{\delta-1} \nu_l'\right)$. The prover computes $s' = \sum_{l=0}^{\delta-1} s_l'$ and $t' = \sum_{l=0}^{\delta-1} t_l'$.*

5. *Invoke the following protocols in parallel:*

   a) *Protocol 1 on public input y_{α_0}, g_3, y_3, $\big((u_l, v_l), (\mu_l, \nu_l)\big)_{l=0}^{\delta-1}$, and private input $(r_l, s_l, t_l)_{l=0}^{\delta-1}$ to show that the chain is a valid chain of encrypted keys.*

   b) *Protocol 2 for $l = 0, \ldots, \delta - 1$ on public input $g_3, y_3, \mu_l, \nu_l \in G_{q_3}$ and $g_2, y_2, h_l, \mu'_l, \nu'_l \in G_{q_2}$, and private input $s_l, t_l \in \mathbb{Z}_{q_3}$ and $s'_l, t'_l \in \mathbb{Z}_{q_2}$.*

   c) *Protocol 2 on public input $g_2, y_2, \mu', \nu' \in G_{q_2}$ and $g_1, y_1, g_1, \mu'', \nu'' \in G_{q_1}$, and private input $s', t' \in \mathbb{Z}_{q_2}$ and $s'', t'' \in \mathbb{Z}_{q_1}$. These two protocols show that (μ'', ν'') is a commitment of the hash value of the public keys in the commitments (μ_l, ν_l).*

   d) *Protocol 4 on common input $g_3, y_3, \mu_\delta, \nu_\delta \in G_{q_3}$, $\bar{g}_1, \bar{h} \in G_{q_3}$, $\bar{u}, \bar{v} \in G_{q_3}$, and private input $t_\delta, s_\delta, r \in \mathbb{Z}_{q_3}$ to show that the $\mathcal{CS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}}$ encryption is an encryption of the value committed to in (μ_δ, ν_δ).*

   e) *Protocol 5 on common input $\bar{g}_1, \bar{g}_2, \bar{c}, \bar{d}, \bar{h} \in G_{q_3}$, $C_\delta = (\bar{u}, \bar{\mu}, \bar{v}, \bar{\nu}) \in G_{q_3}^4$, and private input $r \in \mathbb{Z}_{q_3}$ to show that the $\mathcal{CS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}}$ encryption is correctly formed.*

Lemma 3.6.9. *Protocol 6 is a $\{0,1\}^{\kappa_2} \times \mathbb{Z}_{q_3}$-$\Sigma$-protocol.*

Proof. From Lemma 3.6.5, Lemmas 3.6.6, 3.6.7, 3.6.8 and Observations 1 and 2 it follows that Step 5 may be considered a single combined $\{0,1\}^{\kappa_2} \times \mathbb{Z}_{q_3}$-$\Sigma$-protocol. However, we must show that the extracted values satisfy additional equations.

From the combined protocols we can extract $\tau_l, \zeta_l, \psi_l \in \mathbb{Z}_{q_3}$, $\gamma_l \in G_{q_3}$, ζ'_l, ψ'_l, ζ', $\psi' \in \mathbb{Z}_{q_2}$, ζ'', $\psi'' \in \mathbb{Z}_{q_1}$, ψ^*_δ, ζ^*_δ, $\tau \in \mathbb{Z}_{q_3}$, $\Gamma \in G_{q_2}$ such that for $l = 0, \ldots, \delta - 1$

$$\begin{aligned}
(u_l, v_l) &= E_{(\gamma_l, g_3)}(\gamma_{l+1}, \tau_l) = (\gamma_l^{\tau_l}, g_3^{\tau_l}\gamma_{l+1}) \\
(\mu_l, \nu_l) &= (y_3^{\psi_l} g_3^{\zeta_l}, y_3^{\zeta_l}\gamma_l) \\
(\mu'_l, \nu'_l) &= (y_2^{\psi'_l} g_2^{\zeta'_l}, y_2^{\zeta'_l} h_l^{\gamma_l}) \\
(\mu', \nu') &= (y_2^{\psi'} g_2^{\zeta'}, y_2^{\zeta'}\Gamma) \\
(\mu'', \nu'') &= (y_1^{\psi''} g_1^{\zeta''}, y_1^{\zeta''} g_1^{\Gamma}) \\
(\bar{u}, \bar{v}) &= (\bar{g}_1^{\tau}, \bar{h}^{\tau}\gamma^*_\delta) \ .
\end{aligned}$$

If $\prod_{l=1}^{\delta} h_l^{\gamma_l} \neq \Gamma$, then either $\psi' \neq \sum_{l=1}^{\delta} \psi'_l$ or $\zeta' \neq \sum_{l=1}^{\delta} \zeta'_l$. In either case this implies that we can extract $\log_{g_2} y_2$, which is a contradiction. Under Lemma 3.6.7 it must hold that $\gamma_\delta = \gamma^*_\delta$. Thus, the protocol is special-sound.

To simulate the proof the simulator chooses $u_\delta, v_\delta, \mu_l, \nu_l \in G_{q_3}$, $\mu'_l, \nu'_l \in G_{q_2}$ randomly instead of as defined in the protocol. It is easy to see that these elements are identically distributed to the corresponding elements in an execution of the protocol. The simulator invokes the simulator for the combined proof of knowledge of Step 5. It follows that the protocol is special honest verifier perfect zero-knowledge. □

Proof of Equality of Exponents Over Distinct Groups

In several of our subprotocols, we need to prove relations over an RSA modulus. These proofs differ slightly from proofs over a group of prime order. When performing proofs over an RSA modulus the order of the multiplicative group is not known, and therefore we cannot reduce the exponents.

One way to get around this problem is illustrated in the example below, where we prove knowledge of how to open a Pedersen commitment. Here κ_1, κ_2 and κ_3 are three security parameters. The completeness depends on κ_1, the soundness depends on κ_2 and the amount of information disclosed depends on κ_3.

Zero-knowledge proofs over such moduli have been studied by Fujisaki and Okamoto [32]. Here we use techniques that are similar, although not identical to theirs. In [32] the equivalent to a Pedersen commitment over a composite modulus such as $\mathbb{Z}_N$ is studied. To commit to a number s the prover computes $\mathbf{g}^s\mathbf{y}^r$ where r is drawn from $[0, 2^{\kappa_3}N - 1]$ for a security parameter κ_3. The following two lemmas are proven.

Lemma 3.6.10 (cf. [32]). *There exists a polynomial-time algorithm that takes as input a composite number N and r_1, s_1, r_2, s_2, $r_1 \neq r_2$ and $s_1 \neq s_2$, such that $\mathbf{g}^{r_1}\mathbf{y}^{s_1} = \mathbf{g}^{r_2}\mathbf{y}^{s_2}$ that with high probability outputs the factorization of N.*

Lemma 3.6.11 (cf. [32]). *If $\kappa_3 = \Theta(\log N)$, then $\mathbf{g}^s\mathbf{y}^r$ statistically reveals no information about s.*

First consider the problem of proving equality of exponents over distinct groups. This is used as a bridge between the two parts of our protocol. Two Pedersen commitments are given: one over G_q denoted C and one over $\mathbb{Z}_N$ denoted $\mathbf{C}$. The task is to prove that the committed values are equal when interpreted over the integers. This problem has been studied by Boudot and Traoré [10] as well as by Camenisch and Michels [16] and we follow their example.

Protocol 7 (Equality of Exponents Over Distinct Groups).
COMMON INPUT: $\mathbf{g}, \mathbf{y}, \mathbf{C} \in \mathrm{QR}_N$ *and* $g, y, C \in G_q$*, where* $q < N$.
PRIVATE INPUT: $e \in [0, q-1]$*,* $s \in [0, 2^{\kappa_3}N - 1]$*, and* $s' \in \mathbb{Z}_q$*. such that* $\mathbf{C} = \mathbf{g}^e\mathbf{y}^s$ *and* $C = g^e y^{s'}$.

1. *The prover chooses* $a \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}q - 1], b \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}N - 1]$ *and* $b' \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}q - 1]$ *randomly, computes* $\mathbf{A} = \mathbf{g}^a\mathbf{y}^b$, $A = g^a y^{b'}$*, and hands* $(\mathbf{A}, A)$ *to the verifier.*

2. *The verifier chooses* $c \in [0, 2^{\kappa_2} - 1]$ *and hands it to the prover.*

3. *The prover computes*

$$\begin{aligned} d_1 &= a + ce \mod 2^{\kappa_1+\kappa_2+\kappa_3}q, && (3.16)\\ d_2 &= b + cs \mod 2^{\kappa_1+\kappa_2+\kappa_3}N, && (3.17)\\ d_3 &= b' + cs' \mod 2^{\kappa_1+\kappa_2+\kappa_3}q && (3.18)\end{aligned}$$

and hands (d_1, d_2, d_3) to the verifier.

4. *The verifier checks that $\mathbf{g}^{d_1}\mathbf{y}^{d_2} = \mathbf{C}^c\mathbf{A}$ (in $\mathbb{Z}_N$) and $g^{d_1}y^{d_3} = C^cA$ (in G_q).*

Lemma 3.6.12. *Protocol 7 is a $[0, 2^{\kappa_2}-1]$-Σ-protocol with completeness $1-3\cdot 2^{-\kappa_2}$.*

Proof. If the prover is honest the verifier accepts if there is no reduction in the computation of d_1, d_2 or d_3. By the union bound this happens with probability not more than $3 \cdot 2^{-\kappa_1}$, which gives a completeness of $1 - 3 \cdot 2^{-\kappa_2}$.

To prove that the protocol is special-sound, assume we have $\mathbf{A}, A, c, d_1, d_2, d_3$ as well as $c' \neq c$, d_1', d_2', d_3', each list satisfying the equations of Step 4. Then by solving the equations system consisting of Equations (3.16) to (3.18) over $\mathbb{Z}$ we get $\varepsilon = \frac{d_1-d_1'}{c-c'}$, $\zeta = \frac{d_2-d_2'}{c-c'}$ and $\zeta' = \frac{d_3-d_3'}{c-c'}$.

We now show that ε and ζ' are integers. In [32] the following lemma is proven:

Lemma 3.6.13 (cf. [32]). *Let P^* be an oracle that on input $N, \mathbf{g}, \mathbf{y}$ outputs $\mathbf{u}, \boldsymbol{\mu}$ and two lists (c, d_1, d_2), (c', d_1', d_2') satisfying the equations of Step 4 with $(c - c') \nmid (d_1 - d_1')$ or $(c-c') \nmid (d_2 - d_2')$. Then there exists a polynomial-time machine which on input $\mathbf{C}, N$ and access to P^* outputs $\mathbf{z}, e$ such that $\mathbf{z}^e = \mathbf{C}$ and $e > 1$.*

From the above Lemma it follows that ζ, τ are integers since otherwise a prover that is able to construct such proofs can easily be made into the oracle of Lemma 3.6.13 and hence used to break the strong RSA assumption.

A prover able to construct a proof such that the extracted value ζ' is a non-integer can also compute $\log_g y$, so we conclude that also β is an integer and hence $\mathbf{C} = \mathbf{g}^\varepsilon\mathbf{y}^\zeta$, $C = g^\varepsilon y^{\zeta'}$, which concludes the extraction.

The protocol can be simulated by choosing the challenge $c \in [0, 2^{\kappa_2} - 1]$, and the prover's response $d_1, d_2 \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}q - 1]$, $d_3 \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}N - 1]$. Then A and $\mathbf{A}$ are computed according to the equations of Step 4. This gives the same distribution as an execution of the protocol. □

Zero-knowledge proofs over an RSA modulus

Sometimes it is more convenient to keep the committed number in the base rather than in the exponent. In this case a commitment to a number $\mathbf{z}$ can be computed as $(\mathbf{g}^r\mathbf{y}^s, \mathbf{g}^r\mathbf{z})$ where r, s are chosen at random from $[0, 2^{\kappa_3}N - 1]$.

Protocol 8 (Commitment over $\mathbb{Z}_N$).
COMMON INPUT: $\mathbf{g}, \mathbf{y}, \mathbf{u}, \mathbf{v} \in \mathrm{QR}_N$.
PRIVATE INPUT: $s, t \in [0, 2^{\kappa_3}N - 1]$, $\mathbf{r} \in \mathrm{QR}_N$ *such that* $(\mathbf{u}, \mathbf{v}) = (\mathbf{g}^s\mathbf{y}^t, \mathbf{g}^t\mathbf{r})$.

1. *The prover randomly chooses $a, b \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}N - 1]$, computes $\boldsymbol{\mu} = \mathbf{g}^a\mathbf{y}^b$, and hands $\boldsymbol{\mu}$ to the verifier.*

2. *The verifier chooses $c \in [0, 2^{\kappa_2} - 1]$ randomly and hands it to the prover.*

3. The prover computes

$$\begin{aligned} d_1 &= cs + a \mod 2^{\kappa_1+\kappa_2+\kappa_3}N \ , \\ d_2 &= ct + b \mod 2^{\kappa_1+\kappa_2+\kappa_3}N \end{aligned}$$

and hands (d_1, d_2) to the verifier.

4. The verifier checks that $\mathbf{u}^c\boldsymbol{\mu} = \mathbf{g}^{d_1}\mathbf{y}^{d_2}$.

Lemma 3.6.14. *Protocol 8 is a $[0, 2^{\kappa_2}-1]$-Σ-protocol with completeness $1-2\cdot 2^{-\kappa_1}$.*

Proof. It is easy to check that the verifier accepts when there is no reduction modulo $2^{\kappa_1+\kappa_2+\kappa_3}N$ in the computation of d_1 or d_2. Therefore we want to compute the probability of such a reduction to occur. There are at most $2^{\kappa_2+\kappa_3}N$ values of a (corresponding to $ct = 2^{\kappa_2+\kappa_3}N$) for which a reduction occurs for d_1. With $2^{\kappa_1+\kappa_2+\kappa_3}N$ possible values of a the probability for such a reduction to occur is bounded by $\frac{2^{\kappa_2+\kappa_3}N}{2^{\kappa_1+\kappa_2+\kappa_3}N} = 2^{-\kappa_1}$. Since the same reasoning holds for d_2 the union bound gives an upper bound of $2\cdot 2^{-\kappa_1}$ for a reduction to occur in one of the two computations. Hence the completeness is at least $1-2\cdot 2^{-\kappa_1}$.

For the extraction of s, t and $\mathbf{r}$ to prove special soundness, assume that we have two lists $(\boldsymbol{\mu}, c, d_1, d_2)$ and $(\boldsymbol{\mu}, c', d_1', d_2')$ where $c \neq c'$ which both satisfy the equations in Step 4. By solving the equations in Step 3 (over $\mathbb{Z}$) we get $\zeta = \frac{d_1'-d_1}{c'-c}$ and $\tau = \frac{d_2'-d_2}{c'-c}$. By Lemma 3.6.13 it follows that ζ and τ are integers. We now have that the value of $\mathbf{r}$ can now be computed as $\mathbf{v}\mathbf{g}^{-\tau}$.

The protocol can be simulated by choosing $d_1, d_2 \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}N-1]$ and $c \in [0, 2^{\kappa_2}-1]$ at random and computing $\boldsymbol{\mu} = \mathbf{g}^{d_1}\mathbf{y}^{d_2}\mathbf{u}^{-c}$. This gives a distribution of $\boldsymbol{\mu}, c, d_1, d_2$ equal to that of an execution. □

It is possible to write the protocol without the reduction in the computations of d_1 and d_2. Then we get perfect completeness, but since d_1 and d_2 are not uniformly distributed our simulation will not yield the exact distribution of an execution. It seems that we are forced to choose between perfect zero-knowledge and perfect completeness. It our description we choose perfect zero-knowledge. By choosing κ_1 large enough, say $\kappa_1 = 64$, we can in practice ignore the risk of failure with only a minor increase in running time.

Next we give a protocol that shows that two committed values are equal. The idea of the proof is to show that their ratio is one. The protocol is not a proof of knowledge of the committed value nor of the exponents of the commitments, only of the difference between the exponents. Note that we parametrize the protocol on z to allow for different sizes of the exponents. The different sizes appear when the protocol is applied to products of commitments.

Protocol 9 (Equality of Committed Values over $\mathbb{Z}_N$).
COMMON INPUT: $\mathbf{g}, \mathbf{y}, \in \mathrm{QR}_N$ *and* $(\mathbf{u}, \mathbf{v}), (\mathbf{u}', \mathbf{v}') \in \mathrm{QR}_N^2$.
PRIVATE INPUT: $\mathbf{r} \in \mathrm{QR}_N$ *such that* $(\mathbf{u}, \mathbf{v}) = (\mathbf{g}^s\mathbf{y}^t, \mathbf{g}^t\mathbf{r})$, $(\mathbf{u}', \mathbf{v}') = (\mathbf{g}^{s'}\mathbf{y}^{t'}, \mathbf{g}^{t'}\mathbf{r})$ *for some* $s, t, s', t' \in [-2^{\kappa_3}z+1, 2^{\kappa_3}z-1]$.

1. *The prover chooses* a, b *at random from* $[0, 2^{\kappa_1+\kappa_2+\kappa_3}z - 1]$, *computes* $\boldsymbol{\mu} = (\mathbf{g}^a\mathbf{y}^b)$ *and hands* $\boldsymbol{\mu}$ *to the verifier.*
2. *The verifier randomly selects* $c \in [0, 2^{\kappa_2} - 1]$ *and hands it to the prover.*
3. *The prover computes* $d_1 = c(s-s') + a \mod 2^{\kappa_1+\kappa_2+\kappa_3}z$ *and* $d_2 = c(t-t') + b \mod 2^{\kappa_1+\kappa_2+\kappa_3}z$ *and hands* (d_1, d_2) *to the verifier.*
4. *The verifier checks that* $(\mathbf{u}/\mathbf{u}')^c\boldsymbol{\mu} = \mathbf{g}^{d_1}\mathbf{y}^{d_2}$.

Lemma 3.6.15. *Protocol 9 is a* $[0, 2^{\kappa_2}-1]$*-*Σ*-protocol with completeness* $1-2\cdot 2^{-\kappa_1}$.

Proof. An honest prover fails to convince the verifier if there is a reduction in the computation of d_1 or d_2. For d_1 this happens either if $c(s - s') + a > 2^{\kappa_1+\kappa_2}z - 1$ or $c(s - s') + a < 0$. The first case can happen only if $c(s - s') > 0$, and then with probability at most $2^{-\kappa_1}$. The second case can happen only if $c(s - s') < 0$, and also then with probability at most $2^{-\kappa_1}$. The same reasoning holds for d_2, giving by the union bound a probability of at most $2 \cdot 2^{-\kappa_1}$ for a reduction to happen. Therefore the completeness is $1 - 2 \cdot 2^{-\kappa_1}$.

To show special soundness assume that we have (a, b), c and (d_1, d_2) satisfying the equations of Step 4 as well as $c' \neq c$ and d_1', d_2' satisfying the same equations. By solving the equations of Step 3 we get ζ, τ satisfying the equations $\mathbf{u}/\mathbf{u}' = \mathbf{g}^\zeta\mathbf{y}^\tau$ and $\mathbf{v}/\mathbf{v}' = \mathbf{g}^\tau$. By Lemma 3.6.13 ζ and τ are integers. This shows that $(\mathbf{u}/\mathbf{u}', \mathbf{v}/\mathbf{v}')$ is a commitment of the value 1, and hence that $(\mathbf{u}, \mathbf{v})$ and $(\mathbf{u}', \mathbf{v}')$ commit to the same value.

For the simulation choose d_1, d_2 at random from $[0, 2^{\kappa_1+\kappa_2+\kappa_3}z - 1]$ and c from $[0, 2^{\kappa_2} - 1]$. Compute $\boldsymbol{\mu}, \boldsymbol{\nu}$ to satisfy the equations from Step 4. The distribution we get this way is equal to the distribution from an honest execution. □

The above protocol can also be used to prove that a pair $\mathbf{u}, \mathbf{v}$ is a commitment to a public value $\mathbf{w}$. For clarity we state this as a protocol, also this time parametrized on z:

Protocol 10 (Committed Value over $\mathbb{Z}_N$).
Common Input: $\mathbf{g}, \mathbf{y}, \mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathrm{QR}_N$.
Private Input: $s, t \in [-2^{\kappa_3}z + 1, 2^{\kappa_3}z - 1]$ *such that* $(\mathbf{u}, \mathbf{v}) = (\mathbf{g}^s\mathbf{y}^t, \mathbf{g}^t\mathbf{w})$.

1. *Invoke protocol 9 on public input* $\mathbf{g}, \mathbf{y}, (\mathbf{u}, \mathbf{v}), (\mathbf{1}, \mathbf{w})$ *and private exponents* $s, t, 1, 1$.

Lemma 3.6.16. *Protocol 10 is a* $[0, 2^{\kappa_2}-1]$*-*Σ*-protocol with completeness* $1-2\cdot 2^{-\kappa_1}$.

Proof. Follows directly from Lemma 3.6.15. □

In Protocol 2 we showed how to prove that two committed values have an exponential relation. We need to be able to do this also over $\mathbb{Z}_N$. Also in this case we use a protocol for double-decker expontial relations similar to Protocol

3 as base to construct the following protocol. Once again we use the fact that proving $(u,v) = (g_N^{s'} y_N^{t'}, y_N^{s'} g_N^{\mathbf{r}})$, $(\mathbf{u},\mathbf{v}) = (\mathbf{g}^s\mathbf{y}^t, \mathbf{y}^s\mathbf{r})$ is equivalent to proving that $(\theta,\omega,\boldsymbol{\phi}) = (u^{\mathbf{v}^{-1}}, v^{\mathbf{v}^{-1}}, \mathbf{u}^{-1})$ is on the form $(y_N^{f'} g_N^{e'}, y_N^{e'} g_N^{\mathbf{y}^e}, \mathbf{y}^f\mathbf{g}^e)$.

Protocol 11 (Exponential Relations over $\mathbb{Z}_N$).
COMMON INPUT: $\mathbf{g},\mathbf{y},\mathbf{h} \in \mathrm{QR}_N$, $(\mathbf{u},\mathbf{v}),(\mathbf{u}',\mathbf{v}') \in \mathrm{QR}_N^2$, *and* $g_N, y_N \in G_N$.
PRIVATE INPUT: $s,t,s',t' \in [0, 2^{\kappa_3}N-1]$ *such that* $(\mathbf{u},\mathbf{v}) = (\mathbf{g}^s\mathbf{y}^t, \mathbf{g}^t\mathbf{r})$ *and* $(\mathbf{u}',\mathbf{v}') = (\mathbf{g}^{s'}\mathbf{y}^{t'}, \mathbf{y}^{s'}\mathbf{h}^{\mathbf{r}})$.

1. *The prover generates* $s'',t'' \in \mathbb{Z}_N$, *computes* $(u,v) = (g_N^{s''} y_N^{t''}, g_N^{t''} y_N^{\mathbf{r}})$ *and hands* (u,v) *to the verifier.*

2. *The following two protocols are executed in parallel:*

   a) *Protocol 12 on common input* $\mathbf{g},\mathbf{y},\boldsymbol{\phi} \in \mathrm{QR}_N$ *and* $g_N, y_N, \theta, \omega \in G_N$ *where* $(\theta,\omega,\boldsymbol{\phi}) = (u^{\mathbf{v}^{-1}}, v^{\mathbf{v}^{-1}}, \mathbf{u}^{-1})$ *and private input* $t''\mathbf{v}^{-1}, s''\mathbf{v}^{-1} \in \mathbb{Z}_N$ *and* $-t,-s \in [0, 2^{\kappa_3}N-1]$.

   b) *Protocol 7 on common input* $\mathbf{y},\mathbf{h},\mathbf{v}' \in \mathrm{QR}_N$, $g_N, y_N, v \in G_N$ *and private input* $\mathbf{r}, t', t''$.

The idea behind the above protocol is that the value $\mathbf{r}$ first is “lifted” to G_N. The relation is then shown between an element in G_N and an element in $\mathbb{Z}_N$, as shown in Figure 3.7.

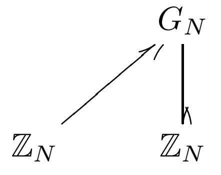

$$\begin{tikzcd} & G_N \\ \mathbb{Z}_N \arrow[ur] & \mathbb{Z}_N \arrow[u] \end{tikzcd}$$

Figure 3.7: Structure of Protocol 11

Lemma 3.6.17. *Protocol 11 is a* $\{0,1\}^{\kappa_2}$*-*Σ*-protocol with completeness* $1-(2\kappa_2+3)2^{-\kappa_1}$.

Proof. Follows from Lemma 3.6.12 and Lemma 3.6.18 by using the union bound for the completeness. □

Protocol 12 (Double-Decker Exponentiation over $\mathbb{Z}_N$).
COMMON INPUT: $\mathbf{g},\mathbf{y},\boldsymbol{\phi} \in \mathrm{QR}_N$ *and* $g_N, y_N, \theta, \omega \in G_N$.
PRIVATE INPUT: $t,s \in [0, 2^{\kappa_3}N-1]$ *and* $t',s' \in \mathbb{Z}_N$ *such that* $(\theta,\omega,\boldsymbol{\phi}) = (y_N^{t'} g_N^{s'}, y_N^{s'} g_N^{\mathbf{y}^s}, \mathbf{y}^t\mathbf{g}^s)$.

1. *The prover chooses* $e_l, f_l \in [0, 2^{\kappa_1+\kappa_3}N - 1]$ *and* $e'_l, f'_l \in \mathbb{Z}_N$ *randomly for* $l = 1, \ldots, \kappa_2$. *Then it computes* $F_{1,l} = y_N^{e'_l} g_N^{f'_l}$, $F_{2,l} = y_N^{f'_l} g_N^{\mathbf{y}^{f_l}}$, $\mathbf{A}_l = \mathbf{y}^{e_l}\mathbf{g}^{f_l}$ *and hands* $(F_{1,l}, F_{2,l}, \mathbf{A}_l)_{l=1}^{\kappa_2}$ *to the verifier.*

2. *The verifier randomly chooses* $b = (b_1, \ldots, b_{\kappa_2}) \in \{0,1\}^{\kappa_2}$ *and hands* b *to the prover.*

3. *The prover computes*

$$\begin{aligned} d_{1,l} &= e_l - b_l t \bmod 2^{\kappa_1+\kappa_3}N \ , \\ d_{2,l} &= f_l - b_l s \bmod 2^{\kappa_1+\kappa_3}N \ , \\ d_{3,l} &= f'_l - b_l \mathbf{y}^{d_{2,l}} s' \bmod N \ , \\ d_{4,l} &= e'_l - b_l \mathbf{y}^{d_{2,l}} t' \bmod N \end{aligned}$$

and hands $(d_{1,l},\ d_{2,l},\ d_{3,l},\ d_{4,l})_{l=1}^{\kappa_2}$ *to the verifier.*

4. *The verifier checks for* $l = 1, \ldots, \kappa_2$ *that*

$$\begin{aligned} \theta^{b_l \mathbf{y}^{d_{2,l}}} y_N^{d_{4,l}} g_N^{d_{3,l}} &= F_{1,l} \ , \\ y_N^{d_{3,l}} (\omega^{b_l} g_N^{1-b_l})^{\mathbf{y}^{d_{2,l}}} &= F_{2,l} \ , \\ \boldsymbol{\phi}^{b_l} \mathbf{y}^{d_{1,l}} \mathbf{g}^{d_{2,l}} &= \mathbf{A}_l \ . \end{aligned}$$

Lemma 3.6.18. *Protocol 12 is a* $\{0,1\}^{\kappa_2}$*-*Σ*-protocol with completeness* $1 - 2\kappa_2 2^{-\kappa_1}$.

Proof. If there is no reduction in the computations of $d_{1,l}$ and $d_{2,l}$ the verifier will accept if the prover is honest. The probability of a reduction in one computation is $2^{-\kappa_1}$. By the union bound this gives that the probability for a reduction in one of the $2\kappa_2$ computation is at most $2\kappa_2 2^{-\kappa_1}$, giving a completeness of' $1 - 2\kappa_2 2^{-\kappa_1}$.

Now we prove special soundness by describing the extraction. For this we follow the proof of Lemma 3.6.6, taking into account that the multiplicative order of $\mathbb{Z}_N$ is unknown.

Suppose that we are given two outputs $(F_{1,l},\ F_{2,l},\ \mathbf{A}_l)_{l=1}^{\kappa_2}$, b, $(d_{1,l}, d_{2,l})_{l=1}^{\kappa_2}$ and b', $(d'_{1,l}, d'_{2,l})_{l=1}^{\kappa_2}$ with $b \neq b'$ that satisfy the equations of Step 4. Thus, for some l, $b_l \neq b'_l$.

Let (ε, τ) and (ψ, ζ) be solutions to the equation systems

$$\left\{ \begin{array}{l} d_{1,l} = e_l - b_l t \\ d'_{1,l} = e_l - b'_l t \end{array} \right\} \quad \text{and} \quad \left\{ \begin{array}{l} d_{2,l} = f_l - b_l s \\ d'_{2,l} = f_l - b'_l s \end{array} \right\} \ ,$$

i.e., $\tau = \frac{d_{1,l} - d'_{1,l}}{b_l - b'_l}$ and $\zeta = \frac{d_{2,l} - d'_{2,l}}{b_l - b'_l}$. Since $|b_l - b'_l| = 1$ this gives integral values of τ, ζ when the system is solved over $\mathbb{Z}$. We now have that $\boldsymbol{\phi} = \mathbf{y}^{\tau}\mathbf{g}^{\zeta}$.

Consider next the equation system

$$\left\{ \begin{array}{l} d_{3,l} = f'_l - b_l \mathbf{y}^{d_{2,l}} s' \\ d'_{3,l} = f'_l - b'_l \mathbf{y}^{d'_{2,l}} s' \end{array} \right\} \ .$$

Note that $b_l \mathbf{y}^{d_{2,l}}$ is zero if $b_l = 0$ and non-zero otherwise, and that the inverse of $\mathbf{y}$ over $\mathbb{Z}_N$ can be found in polynomial time. Thus, the system is solvable. Let (ψ', ζ') be a solution and assume without loss that $b'_l = 0$. Then we have

$$F_{2,l} = y_N^{d_{3,l}} \omega^{\mathbf{y}^{d_{2,l}}} = y_N^{\psi' - \mathbf{y}^{d_{2,l}}\zeta'} \omega^{\mathbf{y}^{d_{2,l}}} = y_N^{\psi' - \mathbf{y}^{\psi-\zeta}\zeta'} \omega^{\mathbf{y}^{\psi-\zeta}}$$
$$F_{2,l} = y_N^{d'_{3,l}} g_N^{\mathbf{y}^{d'_{2,l}}} = y_N^{\psi'} g_N^{\mathbf{y}^{d'_{2,l}}} = y_N^{\psi'} g_N^{\mathbf{y}^{\psi}}$$

Solving for ω gives $\omega = y_N^{\zeta'} g_N^{\mathbf{y}^{\zeta}}$. Finally, let (ε', τ') be the solution to

$$\left\{ \begin{array}{l} d_{4,l} = e'_l - b_l \mathbf{y}^{d_{2,l}} t' \\ d'_{4,l} = e'_l - b'_l \mathbf{y}^{d_{2,l'}} t' \end{array} \right\} .$$

Then we have

$$F_{1,l} = \theta^{\mathbf{y}^{d_{2,l}}} y_N^{d_{4,l}} g_N^{d_{3,l}} = \theta^{\mathbf{y}^{d_{2,l}}} y_N^{\varepsilon' - \mathbf{y}^{d_{2,l}}\tau'} g_N^{\psi' - \mathbf{y}^{d_{2,l}}\zeta'}$$
$$F_{1,l} = y_N^{d'_{4,l}} g_N^{d'_{3,l}} = y_N^{\varepsilon'} g_N^{\psi'}$$

Solving for θ gives $\theta = y_N^{\tau'} g_N^{\zeta'}$. We conclude that the protocol is special-sound. □

Protocol 13 (Knowledge of a Root of a Committed Value over $\mathbb{Z}_N$).
COMMON INPUT: $\mathbf{g}, \mathbf{y}, \mathbf{u}, \mathbf{v}, \mathbf{u}', \mathbf{v}', \mathbf{C} \in \mathrm{QR}_N$.
PRIVATE INPUT: *$s, t, s', t', s'', e \in [0, 2^{\kappa_3}N - 1]$ and $\mathbf{r} \in \mathrm{QR}_N$ such that $(\mathbf{u}, \mathbf{v}) = (\mathbf{g}^s \mathbf{y}^t, \mathbf{g}^t \mathbf{r})$, $(\mathbf{u}', \mathbf{v}') = (\mathbf{g}^{s'} \mathbf{y}^{t'}, \mathbf{g}^{t'} \mathbf{r}^e)$ and $\mathbf{C} = \mathbf{g}^{s''} \mathbf{y}^e$.*

1. *The prover chooses $a, b, f, h, i, j \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}N - 1]$ randomly and computes*

$$\mathbf{A_1} = \mathbf{g}^a \mathbf{y}^b \mathbf{u}^e \tag{3.19}$$
$$\mathbf{A_2} = \mathbf{g}^b \mathbf{v}^e \tag{3.20}$$
$$\mathbf{B_1} = \mathbf{g}^f \mathbf{y}^h \mathbf{u}^i \tag{3.21}$$
$$\mathbf{B_2} = \mathbf{g}^h \mathbf{v}^i \tag{3.22}$$
$$\mathbf{B_3} = \mathbf{g}^j \mathbf{y}^i \ . \tag{3.23}$$

   Then it hands $\mathbf{A_1}, \mathbf{A_2}, \mathbf{B_1}, \mathbf{B_2}, \mathbf{B_3}$ to the verifier. The following protocols are executed in parallel with the protocol below:

   a) *Protocol 9 parameterized with $z = (2^{\kappa_2}N)^2 + 2^{\kappa_1+\kappa_2+\kappa_3}N$ on public input $\mathbf{g}$, $\mathbf{y}$, $(\mathbf{A_1}, \mathbf{A_2})$, $(\mathbf{u}', \mathbf{v}')$ and private input $\mathbf{r}^e$ where the secret exponents are $(se + a, te + b)$ and (s', t').*
   b) *Protocol 8 on public input $\mathbf{g}, \mathbf{y}, (\mathbf{u}, \mathbf{v})$ and private input $s, t, \mathbf{r}$.*
   c) *Protocol 8 on public input $\mathbf{g}, \mathbf{y}, (\mathbf{u}', \mathbf{v}')$ and private input $s', t', \mathbf{r}^e$.*

2. *The verifier chooses $c \in [0, 2^{\kappa_2} - 1]$ randomly and hands it to the prover.*

3. *The prover computes*

$$\begin{aligned} d_1 &= ca + f \mod 2^{\kappa_1+\kappa_2+\kappa_3}N & (3.24)\\ d_2 &= cb + h \mod 2^{\kappa_1+\kappa_2+\kappa_3}N & (3.25)\\ d_3 &= ce + i \mod 2^{\kappa_1+\kappa_2+\kappa_3}N & (3.26)\\ d_4 &= cs'' + j \mod 2^{\kappa_1+\kappa_2+\kappa_3}N \ . & (3.27) \end{aligned}$$

4. *The verifier checks that*

$$\begin{aligned} \mathbf{A_1}^c \cdot \mathbf{B_1} &= \mathbf{g}^{d_1}\mathbf{y}^{d_2}\mathbf{u}^{d_3} & (3.28)\\ \mathbf{A_2}^c \cdot \mathbf{B_2} &= \mathbf{g}^{d_1}\mathbf{y}^{d_2}\mathbf{v}^{d_3} & (3.29)\\ \mathbf{C}^c \cdot \mathbf{B_3} &= \mathbf{g}^{d_4}\mathbf{y}^{d_3} \ . & (3.30) \end{aligned}$$

Lemma 3.6.19. *Protocol 13 is a* $[0, 2^{\kappa_2} - 1]$*-*Σ*-protocol with completeness* $1 - 10 \cdot 2^{-\kappa_1}$.

Proof. The verifier rejects if one of the three subprotocols fail or there is an overflow in the computation of d_1, d_2, d_3 or d_4. Each of the subprotocols has a probability of failure of $2 \cdot 2^{-\kappa_1}$, and the probability for an overflow for each d_i is $2^{-\kappa_1}$. The union bound then gives a completeness of at least $1 - 10 \cdot 2^{-\kappa_1}$.

Extraction of $s, t, s', t', \mathbf{r}$ follows from Lemmas 3.6.14 and 3.6.15. Extraction of s'' and e, assuming two lists ($\mathbf{A_1}$, $\mathbf{A_2}$, $\mathbf{B_1}$, $\mathbf{B_2}$, $\mathbf{B_3}$, c, d_1, d_2, d_3, d_4) and ($\mathbf{A_1}$, $\mathbf{A_2}$, $\mathbf{B_1}$, $\mathbf{B_2}$, $\mathbf{B_3}$, c', d_1', d_2', d_3', d_4') satisfying equations in Step 4, $c \neq c'$, is by solving equations in Step 3 to get ζ, ε such that $\mathbf{C} = \mathbf{g}^{\zeta}\mathbf{y}^{\varepsilon}$. By Lemma 3.6.13. ζ and ϵ are integers. Thus the protocol is special-sound.

We now show that the protocol is zero-knowledge. The protocol can be simulated by choosing randomly $a, b, a', b' \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}N]$ and setting $\mathbf{A_1} = \mathbf{g}^a\mathbf{y}^b$, $\mathbf{A_2} = \mathbf{g}^{a'}\mathbf{y}^{b'}$. Then we pick at random $d_1, d_2, d_3, d_4 \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}N - 1]$ and $c \in [0, 2^{\kappa_2} - 1]$. $\mathbf{B_1}, \mathbf{B_2}, \mathbf{B_3}$ can then be computed from the equations in Step 4. This distribution is equal to the distribution from an honest execution of the protocols. The subprotocols can be simulated using the same c according to Lemmas 3.6.14 and 3.6.15. □

Protocol 14 (Equality of Exponents of Committed Values over $\mathbb{Z}_N$).
COMMON INPUT: $\mathbf{g}, \mathbf{y}, \mathbf{h}, \mathbf{u}, \mathbf{v}, \mathbf{C} \in \mathrm{QR}_N$
PRIVATE INPUT: $r, s, t, w \in [0, 2^{\kappa_3}N - 1]$ *such that* $(\mathbf{u}, \mathbf{v}) = (\mathbf{g}^r\mathbf{y}^s, \mathbf{g}^s\mathbf{h}^w)$ *and* $\mathbf{C} = \mathbf{g}^w\mathbf{y}^t$.

1. *The prover chooses* $a, b, e, f \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}N - 1]$*, sets* $(\boldsymbol{\mu}, \boldsymbol{\nu}) = (\mathbf{g}^a\mathbf{y}^b, \mathbf{g}^b\mathbf{h}^e)$ *and* $\mathbf{B} = \mathbf{g}^e\mathbf{y}^f$ *and hands* $(\boldsymbol{\mu}, \boldsymbol{\nu}), \mathbf{B}$ *to the verifier.*

2. *The verifier randomly chooses* $c \in [0, 2^{\kappa_2} - 1]$ *and hands it to the prover.*

3. *The prover computes*

$$\begin{aligned} d_1 &= cr + a \mod 2^{\kappa_1+\kappa_2+\kappa_3}N \ , \\ d_2 &= cs + b \mod 2^{\kappa_1+\kappa_2+\kappa_3}N \ , \\ d_3 &= ct + e \mod 2^{\kappa_1+\kappa_2+\kappa_3}N \ , \\ d_4 &= cw + f \mod 2^{\kappa_1+\kappa_2+\kappa_3}N \end{aligned}$$

and hands (d_1, d_2, d_3, d_4) *to the verifier.*

4. *The verifier checks that* $\mathbf{u}^c\boldsymbol{\mu} = \mathbf{g}^{d_1}\mathbf{y}^{d_2}$, $\mathbf{v}^c\boldsymbol{\nu} = \mathbf{g}^{d_2}\mathbf{h}^{d_4}$ *and* $\mathbf{C}^c\mathbf{B} = \mathbf{g}^{d_4}\mathbf{y}^{d_3}$

Lemma 3.6.20. *Protocol 14 is a* $[0, 2^{\kappa_2}-1]$*-*Σ*-protocol with completeness* $1-4\cdot2^{-\kappa_1}$.

Proof. An honest verifier will convince the verifier except possibly when there is a reduction in the computation of d_1, d_2, d_3, or d_4. Since the probability of a reduction in the computation of any of these values is $2^{-\kappa_1}$, the union bound gives a completeness of at least $1 - 4 \cdot 2^{-\kappa_1}$.

Now we show that the protocol is special-sound. Assume we have two lists $(\boldsymbol{\mu}, \boldsymbol{\nu})$, $\mathbf{B}$, c, d_1, d_2, d_3, d_4 and $(\boldsymbol{\mu}, \boldsymbol{\nu})$, $\mathbf{B}$, c', d_1', d_2', d_3', d_4' with $c \neq c'$ both satisfying the equations of Step 4. Then we can compute $\rho, \zeta, \tau, \omega$ such that $(\mathbf{u}, \mathbf{v}) = (\mathbf{g}^\rho\mathbf{y}^\zeta, \mathbf{g}^\zeta\mathbf{h}^\omega)$ and $\mathbf{C} = \mathbf{g}^\omega\mathbf{y}^\tau$. By Lemma 3.6.13 $\rho, \zeta, \tau, \omega$ are all integers.

The simulator first chooses d_1, d_2, d_3, d_4 from $[0, 2^{\kappa_1+\kappa_2+\kappa_3} - 1]$ and c from $[0, 2^{\kappa_2} - 1]$. Then $\boldsymbol{\mu}$, $\boldsymbol{\nu}$ and $\mathbf{C}$ are computed by solving the equations of Step 4. This gives a distribution equal to that of an honest execution. □

The following is a protocol (parameterized on k and l) to show that a committed value can be written as $ka + l$ for some a.

Protocol 15 (A Committed Value Can Be Written as $ka + l$ **over** $\mathbb{Z}_N$**).**
COMMON INPUT: $\mathbf{g}, \mathbf{y}, \mathbf{C} \in \mathrm{QR}_N$.
PRIVATE INPUT: $a, t \in [0, 2^{\kappa_3}N - 1]$ *such that* $\mathbf{C} = \mathbf{g}^{ka+l}\mathbf{y}^t$.

1. *The prover selects* $e, f, h \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}N - 1]$, $i \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}kN - 1]$ *at random, computes*

$$\mathbf{A} = \mathbf{g}^a\mathbf{y}^e \tag{3.31}$$
$$\mathbf{B_1} = \mathbf{g}^f\mathbf{y}^h \tag{3.32}$$
$$\mathbf{B_2} = \mathbf{y}^i \tag{3.33}$$

and hands $(\mathbf{A}, \mathbf{B_1}, \mathbf{B_2})$ *to the verifier. Both prover and verifier computes* $\mathbf{C}' = \mathbf{g}^l\mathbf{A}^k/\mathbf{C}$.

2. *The verifier randomly chooses* $c \in [0, 2^{\kappa_2} - 1]$ *and hands it to the prover.*

3. *The prover computes*

$$\begin{aligned} d_1 &= ca + f \mod 2^{\kappa_1+\kappa_2+\kappa_3}N & (3.34)\\ d_2 &= ce + h \mod 2^{\kappa_1+\kappa_2+\kappa_3}N & (3.35)\\ d_3 &= c(ek - t) + i \mod 2^{\kappa_1+\kappa_2+\kappa_3}kN & (3.36)\end{aligned}$$

and hands (d_1, d_2, d_3) *to the verifier.*

4. *The verifier checks that* $\mathbf{A}^c\mathbf{B_1} = \mathbf{g}^{d_1}\mathbf{y}^{d_2}$ *and* $(\mathbf{C}')^c\mathbf{B_2} = y^{d_3}$.

Lemma 3.6.21. *Protocol 15 is an* $[0, 2^{\kappa_2} - 1]$*-*Σ*-protocol with completeness* $1 - 3 \cdot 2^{-\kappa_2}$.

Proof. The prover succeeds to convince the verifier unless there is an overflow in one of the computations of d_i. By the union bound, this probability is at most $3 \cdot 2^{\kappa_2}$, giving a completeness of $1 - 3 \cdot 2^{-\kappa_2}$.

For the extraction assume we have two lists $\mathbf{A}$, $\mathbf{B_1}$, $\mathbf{B_2}$, c, d_1, d_2, d_3 and $\mathbf{A}$, $\mathbf{B_1}$, $\mathbf{B_2}$, c', d_1', d_2', d_3', $c \neq c'$, satisfying the equations of Step 4. From this we can compute $\alpha, \varepsilon, \theta$ such that $\mathbf{A} = \mathbf{g}^\alpha\mathbf{y}^\varepsilon$ and $\mathbf{C}' = \mathbf{y}^\theta$. By Lemma 3.6.13 α, ε and θ are all integers. If we set $\tau = k\varepsilon - \theta$ it holds that $\mathbf{C} = \mathbf{g}^\varepsilon\mathbf{y}^\tau$. This concludes the extraction.

The verifier's view can be simulated by randomly choosing $v \in \mathbb{Z}_N$ and setting $\mathbf{A} = \mathbf{g}^v$. Then $c \in [0, 2^{\kappa_2} - 1]$ is chosen at random together with $d_1, d_2 \in [0, 2^{\kappa_1+\kappa_2}N - 1]$, $d_3 \in [0, 2^{\kappa_1+\kappa_2}N^2 - 1]$. Finally $\mathbf{B_1}, \mathbf{B_2}$ are computed from the equations in Step 4. This gives a distribution equal to that of an honest execution of the protocol. □

We also need the protocol that a committed value lies in an interval by Boudot. Instead of giving the complete protocol, we only give the interface and refer the reader to [9] for complete details.

Protocol Head 16 (A Committed Number Lies in an Interval).
COMMON INPUT: $\mathbf{g}, \mathbf{y}, \mathbf{C} \in \mathrm{QR}_N$ *and* $a, b \in \mathbb{Z}$.
PRIVATE INPUT: $s \in [a, b]$ *and* $r \in [0, 2^{\kappa_3}N - 1]$ *such that* $\mathbf{C} = \mathbf{g}^s\mathbf{y}^r$.

Lemma 3.6.22 (cf. [9]). *Protocol 16 is a* $\{0, 1\}^{\kappa_2}$*-*Σ*-protocol with perfect completeness.*

From these building blocks we can now present the proof that a committed signature is valid.

Protocol 17 (Validity of Committed Signature from Hash).
COMMON INPUT: $\mathbf{g}, \mathbf{y}, \mathbf{h}, \mathbf{z}, \mathbf{u}, \mathbf{v}, \mathbf{u}', \mathbf{v}', \mathbf{C}, \mathbf{C}' \in \mathrm{QR}_N, N, H^{\mathsf{Sh}}, e' \in [2^\kappa, 2^{\kappa+1} - 1]$.
PRIVATE INPUT: $r, s, r', s', t, t' \in [0, 2^{\kappa_3}N - 1]$, $e \in [2^\kappa, 2^{\kappa+1} - 1]$, *and* $w_\alpha \in \mathbb{Z}_{q_2}$

such that

$$\begin{aligned}(\mathbf{u},\mathbf{v}) &= (\mathbf{g}^s\mathbf{y}^r, \mathbf{g}^r\boldsymbol{\sigma})\\ (\mathbf{u}',\mathbf{v}') &= (\mathbf{g}^{s'}\mathbf{y}^{r'}, \mathbf{g}^{r'}\boldsymbol{\sigma}')\\ \mathbf{C} &= \mathbf{y}^t\mathbf{g}^e\\ \mathbf{C}' &= \mathbf{y}^{t'}\mathbf{g}^{w_\alpha}\end{aligned}$$

and the signature is valid, i.e., $\mathsf{Vf}^{\mathsf{cs}}(\mathrm{id}, H^{\mathsf{Sh}}, N, \mathbf{h}, \mathbf{z}, e', w_\alpha, (e, \boldsymbol{\sigma}, \boldsymbol{\sigma}')) = 1$.

1. *Let* $\mathbf{z}'$ *denote* $(\boldsymbol{\sigma}')^{e'}\mathbf{h}^{-w_\alpha}$. *The prover chooses* ζ, τ, ζ', τ', ζ'', τ'', ζ''', τ''', ζ'''', $\tau'''' \in [0, 2^{\kappa_3}N-1]$ *and sets*

$$\begin{aligned}(\boldsymbol{\mu},\boldsymbol{\nu}) &= (\mathbf{g}^{\zeta}\mathbf{y}^{\tau}, \mathbf{g}^{\tau}\boldsymbol{h}^{-w_\alpha})\ ,\\ (\boldsymbol{\mu}',\boldsymbol{\nu}') &= (\mathbf{g}^{\zeta'}\mathbf{y}^{\tau'}, \mathbf{g}^{\tau'}\mathbf{z}')\ ,\\ (\boldsymbol{\mu}'',\boldsymbol{\nu}'') &= (\mathbf{g}^{\zeta''}\mathbf{y}^{\tau''}, \mathbf{g}^{\tau''}\boldsymbol{\sigma}^{e})\ ,\\ (\boldsymbol{\mu}''',\boldsymbol{\nu}''') &= (\mathbf{g}^{\zeta'''}\mathbf{y}^{\tau'''}, \mathbf{y}^{\zeta'''}H^{\mathsf{Sh}}_{(N,\mathbf{g})}(\mathbf{z}'))\ ,\\ (\boldsymbol{\mu}'''',\boldsymbol{\nu}'''') &= (\mathbf{g}^{\zeta''''}\mathbf{y}^{\tau''''}, \mathbf{y}^{\zeta''''}\mathbf{h}^{-H^{\mathsf{Sh}}_{(N,\mathbf{g})}(\mathbf{z}')})\ .\end{aligned}$$

   Then it hands $(\boldsymbol{\mu},\boldsymbol{\nu})$, $(\boldsymbol{\mu}',\boldsymbol{\nu}')$, $(\boldsymbol{\mu}'',\boldsymbol{\nu}'')$, $(\boldsymbol{\mu}''',\boldsymbol{\nu}''')$, *and* $(\boldsymbol{\mu}'''',\boldsymbol{\nu}'''')$ *to the verifier.*

2. *The following protocols are run in parallel (using a common challenge* $c \in [0, 2^{\kappa_2}-1]$*):*

   a) *Protocol 8 on the public input* $\mathbf{g}, \mathbf{y}, (\mathbf{u},\mathbf{v})$ *and private input* $s, r, \boldsymbol{\sigma}$ *to show that the prover knows how to open the commitment* $(\mathbf{u},\mathbf{v})$.

   b) *Protocol 8 on the public input* $\mathbf{g}, \mathbf{y}, (\mathbf{u}',\mathbf{v}')$ *and private input* $s', r', \boldsymbol{\sigma}'$ *to show that the prover knows how to open the commitment* $(\mathbf{u}',\mathbf{v}')$.

   c) *Protocol 14 on public input* $\mathbf{g}, \mathbf{y}, \mathbf{h}, (\boldsymbol{\mu},\boldsymbol{\nu}), (\mathbf{C}')^{-1}$ *and private input* ζ, τ, t', w_α *to show that* $(\boldsymbol{\mu},\boldsymbol{\nu})$ *is a commitment of* $\mathbf{h}^{-w_\alpha}$.

   d) *Protocol 9 with* $z = N + N2^{\kappa+1}$ *on public input* $\mathbf{g}$, $\mathbf{y}$, $(\boldsymbol{\mu}', \boldsymbol{\nu}')$, $(\boldsymbol{\mu}(\mathbf{u}')^{e'}, \boldsymbol{\nu}(\mathbf{v}')^{e'})$ *and private input* ζ', τ', $\zeta + se'$, $\tau + re'$ *to show that* $(\boldsymbol{\mu}',\boldsymbol{\nu}')$ *is a commitment of* $\mathbf{z}'$.

   e) *Protocol 13 on public input* $\mathbf{g}, \mathbf{y}, (\mathbf{u},\mathbf{v}), (\boldsymbol{\mu}'',\boldsymbol{\nu}''), \mathbf{C}$ *and private exponents* s, r, ζ'', τ'', t. *This shows that* $(\boldsymbol{\mu}'',\boldsymbol{\nu}'')$ *hides the value hidden in* $(\mathbf{u},\mathbf{v})$ *to the power of the value hidden in* $\mathbf{C}$.

   f) *Protocol 11 on public input* $\mathbf{g}, \mathbf{y}, \mathbf{g}, (\boldsymbol{\mu}',\boldsymbol{\nu}'), (\boldsymbol{\mu}''',\boldsymbol{\nu}'''), g_N, y_N$ *and* ζ', τ', ζ''', τ''' *as private input to show that* $(\boldsymbol{\mu}''',\boldsymbol{\nu}''')$ *is a commitment of a Shamir hash of* $\mathbf{z}'$.

   g) *Protocol 11 on public input* $\mathbf{g}, \mathbf{y}, \mathbf{h}^{-1}, (\boldsymbol{\mu}''',\boldsymbol{\nu}'''), (\boldsymbol{\mu}'''',\boldsymbol{\nu}''''), g_N, y_N$ *and private input* ζ''', τ''', ζ'''', τ'''' *to show that* $(\boldsymbol{\mu}'''',\boldsymbol{\nu}'''')$ *commits to* $\mathbf{h}$ *to the power of* $H^{\mathsf{Sh}}(\mathbf{z}')$.

h) *Protocol 10 with* $z = 2N$ *on public input* $\mathbf{g}, \mathbf{y}, (\boldsymbol{\mu}''\boldsymbol{\mu}'''', \boldsymbol{\nu}''\boldsymbol{\nu}''''), \mathbf{z}$ *with private input* $\zeta'' + \zeta'''', \tau'' + \tau''''$ *to finally show that the signature is valid.*

i) *Protocol 15 with* $k = 4$ *and* $l = 3$ *on public input* $\mathbf{g}, \mathbf{y}, \mathbf{C}$ *and private input* e, t *to prove that* e *is odd and different from* e'*.*

j) *Protocol 16 on public input* $\mathbf{g}, \mathbf{y}, \mathbf{C}, 2^{\kappa}, 2^{\kappa+1} - 1$*. and private input* e, t *to prove that* e *belongs to the correct interval.*

Lemma 3.6.23. *Protocol 17 is* $[0, 2^{\kappa_2} - 1]$*-*Σ*-protocol with completeness* $1 - (4\kappa_2 + 24)2^{-\kappa_2}$*.*

Proof. By using the union bound on the probability of failure of the subprotocols, we get a completeness of at least $1 - (4\kappa_2 + 24)2^{-\kappa_2}$.

We now describe the extraction to show that the protocol is special-sound. By Lemmas 3.6.14, 3.6.20, and 3.6.19 we can extract $\rho, \zeta, \rho', \zeta', \tau, \tau', \varepsilon$ such that $\mathbf{u} = \mathbf{g}^{\rho}\mathbf{y}^{\zeta}$, $\mathbf{u}' = \mathbf{g}^{\rho'}\mathbf{y}^{\zeta'}$, $\mathbf{C} = \mathbf{g}^{\tau}\mathbf{y}^{\varepsilon}$, and $\mathbf{C}' = \mathbf{g}^{\tau'}\mathbf{y}^{\omega}$ from Steps 2a, 2b, 2c, and 2e. Now also $\boldsymbol{\varsigma}$, $\boldsymbol{\varsigma}'$ such that $\mathbf{v} = \mathbf{g}^{\zeta}\boldsymbol{\varsigma}$ and $\mathbf{v}' = \mathbf{g}^{\zeta'}\boldsymbol{\varsigma}'$ can be computed.

It now remains to be shown that $(\varepsilon, \boldsymbol{\sigma}, \boldsymbol{\sigma}')$ is a valid signature of ω. We do that by checking that the two steps of Algorithm 7 holds. Steps 2i and 2j ensure that Step 1 of the verification algorithm for Cramer-Shoup signatures holds. From Steps 2c and 2d and Lemmas 3.6.20 and 3.6.15 it follows that $(\boldsymbol{\mu}', \boldsymbol{\nu}')$ is a commitment of $\mathbf{z}' = (\boldsymbol{\sigma}')^{e'}\mathbf{h}^{-\omega}$. Step 2e and Lemma 3.6.19 give that $(\boldsymbol{\mu}'', \boldsymbol{\nu}'')$ is a commitment of $\mathbf{h}^{\varepsilon}$, and Steps 2f, 2g with Lemma 3.6.17 shows that $(\boldsymbol{\mu}'''', \boldsymbol{\nu}'''')$ commits to $\mathbf{h}^{H^{\mathsf{Sh}}(\mathbf{z}')}$. Finally Step 2g shows that the equality of Step 2 of Algorithm 7 holds. Hence $(\varepsilon, \boldsymbol{\sigma}, \boldsymbol{\sigma}')$ is valid signature of ω.

Since all subprotocols are Σ-protocols, the constructed protocol can also be simulated. Also all protocols are either of type $[0, 2^{\kappa_2} - 1]$-Σ or $\{0, 1\}^{\kappa_2}$-Σ. Since there is a natural bijection between $[0, 2^{\kappa_2} - 1]$ and $\{0, 1\}^{\kappa_2}$, the resulting protocol is a $[0, 2^{\kappa_2} - 1]$-Σ-protocol.

□

Final Protocol

We are finally ready to give the complete proof of a correct signature corresponding to the proof in Step 3 of Algorithm 9. The common input consists of a chain of cryptotexts and commitments of a $\mathcal{SS}^{\mathsf{cs}}$ signature of the public keys corresponding to the path of the signer in the tree.

Protocol 18 (Valid $\mathcal{HGS}$ Signature).
COMMON INPUT:

- $\mathbf{g}, \mathbf{y}, \mathbf{h}, \mathbf{z} \in \mathrm{QR}_N$
- $e' \in [2^{\kappa}, 2^{\kappa+1} - 1]$

- $(u_l, v_l)_{l=0}^{\delta-1} \in G_{q_3}^{2\delta}$
- $C_\delta \in G_{q_3}^4$
- $(\mathbf{u}, \mathbf{v}) \in \mathrm{QR}_N^2$
- $(\mathbf{u}', \mathbf{v}') \in \mathrm{QR}_N^2$
- $\mathbf{C} \in \mathrm{QR}_N$
- $H = (h_1, \ldots, h_\delta) \in G_{q_2}^\delta$
- $g_1, y_1 \in G_{q_1}$
- $g_2, y_2 \in G_{q_2}$
- $g_3, y_3 \in G_{q_3}$
- $y_{\alpha_0} \in G_{q_3}$
- $Y \in G_{q_3}^5$.

Private Input:

- $(\tau_0, \ldots, \tau_\delta) \in \mathbb{Z}_{q_3}^{\delta+1}$
- $(\gamma_1, \ldots, \gamma_\delta) \in G_{q_3}^\delta$
- $\varepsilon \in [2^{\kappa/2}, 2^{\kappa/2+1} - 1]$,
- $(\tau, \zeta, \tau', \zeta', \psi) \in [0, 2^{\kappa_3} N - 1]^6$

such that

$$\begin{aligned}
\gamma_0 &= y_{\alpha_0} \\
(u_l, v_l) &= E_{(\gamma_l, g_3)}(\gamma_{l+1}, \tau_l) \textit{ for } l = 0, \ldots, \delta - 1 \\
C_\delta &= E_Y^{\mathsf{cs}}(\gamma_\delta, \tau_\delta) \\
\mathbf{u} &= \mathbf{g}^\zeta \mathbf{y}^\tau \\
\mathbf{u}' &= \mathbf{g}^{\zeta'} \mathbf{y}^{\tau'} \\
\mathbf{C} &= \mathbf{g}^\varepsilon \mathbf{y}^\psi
\end{aligned}$$

and $\mathsf{Vf}^{\mathsf{cs}}(H, H_{(\mathbf{g},N)}^{\mathsf{Sh}}, N, \mathbf{h}, \mathbf{z}, e, (\gamma_1, \ldots, \gamma_\delta), (\varepsilon, \mathbf{v}/\mathbf{y}^\tau, \mathbf{v}'/\mathbf{y}^{\tau'})) = 1$.

1. *The prover randomly selects* $s, t \in \mathbb{Z}_{q_2}$, $t' \in [0, 2^{\kappa_3} N - 1]$ *and computes* $(\mu, \nu) = (g_1^t y_1^s, y_1^t g_1^{H(\gamma_1, \ldots, \gamma_\delta)})$, $\mathbf{C}' = \mathbf{g}^{H(\gamma_1, \ldots, \gamma_\delta)} \mathbf{y}^{t'}$. *The prover hands* (μ, nu) *and* $\mathbf{C}'$ *to the verifier.*

2. The following protocols are executed in parallel

a) Protocol 6 on public input g_3, y_3, y_{α_0}, g_2, y_2, g_1, y_1, H, $(u_l, v_l)_{l=0}^{\delta-1}$, (μ, ν), Y, C_δ and private input $\tau_0, \ldots, \tau_l$, $\gamma_1, \ldots, \gamma_\delta$, s, t.

b) Protocol 7 on public input $\mathbf{g}, \mathbf{y}, \mathbf{C}', g_1, y_1, \nu$ and private input $H(\gamma_1, \ldots, \gamma_\delta)$, t, t'.

c) Protocol 17 on public input $\mathbf{g}, \mathbf{y}, (\mathbf{u}, \mathbf{v}), (\mathbf{u}', \mathbf{v}'), \mathbf{C}, \mathbf{C}'$ and private input $\tau, \zeta, \tau', \zeta', \psi, s, \varepsilon$.

Lemma 3.6.24. *Protocol 18 is a $[0, 2^{\kappa_2} - 1] \times \mathbb{Z}_{q_2}$-$\Sigma$-protocol with completeness $1 - (4\kappa_2 + 27)2^{-\kappa_2}$.*

Proof. Since the completeness of Steps 2a, 2b, 2c are 1, $1 - 3 \cdot 2^{-\kappa_2}$ and.$1 - (4\kappa_2 + 24)2^{-\kappa_2}$ respectively, the union bound gives a completeness of at least $1 - (4\kappa_2 + 27)2^{-\kappa_2}$ for the constructed protocol.

Extraction of $\tau_0, \ldots, \tau_\delta$ and $\gamma_1, \ldots, \gamma_\delta$ with the necessary properties follows from Lemma 3.6.9, from which is also follows that ν is a commitment of $H(\gamma_1, \ldots, \gamma_\delta)$. Extraction of $\tau, \zeta, \tau', \zeta', \varepsilon, \psi$ follows from Lemma 3.6.23. By Lemma 3.6.23 we can also extract ξ such that

$$\mathsf{Vf}^{\mathsf{cs}}(\mathrm{id}, H^{\mathsf{Sh}}_{(\mathbf{g},N)}, N, \mathbf{h}, \mathbf{z}, e, \xi, (\varepsilon, \mathbf{v}/\mathbf{y}^{\tau}, \mathbf{v}'/\mathbf{y}^{\tau'})) = 1$$

and $\mathbf{C}'$ is a commitment of ξ.

Finally from Step 2b and Lemma 3.6.12 it follows that $\mathbf{C}'$ and ν are commitments of the same number, i.e., $\xi = H(\gamma_1, \ldots, \gamma_\delta)$. This implies that

$$\mathsf{Vf}^{\mathsf{cs}}(H, H^{\mathsf{Sh}}_{(\mathbf{g},N)}, N, \mathbf{h}, \mathbf{z}, e, (\gamma_1, \ldots, \gamma_\delta), (\varepsilon, \mathbf{v}/\mathbf{y}^{\tau}, \mathbf{v}'/\mathbf{y}^{\tau'})) = 1 \ .$$

Therefore we can conclude that the protocol is special-sound.

The protocol can be simulated since it is constructed from subprotocols that can be simulated. □

3.7 An Alternative Construction

In this section we sketch an alternative provably secure construction. Let $\mathcal{SS} = (\mathsf{Kg}, \mathsf{Sig}, \mathsf{Vf})$ be a signature scheme. For each group manager M_α (or signer S_α), $(spk_\alpha, ssk_\alpha) \leftarrow \mathsf{Kg}(1^\kappa)$, and $(pk_\alpha, sk_\alpha) \leftarrow \mathsf{GMKg}(1^\kappa)$ are generated. Then for each child α of $\beta \in T$, $\sigma_\beta(\alpha) = \mathsf{Sig}_{ssk_\beta}(pk_\alpha, spk_\alpha)$ is computed. Finally, for each $\alpha \in T \backslash \{\rho\}$ set $hpk(\alpha) = (spk_\alpha, pk_\alpha, \sigma_\beta(\alpha))$, where $\alpha \in \beta$, and $hsk(\alpha) = (sk_\alpha)$. For the root ρ we set $hpk(\rho) = (spk_\rho, pk_\rho)$ and $hsk(\rho) = (ssk_\rho, sk_\rho)$.

Consider a signer S_α corresponding to a path $\alpha_0, \ldots, \alpha_\delta$, where $\alpha_0 = \rho$ and $\alpha_\delta = \alpha$. To sign a message m the signer computes

$$C = (C_0, \ldots, C_\delta) = (E_{pk_0}(\sigma_{\alpha_0}(\alpha_1)), \ldots, E_{pk_{\delta-1}}(\sigma_{\alpha_{\delta-1}}(\alpha_\delta)), E_{pk_\delta}(\mathsf{Sig}_{ssk_\alpha}(m))) \ ,$$

and provides a NIZK π that C is formed as above with $pk_0 = pk_\rho$ and $\alpha_0 = \rho$. The signature consists of the pair (C, π).

To verify a signature (C, π) the verifier simply checks the NIZK π. To open a signature, a group manager M_β on depth l first verifies the signature. If it is not valid, it returns $\perp$. Otherwise it computes $\sigma = D_{sk_\beta}(C_l)$. If σ is equal to $\sigma_\beta(\alpha)$ for some $\alpha \in \beta$, then it returns α and otherwise $\perp$.

This construction is a strict generalization of the construction in [7] except that we require that the cryptosystem used is cross-indistinguishable. The construction is provably secure under the existence of a family of trapdoor permutations. However, as part of the proof we must essentially redo the analysis of the CCA2-secure cryptosystem of Sahai [62], and the group signature scheme of Bellare et al. [7], which makes the proof more complex than the proof for the construction we detail in this chapter.

A potential advantage of this scheme is that key generation need not be performed centrally. Each group manager M_β could also be given the secret signature key ssk_β which allows it to generate (spk_α, pk_α) and (ssk_α, sk_α) for a child M_α or S_α by itself. Thus, a group manager could issue keys without interacting with any other group manager. However, as we will see in the next section, it is far from obvious how to define the security of such a scheme.

3.8 Eliminating the Trusted Key Generator

We have defined hierarchical group signatures using a trusted key generator. This is a natural first step when trying to understand a new notion, but there are situations where one would like a (hierarchical) group signature scheme *without* a trusted party.

If there exists a set of parties of which the majority is trusted, general multiparty techniques can be used to replace the trusted key generator by the secure evaluation of a function. However, this introduces a trust hierarchy that is inconsistent with the hierarchy of the group managers and signers.

Consider now the security of the construction when there is no trusted key generator. In this case hierarchical anonymity and hierarchical traceability (full anonymity and full traceability) do not suffice to ensure security. The problem is that the experiments only consider the advantage of an adversary when all keys are generated honestly. Thus, all bets are off if this is not the case. It is however not clear what a definition of security for (hierarchical) group signatures without a trusted key generator should look like.

The adversary should probably have the power to choose its keys adaptively, based on the keys and signatures of honest parties. There are many subtle issues. For example, without a trusted key generator the default for hierarchical group signatures is that not only trees but general acyclic graphs of group managers are allowed. Is this what we want? If only trees are supposed to be allowed, certificates must embed additional information that restricts how a certificate chain may look and the NIZK must consider this as well. Other interesting questions are: Is there a well defined tree? Do all participants know what the tree look like? Who generates the common random string used by the NIZKs?

We believe that the alternative construction described above is well suited to a setting without a trusted key generator. However, without a rigorous definition of security we cannot claim anything, and currently there exists as far as we know not even a rigorous definition of security for group signatures without a trusted party, even less so for hierarchical group signatures.

3.9 Conclusion

We have introduced and formalized the notion of hierarchical group signatures and given two constructions. The first construction is provably secure under general assumptions, whereas the second is provably secure under the DDH assumption, the strong RSA assumption and the 4-Cunningham chain assumption in the random oracle model.

Although the latter construction is practical, i.e., it can be implemented and run on modern workstations, it is still relatively slow. Thus, an interesting open problem is to find more efficient constructions of hierarchical group signatures.

Chapter 4

On the Security of Non-RSA EMV Payment Cards

4.1 Introduction

A large part of today's electronic purchases are made with different kinds of payment cards. The majority of the cards used today have a magnetic stripe where the card data is stored. Over the last years, card skimming, where the content of the magnetic stripe is copied, has become a major problem. The countermeasure is to move from the magnetic stripe to smart-cards where the data is stored on a chip instead. To make sure also smart-card based payment cards will have the same global acceptance as the magnetic stripe, Europay, MasterCard and Visa have together developed the EMV specification.

The preparations for moving from payment cards based on magnetic stripe to smart-card based cards have been going on for more than ten years. Some card issuers have already converted their card base to EMV smart-cards, and more are about to make the switch.

The base for EMV smart-cards is the EMV specifications [29], which define the protocol between the card and the terminal. Payment organizations, in particular Visa and MasterCard, have developed their own extensions to the EMV specifications [48, 47, 73].

In this chapter we will examine a potential problem in the configuration of an EMV card. In particular we will show how to avoid this problem with a card based on Visa's VSDC specification, and that it cannot be avoided when using MasterCard's M/Chip specification.

EMV specifies two possible security levels for cards, Static Data Authentication (SDA) and Dynamic Data Authentication (DDA). The difference lies in that DDA cards must support RSA, whereas an SDA card does not. The issue we discuss in this chapter relates only to SDA cards.

4.2 Smart-cards

A smart-card is a tiny computer with its own CPU and storage. The data stored on the card cannot be read or written directly but only through certain functions. This means that a smart-card may have keys that can be used for encryption but cannot be read in clear. Another possilibity is to have a PIN that must be entered before certain functions can be used. More information about smart-cards can be found in [38].

The fact that the data on the card can only be accessed through predefined functions means that we can define data to be *public* when it can be accessed in clear and *private* when it is used only for internal processing by the card. When analyzing protocols involving smart-cards, a reasonable security model is to assume the data can only be accessed and modified using the predefined functions. The weakness analyzed in this chapter follows that security model.

4.3 The EMV Specification

The EMV specification describes in detail the data flow between the card and the terminal during a transaction. The outcome of an execution the protocol is one of the following

1. Transaction is approved offline.
2. Transaction is denied offline.
3. Transaction is sent to the issuer for online authorization[1]

Since most transaction are either approved offline or sent online, we will consider only these two cases here. The principle is that a transaction can be approved offline *only* if *both* the card *and* the terminal agrees on it, but is sent online if at least one of them requests it.

Both Visa and MasterCard have written their own extensions to EMV. Here they define which of the public EMV parameters that can be used, and also what the internal behavior of the card should be. Visa calls their application VSDC [73]. MasterCard has published two separate documents, one giving the external interface in the form of minimum requirements [47] and one defining the internal behavior by describing the application M/Chip [48]. However, also MasterCard is moving towards a unified document giving both internal and external details.

The Participants

The main participants in an EMV transaction are

[1]In this case the protocol also defines the behavior if no response is received from the issuer, but this will not be discussed here.

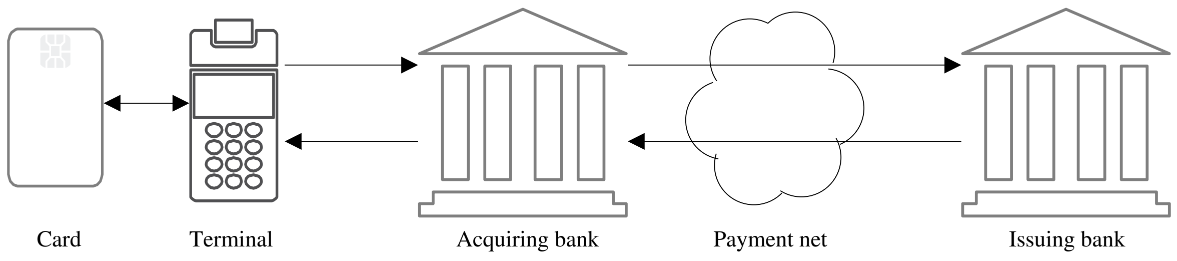


Figure 4.1: Data flow of an EMV transaction

- the card. The card is held by a customer (the cardholder) and contains information (both secret and public) set by the issuer. The card is identified by its card number.
- the terminal. The terminal is located at a merchant and is able to interact with cards. The terminal has internal settings defining when to allow offline transactions and when to require transactions to be processed online.
- the acquirer. The acquirer receives data from the terminal. In this principal description, the acquirer does nothing but forwarding the information to the issuer.
- the issuer. The issuer issues cards to his customers and processes transactions where his cards are involved.
- the payment organization. The payment organization is responsible for forwarding information between issuers and acquirers, and also for issuing public key certificates to the issuers.

In figure 4.1 a schematic overview is given.

Cryptographic Authentication

There are two different forms of authentication in EMV transactions, authentication between the card and the terminal and authentication between the card and the issuer.

Card-Terminal Authentication

In the very beginning of a transaction, the terminal verifies the authenticity of the card. This can be done either with *Static Data Authentication* or with *Dynamic Data Authentication*. In the first case, the card has a (static) signature on some subset of its parameters which is verified by the terminal. In the second case, the terminal verifies that the card also knows a private key by letting it sign a random nonce.

Card-Issuer Authentication

During a transaction, the card may generate one or two MACs. These MACs are generated with a symmetric key known only to the issuer and the card. Therefore the MAC can only be verified by the issuer and not by the terminal or the acquirer.

Card Configuration

SDA and DDA.

EMV gives the option of using low-cost cards without RSA capabilities as well as more expensive RSA enabled cards. Cards without RSA capabilities support only *Static Data Authentication* (SDA) whereas cards with RSA support can handle also *Dynamic Data Authentication* (DDA).

For both SDA and DDA, the issuer receives a certificate from the payment organization. The issuer certificate and the issuer public key, IPK, are stored publicly on the card. For SDA, the issuer signs a set of card parameters of his choice with his private key and places the signature on the card. The signature is called *Signed Static Application Data*, SSAD. In the case of DDA the card is given its own RSA key pair. The card private key is stored internally on the card, but the card public key is signed by the issuer. (Even if the card supports DDA, an SDA signature is usually still put on the card.)

Card parameters.

Apart from the keys and certificates mentioned above, several parameters describing under which circumstances to allow offline transactions are stored on the card. In this thesis we are only interested in one parameter, namely *Lower Consecutive Offline Limit* (LCOL). The LCOL gives the number of transactions that can be performed offline, i.e., without contacting the issuer.

Also the parameters *Application Transaction Counter* (ATC) and *Last Online Application Transaction Counter* (LATC) are stored on the card. The ATC contains the number of transaction the card has performed and the LATC holds the index of latest transaction that executed online. They are both initialized to zero.

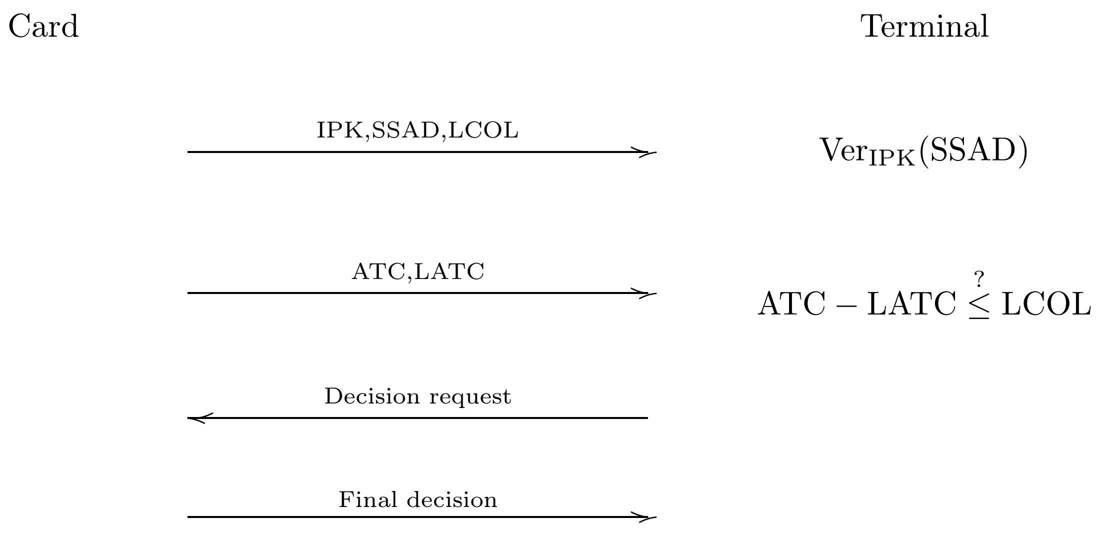
Symmetric keys.

When a card is issued, the issuer generates a symmetric key that is stored on the card and used to generate MACs transaction messages.[2] The key is also stored by issuer, but not disclosed to the merchants or the acquirers.

[2]Up to three other symmetric keys are also stored on the card, but since they play no role to our result, we ignore them.

The Protocol

The following is an overview of the protocol used for deciding the result of the transaction. Here we only describe the case of an SDA card.


Card
Terminal
IPK,SSAD,LCOL
$\mathrm{Ver}_{\mathrm{IPK}}(\mathrm{SSAD})$
ATC,LATC
$\mathrm{ATC} - \mathrm{LATC} \overset{?}{\leq} \mathrm{LCOL}$
Decision request
Final decision

1. The terminal reads all public data from the card.

2. The terminal receives transaction data, including the amount and currency of the transaction.

3. The terminal uses the public key of the payment organization to verify that the issuer certificate is valid and then uses the issuer public key (IPK) to verify that the SDA signature is valid (SSAD). If any of these steps fail, the processing is terminated.

4. The terminal reads the ATC and the LATC off the card and computes the number of offline transactions as LATC − ATC. It compares this value the *Lower Consecutive Offline Limit* (LCOL) read in step 2 and proceeds as follows:

   - If the number of offline transactions is lower than LCOL, the terminal makes a request for an offline transaction to the card.
   - If the number of offline transactions is equal to or higher than LCOL, the temrinal makes a request for an online transaction to the card.
   - If the LCOL is not present, the terminal makes a request for an offline transaction to the card.

5. After receiving the terminal's decision, the card may perform additional internal risk analysis before it makes the final decision. However, if the terminal requested an online transaction the card may not override the decision to make the transaction offline.

- If the card receives a request for an online transaction, it creates a MAC over transaction data and returns to the terminal. This data is sent to the issuer who verifies the authenticity of the MAC.
- If the card receives a request for an offline transaction, then depending on the result of the internal analysis, it either approves the transaction offline or requests an online authorization. In both cases a MAC is created.

4.4 The Problem

In this section we will describe the potential problem, and also how to avoid it where possible.

Making a Pure Online Card

In many cases, it is desirable to have a card that can only function online. There are two ways to achieve this:

- Set the LCOL to zero. This way the terminal will always make a transaction go online.
- Make the internal risk analysis of the card such that it always makes the decision to go online, regardless of what the terminal's decision is.

The most obvious reason to make a card online-only is to make sure the cardholder does not spend money he does not have. However, for an SDA card there is also another reason. Since the MAC cannot be verified offline, someone might copy the card, keeping the original SDA signature, but replace the symmetric key. Then a terminal would accept the card (since the SDA signature is valid), and when (and if) the issuer detects that the MAC is invalid, it is already too late.

The essence of the attack described here is to copy the card and modify the copy in such a way that it will allow offline transactions.

Copying an SDA Card

If we assume that the hardware is secure, the adversary can copy all the public data on a card, but not the internal data. Also, when the card is copied, he can modify data that has not been signed by the issuer, but if he changes the data included in the SDA signature the card will not be accepted. When a card is copied the adversary can change the internal behavior of the card by replacing the original program code by his own.

Making an Online Card Work Offline

As mentioned above, there are two ways to make a card online-only. Here we will discuss different attack scenarios depending on what method is used.

If the LCOL parameter is set to zero, then it may either be signed or not be signed with the SDA signature. If it is not signed, the adversary can simply copy the card and modify the LCOL to contain a non-zero value. The card will be accepted offline, since the SDA signature is still valid. He will not be able to copy the symmetric key, so the copy cannot be used online, but as long as the card only is used offline, it will work. In case the LCOL is signed, it cannot be changed and the attack does not work.

If the LCOL is not present on the card, but the internal risk analysis of the card is used to make all transaction go online, then the attack is a little bit different. When the adversary copies the card, he replaces the card application with an application that always accepts to make the transaction offline. Also here he cannot copy the symmtric key, but he will be able to use the card offline.

Note that for any of these attacks, the adversary only needs access to the card for a few seconds so that he can read all the public data. Since the commands for doing this are standardized, any card-reader could be used for this.

Copying an Offline-enabled Card

If the card has LCOL non-zero, but not in the SDA signature, the adversary can of course use a similar method to get an arbitrary number of offline transactions (with invalid MACs, making it impossible to tie the transaction to the card). However, even if LCOL is signed, the adversary can issue an attack similar to those described above. He can copy all the parameters on the card, but modify the card application to that it always responds that no offline transactions have been performed prior to the current. That way the terminal will always accept to make the transaction offline (since the number of offline transactions is lower than the LCOL) and the issuer will not be able to detect that the MAC is invalid.

Protecting Against the Attack

As we can see, the only way to make the card secure against the proposed attack is to set the LCOL to zero and include it in the SDA signature. In other words, there is no way of making a secure offline SDA card.[3]

However, the specifications for M/Chip [48] (used for MasterCard) don't allow the use of LCOL, leaving only card-based risk analysis for making a card online-only. As we have seen, such an approach is always susceptible to the attack by modifying the application. (The M/Chip specifications do define the LCOL, but only as private parameter used internally by the card.)

[3]This attack does not work for DDA cards.

It can be noted that inclusion of LCOL in the data signed with SDA is not in the published recommendations. One step for reducing the potential threat is to update the recommendations to include the LCOL and also note that it should be set to zero.

4.5 Conclusions and Recommendations

We have demonstrated how EMV cards with a certain configuration can be attacked, and we have also pointed out how to configure a card correctly to avoid this attack. We have seen that cards based on M/Chip cannot be configured in the proposed way, and are therefore always susceptible to the attack. Here the only solution is to move to (more expensive) DDA cards.

One possibility to solve the problem is the change the EMV specification so that a terminal always goes online when a non-DDA EMV card is used. Although the consequence is that issuers using low-cost card cannot benefit from the advantages of offline transaction, from a security perspective this approach would be the most efficient.

Chapter 5

Lattices With Many Cycles Are Dense

5.1 Introduction

The interest in the computational complexity of lattice problems started in the beginning of the 1980s, when van Emde Boas published the first **NP**-completeness result for lattice problems [71]. Several hardness results for different variants of this problems and for different subsets of lattices have followed. One such way of classifying lattices is according to the cycle structure of Abelian group $\mathbb{Z}^n/\Lambda$, which is the main focus of this chapter. Previous results on the complexity of lattice problems that either explicitly or implicitly consider lattices with a certain cycle structure include [1, 13, 56, 67].

The group $\mathbb{Z}^n/\Lambda$ is finite if $\Lambda \subseteq \mathbb{Z}^n$ and full-dimensional. One way to visualize this group is to divide $\mathbb{Z}^n$ into the parallelepipeds spanned by a basis and consider two points equivalent if they lie in the same position in their respective parellelepipeds. In Figure 5.1 one such equivalence class of points is shown. Note how this can be considered a generalization of reduction modulo an integer over $\mathbb{Z}$. It is easy to see that $\mathbb{Z}^n/\Lambda$ is a group under addition, and since addition is commutative, the group is abelian. As with any abelian group, it is isomorphic to the cartesian product of cyclic groups. By writing the cycle lengths in increasing order so that the length of cycle i divides the length of cycle $i+1$, we get a unique representation. For example, instead of writing $\mathbb{Z}_3 \times \mathbb{Z}_5$ we write $\mathbb{Z}_{15}$, and instead of $\mathbb{Z}_2 \times \mathbb{Z}_3 \times \mathbb{Z}_3$ we write $\mathbb{Z}_3 \times \mathbb{Z}_6$.

There are two reasons to study the hardness of certain lattice problems in different subclasses of lattices rather than for general lattices. The first reason is purely theoretical – it gives us a better understanding of how the computational complexity of lattice problems behaves if we restrict ourselves to certain lattice classes. The second reason is more practical – most hardness results are worst-case results for general lattices. The lattices that appear in many applications may have

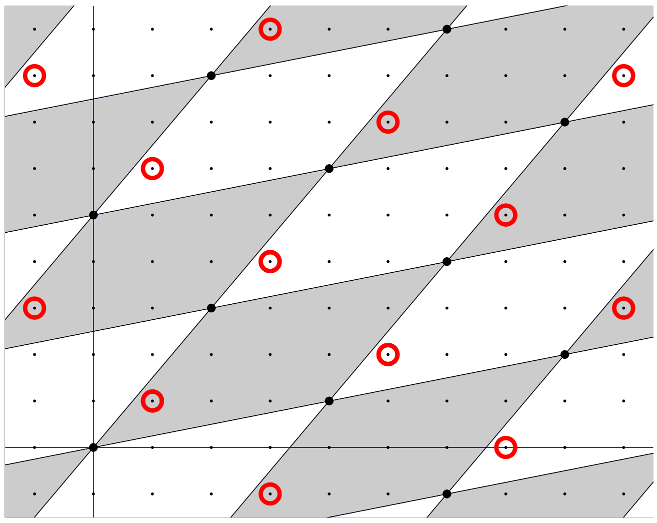


Figure 5.1: Points that are equivalent modulo a lattice

certain structural properties. It would be desired to have results that show that these properties cannot be used to solve lattice problem more efficiently.

The first result on the cycle structure was published by Paz and Schnorr [56]. In their paper it is shown that any lattice can be approximated arbitrarily well by a lattice with one cycle. In other words, the lattices with one cycle form a hard core. On the other hand, the lattices Cai and Nerurkar [13] prove to be hard in the improved version of Ajtai [1] have up to n/c cycles. Although the results are different in nature (the latter is not an **NP**-hardness result), it is interesting to note that they give hardness results for lattices with different cycle structure. This gives rise to the question of the role of the cycle structure in the complexity of lattice problems.

The influence of the cycle structure on the hardness of lattice problems has practical implications. For some crypto systems (e.g., NTRU [37]) there are attacks based on finding short vectors in certain lattices. The lattices used in some of these attacks have a cycle structure that differs from the cycle structure of the lattices that previously have been shown to be **NP**-hard.

Since a lattice with n cycles always can be transformed into a lattice with fewer cycles by a simple rescaling, the maximum number of cycles that is meaningful to analyze is $n-1$. Trolin showed that the exact version SVP under the max-norm is **NP**-complete for n-dimensional lattices with $n-1$ cycles of equal length [67].

In this chapter we investigate the importance of the cycle structure further. Our main result is a polynomial-time transformation that with arbitrary precision approximates any n-dimensional lattice with a lattice that has $n-1$ cycles of equal length, showing that these lattices form a hard core. A consequence of this is that short vectors and close vectors cannot be computed more efficiently in this class of lattices than in general lattices, except possibly for a polynomial factor. As our transformation only changes the size of the coordinates of the basis vectors and not the dimension of the lattice, the transformation is rather tight.

5.2 Background

Lattices

A *lattice* is a discrete additive subgroup $\Lambda \subseteq \mathbb{R}^n$. A lattice Λ can be defined by its basis, a set of independent vectors $\{\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_m}\}$, $\mathbf{b_i} \in \mathbb{R}^n$, such that $\mathbf{u} \in \Lambda$ if and only if there exist integers $t_1, t_2, \ldots, t_m$ such that $\mathbf{u} = \sum_{i=1}^{m} t_i \mathbf{b_i}$. If $m = n$ the lattice is said to be full-dimensional. Only lattices that are subsets of $\mathbb{Q}^n$ (and often $\mathbb{Z}^n$) are considered in this chapter. For each vector $\mathbf{v} \in \mathbb{R}^n$ and $p \geq 1$ the ℓ_p-norm is defined as $\|\mathbf{v}\|_p = \sqrt[p]{\sum_{i=1}^{n} |v_i|^p}$. The ℓ_∞-norm, also called the maximum norm, is defined as $\|\mathbf{v}\|_\infty = \max_{i=1}^{n} |v_i|$. When no index is given, $\|\mathbf{v}\| = \|\mathbf{v}\|_2$.

A *basis matrix* of a lattice is a matrix whose rows form a basis of the lattice. The *determinant* of a lattice is the absolute value of the determinant of a basis matrix. For lattices that are not full-dimensional, the determinant is defined as $\det(\Lambda) = \sqrt{\det\left(\mathbf{B}\mathbf{B}^T\right)}$. It is not difficult to see that the determinant is independent of the choice of basis.

Basis Representations

In different situations different bases may be suitable. Two such representations are the Hermite Normal Form and LLL-reduced bases.

A basis $\{\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_n}\}$ is said to be on Hermite Normal Form (HNF) if the basis matrix is upper triangular, and $b_{ii} > b_{ji} \geq 0$ for $j < i$. The Hermite Normal Form can be computed efficiently [39]. In [51] Micciancio gives some results on the use of HNF in cryptographic applications.

An LLL-reduced basis is defined as follows. Every lattice basis $\{\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_m}\}$ has an associated *orthogonal* basis $\left\{\hat{\mathbf{b}}_\mathbf{1}, \hat{\mathbf{b}}_\mathbf{2}, \ldots, \hat{\mathbf{b}}_\mathbf{m}\right\}$ defined by

$$\hat{\mathbf{b}}_\mathbf{i} = \mathbf{b_i} - \sum_{j=1}^{i-1} \mu_{ij} \hat{\mathbf{b}}_\mathbf{i}$$

where $\mu_{ij} = \left\langle \mathbf{b_i}, \hat{\mathbf{b}}_\mathbf{j} \right\rangle / \left\|\hat{\mathbf{b}}_\mathbf{j}\right\|^2$ for $i > j$. Extending the definition, we let $\mu_{ii} = 1$ and $\mu_{ij} = 0$ for $i < j$. It holds that $\prod_{i=1}^{m} \left\|\hat{\mathbf{b}}_\mathbf{i}\right\| = \det(\Lambda)$. A lattice basis is called *LLL-reduced* (after Lenstra, Lenstra and Lovász) with δ, $1/4 \leq \delta < 1$, if $|\mu_{ij}| \leq 1/2$ for $1 \leq j < i \leq m$ and $\delta \left\|\hat{\mathbf{b}}_\mathbf{i-1}\right\|^2 \leq \left\|\hat{\mathbf{b}}_\mathbf{i}\right\|^2 + \mu_{i,i-1}^2 \left\|\hat{\mathbf{b}}_\mathbf{i-1}\right\|^2$ for $i = 2, \ldots, m$. An LLL-reduced basis can be found in polynomial time [44].

The two most studied lattice problems are the closest vector problem, CVP, and the shortest vector problem, SVP. The input to the closest vector problem is a lattice Λ, $\mathbf{y} \in \mathbb{R}^n$ and $d > 0$. The problem is to determine whether or not there exists $\mathbf{x} \in \Lambda$ such that $\|\mathbf{y} - \mathbf{x}\| < d$. SVP is the homogeneous variant of the same problem, where we want to determine whether or not there exists $\mathbf{x} \in \Lambda$ such that $0 < \|\mathbf{x}\| < d$. As a matter of fact, these are both families of problems, since every norm gives a different problem.

It is known that CVP is **NP**-complete for any ℓ_p-norm (including the max-norm, ℓ_∞) [71] and that it is **NP**-hard to approximate within $n^{\frac{c_p}{\log \log n}}$ for some constants c_p [27]. It is also known that SVP is **NP**-complete in the ℓ_∞-norm [43] and under randomized reductions also for any ℓ_p-norm [2]. Khot has recently shown that SVP is **NP**-hard to approximate within any constant factor in ℓ_p-norm under randomized reductions [41], improving previous results [52, 40]. Dinur has improved the bound for ℓ_∞-norm to $n^{1/\log \log n}$ [26].

The Cycle Structure

In this chapter we focus on the role of the *cycle structure* of a lattice in the complexity of lattice problems. The cycle structure is defined as the algebraic structure of the group $\mathbb{Z}^n/\Lambda$ for a full-dimensional lattice Λ.

Definition 5.2.1 (Cycle Structure). *A lattice Λ is said to have the* cycle structure $k_1 \times k_2 \times \cdots \times k_m$, *if the additive factor group $\mathbb{Z}^n/\Lambda \sim \mathbb{Z}_{k_1} \times \mathbb{Z}_{k_2} \times \cdots \times \mathbb{Z}_{k_m}$ and k_i divides k_{i+1} for $i = 1, 2, \ldots, m-1$.*

Cycles of length one are called trivial. In the cases where it is not clear from the context we specify whether non-trivial cycles should be considered. A lattice with only one non-trivial cycle is called *cyclic*. Depending on context, it may be more convenient to number the cycle lengths in increasing or decreasing order.

There are other ways to look upon the cycle structure that may be useful in certain situations. One is to consider the Smith Normal Form [65] of a basis matrix, and another to examine the set of modular equations whose solutions are precisely the lattice points.

Definition 5.2.2 (Smith Normal Form). *Let $\mathbf{B}$ be an integral square matrix. The* Smith Normal Form, SNF, *of $\mathbf{B}$ is the diagonal matrix $\mathbf{S}$ are such that $\mathbf{S} = \mathbf{UBV}$, with $\mathbf{U}$ and $\mathbf{V}$ integral and $|\det(\mathbf{U})| = |\det(\mathbf{V})| = 1$ and diagonal elements s_i of $\mathbf{S}$ such that s_{i+1}/s_i all are integers.*

Such a diagonal matrix exists for every integral square matrix, see, e.g., [54]. The following theorem from [56] shows the relation between the Smith Normal Form and the cycle structure.

Theorem 5.2.3. *Let Λ be an n-dimensional lattice, and let $\mathbf{B}$ be a basis matrix of Λ. Let $\mathbf{S}$ be the Smith Normal Form of $\mathbf{B}$. Let the diagonal elements of $\mathbf{S}$ be $s_1, s_2, \ldots, s_n$. Then the cycle structure of Λ is $s_1 \times s_2 \times \cdots \times s_n$.*

We also give a theorem showing a connection between the subdeterminants of a lattice and its Smith Normal Form. An i-minor of $\mathbf{B}$ is an $i \times i$ matrix formed by taking i rows and i columns of $\mathbf{B}$.

Theorem 5.2.4. *Let $\mathbf{B}$ be an integral square matrix. Then the diagonal elements of the Smith Normal Form, $s_1, s_2, \ldots, s_n$ can be computed as*

$$s_i = \frac{d_i}{d_{i-1}}$$

where d_i is gcd *of the determinants of all i-minors of $\mathbf{B}$, and $d_0 = 1$.*

Although this method of computing the Smith Normal Form and hence the cycle structure is quite inefficient (we need consider all the i-minors, not only the principal), it turns out to be useful in certain proofs in this chapter. There are other, more efficient methods to compute the Smith Normal Form [39].

Another way to describe the number of cycles of a lattice is to use a different representation of the lattice, namely as a set of modular equations. Every lattice can be described in this way.

Theorem 5.2.5. *Let $\Lambda \subseteq \mathbb{Z}^n$ be a lattice. Then there exist n-dimensional vectors $\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_m}$ and integers $b_1, b_2, \ldots, b_m$, $b_i > 1$, such that*

$$\Lambda = \{\mathbf{x} : \langle \mathbf{a_1}, \mathbf{x}\rangle \equiv 0 \bmod b_1 \wedge \langle \mathbf{a_2}, \mathbf{x}\rangle \equiv 0 \bmod b_2 \wedge \ldots \wedge \langle \mathbf{a_m}, \mathbf{x}\rangle \equiv 0 \bmod b_m\} \ .$$

The essence of this theorem is that any lattice can be expressed as a system of modular linear equations whose solutions form the lattice.

The connection to the cycle structure is that the number of nontrivial cycles is m, and the length of cycle i is b_i, provided that the system of equations has been reduced to minimize the number of equations and that the gcd of the coefficients and the modulus is 1 in each equation.

In the transformations we approximate lattices in $\mathbb{Z}^n$ with lattices in $\mathbb{Q}^n$. The standard definition of cycle structure cannot be applied to general lattices in $\mathbb{Q}^n$. Since multiplication by a constant does not affect lattice problems such as SVP and CVP, we will define the cycle structure of a lattice $\Lambda \subset \mathbb{Q}^n$ as the cycle structure of $k\Lambda$, where k is the smallest integer such that $k\Lambda \subseteq \mathbb{Z}^n$.

We now state three simple lemmas on the cycle structure. They follow directly from Theorem 5.2.3.

Lemma 5.2.6. *Let $\Lambda \subseteq \mathbb{Z}^n$ be a lattice with cycle structure $k_1 \times k_2 \times \cdots \times k_m$. Then $\det(\Lambda) = \prod_{i=1}^m k_i$.*

Lemma 5.2.7. *Let $\Lambda \subseteq \mathbb{Z}^n$ be a lattice with cycle structure $k_1 \times k_2 \times \cdots \times k_n$ (not necessarily all nontrivial). Then the lattice $t \cdot \Lambda$ has cycle structure $t \cdot k_1 \times t \cdot k_2 \times \cdots \times t \cdot k_n$*

Lemma 5.2.8. *Let $\Lambda \subseteq \mathbb{Z}^n$ be a lattice with cycle structure $k_1 \times k_2 \times \cdots \times k_n$, $k_1 \geq k_2 \geq \cdots \geq k_n$. Then the lattice $\frac{1}{k_n} \cdot \Lambda$ has cycle structure $\frac{k_1}{k_n} \times \frac{k_2}{k_n} \times \cdots \times \frac{k_n}{k_n}$.*

Because of the divisibility requirement, the lattice $\frac{1}{k_n}\Lambda$ in Lemma 5.2.8 is in $\mathbb{Z}^n$. Should k_n be greater than one, we can always remove it as shown in the theorem. Hence we can assume without loss of generality that the number of cycles is less than n.

Previous Results on the Cycle Structure

In [56] the following theorem is proved.

Theorem 5.2.9. *Let $\Lambda \subseteq \mathbb{Z}^n$ be a lattice. Then for every $\varepsilon > 0$ we can efficiently construct a linear transformation $\sigma_{\Lambda,\varepsilon} : \Lambda \to \mathbb{Z}^n$ such that $\sigma_{\Lambda,\varepsilon}(\Lambda)$ is a lattice and for some integer k*

1. $\forall \mathbf{u} \in \Lambda : \|\mathbf{u} - \sigma_{\Lambda,\varepsilon}(\mathbf{u})/k\| \leq \varepsilon\|\mathbf{u}\|$
2. *$\sigma_{\Lambda,\varepsilon}(\Lambda)$ is cyclic.*

This theorem implies that if we can solve a lattice problem for cyclic lattices, we can get an approximative solution for the same problem for any with arbitrary precision. In other words, the cyclic lattices form a hard core.

In his celebrated paper [1], Ajtai showed how to generate lattices with a connection between the average case and the worst case of variants of SVP. The lattices in the constructions in Cai's and Nerurkar's improved version of Ajtai's result [13] have n/c cycles. Although this result is not an **NP**-hardness result, it raises the question of whether the hardness of lattice problems does or does not in general decrease with a higher number of cycles. In [67] it is shown that SVP in the maximum norm is **NP**-complete for lattices with $n-1$ cycles, giving further evidence that hardness results of lattice problems extend to many cycle structures. The result of the current chapter gives the main result of [67] as a consequence.

5.3 The Approximation

Let $\Lambda \subseteq \mathbb{Z}^n$ be an arbitrary lattice. To adapt this into a lattice with $n-1$ cycles that is arbitrarily close to the original lattice we go through the following five steps:

1. Inflate the lattice by a factor k and perturb to achieve a lattice with Hermite Normal Form of a certain form.
2. Reduce the sublattice spanned by the first $n-1$ vectors of the Hermite Normal Form using the LLL algorithm.
3. Factor the partly reduced basis matrix into two matrices, where the second has its determinant equal to one.
4. Perform modifications to the first matrix to give it $n-1$ cycles of equal length.

5. Multiply the two matrices to get a basis for an $(n-1)$-cyclic lattice that is close to the original lattice.

In Sections 5.3 to 5.3 these steps are described in detail. It is also shown that the modifications have the desired effect on the cycle structure. In Section 5.3 we analyze the disturbance from the perturbation and show that it does not move a lattice vector more than a small multiple of the original length. All the transformations are linear, and extend through linearity to any point in $\mathbb{R}^n$.

Acquiring a Lattice with a Good Hermite Normal Form

For the modification to work we need the lattice to have a Hermite Normal Form of a certain form. In this section we describe how we efficiently can modify a general lattice slightly to get the Hermite Normal Form we need.

Let $\Lambda \subseteq \mathbb{Z}^n$ be a lattice, and let $\mathbf{H}$ be its basis in Hermite Normal Form. For the coming steps, we need the basis of the lattice to be of the following form:

$$\mathbf{B} = \begin{pmatrix} 1 & 0 & \cdots & 0 & a_1 \\ 0 & 1 & \cdots & 0 & a_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & a_{n-1} \\ 0 & 0 & \cdots & 0 & d \end{pmatrix} \tag{5.1}$$

where $d = \det(\Lambda)$ and $0 \leq a_i < d$. We show how to perturb Λ so that we get a lattice whose Hermite Normal Form as is in equation (5.1). The method we use is based on the following theorem.

Lemma 5.3.1. *Let* $\mathbf{H}$ *be a matrix on Hermite Normal Form, i.e.,*

$$\mathbf{H} = \begin{pmatrix} h_{11} & h_{12} & h_{13} & \ldots & h_{1(n-1)} & h_{1n} \\ 0 & h_{22} & h_{23} & \ldots & h_{2(n-1)} & h_{2n} \\ 0 & 0 & h_{33} & \ldots & h_{3(n-1)} & h_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & h_{(n-1)(n-1)} & h_{(n-1)n} \\ 0 & 0 & 0 & \ldots & 0 & h_{nn} \end{pmatrix}.$$

Then the matrix $\tau(\mathbf{H})$ *given by*

$$\tau(\mathbf{H}) = \begin{pmatrix} h_{11} & h_{12} & h_{13} & \ldots & h_{1(n-1)} & h_{1n} \\ 1 & h_{22} & h_{23} & \ldots & h_{2(n-1)} & h_{2n} \\ 0 & 1 & h_{33} & \ldots & h_{3(n-1)} & h_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & h_{(n-1)(n-1)} & h_{(n-1)n} \\ 0 & 0 & 0 & \ldots & 1 & h_{nn} \end{pmatrix} \tag{5.2}$$

has a Hermite Normal Form as in equation (5.1). The transformation can be computed in time polynomial in the size of the input data.

Proof. We show how to transform the matrix $\tau(\mathbf{H})$ into the Hermite Normal Form using row operations. We begin by placing the topmost vector at the bottom. This gives a matrix that is upper triangular except for the last row. Since all elements on the diagonal are one, we can cancel the $n-1$ first elements of the last row. We then cancel all non-diagonal elements except for the right-most column, which gives a matrix on HNF. Since the determinant is preserved, the bottom right entry must be $\det(\mathbf{H})$. □

We also define the transformation when the input is a vector as

$$\tau_{\Lambda,k}\left(\sum_{i=1}^{n} t_i \mathbf{u_i}\right) = \sum_{i=1}^{n} t_i \mathbf{u'_i} \tag{5.3}$$

where $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_n}$ are the rows of $\mathbf{U}$ and $\mathbf{u'_1}, \mathbf{u'_2}, \ldots, \mathbf{u'_n}$ are the rows of $\tau(k\mathbf{U})$.

As the reader may have noticed, this step actually implies the result from [56], although we not only achieve a cyclic lattice, but a lattice whose Hermite Normal Form is as defined above.

Factoring the Basis

Now that we have a basis with the Hermite Normal Form we need, we proceed by finding a more orthogonal basis and factoring the basis matrix.

Let the operation $\rho(\mathbf{B})$ be defined as follows: First the LLL-reduction is applied to the first $n-1$ vectors of $\mathbf{B}$ using $\delta = 3/4$, keeping the last vector unchanged. Let us call this intermediate step ρ'. Assuming that the input is a basis matrix $\mathbf{B}$ of the form (5.1), this gives a matrix of the form

$$\rho'(\mathbf{B}) = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1(n-1)} & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2(n-1)} & b_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b_{(n-1)1} & b_{(n-1)2} & \cdots & b_{(n-1)(n-1)} & b_{(n-1)n} \\ 0 & 0 & \cdots & 0 & d \end{pmatrix}. \tag{5.4}$$

From the LLL-reduced basis the $(n-1)$'th vector is placed first, keeping the internal order of the other vectors. The complete transformation is called ρ. The matrix $\rho(\mathbf{B})$ can be factored into

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & d \end{pmatrix} \cdot \begin{pmatrix} b_{(n-1)1} & b_{(n-1)2} & \cdots & b_{(n-1)n} \\ b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{(n-2)1} & b_{(n-2)2} & \cdots & b_{(n-2)n} \\ 0 & 0 & \cdots & 1 \end{pmatrix} \tag{5.5}$$

Since the determinant of the right factor is 1, the cycle structure of the product only depends on the left factor. This follows since, as pointed out in [56], unimodular transformations do not change the cycle structure.

Modifying the Cycle Structure

Let $\mathbf{B}_l$ be the left factor in the basis factorization (5.5) and $\mathbf{B}_r$ the right factor. We create a new lattice Λ' by inflating the lattice spanned by $\mathbf{B}_l$ by a factor d^{n-2}. Put differently, the matrix $d^{n-2} \cdot \mathbf{B}_l$ is a basis matrix of Λ'. By Lemma 5.2.7, this lattice has $n-1$ cycles of length d^{n-2} and one cycle of length d^{n-1}.

By modifying the lattice Λ' slightly, we get a new lattice that has $n-1$ cycles of length d^{n-1}. We call the new lattice Λ''. The modification is defined by the function γ':

$$\gamma'_n(d) = \begin{pmatrix} d^{n-2} & d^{n-3} & d^{n-4} & \cdots & d^2 & d & 1 & 0 \\ 0 & d^{n-2} & d^{n-3} & \cdots & d^3 & d^2 & d & 0 \\ 0 & 0 & d^{n-2} & \cdots & d^4 & d^3 & d^2 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & d^{n-2} & d^{n-3} & d^{n-4} & 0 \\ 0 & 0 & 0 & \cdots & 0 & d^{n-2} & d^{n-3} & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & d^{n-2} & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & d^{n-1} \end{pmatrix}.$$

Theorem 5.3.2. *The lattice Λ'' with basis matrix $\gamma'_n(\det(\mathbf{B}_l))$ has $n-1$ nontrivial cycles, each of which has length d^{n-1}.*

Proof. Set $\mathbf{C} = \gamma'(\mathbf{B}_l)$. We describe why this lattice has the cycle structure mentioned above by examining the quotient

$$k_1 = \frac{m_n}{m_{n-1}}$$

where $m_n = |\det(\mathbf{C})| = d^{(n-1)(n-1)}$ and m_{n-1} is the gcd of all $(n-1)$-minors of $\mathbf{C}$. We know that k_1 is the length of the longest cycle.

We determine m_{n-1} by systematically examining the $(n-1)$-minors of $\mathbf{C}$. Let $\mathbf{C}^{i,j}$ be the $(n-1)\times(n-1)$-matrix where the i'th row and the j'th column of $\mathbf{C}$ have been removed. First consider $\mathbf{C}^{i,j}$, where $i < j$. These matrices are triangular with one or more zeroes on the diagonal. Therefore, the determinants of these matrices are all zero. The matrices $\mathbf{C}^{i,i}$ are also triangular, but with non-zero elements on the diagonal. For $i < n$, $\det\left(\mathbf{C}^{i,i}\right) = d^{(n-2)(n-1)+1}$, and $\det\left(\mathbf{C}^{n,n}\right) = d^{(n-2)(n-1)}$. Next we consider $\det\left(\mathbf{C}^{i,j}\right)$ where $n > i > j$. These matrices are block-triangular, as below.

$$\mathbf{C}^{i,j} = \begin{pmatrix} \mathbf{D_{j-1}} & \cdot & \cdot & \cdot \\ \mathbf{0} & \mathbf{L_{i-j}} & \cdot & \cdot \\ \mathbf{0} & \mathbf{0} & \mathbf{D_{n-i-1}} & \cdot \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & d^{n-1} \end{pmatrix}$$

where $\mathbf{D_k}$ is the $k \times k$ triangular matrix

$$\mathbf{D_k} = \begin{pmatrix} d^{n-2} & d^{n-3} & d^{n-4} & \cdots & d^{n-k} & d^{n-k-1} \\ 0 & d^{n-2} & d^{n-3} & \cdots & d^{n-k+1} & d^{n-k} \\ 0 & 0 & d^{n-2} & \cdots & d^{n-k+2} & d^{n-k+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & d^{n-2} & d^{n-3} \\ 0 & 0 & 0 & \cdots & 0 & d^{n-2} \end{pmatrix}$$

and $\mathbf{L_k}$ is the $k \times k$ matrix

$$\mathbf{L_k} = \begin{pmatrix} d^{n-3} & d^{n-4} & d^{n-5} & \cdots & d^{n-k} & d^{n-k-1} & d^{n-k-2} \\ d^{n-2} & d^{n-3} & d^{n-4} & \cdots & d^{n-k+1} & d^{n-k} & d^{n-k-1} \\ 0 & d^{n-2} & d^{n-3} & \cdots & d^{n-k+2} & d^{n-k+1} & d^{n-k} \\ 0 & 0 & d^{n-2} & \cdots & d^{n-k+3} & d^{n-k+2} & d^{n-k+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & d^{n-2} & d^{n-3} & d^{n-4} \\ 0 & 0 & 0 & \cdots & 0 & d^{n-2} & d^{n-3} \end{pmatrix}.$$

To compute $\det(\mathbf{L_k})$, we notice that the last two columns are linearly dependent, since the leftmost column multiplied by d gives the $(k-1)$'th column. This means that $\det(\mathbf{L_k}) = 0$, and that $\det\left(\mathbf{C}^{i,j}\right) = 0$ for $i < j < n$.

What remains to be checked is $\det\left(\mathbf{C}^{n,j}\right)$ for $j = 1, 2, \ldots, n-1$. These matrices, where we have removed the last row, have only zeroes in their right-most column and hence the determinant is 0.

Combining these results, we see that $d^{(n-2)(n-1)}$ is a factor of all the $(n-1)$-minors. Also, there are $(n-1)$-minors whose determinant is precisely $d^{(n-2)(n-1)}$. Hence we have that $m_{n-1} = d^{(n-2)(n-1)}$, and consequently $k_1 = d^{n-1}$. Since gcd of all 1-minors (in other words, all the elements) is 1, $m_1 = 1$. This means that we have $n-1$ cycles whose product is $d^{(n-1)(n-1)}$ (the determinant) and that the longest one has length d^{n-1}. Because of the divisibility requirement on the lengths of the cycles, the only possibility is that there are $(n-1)$ cycles of length d^{n-1}. $\square$

Returning to the Original Representation

Returning to the original representation is just a matter of multiplying by $\mathbf{B}_r$. Since this does not change the cycle structure ($\mathbf{B}_r$ is unimodular), we still have a lattice with the required cycle structure.

We denote the transformation described in Sections 5.3 to 5.3 by γ. More precisely,

$$\gamma(\mathbf{B}) = \gamma'_n\left(\det(\mathbf{B}_l)\right) \cdot \mathbf{B}_r$$

where $\mathbf{B}_l$ and $\mathbf{B}_r$ are the left and right factors of $\rho(\mathbf{B})$ as in (5.5) and n is the dimension of the lattice.

We also define the transformation when applied to a vector $\mathbf{v} = \sum_{i=1}^{n} t_i \mathbf{b_i}$ in a lattice Λ where $\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_n}$ is a basis. The transformation is then defined as

$$\gamma_\Lambda(\mathbf{v}) = \sum_{i=1}^{n} t_i \mathbf{b_i'}$$

where $\mathbf{b_1'}, \mathbf{b_2'}, \ldots, \mathbf{b_n'}$ are the rows of $\gamma(\mathbf{B})$.

Since LLL-reduction can be performed in polynomial time ρ can be computed in polynomial time. It is obvious that also γ' and the factorization in $\mathbf{B}_l$ and $\mathbf{B}_r$ require at most polynomial time. Hence γ can be computed in time polynomial in the size of the input data.

Completing the Approximation

Now we have the necessary steps to complete the approximation. Let $\Lambda \subseteq \mathbb{Z}^n$ be a lattice. Our goal is to prove that for any $\varepsilon > 0$ there exist a transformation $\sigma_{\Lambda,\varepsilon}$ and an integer k such that

1. $\forall \mathbf{u} \in \mathbb{Z}^n : \|\mathbf{u} - \sigma_{\Lambda,\varepsilon}(\mathbf{u})/k\| \leq \varepsilon \|\mathbf{u}\|$.
2. $\sigma_{\Lambda,\varepsilon}(\Lambda)$ has $n-1$ non-trivial cycles of equal length.

The transformations we use are $\tau_{\Lambda,k}$ and γ_Λ as described above. Since the displacement for these transformations (as we will see) depends on the determinant, we need to find an appropriate k that makes the determinant large enough. In the final approximation we will begin by applying τ and then apply γ. This composed transformation is called $\sigma_{\Lambda,\varepsilon}(\mathbf{u})$ and can be computed in polynomial time since both τ and γ can be computed in polynomial time.

We bound the displacement introduced by the two transformations τ and γ described above.

Lemma 5.3.3. *Let Λ be a lattice and let $\tau_{\Lambda,k}$ be defined as in (5.3). Then $\forall \mathbf{u} \in \mathbb{Z}^n : \left\|\mathbf{u} - \frac{1}{k}\tau_{\Lambda,k}(\mathbf{u})\right\| \leq \frac{1}{k} 2^n \|\mathbf{u}\|$.*

Proof. The proof of this lemma follows the proof in [56] closely. Let $\mathbf{B}$ be the basis matrix on HNF of the lattice Λ, let $\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_n}$ be its rows and b_{ij} its elements. Assume $\mathbf{u} = \sum_{i=1}^{n} t_i \mathbf{b_i}$. We first show that

$$|t_i| \leq \|\mathbf{u}\| 2^{i-1} \tag{5.6}$$

for any ℓ_p-norm (including ℓ_∞). Since $\mathbf{B}$ is upper-triangular, $\sum_{i=1}^{j} |t_i|\,|b_{ij}| \geq |u_j|$. Dividing with $|b_{jj}|$ and using the property of the HNF that $b_{ij} \leq b_{jj}$ for $i < j$ we get

$$|t_j| \leq \frac{\sum_{i=1}^{j-1} |t_i|\,|b_{ij}| + |u_j|}{|b_{jj}|} \leq \sum_{i=1}^{j-1} |t_i| + \|\mathbf{u}\| \ .$$

Induction (on j) gives (5.6).

We can now compute the actual displacement for a vector $\mathbf{u} = \sum_{i=1}^{n} t_i \mathbf{b_i}$ as

$$\begin{aligned} \left\| \mathbf{u} - \frac{1}{k}\tau_{\Lambda,k}(u) \right\| &= \frac{1}{k}\left\| \sum_{i=2}^{n} t_i \mathbf{e_{i-1}} \right\| \\ &\leq \frac{1}{k}\sum_{i=2}^{n} 2^{i-1}\|\mathbf{u}\| \\ &\leq \frac{1}{k}2^{n}\|\mathbf{u}\| \end{aligned} \tag{5.7}$$

where $\mathbf{e_i}$ are the unit vectors. □

We need some bounds on the basis (5.4) before we can complete the proof. We give these bounds as two lemmas. The first lemma shows that the coordinates of a vector are bounded in a way similar to Lemma 5.3.3, and the second that the basis vectors are bounded.

Lemma 5.3.4. *Let* $\mathbf{B}$ *be the basis matrix of* Λ *given on the form (5.4), let* $\mathbf{b_1}$, $\mathbf{b_2}$, *…,* $\mathbf{b_n}$ *be its rows. Assume* $\mathbf{u} = \sum_{i=1}^{n} t_i \mathbf{b_i}$. *Then*

$$|t_i| \leq 2^{\frac{3}{2}n-i}\|\mathbf{u}\| \tag{5.8}$$

for $i < n$ *and for any* ℓ_p*-norm (including* ℓ_∞*).*

Proof. Since the first $n-1$ rows of $\mathbf{B}$ are LLL-reduced, there is an orthogonal basis $\hat{\mathbf{b}}_\mathbf{i}$ given by $\hat{\mathbf{b}}_\mathbf{i} = \mathbf{b_i} - \sum_{j=1}^{i-1} \mu_{ij}\hat{\mathbf{b}}_\mathbf{j}$, or $\mathbf{b_i} = \sum_{j=1}^{i} \mu_{ij}\hat{\mathbf{b}}_\mathbf{j}$, where $|\mu_{ij}| \leq 1/2$ except for $\mu_{ii} = 1$. We can rewrite $\mathbf{u}$ as

$$\begin{aligned} \mathbf{u} &= \sum_{i=1}^{n} t_i \mathbf{b_i} \\ &= \sum_{i=1}^{n} t_i \left(\sum_{j=1}^{i} \mu_{ij}\hat{\mathbf{b}}_\mathbf{j} \right) \\ &= \sum_{i=1}^{n} \hat{\mathbf{b}}_\mathbf{i} \left(\sum_{j=i}^{n} t_j \mu_{ji} \right) \\ &= \sum_{i=1}^{n} \hat{t}_i \hat{\mathbf{b}}_\mathbf{i} \end{aligned}$$

where

$$\hat{t}_i = \sum_{j=i}^{n} t_j \mu_{ji} \ .$$

Assume that the lemma is not true, and let i be the largest index such that $|t_i| > 2^{\frac{3}{2}n-i}\|\mathbf{u}\|$. Then

$$\begin{aligned}
\left|\hat{t}_i\right| &= \sum_{j=i}^{n} \mu_{ji} t_j \\
&\geq |t_i| - \left|\sum_{j=i+1}^{n} \mu_{ji} t_j\right| \\
&> 2^{\frac{3}{2}n-i}\|\mathbf{u}\| - \frac{1}{2}\|\mathbf{u}\| \sum_{j=i+1}^{n} 2^{\frac{3}{2}n-j} \\
&\geq 2^{\frac{n}{2}}\|\mathbf{u}\| \left(2^{n-i} - \frac{1}{2}\sum_{j=0}^{n-i-1} 2^j\right) \\
&\geq 2^{\frac{n}{2}}\|\mathbf{u}\|
\end{aligned}$$

However, since $\mathbf{u} = \sum_{i=1}^{n} \hat{t}_i \hat{\mathbf{b}}_\mathbf{i}$, the vectors $\hat{\mathbf{b}}_\mathbf{i}$ are pairwise orthogonal and $\left\|\hat{\mathbf{b}}_\mathbf{i}\right\| \geq 2^{-\frac{i-1}{2}}$, this would imply that $\|\mathbf{u}\| > \|\mathbf{u}\|$. As this is a contradiction, the assumption must be false. This proves the lemma. □

Lemma 5.3.5. *Let* $\mathbf{B}$ *be a basis matrix of the form (5.1), and let* $\mathbf{b_i}$ *be the row vectors of the matrix* $\rho(\mathbf{B})$*. Then it holds that*

$$\|\mathbf{b_i}\| \leq n 2^{\frac{n^2}{8}} \sqrt[4]{d^2 n}$$

for $i = 2, 3, \ldots, n-1$*.*

The idea of the proof is that in an LLL-reduced basis $\mathbf{B}$ the length of every vector except the last one has an upper bound of the order $\sqrt{\det(\mathbf{B})}$. We then need to renumber the vectors since the can only afford the first vector to remain unbounded in order to bound $\gamma(\mathbf{B})$. It is essential that the bound is $o(\det(\mathbf{B}))$ because of the displacement of γ. The full proof is as follows.

Proof. If we are able to prove that the condition holds for $\mathbf{b'_1}, \mathbf{b'_2}, \ldots, \mathbf{b'_{n-2}}$, where $\mathbf{b'_i}$ are the row vectors of $\rho'(\mathbf{B})$, the lemma obviously follows by renumbering.

We are interested in the $(n-1)$-dimensional lattice $\mathbf{S} \subset \mathbb{Z}^n$ spanned by $\mathbf{b'_1}$, $\mathbf{b'_2}$, $\ldots$, $\mathbf{b'_{n-1}}$. A basis (in fact the basis given in (5.1)) of the $(n-1)$-dimensional lattice is

$$\mathbf{C} = \begin{pmatrix} 1 & 0 & \cdots & 0 & a_1 \\ 0 & 1 & \cdots & 0 & a_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & a_{n-1} \end{pmatrix} \tag{5.9}$$

with $0 \leq a_i < d$.

The determinant for this not full-dimensional lattice is given by

$$\det(\mathbf{S}) = \sqrt{\det\left(\mathbf{C}\mathbf{C}^T\right)}\ .$$

Let $\mathbf{G} = \mathbf{C}\mathbf{C}^T$. It holds that

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & \cdots & 0 & a_1 \\ 0 & 1 & \cdots & 0 & a_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & a_{n-1} \end{pmatrix} \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ a_1 & a_2 & \cdots & a_{n-1} \end{pmatrix} =$$

$$= \begin{pmatrix} 1+a_1^2 & a_1a_2 & a_1a_3 & \cdots & a_1a_{n-2} & a_1a_{n-1} \\ a_2a_1 & 1+a_2^2 & a_2a_3 & \cdots & a_2a_{n-2} & a_2a_{n-1} \\ a_3a_1 & a_3a_2 & 1+a_3^2 & \cdots & a_3a_{n-2} & a_3a_{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-2}a_1 & a_{n-2}a_2 & a_{n-2}a_3 & \cdots & 1+a_{n-2}^2 & a_{n-2}a_{n-1} \\ a_{n-1}a_1 & a_{n-1}a_2 & a_{n-1}a_3 & \cdots & a_{n-1}a_{n-2} & 1+a_{n-1}^2 \end{pmatrix} =$$

$$= \mathbf{a}\mathbf{a}^T + \mathbf{I_{n-1}}$$

where $\mathbf{I_{n-1}}$ is the $(n-1)$-dimensional unit matrix. Since

$$\mathbf{G}\cdot\mathbf{a} = \left(\mathbf{a}\mathbf{a}^T + \mathbf{I_{n-1}}\right)\mathbf{a} = \mathbf{a}\mathbf{a}^T\mathbf{a} + \mathbf{a} = \mathbf{a}\left(\mathbf{a}^T\mathbf{a} + 1\right) = \mathbf{a}\left(1 + \sum_{i=1}^{n-1} a_i^2\right)$$

the vector $\mathbf{a}$ is an eigenvector of $\mathbf{G}$ with eigenvalue $\lambda_1 = 1 + \sum_{i=1}^{n-1} a_i^2$. Now let $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{n-2}}$ be $n-2$ linearly independent vectors orthogonal to $\mathbf{a}$. Then

$$\mathbf{G}\cdot\mathbf{v_i} = \left(\mathbf{a}\mathbf{a}^T + \mathbf{I_{n-1}}\right)\mathbf{v_i} = \mathbf{a}\cdot\left(\mathbf{a}^T\mathbf{v_i}\right) + \mathbf{v_i} = \mathbf{a}\cdot 0 + \mathbf{v_i} = \mathbf{v_i}$$

which shows that $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{n-2}}$ also are eigenvectors of $\mathbf{G}$ with eigenvalues $\lambda_i = 1, i = 2, 3, \ldots, n-1$. From this it follows that

$$\det(\mathbf{G}) = \prod_{i=1}^{n-1} \lambda_i = 1 + \sum_{i=1}^{n-1} a_i^2$$

which gives an upper bound for the determinant of $\mathbf{S}$ as

$$\det(\mathbf{S}) = \sqrt{\det\left(\mathbf{C}\mathbf{C}^T\right)} = \sqrt{\det(\mathbf{G})} = \sqrt{1 + \sum_{i=1}^{n-1} a_i^2} \leq d\sqrt{n}\ .$$

Our next step is to prove an upper bound for $\left\|\hat{\mathbf{b'_i}}\right\|$, $i < n-1$, where $\hat{\mathbf{b'_i}}$ are the vectors of the corresponding orthogonal system. Using a contradiction argument, we show that

$$\left\|\hat{\mathbf{b'_i}}\right\| < 2^{\frac{n^2}{8}}\sqrt{\det \mathbf{S}}\ .$$

Assume this is not the case, i.e., that there exists an index $j \in \{1, 2, \ldots, n-2\}$ such that

$$\left\|\hat{\mathbf{b'_j}}\right\| \geq 2^{\frac{n^2}{8}}\sqrt{\det \mathbf{S}}\ .$$

Since the basis is LLL-reduced with $\delta = 3/4$,

$$\frac{1}{2}\left\|\hat{\mathbf{b'_{i-1}}}\right\|^2 \leq \left\|\hat{\mathbf{b'_i}}\right\|^2$$

and

$$\prod_{i=1}^{n-1}\left\|\hat{\mathbf{b'_i}}\right\| = \det(\mathbf{S}) \tag{5.10}$$

Since $\left\|\hat{\mathbf{b'_1}}\right\| = \|\mathbf{b'_1}\| \geq 1$, it holds that $\left\|\hat{\mathbf{b'_i}}\right\| \geq 2^{-\frac{i-1}{2}}$ for $i = 1, 2, \ldots, j-1$ and $\left\|\hat{\mathbf{b'_i}}\right\| \geq 2^{\frac{n^2}{8}}\sqrt{\det(\mathbf{S})}2^{-\frac{i-j}{2}}$ for $i = j, j+1, \ldots, n-1$. Using equation (5.10) we can compute the determinant as

$$\det(\mathbf{S}) = \prod_{i=1}^{n-1}\left\|\hat{\mathbf{b'_i}}\right\| \geq \prod_{i=1}^{j-1}2^{-\frac{i-1}{2}} \cdot \prod_{i=j}^{n-1}2^{\frac{n^2}{8}}\sqrt{\det(\mathbf{S})}2^{-\frac{i-j}{2}} =$$

$$2^{-\frac{(j-2)(j-1)}{4}}(\det(\mathbf{S}))^{\frac{n-j}{2}}2^{\frac{n^2}{8}n-j}2^{-\frac{(n-j)(n-j-1)}{2}}\ .$$

For $\det(\mathbf{S})$ large enough this is a decreasing function, so it takes its minimum over j when $j = n-2$ as

$$2^{-\frac{(n-4)(n-3)}{4}}\det(\mathbf{S})2^{\frac{n^2}{8}2}2^{-\frac{2\cdot 1}{2}} > 2^{-\frac{n^2}{4}}\det(\mathbf{S})2^{\frac{n^2}{4}} = \det(\mathbf{S})$$

which gives the contradiction $\det(\mathbf{S}) > \det(\mathbf{S})$. Hence the assumption must be incorrect and $\|\hat{\mathbf{b'_i}}\| < 2^{\frac{n^2}{8}}\sqrt{\det \mathbf{S}} \leq 2^{\frac{n^2}{8}}\sqrt[4]{d^2 n}$ for $i = 1, 2, \ldots, n-2$.

Since we have the relation that $\mathbf{b'_k} = \hat{\mathbf{b'_k}} + \sum_{i=1}^{k-1}\mu_{ij}\hat{\mathbf{b'_i}}$ and $|\mu_{ij}| \leq 1/2$, it holds that

$$\begin{aligned}
\|\mathbf{b_k}\| &= \left\|\hat{\mathbf{b'_k}} + \sum_{i=1}^{k-1}\mu_{ij}\hat{\mathbf{b'_i}}\right\| \\
&\leq \left\|\hat{\mathbf{b'_k}}\right\| + \sum_{i=1}^{k-1}\left\|\mu_{ij}\hat{\mathbf{b'_i}}\right\| \\
&\leq 2^{\frac{n^2}{8}}\sqrt[4]{d^2 n} + \frac{1}{2}n2^{\frac{n^2}{8}}\sqrt[4]{d^2 n} \\
&\leq n2^{\frac{n^2}{8}}\sqrt[4]{d^2 n}\ .
\end{aligned}$$

from which the lemma follows. □

Now we have the necessary tools to find a bound for the transformation γ_Λ.

Lemma 5.3.6. *Let Λ be an n-dimensional lattice and let γ_Λ be as defined in Section 5.3. Then $\forall \mathbf{u} \in \mathbb{Z}^n$*

$$\left\| \mathbf{u} - \frac{1}{\det(\Lambda)^{n-2}} \gamma_\Lambda(\mathbf{u}) \right\| \leq \frac{n^{9/4} 2^{\frac{3}{2}n + \frac{n^2}{8}}}{\sqrt{\det(\Lambda)}} \|\mathbf{u}\|$$

for $\det(\Lambda) = \Omega\left(2^{n^2}\right)$.

Proof. Let $\mathbf{b_i}$ be the vectors of the partly LLL-reduced basis in (5.4) and let $\mathbf{b'_i} = \gamma_\Lambda(\mathbf{b_i})$ be the vectors of the modified basis. If we let the lattice determinant be d, using Lemma 5.3.4 we get a displacement for the vector $\mathbf{u} = \sum_{i=1}^n t_i \mathbf{b_i}$ of (remember the last two basis vectors are not modified in the transformation)

$$\begin{aligned} \left\| \mathbf{u} - \frac{1}{d^{n-2}} \gamma_\Lambda(\mathbf{u}) \right\| &= \frac{1}{d^{n-2}} \left\| \sum_{i=1}^{n-2} t_i \left(d^{n-3} \mathbf{b_{i+1}} + o\left(d^{n-3}\right) \sum_{j=i+2}^{n-2} \mathbf{b_j} \right) \right\| \\ &\leq \frac{1}{d} 2^{\frac{3}{2}n} \|\mathbf{u}\| \sum_{i=2}^{n-1} \|\mathbf{b_i}\| + o\left(d^{-1}\right) 2^{\frac{3}{2}n} \sum_{i=2}^{n-2} \|\mathbf{b_i}\| \ . \end{aligned} \tag{5.11}$$

To show that this displacement remains bounded, we use Lemma 5.3.5 to get an upper bound for $\|\mathbf{b_i}\|$. Inequality (5.11) can be written as

$$\begin{aligned} \left\| \mathbf{u} - \frac{1}{d^{n-2}} \gamma_{\Lambda,\varepsilon}(\mathbf{u}) \right\| &\leq \frac{1}{d} 2^{\frac{3}{2}n} \|\mathbf{u}\| \sum_{i=2}^{n-1} \|\mathbf{b_i}\| + o\left(d^{-1}\right) 2^{\frac{3}{2}n} \sum_{i=2}^{n-2} \|\mathbf{b_i}\| \\ &\leq \frac{1}{d} 2^{\frac{3}{2}n} \|\mathbf{u}\| n \cdot n 2^{\frac{n^2}{8}} \sqrt[4]{d^2 n} \\ &= \frac{n^{9/4} 2^{\frac{3}{2}n + \frac{n^2}{8}}}{\sqrt{d}} \|\mathbf{u}\| \end{aligned} \tag{5.12}$$

for d large enough, which proves the lemma. □

We combine these two lemmas in order to show a bound for the composed transformation $\sigma_{\Lambda,\varepsilon}$.

Theorem 5.3.7. *Let Λ be an n-dimensional lattice. For every choice of $\varepsilon > 0$ there exist integers k and s, at most of size polynomial in $\log\left(\varepsilon^{-1}\right)$ and n, such that the transformation $\sigma_{\Lambda,\varepsilon} = \gamma_{\tau_s(\Lambda)} \circ \tau_{\Lambda,s}$ generates a lattice with $n-1$ cycles of equal length and for any vector $\mathbf{u}$*

$$\left\| \mathbf{u} - \frac{1}{k} \sigma_{\Lambda,\varepsilon}(\mathbf{u}) \right\| \leq \varepsilon \|\mathbf{u}\|$$

Proof. Let $d = \det(\Lambda)$. According to the triangle inequality, the displacement for $\mathbf{u}$ is at most the sum of the displacement for τ and γ. According to Lemma 5.3.3 τ_s gives a displacement of at most $\frac{1}{s}2^n\|\mathbf{u}\|$ whereas according to Lemma 5.3.6 γ gives a displacement of at most $n^{9/4}2^{\frac{3}{2}n+\frac{n^2}{8}}\sqrt{\det(\Lambda')}\|\mathbf{u}\|$ where Λ' is the lattice $\tau_s(\Lambda)$. Since $\det(\Lambda') = \Omega\left(ds^n\right)$ the total displacement is

$$\left\|\mathbf{u} - \frac{1}{(ds^n)^{n-2}}\sigma_{\Lambda,\varepsilon}(\mathbf{u})\right\| \quad \le \quad \frac{1}{s}2^n\|\mathbf{u}\| + n^2 2^{\frac{3}{2}n+\frac{n^2}{8}}\frac{\sqrt[4]{n}}{\sqrt{\Omega\left(ds^n\right)}}\|\mathbf{u}\|$$

By picking $s = O\left(2^n\varepsilon^{-1}\right)$ and $k = (ds^n)^{n-2}$ we fulfill the approximation requirements.

The requirements on the cycle structure follow from the construction of the transformations. $\square$

5.4 Applications to CVP and SVP

In this section we will outline how the transformation can be used to find a solution to CVP and SVP, should these problems be easier to solve in lattices with many cycles.

In CVP our goal, given a lattice $\Lambda \subseteq \mathbb{Z}^n$ and a point $\mathbf{y} \in \mathbb{Z}^n$, is to find $\mathbf{x} \in \Lambda$ such that $\|\mathbf{x}-\mathbf{y}\|_p$ is minimized in some ℓ_p-norm. If (a slightly perturbed) $\mathbf{x}$ remains the lattice point closest to (a slightly perturbed) $\mathbf{y}$ after the transformation, we can reduce the instance of CVP to an instance of CVP in a lattice with many cycles. The following theorem shows how to choose the transformation parameters. The proof is given in the full version.

Theorem 5.4.1. *Let $\Lambda \subseteq \mathbb{Z}^n$, and let $\mathbf{y} \in \mathbb{Z}^n$. Let $\mathbf{x} \in \Lambda$ and $\mathbf{z} \in \Lambda$. Assume that all coordinates are in the interval $0, \ldots, \det(\Lambda) - 1$. It holds that if*

$$\|\mathbf{x} - \mathbf{y}\|_p < \|\mathbf{z} - \mathbf{y}\|_p$$

then

$$\left\|\frac{1}{k}\sigma_{\Lambda,\varepsilon}(\mathbf{x}) - \frac{1}{k}\sigma_{\Lambda,\varepsilon}(\mathbf{y})\right\|_p < \left\|\frac{1}{k}\sigma_{\Lambda,\varepsilon}(\mathbf{z}) - \frac{1}{k}\sigma_{\Lambda,\varepsilon}(\mathbf{y})\right\|_p$$

for

$$0 < \varepsilon < \frac{1}{2pn^{1+1/p}\det(\Lambda)^{p+1}}$$

if $p < \infty$ and

$$0 < \varepsilon < \frac{1}{2\det(\Lambda)}$$

if $p = \infty$ and k is polynomial in ε^{-1}.

Proof. The influence of the transformation on the distance between $\mathbf{x}$ and $\mathbf{y}$ is

$$\left| \|\mathbf{x}-\mathbf{y}\|_p - \left\| \frac{1}{k}\sigma_{\Lambda,\varepsilon}(\mathbf{x}) - \frac{1}{k}\sigma_{\Lambda,\varepsilon}(\mathbf{y}) \right\|_p \right| = \left| \|\mathbf{x}-\mathbf{y}\|_p - \left\| \frac{1}{k}\sigma_{\Lambda,\varepsilon}(\mathbf{x}-\mathbf{y}) \right\|_p \right| \leq$$

$$\left| \left\| (\mathbf{x}-\mathbf{y}) - \frac{1}{k}\sigma_{\Lambda,\varepsilon}(\mathbf{x}-\mathbf{y}) \right\|_p \right| \leq \varepsilon \|\mathbf{x}-\mathbf{y}\|_p \ .$$

In the same way we can compute $\left| \|\mathbf{z}-\mathbf{y}\|_p - \left\| \frac{1}{k}\sigma_{\Lambda,\varepsilon}(\mathbf{z}) - \frac{1}{k}\sigma_{\Lambda,\varepsilon}(\mathbf{y}) \right\|_p \right| \leq \varepsilon\|\mathbf{z}-\mathbf{y}\|_p$.

Using this we get

$$\left\| \frac{1}{k}\sigma(\mathbf{z}) - \frac{1}{k}\sigma(\mathbf{y}) \right\| - \left\| \frac{1}{k}\sigma(\mathbf{x}) - \frac{1}{k}\sigma(\mathbf{y}) \right\| \geq$$

$$(\|\mathbf{z}-\mathbf{y}\| - \|\mathbf{x}-\mathbf{y}\|) - \varepsilon(\|\mathbf{z}-\mathbf{y}\| + \|\mathbf{x}-\mathbf{y}\|) \ .$$

We want to pick ε to ensure this expression is greater than 0. For $p < \infty$ we have a lower bound for the first part of the expression as

$$\|\mathbf{z}-\mathbf{y}\| - \|\mathbf{x}-\mathbf{y}\| \geq \sqrt[p]{n \cdot \det(\Lambda)^p - 1} - \sqrt[p]{n \cdot \det(\Lambda)^p - 2} \geq \frac{1}{p}\left(n \det(\Lambda)^p\right)^{1/p-1}$$

and an upper bound for the second part as

$$\|\mathbf{z}-\mathbf{y}\| + \|\mathbf{x}-\mathbf{y}\| \leq 2\sqrt[p]{n}\det(\Lambda)$$

we have the necessary condition fulfilled if we pick

$$0 < \varepsilon < \frac{1}{p}\frac{1}{n \cdot \det(\Lambda)^p}\frac{1}{2\sqrt[p]{n}\det(\Lambda)}$$

If, on the other hand, $p = \infty$, we have $\|\mathbf{z}-\mathbf{y}\| - \|\mathbf{x}-\mathbf{y}\| \geq 1$ and $\|\mathbf{z}-\mathbf{y}\| + \|\mathbf{x}-\mathbf{y}\| \leq 2\det(\Lambda)$ so the condition holds if

$$0 < \varepsilon < \frac{1}{2\det(\Lambda)} \ .$$

□

The following two lemmas show how to use Theorem 5.4.1 to reduce CVP to a lattice with $n-1$ cycles. The first lemma follows directly from the fact that every lattice repeats itself in cubes with side $\det(\Lambda)$.

Lemma 5.4.2. *Let* $(\Lambda \subseteq \mathbb{Z}^n, \mathbf{y} \in \mathbb{Z}^n)$ *be an instance of* CVP. *Then for any* $\mathbf{u} \in \mathbb{Z}^n$ $\mathbf{x} \in \Lambda$ *is a solution if and only if* $\mathbf{x} - \det(\Lambda)\cdot\mathbf{u}$ *is a solution of the instance* $(\Lambda, \mathbf{y} - \det(\Lambda)\cdot\mathbf{u})$.

Lemma 5.4.3. *Let* $(\Lambda \subseteq \mathbb{Z}^n, \mathbf{y} \in \mathbb{Z}^n)$ *be an instance of* CVP *such that* $0 \leq y_i < \det(\Lambda)$. *Then* $\mathbf{x} \in \Lambda$ *is a solution if and only if* $\frac{1}{k}\sigma_{\Lambda,\varepsilon}(\mathbf{x})$ *is a solution of the instance* $\left(\frac{1}{k}\sigma_\varepsilon(\Lambda), \frac{1}{k}\sigma_{\Lambda,\varepsilon}(\mathbf{y})\right)$ *for* k *and* ε^{-1} *polynomial in* $\det(\Lambda)$ *and* n.

Proof. The lemma follows directly from Theorem 5.4.1. Using the two lemmas, we can construct the reduction by first reducing the target vector modulo $\det(\Lambda)$ and then apply the transformation with the appropriate value of ε. □

Obviously the same technique can be used to achieve a similar result for SVP. The following lemma follows directly from the above lemmas.

Lemma 5.4.4. *Let* $\Lambda \subseteq \mathbb{Z}^n$ *be an instance of* SVP. *Then* $\mathbf{x} \in \Lambda$ *is a solution if and only if* $\frac{1}{k}\sigma_{\Lambda,\varepsilon}(\mathbf{x})$ *is a solution of the instance* $\frac{1}{k}\sigma_\varepsilon(\Lambda)$ *for* k *and* ε^{-1} *polynomial in* $\det(\Lambda)$ *and* n.

From this we can conclude that the inapproximability results for SVP and CVP from [41] and [27] hold also for lattices with $n-1$ cycles.

Theorem 5.4.5. SVP *in* ℓ_p*-norm is* ***NP****-hard to approximate within any constant factor for* n*-dimensional lattices with* $n-1$ *non-trivial cycles of equal length.*

Theorem 5.4.6. *There exist constants* c_p *such that* CVP *is* ***NP****-hard to approximate within* $n^{\frac{c_p}{\log\log n}}$ *in* ℓ_p*-norm for* n*-dimensional lattices with* $n-1$ *non-trivial cycles of equal length.*

5.5 Conclusions

We have constructed a transformation that given an n-dimensional lattice of any cycle structure produces a lattice with $n-1$ cycles that is arbitrarily close to the original lattice. This closes the question of whether SVP and CVP can be easier to solve in lattices with many cycles. Using the presented result, such a solution would give a solution for the general case that is at most a polynomial factor slower in running time. Also the known inapproximability results for SVP and CVP extend to lattices with $n-1$ cycles.

By previous results, we know that any lattice can be approximated arbitrarily well by a cyclic lattice, and hence that SVP and CVP cannot be easier to solve in cyclic lattices than in general lattices, except possibly for a polynomial factor. We now have the two extremes, for one cycle and for $n-1$ cycles.

From the results by Ajtai and the improvement by others we have a hardness result also for lattices with n/c cycles. Together with our result this gives evidence for the general hypothesis that the cycle structure have little importance in deciding the hardness of SVP and CVP in a certain lattice.

Although it does seem likely that also lattices with m non-trivial cycles form a hard core for $2 \leq m \leq n-2$, we don't have a proof for this. The current proof does not easily extend to these cycle structures. Since our method relies on inflating the lattice by a factor d^t to get a lattice with determinant d^{nt+1} and then making changes to achieve m cycles, the length of each cycle is $d^{(nt+1)/m}$. Naturally t must be chosen so that $(nt+1)/m$ is an integer. In our case, we achieve this by setting $t = n - 2$ and $m = n - 1$. Since the value of t would depend on m and for certain relations between m and n no such t exists at all, our method cannot directly be generalized to create any cycle structure where the non-trivial cycles have equal length.

Even if a transformation into m cycles of equal length for $1 \leq m \leq n - 1$ were found it would still be an open question whether other cycle structures, where the cycles have different lengths, remain easy. Still the current result seems to be a strong indication that the cycle structure does not play an important role for the computational complexity of lattice problems.

Bibliography

[1] M. Ajtai. Generating hard instances of lattice problems. In *28th ACM Symposium on the Theory of Computing (STOC)*, pages 99–108. ACM Press, 1996.

[2] M. Ajtai. The shortest vector problem in ℓ_2 is **NP**-hard for randomized reductions. In *30th ACM Symposium on the Theory of Computing (STOC)*, pages 10–19. ACM Press, 1998.

[3] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270. Springer Verlag, 2000.

[4] G. Ateniese and G. Tsudik. Some open issues and directions in group signatures. In *Financial Cryptography '99*, volume 1648 of *Lecture Notes in Computer Science*, pages 196–211. Springer Verlag, 1999.

[5] L. Babai. Trading group theory for randomness. In *17th ACM Symposium on the Theory of Computing (STOC)*, pages 421–429. ACM Press, 1985.

[6] M. Bellare and O. Goldreich. On defining proofs of knowledge. In *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer Verlag, 1992.

[7] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer Verlag, 2003.

[8] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *20th ACM Symposium on the Theory of Computing (STOC)*, pages 103–118. ACM Press, 1988.

[9] F. Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444. Springer Verlag, 2000.

[10] F. Boudot and J. Traoré. Efficient publicly veriable secret sharing schemes with fast or delayed recovery. In *2nd International Conference on Information and Communication Security (ICICS)*, volume 1726 of *Lecture Notes in Computer Science*, pages 87–102. Springer Verlag, 1999.

[11] S. Brands. Untraceable off-line cash in wallets with observers. In *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 302–318. Springer Verlag, 1994.

[12] E. Brickell, P. Gemmell, and D. Kravitz. Tracing extensions to anonymous cash and the making of anonymous change. In *6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 457–466. ACM Press, 1995.

[13] J-Y. Cai and A. Nerurkar. An improved worst-case to average-case connection for lattice problems. In *38th IEEE Symposium on ACM Symposium on the Theory of Computing (STOC)*, pages 468–477. IEEE Computer Society Press, 1997.

[14] J. Camenisch. Efficient and generalized group signature. In *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 465–479. Springer Verlag, 1997.

[15] J. Camenisch and M. Michels. A group signature scheme with improved efficiency. In *Advances in Cryptology – ASIACRYPT'98*, volume 1514 of *Lecture Notes in Computer Science*, pages 160–174. Springer Verlag, 1999.

[16] J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes. In *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 413–430. Springer Verlag, 1999.

[17] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer Verlag, 1997.

[18] R. Canetti, O. Goldreich, and S. Halevi. The random oracle model revisited. In *30th ACM Symposium on the Theory of Computing (STOC)*, pages 209–218. ACM Press, 1998.

[19] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327. Springer Verlag, 1990.

[20] D. Chaum, E. van Heijst, and B. Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. In *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 470–484. Springer Verlag, 1991.

[21] D. Chaum and E. van Heyst. Group signatures. In *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer Verlag, 1991.

[22] L. Chen and T.P. Pedersen. New group signature schemes. In *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 171–181. Springer Verlag, 1994.

[23] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer Verlag, 1994.

[24] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer Verlag, 1998.

[25] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *6th ACM Conference on Computer and Communications Security (CCS)*, pages 46–51. ACM Press, 1999.

[26] I. Dinur. Approximating SVP_∞ to within almost polynomial factors is **NP**-hard. In *4th Algorithms and Complexity (CIAC)*, volume 1767 of *Lecture Notes in Computer Science*, pages 263–276. Springer Verlag, 2000.

[27] I. Dinur, G. Kindler, R. Raz, and S. Safra. Approximating CVP to within almost-polynomial factors is **NP**-hard. *Combinatorica*, 23(2):205–243, 2003.

[28] T. ElGamal. A public key cryptosystem and a signiture scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[29] EMVCo. *EMV2000 Integrated Circuit Card Specifications for Payment Systems*, December 2000. Available from `http://www.emvco.com` (September 203).

[30] U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero-knowledge proofs under general assumptions. *SIAM Journal of Computing*, 29(1):1–28, 1999.

[31] N.T. Ferguson. Single term off-line coins. In *Advances in Cryptology – EUROCRYPT'93*, volume 765 of *Lecture Notes in Computer Science*, pages 318–328. Springer Verlag, 1993.

[32] E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer Verlag, 1997.

[33] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *21st ACM Symposium on the Theory of Computing (STOC)*, pages 25–32. ACM Press, 1989.

[34] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[35] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal of Computing*, 18(1):186–208, 1989.

[36] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, 17(2): 281–308, 1988.

[37] J. Hoffstein, J. Pipher, and J.H. Silverman. NTRU: a ring based public key cryptosystem. In *Algorithmic Number Theory – ANTS-III*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer Verlag, 1998.

[38] T.M. Jurgensen and S.B Guthery. *Smart Cards: The Developer's Toolkit.* Prentice Hall PTR, 1st edition, 2002. ISBN 0-13-093730-4.

[39] R. Kannan and A. Bachem. Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix. *SIAM Journal of Computing*, 8:499–507, 1979.

[40] S. Khot. Hardness of approximating the shortest vector problem in high l_p norms. In *44th IEEE Symposium on ACM Symposium on the Theory of Computing (STOC)*, pages 290–297. IEEE Computer Society Press, 2003.

[41] S. Khot. Hardness of approximating the shortest vector problem in lattices. In *45th IEEE Symposium on ACM Symposium on the Theory of Computing (STOC)*. IEEE Computer Society Press, 2004.

[42] S. Kim, S. Park, and D. Won. Group signatures for hierarchical multigroups. In *Information Security Workshop – ISW'97*, volume 1396 of *Lecture Notes in Computer Science*, pages 273–281. Springer Verlag, 1998.

[43] J.C. Lagarias. The computational complexity of simultaneous diophantine approximation problems. *SIAM Journal of Computing*, 14:196–209, 1985.

[44] A.K. Lenstra, H.W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[45] A. Lysyanskaya and Z. Ramzan. Group blind digital signatures: A scalable solution to electronic cash. In *Financial Cryptography'98*, volume 1465 of *Lecture Notes in Computer Science*, pages 184–197. Springer Verlag, 1998.

[46] H. Maier. Primes in short intervals. *Michigan Mathematical Journal*, 32(2): 221–225, 1985.

[47] MasterCard International. *Terminal Requirements for Acceptance of Chip Pay Now (Debit) and Pay Later (Credit) Cards, version 4.0*, August 2001. Available from `http://www.mastercardintl.com` (September 2003).

[48] MasterCard International. *M/Chip 4 Card Application Specifications for Credit and Debit version 1.0*, October 2002. Available from `http://www.mastercardintl.com` (September 2003).

[49] R. Merkle. Protocols for public key cryptosystems. In *1980 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1980.

[50] S. Micali, M. Rabin, and J. Kilian. Zero-knowledge sets. In *44th IEEE Symposium on ACM Symposium on the Theory of Computing (STOC)*, pages 80–91. IEEE Computer Society Press, 2003.

[51] D. Micciancio. Improving lattice based cryptosystems using the Hermite normal form. In *Cryptography and Lattices (CaLC) 2001*, volume 2146 of *Lecture Notes in Computer Science*, pages 126–145. Springer Verlag, 2001.

[52] D. Micciancio. The shortest vector in a lattice is hard to approximate within some constant. *SIAM Journal of Computing*, 30:2008–2035, 2001.

[53] G.L. Miller. Riemann's hypothesis and tests for primality. *Journal of Compter and System Sciences*, 13:300–317, 1976.

[54] M. Newman. *Integral Matrices.* Academic Press, 1972. ISBN 0-12-517850-6.

[55] T. Okamoto and K. Ohta. Universal electronic cash. In *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 324–337. Springer Verlag, 1992.

[56] A. Paz and C.P. Schnorr. Approximating integer lattices by lattices with cyclic factor groups. In *14th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 267 of *Lecture Notes in Computer Science*, pages 386–393. Springer Verlag, 1987.

[57] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[58] The prime pages. `http://primes.utm.edu`, March 2004.

[59] The proth search page. `http://www.prothsearch.net`, March 2004.

[60] C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer Verlag, 1991.

[61] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21(2): 120–126, 1978.

[62] A. Sahai. Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security. In *40th IEEE Symposium on ACM Symposium on the Theory of Computing (STOC)*, pages 543–553. IEEE Computer Society Press, 1999.

[63] T. Sander and A. Ta-Shma. Auditable, anonymous electronic cash. In *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 555–572. Springer Verlag, 1999.

[64] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero knowledge. In *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 566–598. Springer Verlag, 2001.

[65] H.J.S. Smith. On systems of linear indeterminate equations and congruences. *Philosophical Transactions of the Royal Society of London*, 151:293–326, 1861.

[66] M. Stadler. Publicly verifiable secret sharing. In *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 190–199. Springer Verlag, 1996.

[67] M. Trolin. The shortest vector problem in lattices with many cycles. In *Cryptography and Lattices (CaLC) 2001*, volume 2146 of *Lecture Notes in Computer Science*, pages 194–205. Springer Verlag, 2001.

[68] M. Trolin. An efficient protocol for electronic cash. Technical Report TRITA-NA-0411, DDepartment of Numerical Analysis and Computer Science, Royal Institute of Technology (KTH), 2004.

[69] M. Trolin. Lattices with many cycles are dense. In *21st Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2996 of *Lecture Notes in Computer Science*, pages 433–444. Springer Verlag, 2004.

[70] M. Trolin and D. Wikström. Hierarchical group signatures. Technical Report TRITA-NA-0419, Department of Numerical Analysis and Computer Science, Royal Institute of Technology (KTH), 2004.

[71] P. van Emde Boas. Another **NP**-complete partition problem and the complexity of computing short vectors in lattices. Technical Report 81-04, Mathematics Department, University of Amsterdam, 1981. Also available at `http://turing.wins.uva.nl/~peter`.

[72] V. Varadharajan, K.Q. Nguyen, and Y. Mu. On the design of efficient RSA-based off-line electronic cash schemes. *Theoretical Computer Science*, 226: 173–184, 1999.

[73] Visa International. *VIS Visa Integrated Circuit Card 1.4.0*, April 2000. Available from `http://international.visa.com` (September 2003).

[74] A. Young and M. Yung. Finding length-3 positive Cunningham chains and their cryptographic significance. In *Algorithmic Number Theory – ANTS-III*, volume 1423 of *Lecture Notes in Computer Science*, pages 289–298. Springer Verlag, 1998.

[75] C. Zamfir, A. Damian, I. Constandache, and V. Cristea. An efficient ecash platform for smart phones. In *E_ COMM_ LINE 2004*, pages 5–9, 2004. ISBN 973-0-03671-3.