KTH Numerical Analysis
and Computer Science

On the Security of Mix-Nets and Related Problems

DOUGLAS WIKSTRÖM

Licentiate Thesis
Stockholm, Sweden 2004

TRITA NA 04-06
ISSN 0348-2952
ISRN KTH/NA/R--04/06--SE
ISBN 91-7283-717-9

KTH Numerisk analys och datalogi
SE-100 44 Stockholm
SWEDEN

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av teknologie licentiatexamen fredagen den 7 maj 2004 i sal D2, Kungliga Tekniska Högskolan, Lindstedtsvägen 5, Stockholm.

Tryck: Universitetsservice US AB

Abstract

We investigate a number of problems related to the notion of a mix-net. A mix-net is a cryptographic protocol that provides anonymity for a set of senders. The primary application for mix-nets is to ensure privacy in electronic elections.

The main contributions of the thesis are the following:

- We introduce a new technique for cryptographic protocols and propose a mix-net that is more efficient than any mix-net in the literature for a certain range of parameters. Currently we are unable to give security proofs for our construction.
- We prove that the security of a single mix-center is equivalent to the semantic security of the underlying cryptosystem.
- We give a number of attacks on a mix-net protocol proposed by Golle, Zhong, Boneh, Jakobsson, and Juels (2002). Two of the attacks may be applied to break the security of protocols by Jakobsson (1998), Mitomo and Kurosawa (2000), and Jakobsson and Juels (2001).
- We extend the attack of Lim and Lee (1997) based on the malicious use of illegal inputs to protocols, and illustrate why their proposed counter-measure to such attacks is insufficient.
- The El Gamal cryptosystem is malleable. This means that cleartexts may be transformed while encrypted. We characterize a class of transformations that can not be computed under encryption.
- We generalize the algorithm of Shallit and Sorenson (1993) for computing the Jacobi symbol, and give an efficient algorithms for computing the cubic residue character and the quartic residue character.

Sammanfattning

Svensk sammanfattning.

Contents

Acknowledgments

The larger part of the research presented in this thesis was conducted at the Adaptive Robust Computing group (ARC) at the Swedish Institute of Computer Science (SICS). I have since transfered to the Theoretical Computer Science group (TCS) of the Department of Analysis and Computer Science (Nada) at the Royal Institute of Technology (KTH), and would like to acknowledge the help and support from both places.

I am most grateful to my advisor Johan Håstad for his advise and support. Johan also helped me invent the attack of Section 4.2. I would further like to thank Michael Goldmann for his advise during Johan's stay at Princeton.

Nada and Johan deserve my deepest gratitude for providing me with the opportunity to finish this thesis and continue my research. My research at Nada was funded by Vetenskapsrådet (VR).

I am grateful to the former leader of the ARC group, Gunnar Sjödin, for his advise and support. Björn Grönvall and Torsten Ekedahl also deserves my gratitude for discussing mix-nets with me. The work in Chapter 7 is joint work with Anders Holst. He patiently listened to my ideas, and then we discussed the remaining problem until we found a solution. My research at SICS was funded by the Swedish Research Institute for Information Technology (SITI) and Verket för innovationssystem (VINNOVA).

Having lunch with the members of the ARC group resulted almost without exception in animated discussions. For this I thank Torsten Ekedahl, Daniel Gillblad, Anders Holst, Björn Levin, and Gunnar Sjödin.

Finally I thank Sofie.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Cryptographic Protocols

It was only 30 years ago that cryptography was considered a subject almost exclusively of military interest. Since then not only the applications have changed, but also the tools.

The original application of cryptography was to allow two parties to communicate secretly over an insecure channel, i.e. no intermediary should be able to deduce the cleartext despite having access to a cryptotext. Today there are complex protocols for key exchange, commerce, auctions, elections, contract signing etc, each with its own set of security properties.

The ground breaking work of Diffie and Hellman [16], and Rivest, Shamir and Adleman, revolutionized the subject of cryptography when they introduced the notion of public key cryptography. Although the notion was discovered first by Merkle [46], his idea was less practical and was not published until later. Furthermore, recent de-classified information claims that British governmental organizations discovered public key cryptography already in the beginning of the 70's (see [63]).

The new notion led to a practical way for two parties to establish a secret channel without any prior agreement. Furthermore, identification schemes and digital signatures were developed to serve as electronic replacements of conventional methods for identification and signing. These techniques are used thousands of times each day to secure and authenticate communication over the Internet.

The new tools were embraced by the scientific community. Then Yao [73] initiated the study of multiparty computation, i.e. the study of complex protocols involving several parties. Some years later Goldreich, Micali and Wigderson [30], and Ben-Or, Goldwasser, Wigderson [5] and Chaum, Crépeau and Damgård [9], showed that any function can be securely computed in two different models. The notion of zero-knowledge proofs was discovered by Goldwasser, Micali and Rackoff [32]. These fundamental results were followed by similar results in various models, and were more rigorously analyzed.

Although the above constructions are very general, they do not give practical protocols. Thus alongside the development of general theory, researchers have been investigating more practical solutions to specific problems. One such problem is how to implement an electronic election scheme.

Electronic Elections

The conventional election procedure has been refined over many years to be robust and to ensure correctness of the result. The cast votes are often recounted by different authorities to verify the result, and even though it may be possible to forge a single vote, it is very hard to forge a sufficient number of votes to change the result notably. Furthermore, in many countries the privacy of the voter is preserved by physical means. The intent is not only to guarantee the privacy of the voter, but to protect the voter against coercing and to counter vote buying.

Many questions arise when we try to replace the conventional election procedure with an electronic protocol. In particular if the goal is to let voters cast their ballot from their home PC. How do we identify the voters? How do we prevent multiple voting? Can the privacy of the voter be guaranteed, and for how long? Are there ways to coerce voters, or to convince them to sell their vote? Who verifies that the protocol was executed correctly?

Some of these problems seem impossible to solve completely in practice, e.g. it can not be guaranteed that there is not a virus on the home PC. Other problems, such as identifying voters, seem easy to solve as standalone problems using standard cryptographic techniques. The challenge is to find practical protocols and give convincing arguments of their security in reasonable models.

Although there are exceptions, most constructions for electronic voting in the literature fall into one of two categories.

The first type of construction is based on the use of a homomorphic cryptosystem. Such cryptosystems have the remarkable property that if several cryptotexts are multiplied, the result is a cryptotext of the product of the cleartexts of the original cryptotexts. This approach has been studied and refined by several authors [10, 3, 4] leading up to the work of Cramer, Gennaro, and Schoenmakers [11]. They give a well analyzed solution based on the El Gamal [19] cryptosystem. Damgård and Jurik [13] and independently O. Baudron, P.-A. Fouque, D. Pointcheval, G. Poupard and J. Stern [6], generalized this approach to use the Paillier [53] cryptosystem. The main advantage of this approach is that counting the votes can be done very efficiently. A drawback of this approach is that the set of candidates must be fixed in advance, i.e. there can be no write-in votes. Furthermore, the voters must give a relatively complex proof that their ballot contains a vote for a candidate on the candidate list. The problem is particularly severe if the El Gamal [19] cryptosystem is used, in which case only a small number of candidates can be used. If instead the Paillier [53] cryptosystem is used, the number of candidates can be large, but the candidates must still be fixed in advance.

The second approach was invented by Chaum [8]. His idea is a straightforward translation of the conventional voting procedure. Each voter encrypts its vote and writes it on a bulletin board. Then a mix-net (or anonymous channel) is used to mix and decrypt the cryptotexts in such a way that it is infeasible to find any correspondence between the input cryptotexts and the output cleartexts. The mix-net ensures the privacy of the voter. The main advantage of this approach is that arbitrary write-in votes can be allowed. A fixed set of candidates can also be used. On the other hand all known mix-net protocols are relatively inefficient. Another principal drawback with write-in votes is that it is easy to coerce a voter to cast a blank vote. The coercer simply chooses a random string and asks the voter to write this on his ballot. Then the coercer verifies that the random string appears in the output. Although this problem already exists for conventional voting procedures the problem is worsened when the resulting cleartexts are stored electronically, since it makes it easier for the coercer to verify that the random string appears in the output. Interestingly, ideas from the second approach have been used by Hirt and Sako [36] to counter vote selling in an election protocol of the first type.

In this thesis we consider the notion of a mix-net and related problems.

1.2 Further Applications and Previous Work on Mix-Nets

Although electronic elections is perhaps the most spectacular application of mix-nets, Chaum proposes the notion as a general means for a group of senders to send messages without revealing their identity.

Chaum's original "anonymous channel" [8, 52] enables a sender to securely send mail to a receiver anonymously, and also to securely receive mail from this recipient without revealing the sender's identity. When constructing election schemes [8, 24, 54, 59, 51] the mix-net is used to ensure that the vote of a given voter can not be revealed. Also in the construction of electronic cash systems [39] mix-nets have been used to ensure privacy. Thus a mix-net is a useful primitive in constructing cryptographic protocols.

Abe gives an efficient construction of a general mix-net [1], and argues about its properties. Jakobsson has written (partly with Juels) a number of more general papers on the topic of mixing [38, 40, 41] also focusing on efficiency, of which the first appeared at the same time as Abe's construction.

Recently two groups have proposed efficient zero-knowledge proofs of a correct shuffle in the random oracle model. Furukawa and Sako [25] and Neff [50] have both found efficient ways to compute such proofs. Groth [33] has refined Neff's ideas and performed a more thorough analysis.

Golle et al. [34] presents a new idea for providing robustness.

Desmedt and Kurosawa [15] describe an attack on a protocol by Jakobsson [38]. Similarly Mitomo and Kurosawa [49] exhibit a weakness in another protocol by Jakobsson [40]. Pfitzmann has given some general attacks on mix-nets [58, 57], and Michels and Horster give additional attacks in [48]. Abe and Imai [2] presents attacks for the protocol of Golle et al. [34] and also for the protocol of Jakobsson and Juels [42]. Their attacks for [34] are similar to two of the attacks of Chapter 4, but the attack they give for [42] is unrelated to the attack we give in Chapter 4 for this construction.

1.3 What is a Mix-Net?

Suppose a set of senders $S_1, \ldots, S_N$ each have an input m_i, and want to compute the *unordered* set $\{m_1, \ldots, m_N\}$, but keep the identity of the sender of any particular message m_i secret.

A trusted party T can provide the service required by the senders. First it collects all messages. Then it randomly re-orders the inputs and outputs the result. This is illustrated in Figure 1.1.

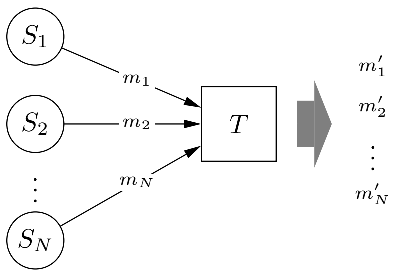

Figure 1.1: The trusted party T receives a message m_i from each sender S_i. Then it orders $m_1, \ldots, m_N$ randomly to form a list $(m'_1, \ldots, m'_N)$ that it outputs.

A protocol, i.e. a set of machines $M_1, \ldots, M_k$, that emulates the service of the trusted party as described above is called a *mix-net*, and the parties $M_1, \ldots, M_k$ are referred to as *mix-servers*. Some authors use the terms: anonymous channel, mix, or shuffle-network and mix-centers or mixers instead of mix-net and mix-server respectively.

The usual assumption is that each sender S_i trusts that a certain fraction of these are honest. The sender may not want to decide upon any particular parties M_j to trust, but only how many. This type of trust is common in the real world. For example the board of a company consists of a group of individuals. The stockholder need not trust each individual to trust the board as a whole.

Theoretical work shows that a secure mix-net can be constructed (cf. [29]). However, in practice the mere existence of a mix-net is not enough. It must also be *efficient*.

The Basic Approach

The notion of a mix-net was first investigated by Chaum [8]. His approach can be summarized as follows. Each mix-server M_j has a public key pk_j and a private key sk_j. We write $c = E_{\mathrm{pk}}(m)$ for the probabilistic encryption of a message m using the public key pk and $D_{\mathrm{sk}}(c) = m$ for the decryption of a cryptotext c using the private key sk. To send a message m_i the sender S_i forms $c_{0,i} = E_{\mathrm{pk}_1}(E_{\mathrm{pk}_2}(\ldots E_{\mathrm{pk}_k}(m_i)\ldots))$ and writes this on a bulletin board. This gives a list $L_0 = (c_{0,1}, \ldots, c_{0,N})$ of cryptotexts. Then for $j = 1, \ldots, k$ the j:th mix-server M_j decrypts each element $c_{j-1,i}$ of L_{j-1} by computing $D_{\mathrm{sk}_j}(c_{j-1,i})$ and then forms a new list $L_j = (c_{j,1}, \ldots, c_{j,N})$ consisting of the decrypted elements, but in random order. By construction the output of M_k is the list of cleartexts randomly permuted. If at least one mix-server keeps its part of the permutation secret, essentially no information of the full permutation is revealed and the anonymity of the senders is ensured. Below we discuss how a bulletin board may be implemented electronically.

A drawback of the direct implementation of the above method is that the size of the cryptotext $c_{0,i}$ computed by the sender grows linearly with the number of mix-servers k. This may be avoided by use of a homomorphic cryptosystem.

Another, more important, problem is that there is no obvious way to verify that the mix-servers behave honestly. Indeed any of the mix-servers could replace all of the cryptotexts with other cryptotexts of its choice. It is important that the sender can trust the result of a mix-net. A mix-net is said to be robust if it outputs the correct result as long as a certain fraction, e.g the majority of the mix-servers, are honest.

Chaum's mix-net can be said to be decryption-based. Later constructions use almost without exception the El Gamal cryptosystem and are re-encryption based. Suppose we write $c = E_y(m, r)$ for the encryption of a message m using a public key y and randomness r. The homomorphic property of the El Gamal cryptosystem implies that there is a random re-encryption function F such that $F_y(c, s) = E_y(m, r + s)$. In effect anybody with knowledge of the public key can add randomness to a cryptotext. This property is used heavily in re-encryption based mix-nets.

In a re-encryption mix-net there is a joint public key y such that the corresponding private key is shared by the mix-servers. Thus a quorum of the mix-servers can decrypt a cryptotext, but any individual mix-server gets no information on its own. The sender S_i simply encrypts his message m_i using the public key y, $c_{0,i} = E_y(m_i)$, and writes $c_{0,i}$ on the bulletin board. This gives a list $(c_{0,i}, \ldots, c_{0,N})$ of cryptotexts. Then for $j = 1, \ldots, k$ the j:th mix-server M_j re-encrypts each element $c_{j-1,i}$ of L_{j-1} by computing $F_y(c_{j-1,1}, r_{j,i})$ with fresh randomness $r_{j,i}$ and then forms a new list $L_j = (c_{j,1}, \ldots, c_{j,N})$ consisting of the re-encrypted elements, but in random order. Finally the mix-servers jointly decrypt all elements in L_k to find the result. Thus the sequential decryptions are replaced by sequential re-encryptions. This approach is illustrated in Figure 1.2.

El Gamal based mix-nets avoid the first drawback of Chaums approach, i.e.

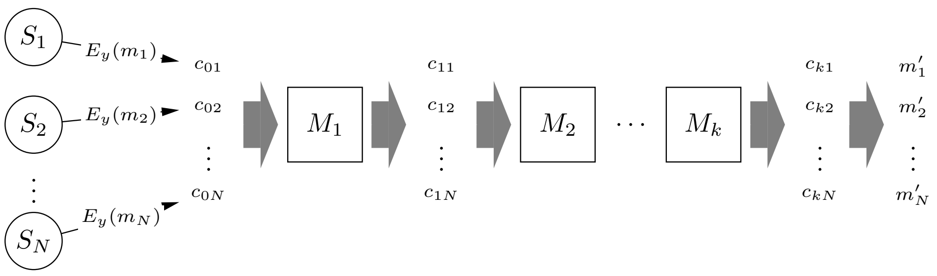

Figure 1.2: The figure illustrates the re-encryption and permutation in a Mix-Net. Each sender S_i sends an encrypted message c_{0i}. The list of messages $(c_{01}, \ldots, c_{0N})$ is repeatedly re-encrypted and permuted by the mix-servers $M_1, \ldots, M_k$. Finally the mix-servers jointly decrypt the list $(c_{k,1}, \ldots, c_{k,N})$.

that the cryptotext computed by a sender grows with the number of mix-servers.

There are mainly two different approaches for providing robustness for such a mix-net. The first approach requires each mix-server to prove in zero-knowledge that it performs its actions correctly. The second approach is more optimistic and tries to find more efficient tests for the overall correctness of an execution. Only in the event of detected cheating more careful tests are performed to identify cheaters.

1.4 Preliminaries

In this section we introduce general notation and conventions used throughout the thesis. We also review a number of definitions and constructions of some primitives.

Notation

Throughout, the parties of any mix-net protocol are: senders denoted by S_i, and mix-servers denoted by M_j. The protocols under discussion differ with the chapters.

We use PC to denote the set of polynomial size circuit families. The notation 1^n is used to denote n in unary basis, i.e. a sequence of n ones.

We consistently use the expression “chosen randomly” instead of “chosen uniformly and independently at random”. Unless otherwise stated, all random variables are independent of all other random variables.

Let X be a random variable with probability function $p_X : \{0,1\}^n \to [0,1]$, and let M be a string describing a probabilistic circuit or Turing machine. We use the notation $M(X)$ for the induced random variable resulting when M is run on input X.

Let Σ_N be the group of permutations on N elements. If $N(n)$ is a function, we abuse notation and write Σ_N for the family $\{\Sigma_{N(n)}\}$.

Throughout we denote by $G_q = \langle g \rangle$ a group of prime order q with generator g. To make assumptions on the complexity of computational problems in G_q we must consider families of such groups and consider specific representations of these groups. When nothing else is explicitly stated, the reader may think of G_q as a large subgroup of the multiplicative group modulo a prime p, $\mathbb{Z}_p^*$. Let $q_1, q_2, \ldots$ be an increasing sequence of prime numbers, where $\lceil \log_2 q_n \rceil = n$, and let $G_{q_1}, G_{q_2}, G_{q_3}, \ldots$ be a corresponding sequence of groups. As usual in the cryptographic literature we abuse notation and write $q = \{q_n\}$, $\mathbb{Z}_q = \{\mathbb{Z}_{q_n}\}$, and $G_q = \{G_{q_n}\}$ to denote these families. Thus whenever we refer to G_q as a group we are really referring to G_{q_n} for some particular security parameter n.

This convention greatly simplifies the exposition and is used consistently also for other families of objects, e.g. if we should write "for each n, X_n is a random variable distributed over $\mathcal{M}_n$", we instead write "$X = \{X_n\}$ is uniformly distributed over $\mathcal{M}$".

The Definition of a Cryptosystem and GM-Security

Let $\mathcal{M}_n = \{0,1\}^n$, and $\mathcal{M} = \{\mathcal{M}_n\}$. The following definition of a cryptosystem is given by Micali, Rackoff, and Sloan in [47].

Definition 1.1 (Public Key Cryptosystem, cf. [47]). A *Public Key Cryptosystem* is a probabilistic Turing machine C running in expected polynomial time that on input 1^n outputs the description of two probabilistic circuits E_n and D_n of polynomial size in n such that for a polynomial $\kappa(n)$:

1. The encryption circuit E_n has n inputs and $\kappa(n)$ outputs.
2. The decryption circuit D_n has $\kappa(n)$ inputs and n outputs.
3. $\forall m \in \mathcal{M}$, $\forall c > 0$, $\exists n_0$ such that for $n > n_0$:

$$\Pr[D(E(m)) = m] > 1 - \frac{1}{n^c} \ .$$

The definition of semantic security given in [47] is a slightly changed version of a definition given by Goldwasser and Micali in [31]. Together these two papers give a proof of equivalence of the definition of semantic security of a cryptosystem and Definition 1.2 below.

Definition 1.2 (GM-security, cf. [47]). Let $(E, D) = \{(E_n, D_n)\} = \{C(1^n)\}$, where C is a public key cryptosystem, and let b be uniformly and independently distributed in $\{0,1\}$. C is *GM-secure* if $\forall m_0, m_1 \in \mathcal{M}$, $\forall T \in \mathrm{PC}$ and $\forall c > 0$, $\exists n_0$ such that $\forall n > n_0$:

$$\left| \Pr[T(E, m_0, m_1, E(m_b)) = m_b] - \frac{1}{2} \right| < \frac{1}{n^c} \ .$$

The Decision Diffie-Hellman Assumption

We use the El Gamal cryptosystem described in Section 1.4 extensively. The security of this cryptosystem rests on the Decision Diffie-Hellman assumption (DDH-assumption) described below.

Definition 1.3 (Decision Diffie-Hellman assumption). Let $\alpha, \beta, \gamma \in \mathbb{Z}_q$ be randomly chosen. The (non-uniform) Decision Diffie-Hellman assumption for G_q states that $\forall A \in \mathrm{PC}$, $\forall c > 0$, $\exists n_0$, such that for $n > n_0$

$$|\Pr[A(g^\alpha, g^\beta, g^\gamma) = 1] - \Pr[A(g^\alpha, g^\beta, g^{\alpha\beta}) = 1]| < \frac{1}{n^c} \ .$$

The DDH-assumption is a well founded conjecture, but unproven. It is believed to hold whenever G_q is a subgroup of the multiplicative group of a field or a "cryptographically secure" elliptic curve. For concreteness we often assume that G_q is a subgroup of the multiplicative group modulo a prime p, i.e. $\mathbb{Z}_p^*$, for some prime p. For a more detailed introduction to this assumption we suggest the survey of Boneh [7].

The El Gamal Cryptosystem

The El Gamal cryptosystem [19] is employed in a group G_q of the type discussed above.

The private key $x \in \mathbb{Z}_q$ is chosen randomly, and then the corresponding public key (g, y) is defined by $y = g^x$. Sometimes g is a system-wide parameter, in which case we refer to y as the public key.

Encryption of a message $m \in G_q$ using the public key y is given by

$$E_{(g,y)}(m, r) = (g^r, y^r m) \ ,$$

where $r \in \mathbb{Z}_q$ is chosen randomly, and decryption of a cryptotext on the form $(u, v) = (g^r, y^r m)$ using the private key x is given by

$$D_x(u, v) = u^{-x} v = m \ .$$

Tsionis and Yung [65] shows that the El Gamal cryptosystem is GM-secure [31, 47] under the DDH-assumption.

Note that messages are assumed to be in G_q. This means that to meet the formal requirements of Definition 1.1, an additional assumption on G_q is needed. There must be a way to encode an arbitrary string uniquely as an element in G_q. In practice this is not a problem for a multiplicative group modulo a prime, or for an elliptic curve group.

When we do not use the random input r explicitly, we write $E_{(g,y)}(m)$, and when g is a system-wide parameter we write $E_y(m, r)$ or $E_y(m)$. Furthermore, if $m = (m_1, \ldots, m_N)$ is a list of elements $m_i \in G_q$, we extend our notation in the natural way and write $E_y(m)$ to denote $(E_y(m_1), \ldots, E_y(m_N))$. This convention is used also for the decryption function and the re-encryption function defined below.

The El Gamal system is said to be homomorphic or have the re-encryptability property. This means that if $(u_0, v_0) = E_y(m_0, r_0)$ and $(u_1, v_1) = E_y(m_1, r_1)$, then $(u_0u_1, v_0v_1) = E_y(m_0m_1, r_0 + r_1)$, i.e. the component-wise product of two cryptotexts is a cryptotext of the product of the cleartexts. The homomorphic property also allows random re-encryption defined as follows

$$F_{(g,y)}(u, v, r) = (g^r u, y^r v) \ .$$

If (u, v) is an encryption of m, then $F(u, v, r)$ is another randomly chosen encryption of m. This is a very useful property in the construction of protocols.

The following two functions allow easy description of a distributed El Gamal cryptosystem.

$$P_x(u, v) = u^{-x}, \quad \text{and} \quad D_P(u_x, (u, v)) = vu_x \ .$$

If $(u, v) = E_{(g,y)}(m, r)$ we have that $D_P(P_x(u, v), (u, v)) = m$. More interestingly, if $x = x_1 + x_2$, and $y = y_1y_2$, where $y_1 = g^{x_1}$, and $y_2 = g^{x_2}$, we have that $P_{x_1}(u, v)P_{x_2}(u, v) = u^{-x_1}u^{-x_2} = u^{-x}$. This gives the following useful identity $D_P(P_{x_1}(u, v)P_{x_2}(u, v), (u, v)) = m$.

Thus we may think of the function P_x as computing partial decryption factors, and the function D_P to simply perform the decryption given the appropriate decryption factor.

Bulletin Board

TODO: Diskutera antagande och implementationer? We assume the existence of a bulletin board. Each party may write on a special part of the bulletin board, but no party can erase anything from the bulletin board. The bulletin board is modeled as a trusted party.

We use the term "publish" to denote that a party writes something on the bulletin board.

1.5 Our Results

This thesis is based on the following papers.

1. D. Wikström, *An Efficient Mix-Net*, Swedish Institute of Computer Science (SICS) Technical Report T2002:21, ISSN 1100-3154, SICS-T-2002/21-SE, february 2002, `http://www.sics.se`.

   The paper proposes a practical and efficient mix-net based on a combination of the notion of "repetitition robustness" introduced by Jakobsson [40] and "double encryption" introduced here.

2. D. Wikström, *The Security of a Mix-Center Based on a Semantically Secure Cryptosystem*, Indocrypt 2002, LNCS 2551, pp. 368-381, 2002.

   The paper investigates relations between the semantic security of a cryptosystem, and the security of a mix-center based on such a cryptosystem.

3. D. Wikström, *Five Practical Attacks for "Optimistic Mixing for Exit-Polls"*, proceedings of the Tenth Annual Workshop on Selected Areas in Cryptography (SAC '03), LNCS 3006, pp. 2004.**TODO: fixa detta.**

   The paper gives five practical attacks for the mix-net construction proposed by Golle, Zhong, Boneh, Jakobsson and Juels [34]. Two of our attacks break two other mix-nets [40, 49] and [42] in the literature.

4. D. Wikström, *Elements in $\mathbb{Z}_p^*\backslash G_q$ are Dangerous*, Swedish Institute of Computer Science (SICS) Technical Report: T2003:05, ISSN: 1100-3154, ISRN: SICS-T-2003/05-SE, february 2003, `http://www.sics.se`.

   The paper investigates how certain corrupt inputs may be used maliciously when secure subprotocols are applied without verifying the form of inputs explicitly.

5. D. Wikström, *A Note on the Malleability of the El Gamal Cryptosystem*, Indocrypt 2002, LNCS 2551, pp. 176-184, 2002.

   The paper investigates in what ways the cleartext of an El Gamal cryptotext may be transformed under encryption. It gives a class of hard functions.

6. D. Wikström, A. Holst, *Algorithms for the Cubic and Quartic Residue Characters*, Manuscript, may 2003.

   The paper describes how the binary algorithm for computing the Jacobi symbol of Shallit and Sorenson [62], can be generalized to the cubic and quartic case.

   The author of this thesis constructed the generalized algorithms using the standard norms in $\mathbb{Z}[\omega]$ and $\mathbb{Z}[i]$, but the special norms that we present were discovered during discussions with Anders Holst.

Chapter 2

An Efficient Mix-Net

In many applications it is expected that mix-servers hardly ever behave maliciously, since the threat of bad publicity in the event of detection is too great. Thus it is worthwhile to optimize the execution of a mix-net in the case that all mix-servers behave honestly. We target such applications where the number of senders N is very large, e.g. $N = 10^5$, and the number of mix-servers k is relatively large as well, e.g. $k = 300$, but $k^2 < N$.

We assume that there exists a slow backup mix-net. Then we propose a mix-net that is very efficient if all mix-servers behave honestly. If on the other hand cheating is detected, the mix-servers resort to the slower backup mix-net.

The efficiency of our construction is based on the combination of the notion of repetitive robustness, introduced by Jakobsson [40], and the notion of *double encryption* introduced here.

The combination of double encryption and repetitive robustness enables us to avoid the large number of zero-knowledge proofs of knowledge required in most mix-net constructions. Given that the mix-servers precompute exponentiations we estimate the running time of our protocol to $(10\sigma + 7)N$ exponentiations. The factor σ is a small integer, e.g. $\sigma = 3$ for the values of N and k given above.

The running time of our construction should be contrasted with previous mix-nets which require at least $\Omega(kN)$ exponentiations, where k is the number of mix-servers. Thus when N is very large and k is fairly large our construction is practical, whereas other constructions are impractical.

Currently we are unable to prove the security of our construction. We only argue informally and explain the underlying ideas on which the security is believed to rest. It seems that new tools must be developed to prove the security of the protocol. Since we do not have a proof of security even when the subprotocols are modeled as trusted parties we do not consider the problem of realizing subprotocols such that they compose well with our protocol. We only give some references for how these subprotocols have been realized in the literature.

The exposition in this chapter differs from a previous version [70] of this work in that we assume the existence of trusted parties as a replacement of subprotocols. This simplifies the exposition of the protocol and allows us to focus our main contribution.

We note that Golle et al. [34] independently introduced the notion of "double enveloping".

2.1 Additional Notation

We introduce some additional notation used only in this chapter.

Let $L = (u_1, \ldots, u_s) \in G_q^s$ be a list of elements. If $s = tN$ we may view L as a list $L = \{(u_{i,1}, \ldots, u_{i,t})\}_{i=1}^N$, of N t-tuples such that $(u_{i,1}, \ldots, u_{i,t}) \in G_q^t$, where we set $u_{i,j} = u_{(i-1)t+j}$. Each notation we describe below has a parameter t, either explicitly or implicitly, that indicates how we view the list L. This allows us to treat the list L as a list of differently sized tuples depending on the context.

Let $\pi \in \Sigma_N$ be a permutation. We abuse notation, and write πL for the list $\{(u_{\pi^{-1}(i),1}, \ldots, u_{\pi^{-1}(i),t})\}_{i=1}^N$ where the tuples have been permuted. The size t of the permuted tuples is implicitly determined by $t = s/N$, since $\pi \in \Sigma_N$.

We need a notation for a list of tuples sorted lexicographically. We let $\mathrm{Sort}_t(L)$ be the function that returns a pair (L', π), where L' is the list consisting of the t-tuples of L sorted lexicographically, and π is the permutation such that $L' = \pi L$.

We define $\mathrm{Expand}_\sigma(L)$ to be the list where each element of L has been duplicated σ times. We let $\mathrm{Compact}_{t,\sigma}(L)$ be a function that returns (L', A) defined as follows. L' is a list of unique t-tuples of L for which there exists exactly σ copies in L. A is the set of indices of t-tuples in L of which there does not exist exactly σ copies.

Let $R, R' \in \mathbb{Z}_q^s$, where we let $R = \{r_i\}_{i=1}^s$ and $R' = \{r'_i\}_{i=1}^s$. We denote by $R + R'$ the component-wise sum $R + R' = \{r_i + r'_i\}_{i=1}^s$.

Commitment Scheme

We assume the existence of an ideal commitment scheme. This is modeled as a trusted party, to which the commiter sends its value. When the commiter decides to open its commitment it informs the trusted party, which reveals the commited value to all other parties.**TODO: Kolla stavning commitments!**

Commitment schemes can be implemented under a variety of standard assumptions. **TODO: referenser?**

In the random oracle model the simplest commitment scheme is the following**TODO: Referens för detta commitment-scheme?**. To commit to a string s, the commiter chooses $r \in \mathbb{Z}_q$ randomly and computes $c = h(s, r)$. Then c is handed to the receiver (or in our protocol written on the bulletin board). To open the commitment the commiter hands s and r to the receiver (or bulletin board). This allows the receiver to verify the relation $c = h(s, r)$.

This scheme is very practical when the random oracle h is implemented by one of the standard cryptographic hash functions, e.g. SHA [22, 23], and intuitively the potential problem of leakage of the commited bits is particularly small when the string s is very long as it is in our application.

Key Generation

The protocol requires two distributed El Gamal systems, whose public keys are related in a special way. We refer to these systems as the outer and inner system respectively. We begin by describing the keys of the outer system.

For each $j = 1, \ldots, k$, let $x_{j,\mathrm{out}} \in_R \mathbb{Z}_q$ be the randomly chosen secret key of M_j of the outer El Gamal system and let $(g, y_{j,\mathrm{out}})$ be the corresponding public key. Then define

$$x_{\mathrm{out}} = \sum_{j=1}^{k} x_{j,\mathrm{out}} \ , \qquad \text{and} \qquad y_{\mathrm{out}} = \prod_{j=1}^{k} y_{j,\mathrm{out}} \ .$$

By construction x_{out} and (g, y_{out}) make a key pair of a joint El Gamal system. Anybody may compute y_{out}, but x_{out} remains secret.

The inner El Gamal system is not based on the generator g, but uses y_{out} as a generator instead. The element y_{out} is a generator since the order q of G_q is prime. Thus for each $j = 1, \ldots, k$, let $x_{j,\mathrm{in}} \in_R \mathbb{Z}_q$ be the randomly chosen secret key of M_j of the inner El Gamal system and define $y_{j,\mathrm{in}} = y_{\mathrm{out}}^{x_{j,\mathrm{in}}}$. Then $(y_{\mathrm{out}}, y_{j,\mathrm{in}})$ is a public key corresponding to the secret key $x_{j,\mathrm{in}}$. Then define

$$x_{\mathrm{in}} = \sum_{j=1}^{k} x_{j,\mathrm{in}} \ , \qquad \text{and} \qquad y_{\mathrm{in}} = \prod_{j=1}^{k} y_{j,\mathrm{in}} \ .$$

Similarly as for x_{out} and (g, y_{out}), we have that x_{in} and $(y_{\mathrm{out}}, y_{\mathrm{in}})$ make a key pair of a joint El Gamal system.

We assume a trusted party that generates and distributes the keys to the mix-servers. If more than a majority of the mix-servers asks the trusted party to reveal a certain part $x_{j,\mathrm{out}}$ (or $x_{j,\mathrm{out}}$) of the key it does so. Thus the trusted party models a distributed key generation in which any key can be reconstructed by a majority.

To replace the trusted party with a cryptographic protocol is non-trivial. A well analyzed construction for this purpose is given by Gennaro et al [28].

Zero-Knowledge Proof of Knowledge

We assume the existence of two types of ideal zero-knowledge proofs of knowledge.

In the first type, a tuple $[g, y_{\mathrm{in}}, y_{\mathrm{out}}, \alpha]$ where we have $\alpha = E_{y_{\mathrm{out}}}(E_{y_{\mathrm{in}}}(m)) = [(\mu_1, \mu_2), (\nu_1, \nu_2)]$ for some $m \in G_q$ is given. The encryptor is required to prove knowledge of $r, s, t \in \mathbb{Z}_q$ such that $\mu_1 = g^r$, $\mu_2 = y_{\mathrm{out}}^s$, $\nu_1 = g^t$, i.e. the encryptor must prove knowledge of m.

In the second type a tuple $(y, \{(u_i, v_i), \lambda_i\}_{i=1}^{l})$, where $\alpha_i, \lambda_i \in G_q$ is given and a party is required to prove knowledge of x such that $y = g^x$ and $\{\lambda_i\}_{i=1}^{l} = P_x(\{(u_i, v_i)\}_{i=1}^{l})$, i.e. that $\lambda_i = u_i^{-x}$.

In both cases we model the proof of knowledge as a trusted party to which the prover sends the tuple and his witness. A verifier may query the trusted party on a certain prover and tuple. If the trusted party previously received a valid witness from the prover in question it returns true, and otherwise false.

The above proofs of knowledge can be implemented using standard Schnorr-signature similar methods as outlined below. To conclude that these protocols are zero-knowledge proofs of knowledge in the random oracle model the results in e.g. Schnorr [60] or Tsionis and Yung [65] need only be modified slightly.

Proof of Knowledge of Cleartext

1. The prover chooses $\gamma_1, \gamma_2, \gamma_3 \in \mathbb{Z}_q$ randomly and computes $\beta_1 = g^{\gamma_1}$, $\beta_2 = y_{\mathrm{out}}^{\gamma_2}$, and $\beta_3 = g^{\gamma_3}$. Then it computes $c = h(g, y_{\mathrm{in}}, y_{\mathrm{out}}, \alpha, \beta_1, \beta_2, \beta_3)$ and $d_1 = cr + \gamma_1$, $d_2 = cs + \gamma_2$, and $d_3 = ct + \gamma_3$. The proof consists of the tuple (c, d_1, d_2, d_3).

2. The verifier accepts the proof if $c = h(g, y_{\mathrm{in}}, y_{\mathrm{out}}, \alpha, g^{d_1}\mu_1^{-c}, y_{\mathrm{in}}^{d_2}\mu_2^{-c}, g^{d_3}\nu_1^{-c})$.

The verifier computes 6 exponentiations to verify a proof.

Proof of Correct Decryption

1. The prover chooses $\gamma \in \mathbb{Z}_q$ randomly and computes $\beta = g^{\gamma}$ and $\beta_i = u_i^{\gamma}$ for $i = 1, \ldots, l$. Then it computes $c = h(g, y, \{(u_i, v_i)\}_{i=1}^{l}, \beta, \{\beta_i\}_{i=1}^{l})$ and $d = cx + \gamma$. The proof consists of the tuple (c, d).

2. The verifier accepts the proof if $c = h(g, y, \{(u_i, v_i)\}_{i=1}^{l}, g^d y^{-c}, \{u_i^d \lambda_i^c\}_{i=1}^{l})$.

The prover computes $l + 1$ exponentiations to construct a proof, and it only needs $\log_2 q$ random bits. The verifier computes $2(l + 1)$ exponentiations to verify a proof.

Backup Mix-Net

We assume the existence of a backup mix-net that given a list L of El Gamal cryptotexts outputs the cleartexts in random order. We model this as a trusted party that cooperates with the key generator. The trusted party waits until it receives identical lists L from a majority of the mix-servers. Then it asks the key generator for the joint inner secret key x_{in} and outputs $\mathrm{Sort}_1(D_{x_{\mathrm{in}}}(L))$.

It may seem odd that we assume the existence of a mix-net, but recall that our goal is to construct an *efficient* mix-net in the case where there are many senders and mix-servers and all mix-servers behave honestly.

There are numerous proposals for efficient mix-nets. One of the best analyzed is the protocol of Groth [33]. Groth generalizes the construction of Neff [50] and gives a more rigorous analysis.

2.2 The Protocol

The protocol described in detail below may seem complicated, but it is based on two simple ideas: a variant of repetition robustness introduced by Jakobsson [40], and the use of "double encryption" introduced here.

Overview of the Protocol

Before we give any details we explain the key steps of the protocol.

At the start of the protocol each sender encrypts its message twice. First it encrypts its cleartext using the inner layer public key $(y_{\text{out}}, y_{\text{in}})$ and then it encrypts the resulting cryptotext using the outer layer public key (g, y_{out}). Finally it constructs a proof of knowledge of the randomness used in both layers of encryption as described in Section 2.1.

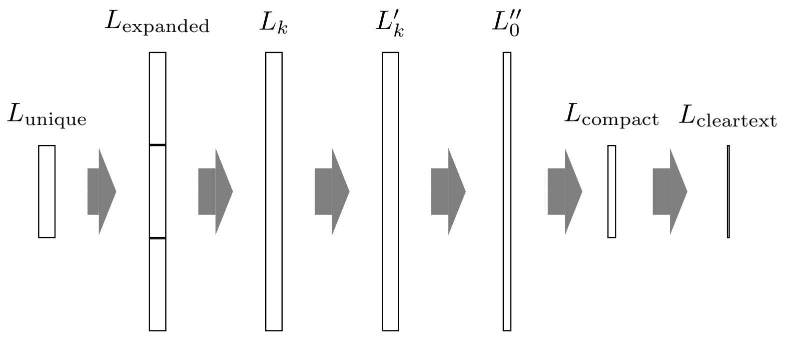

Figure 2.1: The figure illustrates the key phases of the protocol.

At the start of the protocol each mix-server constructs the list L_{unique} of correctly formed cryptotexts written on the bulletin board by senders. A cryptotext is considered correct if it is accompanied by a proof of knowledge of the randomness used during encryption. Any duplicate cryptotexts are also removed, and considered incorrect.

Then an expanded list L_{expanded} is formed consisting of σ identical copies of each cryptotext from the original list L_{unique}. Recall that N is the number of senders. The parameter σ is chosen such that $1/N^{\sigma-1}$ is negligible.

The mix-servers jointly re-encrypt and permute the list L_{expanded} to form L_k, but only the outer layer is re-encrypted (recall that each sender double encrypted its message). This is done sequentially; each mix-server re-encrypts and permutes the output of the previous mix-server as illustrated in Figure 1.2.

This is followed by another round of re-encryption and permutation, where only the outer layer is re-encrypted. The result is the list L'_k.

Next the mix-servers jointly decrypt each cryptotext in the list L'_k to form the list of inner cryptotexts L''_0. We stress that only the outer layer of each cryptotext is decrypted. Thus L''_0 consists of normal El Gamal cryptotexts.

The mix-servers then verify explicitly that all mix-servers followed their program during the first re-encryption and permutation phase. They also verify that there are exactly σ copies of each inner cryptotext, and form the list L_{compact} where the redundant $\sigma - 1$ copies of each cryptotext are removed.

If the verifications proceeds without complaints, each mix-server reveals its private key to the inner cryptosystem. The validity of the revealed private keys are verified against the corresponding public keys. This allows each mix-server to form the list $L_{\mathrm{cleartext}}$ of cleartexts. If the verifications failed we give up and run a slower back-up mix-net protocol.

The idea is that an adversary must alter all copies of each occurring inner cryptotext in the same way. The assumption is that the adversary must guess the position of all copies of any cryptotext that is to be altered, and this can only be done with very low probability. To avoid that the adversary replaces all cryptotexts, dummy cryptotexts are introduced.

We are now ready to describe the details of the protocol.

Sending a Message

To send a message m_i, the sender S_i first computes

$$(u_i, v_i) = E_{(y_{\mathrm{out}}, y_{\mathrm{in}})}(m_i), \quad \text{and} \quad \alpha_i = (E_{(g, y_{\mathrm{out}})}(u_i), E_{(g, y_{\mathrm{out}})}(v_i)) \ .$$

Thus α_i is on the form $[(g^r, {y_{\mathrm{out}}}^{r+s}), (g^t, {y_{\mathrm{out}}}^t({y_{\mathrm{in}}}^s m_i))]$. Then the sender gives a zero-knowledge proof of knowledge of r, $r+s$, and t as described in Section 2.1. The relation between the keys to the outer and inner El Gamal systems simplifies the construction of such a proof. Finally the sender writes α_i to the bulletin board.

The Execution of the Mix-Servers

To improve the following exposition we have interlaced informal descriptions of each step in italics. The informal descriptions do not give a complete description by themselves and are only meant to simplify the exposition.

We assume that each mix-server explicitly verifies that its input in each step is on the correct format. This includes verifying that all lists are of the expected length and contain elements from the expected set of strings, e.g. G_q. If the output of a mix-server is not on the correct format, it is labeled as a cheater and Step 15 is executed. The importance of this is illustrated in Section 4.2 of Chapter 4 and in Chapter 5, where we also discuss ways of avoiding explicit verifications.

Protocol 2.1 (An Efficient Mix-Net).

Preliminaries

1. Dummy Elements. For $j = 1, \ldots, k$ M_j chooses $m_{j,D} \in G_q$ randomly, and simulates an honest sender on input $m_{j,D}$. Let $\{\alpha_j\}_{j=1}^{k}$ be the list of these tuples.

   If the proof of knowledge corresponding to α_l is invalid, then M_l is cheating. Go to Step 15.

   Each mix-server introduces a dummy message in the input to the mix-net.

2. Sending Messages. Each sender S_i sends a message m_i as outlined in Section 2.2 above.

3. Verification of Proofs of Knowledge. Let $\{\alpha_i\}$ be the list of tuples written on the bulletin board by the senders.

   Each M_j verifies for each α_l that the corresponding proof of knowledge is valid.

   To simplify we denote by $L_{\mathrm{unique}} = \{\alpha_i\}_{i=1}^{N_0}$ the list of all α_i with a valid corresponding proof, both the dummy tuples and tuples sent by senders, where all duplicates are removed. Thus if all senders are honest we have $N_0 = N + k$.

   Each mix-server filters the cryptotexts and removes all cryptotexts that do not have a valid corresponding proof of knowledge. It also ensures that there are no duplicates.

4. Expanding the List. Let $L_{\mathrm{expanded}} = \mathrm{Expand}_\sigma(L_{\mathrm{unique}})$. To simplify notation we define $N_1 = \sigma N_0$.

   The mix-servers compute an expanded list L_{expanded} where each cryptotext in L_{unique} is duplicated σ times.

Re-encryption

5. First Re-Encryption. For $j = 1, \ldots, k$, M_j chooses $R_j = \{r_{j,i}\}_{i=1}^{N_1}$ randomly, where $r_{j,i} \in \mathbb{Z}_q^2$, computes

   $$(L_j, \pi_j) \quad = \quad \mathrm{Sort}_4(F_{(g,y_{\mathrm{out}})}(L_{j-1}, R_j)) \ ,$$

   and publishes L_j.

   Each mix-server re-encrypts the list it got from the previous mix and sorts the resulting 4-tuples lexicographically. It stores the permutation corresponding to the sorting.

6. SECOND RE-ENCRYPTION. Let $L'_0 = L_k$. For $j = 1, \ldots, k$, M_j chooses $R'_j = \{r'_{j,i}\}_{i=1}^{N_1}$ randomly, where $r'_{j,i} \in \mathbb{Z}_q^2$, computes

$$(L'_j, \pi'_j) \quad = \quad \mathrm{Sort}_4(F_{(g,y_{\mathrm{out}})}(L'_{j-1}, R'_j)) \enspace ,$$

and publishes L'_j.

Each mix-server re-encrypts the list it got from the previous mix and sorts the resulting 4-tuples lexicographically. It stores the permutation corresponding to the sorting.

Outer Decryption

7. PARTIAL DECRYPTION FACTORS. Each M_j computes

$$\Lambda_j \quad = \quad P_{x_{j,\mathrm{out}}}(L'_k) \enspace ,$$

and commits to Λ_j as explained in Section 2.1. When all mix-server have commited, each M_j opens its commitment.

Each mix-server computes the decryption factors corresponding to its share of the joint key, and commits to its decryption factors. When all mix-servers has commited, each mix-server opens its commitment.

8. JOINT EL GAMAL DECRYPTION. Each M_j computes the component-wise product $\Lambda = \prod_{j=1}^{k} \Lambda_j$ and the list $L''_0 = D_P(\Lambda, L'_k)$.

Each mix-server computes the joint decryption factors and performs the outer decryption.

Verifications of the Outer Decryption and Re-encryptions

9. VERIFICATION OF FIRST RE-ENCRYPTION. Each M_j publishes π_j and R_j. When all mix-servers have published these values, each M_j computes the aggregate re-encryption factors

$$\begin{aligned} R^a \quad &= \quad R_1 + \pi_1^{-1}(R_2 + \pi_2^{-1}(R_3 + \pi_3^{-1}(R_4 + \ldots))) \quad \text{and} \\ (L_k^a, \pi^a) \quad &= \quad \mathrm{Sort}_4(F_{(g,y_{\mathrm{out}})}(L_0, R^a)) \enspace , \end{aligned}$$

and verifies that $L_k^a = L_k$. If equivalence does not hold, somebody cheated during the first re-encryption. Then go to Step 15.

Each mix-server reveals all secret re-encryption exponents and the permutations it used in the first re-encryption. This allows fast explicit verification of the first re-encryption step.

10. VERIFICATION OF DUMMIES IN SECOND RE-ENCRYPTION. Define the indices $\gamma_{0,i} = (\pi^a)^{-1}(i)$ for $i = 1, \ldots, \sigma k$ (i.e. the indices in L'_0 of all copies of dummy

tuples). For $j = 1, \ldots, k$, M_j computes the list $\{\gamma_{j,i}\}_{i=1}^{\sigma k} = \{\pi_j'^{-1}(\gamma_{j-1,i})\}_{i=1}^{\sigma k}$, and publishes $\{\gamma_{j,i}, r'_{j,\gamma_{j-1,i}}\}_{i=1}^{\sigma k}$. We write $L'_j = \{\alpha_{j,i}\}_{i=1}^{N_1}$.

Then each M_j verifies that for $l = 1, \ldots, k$ and $i = 1, \ldots, \sigma k$

$$F_{(g,y_{\mathrm{out}})}(\alpha'_{l-1,\gamma_{l-1,i}}, r'_{l,\gamma_{l-1,i}}) = \alpha'_{l,\gamma_{l,i}} \quad .$$

If equality does not hold, then M_l cheated. Go to Step 15.

The mix-servers trace the dummies through the second re-encryption and explicitly verify each re-encryption step.

11. Verification of Dummies in Outer Decryption. Let $\Lambda_j = \{\lambda_{j,i}\}_{i=1}^{N_1}$. Each M_j gives a zero-knowledge proof that

$$y_j = g^{x_j}, \quad \text{and} \quad \{\lambda_{k,\gamma_{k,i}}\}_{i=1}^{\sigma k} = P_{x_j}(\{\alpha'_{k,\gamma_{k,i}}\}_{i=1}^{\sigma k}) \ ,$$

as outlined in Section 2.1. Then it verifies the proofs of all other mix-servers M_l. If the proof of M_l is not valid it is cheating. Go to Step 15.

The mix-servers prove that they decrypted all dummy tuples correctly.

12. Verification of Tuples. Each M_j first removes the dummies at indices $\gamma_{k,1}, \ldots, \gamma_{k,\sigma k}$ from L''_0. Let L'''_0 be the resulting list. Then L'''_0 should contain $N\sigma$ El Gamal pairs.

Then each mix-server computes $(L_{\mathrm{compact}}, A) = \mathrm{Compact}_{2,\sigma}(L'''_0)$. If $A \neq \emptyset$, somebody is cheating. Go to step 15.

Each mix-server verifies that there are exactly σ copies of each inner cryptotext and removes all dummies and duplicates.

Inner Decryption

13. Inner Decryption. Each M_j publishes $x_{j,\mathrm{in}}$, and verifies that $y_{l,\mathrm{in}} = g^{x_{l,\mathrm{in}}}$ for $l = 1, \ldots, k$. If it does not hold for some l, then M_l cheated. The honest mix-servers then ask the key generator for $x_{l,\mathrm{in}}$.

Finally each M_j computes $x_{\mathrm{in}} = \sum_{l=1}^{k} x_{l,\mathrm{in}}$, and $L_{\mathrm{cleartexts}} = D_{x_{\mathrm{in}}}(L_{\mathrm{compact}})$.

If no cheating occurred, all mixes reveal their private keys of the inner cryptosystem and decrypt the inner cryptotexts.

14. End Of Protocol. When we reach this step, the execution ends and the set of mixed cleartexts is contained in $L_{\mathrm{cleartexts}}$.

Run Back-Up Mix-Net

15. Cheating Occurred. If we reach this step, at least one mix-server is malicious. Each M_j publishes $x_{j,\mathrm{out}}$ and $\{r_{j,i}, r'_{j,i}\}_{i=1}^{N}$.

Each M_j verifies that $y_{l,\text{out}} = g^{x_{l,\text{out}}}$ for all $l \neq j$. If this is not the case, the honest mix-servers ask the key generator for the correct $x_{j,\text{out}}$.

If the cheater is not already identified, this allows anybody to identify the cheater. It also allows explicit reconstruction of x_{out}.

Finally the dummies are removed from L_{unique} to form the list L'_{unique}. The backup mix-net is then invoked on $D_{x_{\text{out}}}(L'_{\text{unique}})$ and the result is denoted $L_{\text{cleartexts}}$.

Each mix-server reveals all its secret inputs. Then the outer layer is stripped off from the set of valid inputs and the result is handed to the backup mix-net.

2.3 Security

Currently we are unable to prove the security of our protocol. We are only able to argue informally that it is secure. To prove the security of our protocol it seems that new tools must be developed.

Robustness

First we consider in which steps the adversary can possibly alter the output of the mix-net without detection. The adversary can not alter the output in the first re-encryption step, since this step is verified explicitly before the inner cryptotexts are decrypted. Neither can it alter the output in any of the verification steps, since these to do not influence the output. Finally, the inner decryption step is explicitly verified by each mix-server individually, so the adversary can not alter the input in this step.

We conclude that if the adversary is able to alter the output, it must do this during the second re-encryption step or during the outer decryption step. Next we argue informally why it is reasonable to expect that the adversary can not alter the output in these steps without detection, and thus that the protocol is robust.

Note that we may now assume that the input to the second re-encryption step is formed correctly, i.e. it is a randomly re-encrypted and permuted copy of the list of cryptotexts with correct proofs of knowledge.

Consider an adversary that, contrary to our belief, is able to alter the output during the execution of the protocol. To avoid detection during the verification of tuples it must ensure that the output of the outer decryption step contains σ copies of each inner cryptotexts. This is easy to accomplish by replacing all cryptotexts. However, to avoid detection during the verification of dummies in the second re-encryption step and the verification of dummies in the outer decryption step, it must be able to reveal the correspondence between the dummy inner cryptotexts in its input and the dummy inner cryptotexts in its output. This implies that the adversary must not only leave the dummy inner cryptotexts unaltered, but must re-encrypt and permute them honestly.

Thus the adversary must somehow transform its input in such a way that no dummy inner cryptotext is altered and such that if a non-dummy inner cryptotext is altered then all copies of it are altered in the same way.

It seems that to do this the adversary must point out the position in its input of the dummies and/or all copies of the cryptotexts it will alter. The results in Chapter 3 implies that the probability of guessing the position of all copies of a specific inner cryptotext correctly is approximately $1/N^{\sigma}$. Since we have chosen σ such that $N^{\sigma-1}$ is negligible, the probability of guessing the position of all copies of any cryptotext is negligible.

On the other hand, it seems that a proof of this conjecture requires a formal assumption on the malleability properties of the El Gamal cryptosystem, since there may exist some other way to transform the cryptotexts. Section 3.4 describes an example of a set of special cleartexts for which there is a transformation that may be computed without finding the positions of individual cryptotexts. However, this is a very special case and no attack along these lines is applicable to our protocol.

In Chapter 6 we take a first step in the program of characterizing the malleability properties of the El Gamal cryptosystem.

Privacy

The privacy of the mix-net is ensured if the adversary can not find any correspondence between any individual cryptotext in the input to the mix-net and any individual cleartext in the output of the mix-net.

Since the randomness used in the first re-encryption step is revealed and the inner cryptotexts are never re-encrypted, the privacy of the mix-net depends solely on the second re-encryption step. Suppose that each mix-server M_j forms its output L_j as some re-encryption and permutation of the output L_{j-1} of the previous mix-server M_{j-1}. Then the results in Chapter 3 and the fact that each sender proves knowledge of its cleartext seem to imply the privacy of the mix-net.

The problem if we try to develop a security proof is to show that the adversary can not form its input in some other way. In Section 4.2 we present an attack for a mix-net in the literature, where the adversary forms its output in a malicious way, but such that the cleartexts are not altered. Our protocol is not vulnerable to this particular type of attack, since the adversary is forced to explicitly reveal how it re-encrypted the dummies. On the other hand it is conceivable that there exists a transformation that avoids this.

2.4 Complexity

Our protocol performs particularly well compared with other mix-nets on a certain range of parameters. To emphasize this and to simplify our estimates we assume that we run the protocol with such parameters. We assume that the number of mix-servers k, the security parameter n, and the number senders N satisfy $k < n$

and $k < \sqrt{N_0}$. We also ignore the cost of additions and sorting, since they are relatively inexpensive.

Each mix-server computes $2 \cdot 4N_1 + 2kN_1$ multiplications. We estimate the cost of one exponentiation to at least the cost of n multiplications and count these multiplications as $2N_1$ exponentiations.

The major part of the computations are performed during the re-encryption, and the decryption phases, but most of this work need not be done in real time. In Table 2.1 we summarize the complexity estimates of the protocol. These figures follows by examination of the protocol.

Phase	Precomputed	Real time	Random bits
Multiplications		$2N_1$	
Verification of POK:s		$6N_0$	
Re-encryptions	$8N_1$		$4N_1 \log_2 q$
Verify 1:st re-encryption		$4N_1$	
Decryptions		$2N_1 + N + k$	
Verify Dummies		$4\sigma k + 2\sigma k^2$	
Total:	$O(8\sigma N)$	$O((10\sigma + 7)N)$	$4N \log_2 q$

Table 2.1: Complexity estimates of Protocol 2.1 in terms of exponentiations.

The total number of exponentiations is $O((18\sigma + 7)N)$ of which $O(8\sigma N)$ can be computed in advance, and $O((10\sigma+7)N)$ must be computed in real time. In practice it is possible to compute a large part of the computations during the verification of the proofs of knowledge of the senders as they appear on the bulletin board. This reduces the number of exponentiations computed in real time by $O(6N_0)$. The exponentiations performed in the verification of the first re-encryption are all relative to a fixed base. This allows further speedups as explained in [45].

An important observation is that essentially all of the computations may be done concurrently by the mix-servers. Thus the total execution time of the protocol given that precomputed values are used, is $O((10\sigma + 7)N)$ exponentiations.

This estimate does not take into account the communication costs, which may be significant when the number of mix-servers is large and there is no ideal bulletin board like a physical broadcast channel.

For all previous constructions the number of exponentiations computed by each mix-server grows at least linearly with k. This is a consequence of the zero-knowledge proof based paradigm; each mix-server must verify the proofs of the other mix-servers.

Practical Parameters and Consequences

In practice σ should be chosen such that $N^{\sigma-1}$ is very small. For $N = 10^5$, $\sigma = 3$ should suffice. Suppose we let $p = 2q + 1$ and q be primes with $\log_2 p \approx 1000$ and let G_q be the subgroup of $\mathbb{Z}_p^*$ of order q. Suppose further that the mix-servers are communicating over a high speed wireless network, i.e. there is a physical broadcast channel. Then our protocol accommodates a relatively large number of mix-servers, e.g. $k = 300$, something that is highly impractical for other constructions.

2.5 Future Work

We have presented an efficient mix-net. We have introduced the notion of double encryption and shown how it combined with the notion of repetitive robustness can be used to make the number of exponentiations essentially independent of the number of mix-servers.

Currently we are not able to prove the security of our construction. We believe that the construction is secure and hope to prove that in the future. Such a proof seems to require new results and/or assumptions on the malleability properties of El Gamal, and perhaps new proof techniques.

Chapter 3

On the Security of Mix-Servers

We introduce a definition of a re-encryption mix-server, and a definition of security for such a mix-server. Then we prove that any semantically secure public key cryptosystem, which allows re-encryption, can be used to construct a secure mix-server.

3.1 Previous Results on Mix-Servers

Jakobsson [40] presents an efficient mix-net protocol. He claims that the protocol is secure and gives a proof sketch of this claim. The work in this section started with an attempt at writing down a formal proof of Jakobssons Lemma 1a [40] given in a slightly simplified form below.

Lemma 1a. (Jakobsson) *If the adversary can, with a non-negligible advantage ϵ over a guess uniformly at random, match any input of a mix-server to its corresponding output, then this adversarial strategy can be used as a black box to break the Decisional Diffie-Hellman assumption with a probability poly(ϵ).*

One problem is that it assumes all message variables identically and independently distributed. This model does not mirror the real world, where it is common that the adversary has some prior knowledge about the distribution of messages sent by a given party, and not all sender's should be approximated by the same distribution. Similarly it is probable that some message variables are dependent. Consider for example elections, where the votes of spouses mostly are dependent. Some problems with arbitrarily distributed messages follow.

Firstly, it is no longer clear how to state the lemma formally, since it is not clear what it should mean to "guess uniformly at random". Since the adversary knows the order of the input elements of the first mix-server he may be able to guess in different ways giving vastly different success probabilities. This is described in full detail when we argue about Definition 3.5.

Secondly Jakobsson assumes that the outcomes of the different copies of the message variables are all different. This allows him to say that the probability of randomly guessing a matching pair is $\frac{1}{N+2}$. This is no longer true if the number of possible messages is small. Additionally, taking this into consideration, it is not possible to argue like Jakobsson does in the argument about the $N+2$:th hybrid. He claims that if we pick new elements from the “message distribution” the $N+2$:th hybrid will have no advantage. Consider a uniformly distributed variable over a set of only two messages. When the lists are very large it is likely that replacing all sent messages by new outcomes of the message variable, does not change the lists much, and one can not conclude that the hybrid has no advantage.

Thirdly, the proof sketch of the security of the complete mix-net of Jakobsson breaks down if we do not assume uniformly and independently distributed message variables, since he applies his lemma also to the first mix-server in the first re-encryption phase. This follows since, in the proof he permutes the input to the adversary $\mathcal{A}$ randomly, and this is not the case in the protocol, where the first mix-server in the first re-encryption phase may have partial knowledge about the distribution of the message variables.

Another problem is that Jakobsson uses Lemma 1a in his proof sketch of his Theorem 1. We discuss this issue in Section 3.4.

Contribution

We provide a definition of security for a single re-encryption mix-server and show in Theorem 3.6 that any semantically secure re-encryption public key system can be used to construct a secure mix-server. We have restricted ourselves to mix-servers based on the random re-encryption paradigm.

We do *not* claim to give a definition of the privacy of a mix-net, since a definition of security of a complete mix-net must involve several other aspects. We highlight this in Section 3.4, where we explain two phenomena related to our theorem that occur naturally in the construction of a mix-net. One of these phenomena illustrates a misuse of Theorem 3.6 in the literature that to our knowledge was undetected until now.

3.2 A Variant Definition of GM-Security

The following is a variant definition of GM-security that can be proven equivalent to Definition 1.2 given in Section 1.4. This may be considered folklore, but we provide a proof for completeness.

Definition 3.1 (GM-security*). Let $(E, D) = \{(E_n, D_n)\} = \{C(1^n)\}$, where C is a public key cryptosystem, and let b be uniformly and independently distributed in $\{0, 1\}$. C is *GM-secure** if $\forall m_0, m_1 \in \mathcal{M}$, $\forall T \in \mathrm{PC}$ and $\forall c > 0$, $\exists n_0$ such that

$\forall n > n_0$:

$$\left|\Pr[T(E, m_0, m_1, E(m_b), E(m_{1-b})) = b] - \frac{1}{2}\right| < \frac{1}{n^c} \ .$$

The following lemma is "folklore" knowledge, but for completeness we give a proof.

Lemma 3.2. *Definition 3.1 is equivalent to Definition 1.2.*

Proof. Suppose that a PKC C is not secure according to Definition 1.2, and let $T = \{T_n\}$ be the family of circuits that shows this. Then $\exists m_0, m_1 \in \mathcal{M}$, and an infinite index set $\mathcal{N}$, such that for each $n \in \mathcal{N}$ we have

$$\left|\Pr[T(E, m_0, m_1, E(m_b)) = m_b] - \frac{1}{2}\right| \geq \frac{1}{n^c}$$

Then the family of circuits $T' = \{T'_n\}$, where T'_n on input (E, m_0, m_1, c_0, c_1) gives as output the output of $T_n(E, m_0, m_1, c_0)$ clearly contradicts Definition 3.1.

For the other direction, suppose that a PKC C is not secure according to Definition 3.1. Then $\exists m_0, m_1 \in \mathcal{M}$, $\exists T' \in \mathrm{PC}$, and an infinite index set $\mathcal{N}$, such that for each $n \in \mathcal{N}$, if we define

$$p_{bd} \quad = \quad \Pr[T'(E, m_0, m_1, E(m_b), E(m_d)) = 1]$$

then

$$\frac{1}{n^c} \quad \leq \quad |p_{01} - p_{10}| = |p_{01} - p_{11} + p_{11} - p_{10}| \leq 2|p_{t,1-t} - p_{11}|$$

for some $t = \{t_n\}$, where $t_n \in \{0, 1\}$. Set $\gamma_t = \alpha$ and $\gamma_{1-t} = E(m_1)$. Then T runs $b = T'(E, m_0, m_1, \gamma_0, \gamma_1)$ and returns m_b. It follows that T breaks the security according to Definition 1.2.

3.3 The Security of a Mix-Server

To be able to formally prove anything about a mix-server we first define the concept of a mix-server and the right notion of security.

Definitions

The following definition captures that cryptotexts can be re-encrypted without knowledge of the private key. This property is closely related to the homomorphic property used in many papers (e.g. [36]). The by now classical El Gamal cryptosystem [19], and the recently discovered Paillier cryptosystem [53] are examples of systems that fit this definition.

Definition 3.3 (Re-Encryption PKC). A *Re-Encryption Public Key Cryptosystem (RPKC)* is a public key cryptosystem C that on input 1^n in addition to descriptions of E_n and D_n also outputs the description of a circuit F_n of polynomial size in n such that:

1. F_n has $\kappa(n)$ inputs and $\kappa(n)$ outputs.
2. For all $m \in \mathcal{M}$ and all $\alpha, \alpha' \in E(m, \mathcal{R})$ we have:
   $\Pr[\alpha' = F(\alpha)] = \Pr[\alpha' = E(m)]$.

The function F above is called the "re-encryption function". Without loss of generality we can assume that E_n, D_n and F_n use an equal number of random bits, i.e. we assume that all of the circuits use the same polynomial number of bits in n.

Formally Definition 1.2 and 3.1 are not applicable to an RPKC. The reason is that A and T in Definition 1.2 and 3.1 respectively are given only E and not F as input. To see that this is an important detail, consider an RPKC C such that if we ignore F in the output it is GM-secure. Clearly C can encode the description of D into the description of F, which makes C easy to break using the knowledge of F. It is however trivial to extend the definitions to be applicable also to an RPKC such that the equivalence of the definitions still holds. Thus we use the definitions as if they were defined properly for the case at hand, i.e. A and T in Definition 1.2 and 3.1 respectively take as additional input F from the output (E, D, F) of C.

Definition 3.4 (Re-Encryption Mix-Server). A *Re-Encryption Mix-Server (RMS)* is a probabilistic Turing machine C_H running in expected polynomial time that on input $(1^n, N)$, N is a polynomial $N(n)$ in n, outputs descriptions of probabilistic circuits E_n, D_n, F_n, and H_n of polynomial size in n such that:

1. The probabilistic Turing machine $K_N(C_H)$ that, given input 1^n, simulates C_H on input $(1^n, N)$ and outputs descriptions of E_n, D_n, F_n is an RPKC.
2. H_n has $\kappa(n) \times N$ inputs and $\kappa(n) \times N$ outputs.
3. $H(\alpha) = \Pi_N F(\alpha)$, where Π_N is uniformly distributed in Σ_N.

We use the notation $\pi_N F(\alpha, r) = H(\alpha)$ when we want to make explicit H's probabilistic input, i.e. π_N and r.

Note that the above is a definition of a *re-encryption* mix-server. In Chaum's [8] original construction each mix-server performed a partial decryption, and not a re-encryption. In Chaum's construction the number of input bits is not equal to the number of output bits. Also one could imagine that a mix-server received input encrypted with one cryptosystem, and produced output using another cryptosystem.

A Definition of a Secure RMS

We now introduce a notion of security for an RMS. Define a predicate ρ with regard to a given RMS taking as input a pair of lists and a pair of indices. Let $l = E(m, r)$

and $l' = \pi_N F(l, r')$, where $m = (m_1, \dots, m_N) \in \mathcal{M}^N$, $r, r' \in \mathcal{R}^N$, and $\pi_N \in \Sigma_N$. Let (i, j) be a pair of indices $1 \leq i, j \leq N$. We let $\rho(l, l', i, j) = T$ if and only if it holds that $m_i = m_{\pi_N^{-1}(j)}$. The predicate is true if the encryption at index i in l and the encryption at index j in l' both encrypt the same message. It is clearly possible that there exist several pairs $(i, j_1), (i, j_2), ..., (i, j_k)$ for which $\rho(l, l', i, j_t) = T$.

The following definition says that given a secure RMS it is impossible to find a pair of indices (i, j) such that $\rho(l, l', i, j)$ holds with respect to the input l and output l' of the RMS notably better than guessing cleverly.

Definition 3.5 (Security of an RMS). Let C_H be an RMS, define the family $(E, D, F, H) = \{(E_n, D_n, F_n, H_n)\} = \{C_H(1^n)\}$, and let $A \in \mathrm{PC}$.

Let M be arbitrarily but independently distributed over $\mathcal{M}^N$, and let $J = \{J_n\}$, where J_n is uniformly and independently distributed over $\{1, \dots, N(n)\}$. Define the random variables:

$$L = E(M), \quad L' = H(L), \text{ and } \quad (I_A, J_A) = A(E, F, L, L') \ .$$

C_H is *secure* if for all M and A as above $\forall c > 0$, $\exists n_0$ such that $\forall n > n_0$:

$$|\Pr[\rho(L, L', I_A, J_A) = T] - \Pr[\rho(L, L', I_A, J) = T]| < \frac{1}{n^c} \ .$$

We argue that this is the right definition of a secure RMS as follows. Suppose that the underlying cryptosystem $K_N(C_H)$ is in some magical way perfect. That is, a cryptotext gives no information in an information theoretical sense about the encrypted message. Then an adversary clearly can not pick the second component of its output better then picking a uniformly chosen index, since the permutation Π_N, unknown to the adversary, is uniformly and independently distributed.

On the other hand the first component can still be chosen cleverly to bias the success probability. Consider for example the case where all M_i are constant, and all but one equals m_i. Then the success probability depends heavily on how the first component is chosen.

The definition states that given an adversary A that has a certain success probability, we get almost the identical success probability by using the first component of A's output and picking the second component randomly. Since we pick the second index randomly this amounts to clever guessing.

Results on the Security for an RMS

Let $K_N(C_H)$ denote the probabilistic Turing machine that given input 1^n simulates C_H on input $(1^n, N)$ to get (E_n, D_n, F_n, H_n) and outputs (E_n, D_n, F_n). We are able to prove the following theorem of which Jakobssons Lemma 1a [40] could be said to be a special case.

Theorem 3.6. *C_H is a secure RMS if and only if for all polynomials $N(n)$ in n, $K_N(C_H)$ is a semantically secure RPKC.*

The theorem implies that if there exists a semantically secure RPKC, then the construction given in Definition 3.4 gives a secure mix-server according to Definition 3.5. We implicitly use the generalization of Definition 1.2 and 3.1 to re-encryption public key cryptosystems, as discussed in Section 3.3.

Note that the presence of the quantification over the variable N in Theorem 3.6 is necessary. Without it there could exist some N for which $K_N(C_H)$ outputs trivial (E, D, F). We also need that N is polynomial in n since we otherwise would be unable to perform a hybrid argument in the proof.

Before we prove Theorem 3.6 we prove some lemmas. Denote by $\pi^{(0)}$ the identity permutation and $\pi^{(1)} = \pi$ for any permutation π. Consider the following generalization of the GM-security*, i.e. Definition 3.1. Throughout this section we assume that $(E, D) = \{(E_n, D_n)\} = \{C(1^n)\}$, when we write E or D.

Definition 3.7 (Generalized GM-security* (gGM)). Let C be a public key cryptosystem, let $(E, D) = \{(E_n, D_n)\} = \{C(1^n)\}$, and let b be uniformly distributed in $\{0, 1\}$. C is *gGM-secure* if for all polynomials N in n, $\forall \pi_N \in \Sigma_N$, $\forall m \in \mathcal{M}^N$, $\forall T \in \mathrm{PC}$, $\forall c > 0$, $\exists n_0$ such that $\forall n > n_0$:

$$\left| \Pr[T(E, m, \pi_N^b E(m)) = b] - \frac{1}{2} \right| < \frac{1}{n^c} \enspace .$$

Lemma 3.8. *A public key cryptosystem C is GM-secure iff it is gGM-secure.*

Proof. We see that gGM-security immediately implies GM-security*, since we may take the polynomial $N(n) = 2$ in the definition of gGM-security.

To prove the opposite direction of the lemma, we assume it is false. Then there exists a GM-secure cryptosystem C, and a polynomial N, $\exists m \in \mathcal{M}$, $\exists T \in \mathrm{PC}$, $\exists \pi_N \in \Sigma_N$, $\exists c > 0$, and an infinite set $\mathcal{N}$ such that for $n \in \mathcal{N}$:

$$\left| \Pr[T_n(E_n, m_n, \pi_{N,n}^b E_n(m_n)) = b] - \frac{1}{2} \right| \geq \frac{1}{n^c} \enspace .$$

We now define an $A = \{A_n\} \in \mathrm{PC}$ that breaks the GM-security* of C. Consider a fixed $n \in \mathcal{N}$. Note that for any permutation, in particular for $\pi_{N,n}$, there exists a chain of permutations $\mathrm{id} = \pi^{(1)}, \pi^{(2)}, \ldots, \pi^{(N)} = \pi_{N,n}$, such that $\pi^{(i+1)}$ and $\pi^{(i)}$ differ only by a transposition. We get the following hybrid argument:

$$\tau_i = \Pr[T_n(E_n, m_n, (\pi^{(i)})^b E_n(m_n)) = b], \quad \frac{1}{n^c} \leq |\tau_N - \tau_1| \leq \sum_{i=1}^{N-1} |\tau_{i+1} - \tau_i| \enspace .$$

where $\tau_1 = \frac{1}{2}$ since $(\pi^{(1)})^b = \mathrm{id}$. This implies $|\tau_{t+1} - \tau_t| \geq \frac{1}{Nn^c}$ for some $1 \leq t < N$. Let k_0 and k_1 be the two indices such that $\pi^{(t)}(k_0) = \pi^{(t+1)}(k_1)$ and $\pi^{(t+1)}(k_0) = \pi^{(t)}(k_1)$. Let $(E_n, m_{n,k_0}, m_{n,k_1}, \alpha_0, \alpha_1)$ be the input to A_n, where b is randomly chosen, and $(\alpha_0, \alpha_1) = (E_n(m_{n,k_b}), E_n(m_{n,k_{1-b}}))$. The circuit A_n:

1. Computes $\alpha = \pi^{(t)} E_n(m_n)$.

2. Replaces the elements of α at positions $\pi^{(t)}(k_0)$ and $\pi^{(t)}(k_1)$ by the elements α_0 and α_1 respectively. Let the resulting vector be α'.

3. Runs $b = T_n(E_n, m_n, \alpha')$, and returns b.

It follows that the GM-security* of C is broken.

Corollary 3.9. *If C is gGM-secure then, $\forall j = \{j_n\}$, where $j_n \in \{1, \ldots, N(n)\}$, $\forall \pi_N, \psi_N \in \Sigma_N$, $\forall m \in \mathcal{M}^N$, $\forall T \in \mathrm{PC}$, $\forall c > 0$, $\exists n_0$ such that $\forall n > n_0$:*

$$|\Pr[T(E, m, \pi_N E(m)) = j] - \Pr[T(E, m, \psi_N E(m)) = j]| < \frac{1}{n^c} \ .$$

Proof. Assume the contrary. Then there exists j, π_N, ψ_N, and $c > 0$ such that for $n \in \mathcal{N}$ the inequality above does not hold. Consider a fixed $n \in \mathcal{N}$. If we set $\tau_{\pi_N} = \Pr[T_n(E_n, m_n, \pi_{N,n} E_n(m_n)) = j_n]$, we have $|\tau_{\pi_N} - \tau_{\psi_N}| \geq \frac{1}{n^c}$. Without loss we assume that $\tau_{\pi_N} < \tau_{\psi_N}$

We now construct a $B = \{B_n\} \in \mathrm{PC}$ that breaks the gGM-security of C. B_n takes input (E_n, m_n, α), where $\alpha = (\pi_{N,n}^{-1} \psi_{N,n})^b E_n(m_n)$, and $b \in \{0, 1\}$ is randomly chosen.

The circuit B_n does the following. It computes $\alpha' = \pi_{N,n} F_n(\alpha)$, and runs $T_n(E_n, m_n, \alpha')$. If it returns j_n then B_n outputs 1. Otherwise it outputs a random bit. If we set $A_n(b) = B_n(E_n, m_n, (\pi_{N,n}^{-1} \psi_{N,n})^b E_n(m_n))$ then we have

$$\left|\Pr[A_n(b) = b] - \frac{1}{2}\right| = \frac{1}{2} \left|\Pr[A_n(0) = 1] - \Pr[A_n(1) = 1]\right|$$

$$= \frac{1}{2} \left| \tau_{\pi_N} + \sum_{l \neq j_n} \frac{1}{2} - \tau_{\psi_N} - \sum_{l \neq j_n} \frac{1}{2} \right| = \frac{1}{2} |\tau_{\pi_N} - \tau_{\psi_N}| \geq \frac{2}{n^c} \ ,$$

which is a contradiction.

Lemma 3.10. *For $m_n \in \mathcal{M}_n^N$, denote by $\Delta_i(m_n)$ the set $\{j | m_{n,j} = m_{n,i}\}$, and let Π_N be uniformly distributed in Σ_N. If C is GM-secure then $\forall i = \{i_n\}$, where $i_n \in \{1, \ldots, N(n)\}$, $\forall m \in \mathcal{M}^N$, $\forall T \in \mathrm{PC}$, $\forall c > 0$, $\exists n_0$ such that $\forall n > n_0$:*

$$\left| \Pr[T(E, m, \Pi_N E(m)) \in \Pi_N(\Delta_i(m))] - \frac{|\Delta_i(m)|}{N} \right| < \frac{1}{n^c} \ .$$

Proof. Intuitively the following proof is clear. Formally we proceed as follows. Let $\epsilon_{\pi_N, j} = \Pr[J_T = j | \Pi_N = \pi_N] - \Pr[J_T = j | \Pi_N = \mathrm{id}]$, where we let $J_T =$

$T(E, m, \Pi_N E(m))$. We write Δ_i for $\Delta_i(m)$, and have:

$$\Pr[J_T \in \Pi_N(\Delta_i)] = \sum_{\pi_N \in \Sigma_N} \frac{1}{N!} \Pr[J_T \in \Pi_N(\Delta_i) | \Pi_N = \pi_N]$$

$$= \sum_{j=1}^{N} \sum_{\pi_N \in \Sigma_N} \frac{1}{N!} \Pr[J_T = j | \Pi_N = \pi_N] \Pr[J_T \in \Pi_N(\Delta_i) | \Pi_N = \pi_N, J_T = j]$$

$$= \sum_{j=1}^{N} \Pr[J_T = j | \Pi_N = \mathrm{id}] \sum_{\pi_N \in \Sigma_N} \frac{1}{N!} \Pr[J_T \in \Pi_N(\Delta_i) | \Pi_N = \pi_N, J_T = j]$$

$$+ \sum_{j=1}^{N} \sum_{\pi_N \in \Sigma_N} \frac{1}{N!} \epsilon_{\pi_N, j} \Pr[J_T \in \Pi_N(\Delta_i) | \Pi_N = \pi_N, J_T = j]$$

$$= \frac{|\Delta_i|}{N} + \sum_{j=1}^{N} \sum_{\pi_N \in \Sigma_N} \frac{1}{N!} \epsilon_{\pi_N, j} \Pr[J_T \in \Pi_N(\Delta_i) | \Pi_N = \pi_N, J_T = j]$$

since $\sum_{\pi_N \in \Sigma_N} \frac{1}{N!} \Pr[J_T \in \Pi_N(\Delta_i) | \Pi_N = \pi_N, J_T = j] = \frac{|\Delta_i|}{N}$. Thus we have:

$$\left| \Pr[J_T \in \Pi_N(\Delta_i)] - \frac{|\Delta_i|}{N} \right| \leq \sum_{j=1}^{N} \sum_{\pi_N \in \Sigma_N} \frac{1}{N!} |\epsilon_{\pi_N, j}| \leq N \max_{\pi_N, j} \{ |\epsilon_{\pi_N, j}| \}$$

which is negligible since $N(n)$ is polynomial and $\max_{\pi_N, j}\{|\epsilon_{\pi_N, j}|\}$ by Corollary 3.9 is negligible.

We are now ready to give the proof of Theorem 3.6.

Proof of Theorem 3.6. First the easy direction of the proof. Suppose that C_H is a secure RMS, but $C = K_N(C_H)$ is not a GM-secure RPKC for some polynomial N. Then $\exists m_0, m_1 \in \mathcal{M}$, $\exists T \in \mathrm{PC}$, $\exists c > 0$ and an infinite index set $\mathcal{N}$ such that for $n \in \mathcal{N}$:

$$\left| \Pr[T(E, m_0, m_1, E(m_b), E(m_{1-b})) = b] - \frac{1}{2} \right| \geq \frac{1}{n^c} \ .$$

The family $A = \{A_n\}$, where A_n given input $(E_n, F_n, E_n(m_{n,0}, m_{n,1}), (\alpha_0, \alpha_1))$ returns the pair $(0, T_n(E_n, m_{n,0}, m_{n,1}, \alpha_0, \alpha_1))$ shows that C_H is not secure.

To prove the other direction, we assume that $K_N(C_H)$ is semantically secure for all polynomials N, but C_H is not secure. Then, using the notation of Definition 3.5, there exists an $A \in \mathrm{PC}$, an infinite index set $\mathcal{N}$, and a $c > 0$ such that for $n \in \mathcal{N}$:

$$|\Pr[\rho(L_n, L'_n, I_{A_n}, J_{A_n}) = T] - \Pr[\rho(L_n, L'_n, I_{A_n}, J_n) = T]| \geq \frac{1}{n^c} \ .$$

We abuse notation and write $\rho(I, J)$ instead of the correct $\rho(L_n, L'_n, I, J)$. A probabilistic argument gives that there exists a fixed $m \in \mathcal{M}^N$ such that for $n \in \mathcal{N}$: $|\Pr[\rho(I_{A_n}, J_{A_n}) = T | M_n = m_n] - \Pr[\rho(I_{A_n}, J_n) = T | M_n = m_n]| \geq \frac{1}{n^c}$.

We define: $\tau_{A,i} = \Pr[\rho(I_{A_n}, J_{A_n}) = T | M_n = m_n, I_{A_n} = i]$ and similarly $\tau_i = \Pr[\rho(I_{A_n}, J_n) = T | M_n = m_n, I_{A_n} = i]$ to simplify notation in the following.

For some $1 \leq t \leq N(n)$ we have:

$$\begin{aligned} \frac{1}{n^c} &\leq |\Pr[\rho(I_{A_n}, J_{A_n}) = T | M_n = m_n] - \Pr[\rho(I_{A_n}, J_n) = T | M_n = m_n]| \\ &= \left| \sum_{i=1}^{N(n)} p_{I_{A_n}}(i)(\tau_{A,i} - \tau_i) \right| \leq N(n) p_{I_{A_n}}(t) |\tau_{A,t} - \tau_t| \ . \end{aligned}$$

We construct a $T \in \mathrm{PC}$ that contradicts Lemma 3.10. The circuit T_n gets input (E_n, F_n, m_n, l'_n), where l'_n is an outcome of L'_n, computes $l_n = E(m_n)$, runs $(i, j) = A_n(E, F, l_n, l'_n)$, and if $i = t$ it returns j, and otherwise it returns the outcome of a random variable J_n, which is uniformly and independently distributed over $\{1, \ldots, N(n)\}$.

Using the notation of Lemma 3.10 we have $\tau_t = \frac{|\Delta_t|}{N}$ which gives:

$$\begin{aligned} \Pr[J_{T_n} \in \Pi_{N(n)}(\Delta_t)] &= \sum_{i \neq t} p_{I_{A_n}}(i) \frac{|\Delta_t|}{N} + p_{I_{A_n}}(t) \tau_{A,t} \\ &= \frac{|\Delta_t|}{N} + p_{I_{A_n}}(t)(\tau_{A,t} - \tau_t) \ . \end{aligned}$$

Thus $|\Pr[J_{T_n} \in \Pi_{N(n)}(\Delta_t)] - \frac{|\Delta_t|}{N}| = p_{I_{A_n}}(t) |\tau_{A,t} - \tau_t| \geq \frac{1}{N(n) n^c}$, which contradicts Lemma 3.10.

In the proof above we implicitly use an extended version of Definition 3.7 that is applicable to an RPKC, and use that Lemma 3.8, Corollary 3.9, and Lemma 3.10 hold correspondingly (see Section 3.3).

3.4 The Definition is Not Sufficient for a Mix-Net

Our results give strong evidence for the security of many constructions of mix-nets in the literature. However they do *not* imply that the mix-nets proposed in the literature are secure, since there is not even a formal definition of security of a mix-net. Neither is Definition 3.5 intended to serve as a definition of security of a mix-net.

To emphasize this fact we give a generalization of an attack on mix-nets of which special cases has been described by Pfitzmann [57], and Jakobsson [38]. Jakobsson also gives a solution on how to prevent this attack. We also give an example of a situation, where our results seem to be applicable but are not.

Using Malleability to Break Anonymity.

The notion of non-malleability was introduced by Dolev, Dwork and Naor [18]. Informally a cryptosystem is non-malleable if, given a cryptotext $\alpha_i = E(m_i)$ of a

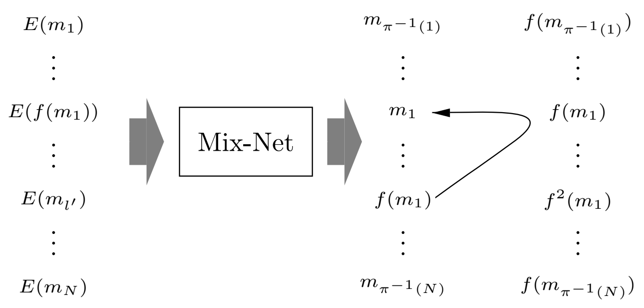

Figure 3.1: The figure illustrates the generic relation attack. The adversary forms an encryption $E(f(m_1))$ of a cleartext $f(m_1)$ related to the cleartext m_1 of the first sender. Then it instructs a corrupted sender to send this cryptotext. Finally it identifies the cleartext m_1 by finding a pair of identical cleartexts in the output and the output transformed with the function f.

message m_i, it is impossible to construct $\alpha'_i = E(m'_i)$, where m'_i has some non-trivial relation to m_i.

Suppose that we have a mix-net that viewed as a single mix-server is secure by Definition 3.5, and that the cryptotexts given as input to the mix-net are encrypted using a malleable cryptosystem. Let $\alpha_1 = E(m_1)$ be the cryptotext of a message m_1 sent to the mix-net by Alice. The adversary, wishes to break the privacy of Alice. To do this it constructs $\alpha'_i = E(f(m_i))$, for some function f, by using the malleability of the cryptosystem. Then it sends $\alpha_j = \alpha'_i$ to the mix-net.

Thus the input to the mix-net can be written as $\alpha_1, \ldots, \alpha_N$, where α_1 is the cryptotext of Alice and $\alpha_j = \alpha'_i$ is the contrived cryptotext. The output of the mix-net has the form $(m'_1, \ldots, m'_N) = (m_{\pi^{-1}(1)}, \ldots, m_1, \ldots, f(m_1), \ldots, m_{\pi^{-1}(N)})$. If we apply the transformation f on each of these elements we get a list on the form $(m''_1, \ldots, m''_N) = (f(m_{\pi^{-1}(1)}), \ldots, f(m_1), \ldots, f^2(m_1), \ldots, f(m_{\pi^{-1}(N)}))$. Note that that there is an i and an l such that $m''_l = m'_i$. If f is chosen from an appropriate family it is likely that the pair (i, l) is unique. This allows the adversary to identify m_l as the message of Alice. The attack is illustrated in Figure 3.1.

Depending on the transformation f the probability of getting an ambiguous answer is higher or lower, and several attackers using "independent" relations may increase the probability of a correct guess.

Jakobssons [38] transformation is the identity, and Pfitzmann [57] assumes an El Gamal cryptosystem where she uses the transformation $f(m) = m^{x_f}$ for some random x_f. The attack clearly fails if we use a non-malleable cryptosystem and check for identical cryptotexts, and this is what Jakobsson proposes.

The conclusion is that Definition 3.5 is inappropriate to define the privacy of a complete mix-net. A definition of privacy of mix-nets must allow active attacks like the above, and must be defined in a multi-party setting.

Using Malleability to Break Robustness.

One method for producing an efficient and robust mix-net protocols is repetition. We apply this method for our mix-net construction in Chapter 2. Below we discuss why this method must be applied carefully.

Consider the following thought experiment, where we let the underlying cryptosystem be the El Gamal system.

Let $m = (m_1, \ldots, m_N)$ be an array of cleartexts, let $\alpha = E(m)$ be the corresponding array of cryptotexts, and let H be the output of a secure RMS. Let α' be the concatenation of σ copies of the list α, and set $\alpha'' = H(\alpha')$.

Note that α'' contains a multiple of σ different cryptotexts of each m_i. Suppose we are given α'' and the goal is to replace all cryptotexts of any single arbitrarily chosen message m_i, with encryptions of some other message m'_i, but let the remaining set of encrypted messages be fixed. That is we must, given α'' construct an α''' such that it contains a multiple of σ cryptotexts of each m_i except one m_j for which we have replaced all its cryptotexts by encryptions of some $m'_j \neq m_j$. Is this problem feasible to solve?

At first it seems that if all m_i are different, then since the RMS is secure the probability should be something like the probability of guessing the position of all σ copies of cryptotexts of m_i. Indeed an argument similar to this is used by Jakobsson [40] in the proof sketch of his Theorem 1.

Unfortunately this is not true in general, not even for uniformly (dependently) distributed messages, as the following example shows. Suppose that the cryptosystem in use is the El Gamal system [19] over a group G_q as described in Section 1.4. Let m_1 be uniformly and independently distributed in G_q, let $k_1 \in G_q$, and $k_2 \in \mathbb{Z}_q$ be fixed and set $m_i = k_1 m_{i-1}^{k_2}$ for all $i \neq 1$. Given an El Gamal cryptotext $\alpha_i = E(m_i)$ of m_i it is easy to compute $f(\alpha_i) = E(k_1 m_i^{k_2})$ without knowledge of the private key or the cleartext messages. Thus to succeed in the thought experiment we need only compute the list $f(\alpha)$, where we let f be defined element-wise. This maps cryptotexts of m_{i-1} onto cryptotexts of m_i except for m_N, which is mapped to an element $m'_1 \neq m_i$ for all i. Thus we have in effect replaced cryptotexts of m_1 with cryptotexts of m'_1 *without* identifying what cryptotexts to change.

However, the distribution of the cleartexts is very special. Indeed if all messages are independently distributed it seems infeasible to solve the thought experiment better than guessing. The protocol of Chapter 2 is based on a problem related to the thought experiment above. One important difference is that inner cryptotexts play the role of the messages that must be replaced. All honest senders have unique and independently chosen inner cryptotexts. Furthermore we force the adversary to behave honestly on at least the dummies, i.e. it does not suffice for the adversarial transformation to keep the dummies in the set of cleartexts. It must re-encrypt them honestly. In Chapter 6 we investigate another related problem in what ways a single El Gamal cryptotext may be transformed.

Chapter 4

Some Practical attacks for Mix-Nets

Golle, Zhong, Boneh, Jakobsson, and Juels [34] propose an efficient mix-net, which they claim to be both robust and secure. We present five practical attacks for their mix-net, and break both its privacy and robustness.

The first attack breaks the privacy of any given sender *without corrupting any mix-server*. The second attack requires that the first mix-server is corrupted. Both attacks are adaptations of the "relation attack" introduced by Pfitzmann [58, 57].

The third attack is similar to the attack of Desmedt and Kurosawa [15] and breaks the privacy of *all* senders. It requires that all senders are honest and that the last mix-server is corrupted.

The fourth attack may be viewed as a novel combination of the ideas of Lim and Lee [44] and Pfitzmann [58, 57]. Johan Håstad helped me find this attack. The attack breaks the privacy of any given sender, and requires that the first and last mix-servers are corrupted. This attack breaks also Jakobsson [40], including the fixed version of Mitomo and Kurosawa [49].

The fifth attack breaks the robustness in a novel way. It requires corruption of some senders and the first mix-server. This attack breaks also the hybrid mix-net of Jakobsson and Juels [42].

Abe and Imai [2] independently discovered attacks similar to the first two of our attacks. They also give an attack for the protocol of Jakobsson and Juels [42] that is unrelated to our attack for that construction.

4.1 Review of "Optimistic Mixing for Exit-Polls"

We present a short review of the relevant parts of the protocol of Golle et al. [34]. The description given here is as close as possible to the original, but we avoid details irrelevant to our attacks and change some notation to simplify the exposition of the attacks. For details we refer the reader to [34].

Parties and Setup

The protocol assumes the existence of a bulletin board as outlined in Section 1.4.

The parties of the protocol are N *senders*, and a relatively small number of *mix-servers*, $M_1, \ldots, M_k$. Each sender encrypts its message, and writes it on the bulletin board. The mix-servers then execute the mix-net protocol.

The protocol employs an El Gamal [19] cryptosystem in a subgroup G_q of prime order q of the multiplicative group modulo a prime p, i.e. $\mathbb{Z}_p^*$. The El Gamal cryptosystem is described in Section 1.4.

In the setup stage each mix-server M_j is somehow given a random $x_j \in \mathbb{Z}_q$, and $y_l = g^{x_l}$ for $l \neq j$. The value x_j is also shared with the other mix-servers using a threshold verifiable secret sharing scheme. Golle et al. [34] discuss different variants for sharing keys, but we choose to present a simple variant, since it has no impact on our attacks. If any mix-server M_j is deemed to be cheating the other mix-servers can verifiably reconstruct its private key x_j. The mix-servers can also compute $y = \prod_{j=1}^{k} y_j$, which gives a joint public key (g, y), with secret corresponding private key $x = \sum_{j=1}^{k} x_j$.

A practical advantage of the mix-net is that it can be used to execute several mix-sessions using the same set of keys, i.e. the El Gamal keys are not changed between mix-sessions. To be able to do this the proofs of knowledge below are bound to a mix-session identifier id that is unique to the current mix-session.

Sending a Message to the Mix-Net

A typical honest sender, Alice, computes the following to send a message m to the mix-net:

$$\begin{aligned} (u, v) = E_{(g,y)}(m), \qquad & w = h(u, v), \text{ and} \\ \alpha = [E_{(g,y)}(u), E_{(g,y)}(v), E_{(g,y)}(w)] \quad = \quad & [(\mu_1, \mu_2), (\nu_1, \nu_2), (\omega_1, \omega_2)] \ , \end{aligned}$$

where $h : \{0,1\}^* \to G_q$ is a hash function modeled by a random oracle. Note that this is similar to our protocol described in Chapter 2, but Golle et al. introduce a hash value and use identical keys for the inner and outer cryptosystems.

Then Alice computes a zero-knowledge proof of knowledge $\pi_{\mathrm{id}}(u, v, w)$, in the random oracle model of u, v and w, which depends on the current mix-session identifier id as outlined in Section 2.1. Finally Alice writes $(\alpha, \pi_{\mathrm{id}}(u, v, w))$ on the bulletin board. We reserve the notation above for the tuple of Alice and use it in the attacks below.

Execution of the Mix-Net

First the mix-servers remove any duplicate inputs to the mix-net, and discard input tuples that contain components not in the subgroup G_q. The mix-servers then discard all input tuples where the proof of knowledge is not valid for the current mix-session. Let $L_0 = \{[(a_{0,i}, b_{0,i}), (c_{0,i}, d_{0,i}), (e_{0,i}, f_{0,i})]\}_{i=1}^N$ be the resulting list of triples of El Gamal pairs. The mixing then proceeds in the following stages.

First Stage: Re-Randomization and Mixing

This step proceeds as in all re-randomization mix-nets based on El Gamal. One by one, the mix-servers $M_1, \ldots, M_k$ randomize all the inputs and their order. (Note that the components of triples are not separated from each other during the re-randomization.) In addition, each mix-net must give a proof that the product of the plaintexts of all its inputs equals the product of the plaintexts of all its outputs. The protocol proceeds as follows.

1. Each mix-server M_j reads from the bulletin board the list L_{j-1} output by the previous mix-server.
2. The mix-server then chooses $r_{ji}, s_{ji}, t_{ji} \in \mathbb{Z}_q$, for $i = 1, \ldots, N$, randomly and computes the re-randomized list

   $$\{[(g^{r_{ji}} a_{j-1,i}, y^{r_{ji}} b_{j-1,i}), (g^{s_{ji}} c_{j-1,i}, y^{s_{ji}} d_{j-1,i}), (g^{t_{ji}} e_{j-1,i}, y^{t_{ji}} f_{j-1,i})]\}_{i=1}^N$$

   of triples. The above list of triples is then randomly permuted, and the resulting list: $L_j = \{[(a_{j,i}, b_{j,i}), (c_{j,i}, d_{j,i}), (e_{j,i}, f_{j,i})]\}_{i=1}^N$ is written on the bulletin board.
3. Define $a_j = \prod_{i=1}^N a_{j,i}$, and define b_j, c_j, d_j, e_j, and f_j correspondingly. The mix-server proves in zero-knowledge that $\log_g a_j/a_{j-1} = \log_y b_j/b_{j-1}$, $\log_g c_j/c_{j-1} = \log_y d_j/d_{j-1}$, and $\log_g e_j/e_{j-1} = \log_y f_j/f_{j-1}$. This implies that $D_x(a_j, b_j) = D_x(a_{j-1}, b_{j-1})$, and similarly for the pairs (c_j, d_j) and (e_j, f_j), i.e. the component-wise product of the inner triples remains unchanged by the mix-server.

Remark 4.1. Since $\log_y b_j/b_{j-1} = \log_g a_j/a_{j-1} = \sum_{i=1}^N r_{ji}$, and M_j knows the latter sum, the proof in Step 3) can be implemented by a zero-knowledge proof of knowledge in the random oracle model as outlined in Section 2.1, and similarly for the pairs (c_j, d_j), and (e_j, f_j).

Second Stage: Decryption of the Inputs

1. A quorum of mix-servers jointly decrypt each triple of cryptotexts in L_k to produce a list L on the form $L = \{(u_i, v_i, w_i)\}_{i=1}^N$. Since the method used to do this is irrelevant to our attacks, we do not present it here.
2. All triples for which $w_i = h(u_i, v_i)$ are called *valid*.
3. Invalid triples are investigated according to the procedure described below. If the investigation proves that all invalid triples are *benign* (only senders cheated), we proceed to Step 4. Otherwise, the decryption is aborted, and we continue with the back-up mixing.

4. A quorum of mix-servers jointly decrypt the cryptotexts (u_i, v_i) for all valid triples. This successfully concludes the mixing. The final output is defined as the set of plaintexts corresponding to valid triples.

Special Step: Investigation of Invalid Triples

The mix-servers must reveal the path of each invalid triple through the various permutations. For each invalid triple, starting from the last server, each server reveals which of its inputs corresponds to this triple, and how it re-randomized this triple. One of two things may happen:

- **Benign case (only senders cheated):** if the mix-servers successfully produce all such paths, the invalid triples are known to have been submitted by users. The decryption resumes after the invalid triples have been discarded.

- **Serious case (one or more servers cheated):** if one or more servers fail to recreate the paths of invalid triples, these mix-servers are accused of cheating and replaced, and the mix-net terminates producing no output. In this case, the inputs are handed over to the back-up mixing procedure below.

Back-Up Mixing

The *outer-layer* encryption of the inputs posted to the mix-net is decrypted by a quorum of mix-servers. The resulting list of *inner-layer* cryptotexts becomes the input to a standard re-encryption mix-net based on El Gamal (using, for example Neff's scheme described in [50]). Then the output of the standard mix-net is given as output by the mix-net.

Remark 4.2. It is infeasible to find two lists $\{(u_i, v_i)\}_{i=1}^N \neq \{(u'_i, v'_i)\}_{i=1}^N$ such that $\prod_{i=1}^N h(u_i, v_i) = \prod_{i=1}^N h(u'_i, v'_i)$, if the product is interpreted in a group where the discrete logarithm problem is hard. This is stated as a theorem by Wagner [66], who credits Wei Dai with this observation, and appears as a lemma in Golle et al. [34].

During the re-encryption and mixing stage each mix-server proves in zero-knowledge that it leaves the component-wise product $(\prod u_i, \prod v_i, \prod w_i)$, of the inner triples (u_i, v_i, w_i) unchanged, but individual triples may still be corrupted. Then invalid triples are traced back. This leaves only valid inner triples in the output and the proofs of knowledge of each server are used to conclude that the component-wise product of these valid inner triples was left unchanged by the mix-net. Golle et al. [34] then refer to the lemma and conclude that the set of valid triples in the output is identical to the set of valid triples hidden in the double encrypted input to the mix-net.

Unfortunately, this intuitively appealing construction is flawed as we explain in Section 4.2. Furthermore, our third attack in Section 4.2 shows that it is possible to behave maliciously without changing the set of inner triples.

4.2 The Attacks

The goal of the adversary is to break the privacy of our typical honest sender Alice and to alter the output without detection. Each of our attacks illustrates a separate weakness of the protocol. The first two attacks are adaptations of the "relation attack", introduced by Pfitzmann [58, 57], to the setting with double enveloping. The idea of the "relation attack" is that to break the privacy of Alice, the adversary computes a cryptotext of a message related to Alice's message. Then the mix-net is run as usual. The output of the mix-net contains two messages related in a way chosen by the adversary. Some relations enable the adversary to determine the message sent by Alice. We illustrate a slightly generalized variant of Pfitzmann's attack in Figure 3.1 in Section 3.4. The third attack is similar to the attack of Desmedt and Kurosawa [15] in that it exploits intermediate results of the protocol and fools a "product test". The fourth attack may be viewed as a novel combination of the ideas of Lim and Lee [44] and Pfitzmann [58, 57]. The fifth attack seems unrelated to any previous attacks.

First Attack: Honest Mix-Servers

We show that the adversary can break the privacy of the typical sender Alice. All that is required is that it can send two messages to the mix-net, which is a natural assumption in most scenarios. In the following we use the notation for the cryptotext of Alice introduced in Section 4.1. The attack is illustrated in Figure 4.1, and the details are as follows. Recall that $[(\mu_1, \mu_2), (\nu_1, \nu_2), (\omega_1, \omega_2)]$ is the tuple sent by Alice. The adversary does the following:

1. It chooses δ and γ randomly in $\mathbb{Z}_q$, and computes:

$$\begin{aligned} w_\delta &= h(\mu_1^\delta, \mu_2^\delta), & \alpha_\delta &= (E_{(g,y)}(\mu_1^\delta), E_{(g,y)}(\mu_2^\delta), E_{(g,y)}(w_\delta)) \ , \text{ and} \\ w_\gamma &= h(\mu_1^\gamma, \mu_2^\gamma), & \alpha_\gamma &= (E_{(g,y)}(\mu_1^\gamma), E_{(g,y)}(\mu_2^\gamma), E_{(g,y)}(w_\gamma)) \ . \end{aligned}$$

   Then it computes the corresponding proofs of knowledge $\pi_{\mathrm{id}}(\mu_1^\delta, \mu_2^\delta, w_\delta)$ and $\pi_{\mathrm{id}}(\mu_1^\gamma, \mu_2^\gamma, w_\gamma)$. This gives the adversary two perfectly valid input tuples $(\alpha_\delta, \pi_{\mathrm{id}}(\mu_1^\delta, \mu_2^\delta, w_\delta))$, $(\alpha_\gamma, \pi_{\mathrm{id}}(\mu_1^\gamma, \mu_2^\gamma, w_\gamma))$, that it sends to the bulletin board (possibly by corrupting two senders).

2. It waits until the mix-net has successfully completed its execution. During the execution of the mix-net the mix-servers first jointly decrypt the "outer layer" of the double encrypted messages. After benign tuples have been removed the result is a list of valid triples

$$((u_1, v_1, w_1), \ldots, (u_N, v_N, w_N)) \ . \tag{4.1}$$

   The final output of the mix-net is the result of decrypting each inner El Gamal pair (u_i, v_i) and results in a list of cleartext messages $(m_1, \ldots, m_N)$.

3. It computes the list

$$(m'_1, \ldots, m'_N) = (m_1^{\delta/\gamma}, \ldots, m_N^{\delta/\gamma}) \ ,$$

and then finds a pair (i, j) such that $m_i = m'_j$. From this it concludes that with very high probability $m_j = u^\gamma$. Then it computes $z = m_j^{1/\gamma}$, and finds a triple (u_l, v_l, w_l) in the list (4.1) such that $z = u_l$. Finally it concludes that with very high probability m_l was the message sent by Alice to the mix-net.

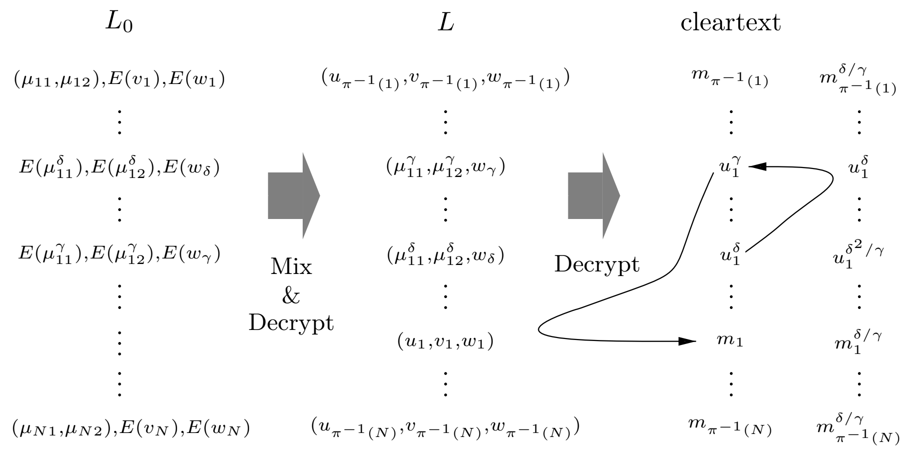
L_0
L
cleartext
$(\mu_{11},\mu_{12}),E(v_1),E(w_1)$
$E(\mu_{11}^\delta),E(\mu_{12}^\delta),E(w_\delta)$
$E(\mu_{11}^\gamma),E(\mu_{12}^\gamma),E(w_\gamma)$
$(\mu_{N1},\mu_{N2}),E(v_N),E(w_N)$
Mix
&
Decrypt
$(u_{\pi^{-1}(1)},v_{\pi^{-1}(1)},w_{\pi^{-1}(1)})$
$(\mu_{11}^\gamma,\mu_{12}^\gamma,w_\gamma)$
$(\mu_{11}^\delta,\mu_{12}^\delta,w_\delta)$
(u_1,v_1,w_1)
$(u_{\pi^{-1}(N)},v_{\pi^{-1}(N)},w_{\pi^{-1}(N)})$
Decrypt
$m_{\pi^{-1}(1)}$
u_1^γ
u_1^δ
m_1
$m_{\pi^{-1}(N)}$
$m_{\pi^{-1}(1)}^{\delta/\gamma}$
u_1^δ
$u_1^{\delta^2/\gamma}$
$m_1^{\delta/\gamma}$
$m_{\pi^{-1}(N)}^{\delta/\gamma}$

Figure 4.1: The figure illustrates the first attack. The leftmost column L_0 is the list of tuples with valid proofs of knowledge of the inner triple, the middle column L is the output of the outer decryption stage, and the rightmost columns, are the cleartext and the exponentiated cleartexts respectively. The adversary follows the arrows to identify the cleartext m_1 belonging to Alice.

Remark 4.3. At additional computational cost it suffices for the adversary to send 2 messages to break the privacy of t senders. Suppose for example that the adversary wants to break the privacy also of Bob and Camilla.

We assume that Bob sent m' encrypted as

$$\begin{aligned}&(u', v') = E_{(g,y)}(m'), \quad w' = h(u', v'), \quad \text{and}\\ &\alpha' = [E_{(g,y)}(u'), E_{(g,y)}(v'), E_{(g,y)}(w')] = [(\mu'_1, \mu'_2), (\nu'_1, \nu'_2), (\omega'_1, \omega'_2)] \ ,\end{aligned}$$

and that Camilla sent m'' encrypted as

$$\begin{aligned}&(u'', v'') = E_{(g,y)}(m''), \quad w'' = h(u'', v''), \quad \text{and}\\ &\alpha'' = [E_{(g,y)}(u''), E_{(g,y)}(v''), E_{(g,y)}(w'')] = [(\mu''_1, \mu''_2), (\nu''_1, \nu''_2), (\omega''_1, \omega''_2)] \ ,\end{aligned}$$

The first step of the attack is unchanged except that the adversary replaces (μ_1, μ_2) by $(\mu_1^\eta(\mu_1')^{\eta'}\mu_1'', \mu_2^\eta(\mu_2')^{\eta'}\mu_2'')$, where $\eta, \eta' \in \mathbb{Z}_q$ are randomly chosen. The adversary proceeds with the attack as before until it has computed z.

At this point in the original attack the adversary identifies an inner triple (u_l, v_l, w_l) such that $z = u_l$ and concludes that m_l was the message sent by Alice.

In the generalized attack the adversary instead identifies three triples (u_l, v_l, w_l), $(u_{l'}, v_{l'}, w_{l'})$ and $(u_{l''}, v_{l''}, w_{l''})$ such that $z = u_l^\eta u_{l'}^{\eta'} u_{l''}$. Then it concludes that m_l was the message sent by Alice, $m_{l'}$ the message sent by Bob, and $m_{l''}$ the message sent by Camilla.

The approach is generalized to higher dimensions in the natural way to break the privacy of t senders.

Why the Attack is Possible

The attack exploits two different flaws of the protocol. The first is that the sender of a message, e.g. Alice, proves only knowledge of the inner El Gamal pair (u, v) and the hash value $w = h(u, v)$, and not knowledge of the message m. The second flaw is that identical El Gamal keys are used for both the inner and outer El Gamal system.

Anybody can compute a single encrypted message $(\mu_1^\delta, \mu_2^\delta) = (g^{r\delta}, y^{r\delta}u^\delta) = E_{(g,y)}(u^\delta, r\delta)$ of a power u^δ of the first component u of the *inner* El Gamal pair (u, v) of the triple α sent by Alice. Anybody can also compute a proof of knowledge of $(\mu_1^\delta, \mu_2^\delta)$ and $w_\delta = h(\mu_1^\delta, \mu_2^\delta)$ (and similarly for $(\mu_1^\gamma, \mu_2^\gamma)$ and ω_γ).

The first flaw allows the adversary to input triples of El Gamal pairs with such proofs of knowledge to the mix-net. The second flaw allows the adversary to use the mix-net to decrypt $(\mu_1^\delta, \mu_2^\delta)$, and thus get its hands on u^δ (and similarly for u^γ). The adversary can then identify (u, v) as the inner El Gamal pair of Alice and break her privacy.

Second Attack: Different Keys and Corrupt Mix-Server

Suppose we change the protocol slightly by requiring that the mix-servers generate separate keys for the outer and inner El Gamal systems, to avoid the first attack of Section 4.2. We assume that there are two different key pairs $((g, y_{\text{in}}), x_{\text{in}})$ and $((g, y_{\text{out}}), x_{\text{out}})$, for the inner and outer El Gamal layers respectively. We also assume that these keys have been shared similarly as the original key pair $((g, y), x)$. This is the type of double enveloping proposed in Chapter 2. For the second attack to succeed we need some additional assumptions.

Unclear Details and Additional Assumptions

We start by quoting Section 5, under “Setup.” point 4 of Golle et al. [34], which presents the proof of knowledge $\pi_{\text{id}}(u, v, w)$ of the sender Alice:

4. This proof of knowledge should be bound to a unique mix-session identifier to achieve security over multiple invocations of the mix. Any user who fails to give the proof is disqualified, and the corresponding input is discarded.

If different keys are used for each mix-session, then the above makes no sense, since the proof of knowledge of u, v and w already depends on the public key of the outer El Gamal system. There is clearly practical value in not changing keys between mix-sessions. We assume that the keys are not changed between mix-sessions even if a mix-server is found to be cheating. If a mix-server is found to be cheating, its shared keys are instead reconstructed by the remaining mix-servers using the secret sharing scheme, and in later mix-sessions the actions of the cheating mix-server are performed in the open (the details of this does not matter to our attack). Under these assumptions we can give an attack on the protocol.

The original paper of Golle et al. [34] does not explicitly say if the discovery of the corrupted mix-server results in a new execution of the key generation protocol. Apparently the intention of the authors is to let the remaining mix-servers generate a new set of keys if any cheating is discovered [35].

The attack is interesting even though this interpretation is not the one intended by the authors, since it shows the importance of explicitly defining all details of protocols and highlights some issues with running several concurrent mix-sessions using the same set of keys.

The Attack

Apart from the above assumptions, the attack only requires that the first mix-server in the mix-chain is corrupted. The attack is employed during two mix-sessions using the same keys and the corrupted mix-server is identified as a cheater in the first mix-session. In the following we describe the actions of the adversary during the first and second mix-sessions, respectively.

The First Mix-Session. We assume that Alice and some other arbitrary sender Bob have sent inputs to the mix-net (and use the notation of Remark 4.3 for the input of Bob). The adversary corrupts M_1. It then replaces α and α' with:

$$[E_{y_{\mathrm{out}}}(u), E_{y_{\mathrm{out}}}(v), E_{y_{\mathrm{out}}}(w')], \quad \text{and} \quad [E_{y_{\mathrm{out}}}(u'), E_{y_{\mathrm{out}}}(v'), E_{y_{\mathrm{out}}}(w)]$$

respectively, in its input list, i.e. the third components of the two triples are shifted. Then it forces M_1 to simulate a completely honest mix-server on the resulting altered list

$$L'_0 = \{[(a'_{0,i}, b'_{0,i}), (c'_{0,i}, d'_{0,i}), (e'_{0,i}, f'_{0,i})]\}_{i=1}^{N} \ .$$

Note that $\prod_{i=1}^{N} a'_{0,i} = a_0$, and similarly for b_0, c_0, d_0, e_0, and f_0. Thus the simulated honest mix-server outputs perfectly valid zero-knowledge proofs that the product of the inner triples are unchanged.

At the end of the mixing the mix-servers verify the tuples and discover the invalid tuples (u, v, w') and (u', v', w). These tuples are traced back all the way to the first mix-server, which is revealed as a cheater. In this process the adversary is able to link Alice to (u, v) (and Bob to (u', v')). Finally the honest mix-servers finish the protocol by using the general constructions based on the work by Neff [50] as suggested in Golle et al. [34].

The Second Mix-Session. To allow the mix-net to execute a second mix-session using the same set of keys, the cheaters key is reconstructed and revealed by a quorum of the mix-servers.

To determine the contents of the El Gamal pair (u, v), the adversary waits for the second mix-session using the same set of keys. Then it uses a "relation attack" [58, 57, 38] in the second mix-session to decrypt (u, v), i.e. it does the following:

1. It chooses δ and γ randomly in $\mathbb{Z}_q$, and computes:

$$\begin{aligned} w_\delta &= h(u^\delta, v^\delta), \quad \alpha_\delta = (E_{y_{\mathrm{out}}}(u^\delta), E_{y_{\mathrm{out}}}(v^\delta), E_{y_{\mathrm{out}}}(w_\delta)), \quad \text{and} \\ w_\gamma &= h(u^\gamma, v^\gamma), \quad \alpha_\gamma = (E_{y_{\mathrm{out}}}(u^\gamma), E_{y_{\mathrm{out}}}(v^\gamma), E_{y_{\mathrm{out}}}(w_\gamma)) \ . \end{aligned}$$

   Then it computes the corresponding proofs of knowledge $\pi_{\mathrm{id}}(u^\delta, v^\delta, w_\delta)$ and $\pi_{\mathrm{id}}(u^\gamma, v^\gamma, w_\gamma)$. This gives two perfectly valid pairs $(\alpha_\delta, \pi_{\mathrm{id}}(u^\delta, v^\delta, w_\delta))$ and $(\alpha_\gamma, \pi_{\mathrm{id}}(u^\gamma, v^\gamma, w_\gamma))$, which it sends to the bulletin board (possibly by corrupting two senders).

2. It waits until the mix-net has successfully completed its execution. The final output of the mix-net is a list of cleartext messages $(m_1, \ldots, m_N)$.

3. Note that $m_i = m^\delta$ and $m_j = m^\gamma$ for some i and j. The adversary computes $\delta\gamma^{-1} \bmod q$, computes the list

$$(m'_1, \ldots, m'_N) = (m_1^{\delta/\gamma}, \ldots, m_N^{\delta/\gamma}) \ ,$$

   and finally finds a pair (i, j) such that $m_i = m'_j$. Then it concludes that with high probability $m_j^{1/\gamma}$ is the message sent by Alice to the mix-net in the *first* mix-session.

Remark 4.4. The attack is easily generalized to break the privacy of several senders by using a circular shift of the third components during the first mix-session. It is also easy to see that Remark 4.3 can be applied to reduce the number of messages sent by the adversary during the second mix-session.

Why the Attack is Possible

The attack exploits that the sender of a message only proves knowledge of the inner triple (u, v, w). At the cost of detected cheating the adversary finds a (u, v) corresponding to Alice, and then uses the second mix-session as a decryption oracle to get its hands on m.

A Note on Concurrent Mix-Sessions

Ignoring the other attacks, a simple counter-measure to the second attack above, is to stipulate that if a cheating mix-server is identified new keys must be generated for the next mix-session.

A disadvantage of this counter-measure is that the mix-net can not be allowed to execute several concurrent mix-sessions using the same keys. If one mix-session is still receiving messages while another mix-session discovers a cheating mix-server the second attack of Section 4.2 can still be applied. The problem is not solved by running the back-up mix-net of Neff [50] on all mix-sessions using the same keys at this point.

This problem of concurrency may seem academic, since in most election scenarios it is not very cumbersome to have different keys for each mix-session, but in future applications of mix-nets it may be useful to run several concurrent mix-sessions using the same keys.

Third Attack: Honest Senders and One Corrupt Mix-Server

In this section we assume that all senders and all mix-servers, except the last mix-server M_k, are honest. The last mix-server M_k is corrupted by the adversary and performs the attack. The attack breaks both the robustness and the privacy.

To simplify our notation we write $L_0 = \{\alpha_i\}_{i=1}^N$ for the input list, where we define $\alpha_i = [(a_{0,i}, b_{0,i}), (c_{0,i}, d_{0,i}), (e_{0,i}, f_{0,i})]$ to be the tuple sent by sender S_i. Instead of following the protocol, M_k proceeds as follows in the first stage.

1. It computes the following tuples

$$\begin{aligned}(a', b', \ldots, f') &= (a_{k-1}/a_0, b_{k-1}/b_0, \ldots, f_{k-1}/f_0), \quad \text{and} \\ \alpha'_1 &= [(a'a_{0,1}, b'b_{0,1}), (c'c_{0,1}, d'd_{0,1}), (e'e_{0,1}, f'f_{0,1})] \ .\end{aligned}$$

2. Then it forms the list

$$L'_{k-1} = \{\alpha'_1, \alpha_2, \ldots, \alpha_N\} \ ,$$

   i.e. a copy of L_0 with the first tuple α_1 replaced by α'_1.

3. When M_k is supposed to re-randomize and permute the output L_{k-1} of M_{k-1} it instead simulates the actions of an honest mix-server on the corrupted input L'_{k-1}. The output list written to the bulletin board by the simulated mix-server is denoted L_k.

4. It waits until the inner layer has been decrypted and uses its knowledge of the permutation that relates L_k to L_0 to break the privacy of all senders.

We show that the attack goes undetected, i.e. the mix-servers decrypt the inner layer. This implies that the attack succeeds.

Firstly, consider the proof of knowledge that mix-server M_k produces during the re-encryption and mixing stage. Define

$$a'_{k-1} = (a' a_{0,1}) \prod_{i=2}^{N} a_{0,i} \enspace ,$$

and similarly for for b'_{k-1}, c'_{k-1}, d'_{k-1}, e'_{k-1}, and f'_{k-1}. In Step 3 above, the simulated honest mix-server outputs proofs of knowledge of the following equalities of logarithms

$$\begin{aligned} \log_g a_k/a'_{k-1} &= \log_y b_k/b'_{k-1} \enspace , \\ \log_g c_k/c'_{k-1} &= \log_y d_k/d'_{k-1} \enspace , \text{ and} \\ \log_g e_k/e'_{k-1} &= \log_y f_k/f'_{k-1} \enspace . \end{aligned}$$

By construction we have that

$$a'_{k-1} = (a' a_{0,1}) \prod_{i=2}^{N} a_{0,i} = a' \prod_{i=1}^{N} a_{0,i} = \frac{a_{k-1}}{a_0} a_0 = a_{k-1} \enspace ,$$

and similarly for b_{k-1}, c_{k-1}, d_{k-1}, e_{k-1}, and f_{k-1}. This implies that the proof of knowledge produced by M_k is deemed valid by the other mix-servers.

Secondly, consider the investigation of invalid inner triples. Tracing back invalid triples is difficult to M_k, since it does not know the re-encryption exponents and the permutation relating L_{k-1} and L_k. We show that there are no invalid inner triples. Suppose we define the sums

$$r = \sum_{j=1}^{k-1} \sum_{i=1}^{N} r_{ji}, \quad s = \sum_{j=1}^{k-1} \sum_{i=1}^{N} s_{ji}, \quad \text{and} \quad t = \sum_{j=1}^{k-1} \sum_{i=1}^{N} t_{ji} \enspace .$$

i.e. we form the sum of all re-encryption exponents used by all mix-servers except the last, for the first second and third El Gamal pairs respectively. Since all mix-servers except M_k are honest, we have

$$(a', b', c', d', e', f') = (g^r, y^r, g^s, y^s, g^t, y^t) \enspace ,$$

which implies that α'_1 is a valid re-encryption of α_1. Thus M_k does not corrupt any inner triple by simulating an honest mix-server on the input L'_{k-1}. Since all senders are honest and the set of inner triples hidden in L_0 and L'_{k-1} are identical, there are no invalid inner triples. Thus cheating is not detected and robustness is broken.

We conclude that the mix-servers decrypt the inner triples, i.e. the privacy of *all* senders is broken.

Why the Attack is Possible

The third attack above exploits that the last mix-server M_k is not forced to take the output L_{k-1} of the next to last mix-server as input. This allows M_k to use a slightly modified version of L_0 instead, which breaks the privacy of all senders.

Fourth Attack: Two Corrupt Mix-Servers

In this section we assume that the first and last mix-servers, M_1 and M_k, are corrupted. We give an attack that breaks the privacy of any given sender Alice. Let $L_0 = \{\alpha_i\}_{i=1}^N$, where $\alpha_i = [(a_{0,i}, b_{0,i}), (c_{0,i}, d_{0,i}), (e_{0,i}, f_{0,i})]$. Without loss we let α_1 and α_2 be the tuples sent by Alice and Bob respectively. Let $\mathbb{Z}_p^* = G_q \times G_\kappa$ and let $1 \neq \zeta \in G_\kappa$. Thus ζ is an element *outside* of the group G_q. The adversary corrupts M_1 and M_k, and they proceed as follows.

1. M_1 computes two modified elements

$$\begin{aligned} \alpha'_1 &= [(\zeta a_{0,1}, b_{0,1}), (c_{0,1}, d_{0,1}), (e_{0,1}, f_{0,1})] \ \text{, and} \\ \alpha'_2 &= [(\zeta^{-1} a_{0,2}, b_{0,2}), (c_{0,2}, d_{0,2}), (e_{0,2}, f_{0,2})] \ . \end{aligned}$$

   Then it forms the modified list $L'_0 = \{\alpha'_1, \alpha'_2, \alpha_3, \ldots, \alpha_N\}$ and instructs M_1 to simulate an honest mix-server on input L'_0. Note that the first component of α'_1 and α'_2 respectively is no longer contained in G_q.

2. M_k simulates an honest mix-server on input L_{k-1}, but it does not write the output L_k of this simulation on the bulletin board. Let $L_k = \{\beta_i\}_{i=1}^N$, where $\beta_i = [(a_{k,i}, b_{k,i}), (c_{k,i}, d_{k,i}), (e_{k,i}, f_{k,i})]$. M_k finds $l, l' \in \{1, \ldots, N\}$ such that

$$a_{k,l}^q = \zeta^q, \quad \text{and} \quad a_{k,l'}^q = \zeta^{-q} \ .$$

   Then it computes the tuples

$$\begin{aligned} \beta'_l &= [(\zeta^{-1} a_{k,l}, b_{k,l}), (c_{k,l}, d_{k,l}), (e_{k,l}, f_{k,l})] \ \text{, and} \\ \beta'_{l'} &= [(\zeta a_{k,l'}, b_{k,l'}), (c_{k,l'}, d_{k,l'}), (e_{k,l'}, f_{k,l'})] \ , \end{aligned}$$

   form the list

$$L'_k = \{\beta_1, \ldots, \beta_{l-1}, \beta'_l, \beta_{l+1}, \ldots, \beta_{l'-1}, \beta'_{l'}, \beta_{l'+1}, \ldots, \beta_N\} \ ,$$

   and writes L'_k on the bulletin board.

3. The mix-net outputs $(m_1, \ldots, m_N)$ and the adversary concludes that Alice sent m_l.

Since all mix-servers except M_1 and M_k are honest there exists $l, l' \in \{1, \ldots, N\}$ and $r, r' \in \mathbb{Z}_q$ such that

$$a_{k,l} = g^r \zeta a_{0,1}, \quad \text{and} \quad a_{k,l'} = g^{r'} \zeta^{-1} a_{0,2} \ .$$

This implies that

$$a_{k,l}^q = \zeta^q (g^r a_{0,1})^q = \zeta^q , \quad \text{and} \quad a_{k,l'}^{-q} = \zeta^{-q} \ .$$

since $\beta^q = 1$ for any $\beta \in G_q$. We have $\zeta^q \neq 1 \neq \zeta^{-q}$, since the order of ζ does not divide q. On the other hand we have $a_{k,i}^q = 1$ for $i \neq l, l'$, since $a_{k,i} \in G_q$ for $i \neq l, l'$. Thus the adversary successfully identifies Alice's cryptotext if the cheating of M_1 and M_k is not detected.

Clearly, the values of b_1, c_1, d_1, e_1, and f_1 are not changed by replacing L_0 with L_0' in Step 1. Neither is a_1, since

$$(\zeta a_{0,1})(\zeta^{-1} a_{0,2}) \prod_{i=3}^{N} a_{0,i} = \prod_{i=1}^{N} a_{0,i} = a_1 \ .$$

Similarly, b_k, c_k, d_k, e_k, and f_k are not changed by replacing L_k with L_k' in Step 2. Neither is a_k, since $(\zeta^{-1} a_{k,l})(\zeta a_{k,l'}) \prod_{i \neq l,l'}^{N} a_{k,i} = \prod_{i=1}^{N} a_{k,i}$. Similarly as in the second attack of Section 4.2, we conclude that the proofs of knowledge produced by M_1 and M_k are deemed valid by the other mix-servers. If we assume that Alice and Bob are honest, their inner triples are never traced back and cheating is not detected.

If $\zeta = \zeta^{-1}$ the adversary can only conclude that Alice sent a message from the set $\{m_l, m_{l'}\}$. This breaks the privacy of Alice, but the adversary can also identify Alice's message uniquely by choosing Bob to be a corrupted sender.

Remark 4.5. Our attack may be viewed as a novel combination of the ideas in Lim and Lee [44] and Pfitzmann [58, 57] in that we use elements in $\mathbb{Z}_p^* \backslash G_q$ to execute a "relation attack". In Section 5.2 we discuss the work of Lim and Lee [44] further, and explain why the counter-measure they propose is insufficient.

Why the Attack is Possible

The attack exploits that a mix-server M_j does not verify that all elements in its input L_{j-1} are in G_q. M_1 uses this to "tag" elements in L_0, which lets them be identified by the last mix-server M_k.

Fifth Attack: One Corrupt Mix-Server

In Proposition 3 of Golle et al. [34] the authors claim that their protocol satisfies the following strong definition of public verifiability if there is a group G_q in which the discrete logarithm problem is hard.

Definition 1. *(Public Verifiability (cf. [34])) A mix net is* public verifiable *if there exists a polynomially bounded verifier that takes as input the transcript of the mixing posted on the bulletin board, outputs "valid" if the set of valid outputs is a permuted decryption of all valid inputs, and otherwise outputs "invalid" with overwhelming* *probability. Note that to prove public verifiability, we consider an adversary that can control all mix-servers and all users.*

Unfortunately, Proposition 3 in [34] is false. The following is a counter-example.

Suppose that there are 4 senders, and that the adversary corrupts two of the senders and the first mix-server M_1. Let

$$\begin{aligned}(u_i, v_i, w_i) &= (g^{r_i}, y^{r_i} m_i, h(u_i, v_i)), \quad \text{and} \\ \alpha_i &= ((a_{0,i}, b_{0,i}), (c_{0,i}, d_{0,i}), (e_{0,i}, f_{0,i})) = (E_y(u_i), E_y(v_i), E_y(w_i))\end{aligned}$$

for $i = 1, 2, 3, 4$. Then define

$$\begin{aligned}\alpha'_3 &= ((a_{0,3}, b_{0,3}), (c_{0,3}, ad_{0,3}), (e_{0,3}, f_{0,3})), \quad \text{and} \\ \alpha'_4 &= ((a_{0,4}, b_{0,4}), (c_{0,4}, a^{-1}d_{0,4}), (e_{0,4}, f_{0,4}))\end{aligned}$$

for some $1 \neq a \in G_q$. Suppose that α_1 and α_2 are sent to the mix-net by honest senders, and α'_3 and α'_4 are sent to the mix-net by the corrupted senders, with corresponding proofs of knowledge.

M_1 replaces α'_3, and α'_4 with α_3 and α_4. This does not change the value of the component-wise product $d_1 = v_1 v_2 v'_3 v'_4$ since $v'_3 v'_4 = v_3 v_4$, and the cheating is not detected, since α_3 and α_4 corresponds to valid inner triples, and thus not traced back. On the other hand the tuples α'_3 and α'_4 correspond to invalid inner triples

$$(u_3, v_3, aw_3), \quad \text{and} \quad (u_4, v_4, a^{-1}w_4) \ .$$

We conclude that the sets of valid inner triples in the input and output respectively differ, public verifiability is broken, and Proposition 3 of [34] is false.

Some may argue that this is not important since the adversary may only choose to correct invalid messages which it has previously prepared for this particular purpose. However, note that the adversary may *choose* whether to correct α'_3 and α'_4. If it chooses not to correct invalid triples they are simply traced back and considered benign.

The following application shows the importance of this subtlety. We use the mix-net construction to run two independent elections (using different keys). First all votes for both elections are collected, and after some time both elections are closed. Then the mix-net is executed for the first election. Finally the mix-net is executed for the second election. In this scenario the adversary can insert specially prepared invalid triples (i.e. votes) in the second election, and then decide whether to correct these triples *based on the outcome* of the first election. This should clearly not be allowed, but may be acceptable in certain non-voting scenarios.

Why the Attack is Possible

The attack exploits the fact that the first mix-server can choose whether to correct specially prepared invalid inner triples or not without detection.

4.3 Further Applications

All the attacks described above work also if $\mathbb{Z}_p^* \supset G_q$ are replaced by any other similarly related groups, e.g. elliptic curves.

The attacks of sections 4.2, 4.2, and 4.2 all exploit the particular structure of the protocol of Golle et al. [34]. We have found no other protocol vulnerable to these attacks. In particular the protocol by Jakobsson [40] with similar structure is not vulnerable to the attack of Section 4.2.

In Section 4.3 below we sketch how the attack of Section 4.2 can be applied to break the privacy of "Flash Mix" by Jakobsson [40], including the fixed protocol proposed by Mitomo and Kurosawa [49]. We note that the latter proposal is given without security claims.

In Section 4.3 below we sketch how the attack of Section 4.2 breaks the robustness of Jakobsson and Juels [42].

In Chapter 5 we investigate a more complicated variant of the attack of Section 4.2 and more generally the malicious use of elements in $\mathbb{Z}_p^* \backslash G_q$.

Attack for "Flash Mix"

In this section we assume that the reader is familiar with the protocol by Jakobsson [40] and sketch how this protocol can be broken by a natural adaptation of the novel attack of Section 4.2.

The attack is employed during the "second re-encryption". The adversary corrupts the first and the last mix-servers, M_1 and M_k, in the mix-chain. During the "second re-encryption", M_1 "tags" two arbitrary El Gamal pairs in its input by multiplying their first components with ζ and ζ^{-1} respectively (similarly as in Section 4.2). Then the honest mix-servers perform their re-encryption and mixing. When the last mix-server M_k is about to re-encrypt and mix the output of the previous mix-server M_{k-1}, it identifies the "tagged" El Gamal pairs, removes the "tags" by multiplying the first components by ζ^{-1} and ζ respectively, and then finally re-encrypts and mix the resulting list honestly.

After the verification of the "first re-encryption", this breaks the privacy of some randomly chosen sender, if the cheating goes undetected. Although the attack is weak, it does break privacy.

Cheating is detected if one of two things happen; the adversary by chance chooses a "dummy element" that is later traced back through the mix-chain, or if M_1 or M_k fails to play its part in the computation of the "relative permutations" correctly. The first event is very unlikely since by construction there are very few "dummy elements". Since the "tags" are removed by M_k, and both M_1 and M_k follow the protocol except for the use of the tags, it follows that the cheating is not detected. It is easy to see that the changes introduced by Mitomo and Kurosawa [49] do not prevent the above attack.

Attack for "An optimally robust hybrid mix network"

Jakobsson and Juels [42] presents a hybrid mix-net. We assume familiarity with their protocol and sketch how it is broken using the attack of Section 4.2.

Suppose that there are four senders, and that the i:th sender forms a cryptotext $(c_0^{(i)}, \mu_0^{(i)}, y_0^{(i)})$ of a message m_i. The adversary corrupts the last two senders and modifies their cryptotexts as follows before they hand them to the mix-net. It replaces $y_0^{(3)}$ by $\overline{y}_0^{(3)} = ay_0^{(3)}$ by and $y_0^{(4)}$ by $\overline{y}_0^{(4)} = a^{-1}y_0^{(4)}$ for some $a \neq 1$.

The adversary corrupts M_1. M_1 then replaces $\overline{y}_0^{(3)}$ by $y_0^{(3)}$ and $\overline{y}_0^{(4)}$ by $y_0^{(4)}$ and simulates an honest M_1 on the modified input. Similarly as in the original attack this does not change the component-wise product $P_0 = y_0^{(1)}y_0^{(2)}\overline{y}_0^{(3)}\overline{y}_0^{(4)} = y_0^{(1)}y_0^{(2)}y_0^{(3)}y_0^{(4)}$. The VerifyComplaint procedure is never invoked, since all cryptotexts are valid. Thus the cheating is not detected.

We conclude that the set of cleartext messages corresponding to the set of valid cryptotexts in the input differs from the set of cleartext messages in the output of the mix-net. This breaks the robustness, i.e. Definition 4(b) of [42].

4.4 Future Work

Further mix-nets may be vulnerable to the attacks. The attack of Section 4.2 is of special interest since it seems unrelated to other attacks in the literature. In particular it should be verified for other hybrid mix-nets that they are not vulnerable to variants of this attack.

Chapter 5

Elements in $\mathbb{Z}_p^* \backslash G_q$ are Dangerous

The group G_q, of prime order q, for which the discrete logarithm problem is assumed hard is often a subgroup of a larger group, e.g. $\mathbb{Z}_p^*$ for some prime $p = \kappa q + 1$. In this section we investigate how elements from $\mathbb{Z}_p^* \backslash G_q$ may be used to attack cryptographic protocols that fail to verify their inputs properly.

Our work extends the work by Lim and Lee [44]. They give a key recovery attack based on similar observations, and propose an ad hoc counter-measure. We give an example of a protocol where their counter-measure is not sufficient. We also discuss some special cases of groups G_q for which the problem can be solved easily, and outline a generic recipe, for *all* groups G_q, for how to efficiently counter any malicious use of elements from $\mathbb{Z}_p^* \backslash G_q$.

The presentation given here differs from the original presentation given in [71]. Section 5.3 on special groups in which the problem is trivially solved has been changed to take into account the results of Chapter 7. A discussion on related results by Lim and Lee [44], and how they differ from ours has been added. Since the original version we have realized that the protocol of Furukawa and Sako [25] is not zero-knowledge as claimed. Thus we have eliminated those of our results that depend on this result, but we still use their construction as an example of a protocol for which the counter-measure of Lim and Lee does not suffice.

We thank Andy Neff for correcting an error and for helpful discussions that improved this chapter. We also thank Jun Furukawa for helping us understand their protocol.

5.1 Notation

In this chapter we let $p = \kappa q + 1$, and q be prime numbers, such that κ is small, e.g. bounded by a constant. We write $p_1, \ldots, p_s$ for the prime factors of κ, and let G_q and G_κ be the unique subgroups of $\mathbb{Z}_p^*$, of order q and κ respectively, i.e. $\mathbb{Z}_p^* = G_q \times G_\kappa$. We also identify the elements of the field $\mathbb{Z}_q$ with the elements of the set $[q] = \{0, \ldots, q-1\}$, i.e. we use a fixed representative $a \in [q]$ for each coset

$a+(q)$ of the ideal (q) in $\mathbb{Z}$. This is mostly the case in practice. Throughout this chapter h denotes a hash function modeled as a random oracle, but the range of h differs depending on the section.

Each protocol we consider may be viewed as a public coin zero-knowledge proof of knowledge for which the Fiat-Shamir heuristic is applied, and the standard formalization of this heuristic is the random oracle model. Neither the attacks we consider, nor the counter-measures we suggest depend in principle on the random oracle model, but the attacks we present are stronger in the random oracle model setting due to the limited interaction between the prover and the verifier.

5.2 Two Observations

Although the observations below are fairly obvious, it seems that some important consequences of these observations have been overlooked in the literature.

1. When G_q is used for cryptographic purposes the arithmetic of this group is usually not implemented directly. Instead a library for arithmetics in the group $\mathbb{Z}_p^*$ and a library for arithmetic in the field $\mathbb{Z}_q$ are used. Together these libraries provide efficient arithmetic in G_q. A side effect of this design is that implementations normally are able to perform arithmetic for *all* of $\mathbb{Z}_p^*$, at least with regard to certain operations.

2. In some protocols in the literature based on arithmetic in G_q the parties do not verify their input properly. Even if elements in the input are expected to belong to G_q, it is only verified that they belong to $\mathbb{Z}_p^*$. The reason is probably that it is easy to verify membership in $\mathbb{Z}_p^*$, but costly to verify membership in G_q.

The two observations indicate that the adversary may be able to hand the parties corrupt inputs from $\mathbb{Z}_p^*\backslash G_q$ without detection. Depending on the particular protocol under attack, the implications of the additional power of the adversary are more or less severe. The adversary may use the additional power in several ways, e.g. it may introduce elements from $\mathbb{Z}_p^*$ to corrupt the output of an execution, but it may also try to extract information by using corrupt input in the interaction with honest parties. The particular protocol under attack decides what is appropriate.

Related Work

Lim and Lee [44] present a key recovery attack based on the above observations. The idea of the attack of Lim and Lee may be summarized as follows.

An honest party is in possession of a secret key x. Suppose that the adversary is able to make the honest party compute and output a^x for some $a \in \mathbb{Z}_p^*\backslash G_q$. We have $a = a'\zeta$, where $a' \in G_q$ and $\zeta \in G_\kappa$. Thus the adversary may compute $\beta = (a^x)^q = \zeta^{qx}$. Since κ is small it is feasible to compute the discrete logarithm $\log_\zeta \beta$, i.e. $qx \mod \kappa$. This implies that the adversary is able to deduce approximately

$\log_2 \kappa$ bits of the secret key x. Thus the attack is dangerous in practice when κ is relatively large, since this notably decreases the search space for the secret key.

Lim and Lee give examples of protocols vulnerable to the above attack. They also propose that to avoid costly verifications, p may be chosen such that $(p-1)/2$ only has large prime factors. The idea is that at most a single bit of the secret key is leaked since solving discrete logarithms in any subgroup of other order than 2 is infeasible.

Our work may be viewed as an extension of their work in several directions. Lim and Lee give a particular key recovery attack. We instead point out that there is a broad class of attacks based on the malicious use of elements in $\mathbb{Z}_p^* \backslash G_q$. We also illustrate the more general problem by giving particular attacks that are not prevented by the counter-measure proposed by Lim and Lee. Finally we discuss efficient counter-measures that prevent *any* malicious use of elements of $\mathbb{Z}_p^* \backslash G_q$, whereas they only consider how to prevent their particular key recovery attack.

5.3 Explicit Verifications and How to Avoid Them

The naïve method for removing the additional power of the adversary is that each subprotocol independently verifies that its inputs belong to G_q. First we identify three special types of groups for which this is a natural alternative. Then we describe a generic recipe for how to modify protocols such that no explicit verifications are necessary.

We note that it is easy to verify that a string of bits s represents an element $a \in \mathbb{Z}_p^*$. In the most common representation this amounts to checking that if s is interpreted as an integer a in base two, we have $0 < a < p$. Similarly it is easy to verify membership in $\mathbb{Z}_q$. The corresponding verification for a general finite field is also easy. The verification for an elliptic curve group is slightly more complicated, but not a problem.

Throughout this chapter we assume that all parties explicitly verify that their inputs belong to $\mathbb{Z}_p^*$ or $\mathbb{Z}_q$ as appropriate.

Special Groups

Consider $\mathbb{Z}_p^*$, for $p = \kappa q + 1$. When $\kappa = 2$ the Legendre symbol $(\frac{a}{p})$ equals $a^q \bmod p$ (if properly interpreted). Shallit and Sorenson [62] present an efficient algorithm to compute the Jacobi symbol. Previous versions of the work in this section erroneously claimed that fast checking of membership in G_q requires an exponentiation. Andy Neff pointed out to us that this is false if $\kappa = 2$. In Section 7.6 we discuss how membership in G_q can be determined efficiently also for $\kappa \in \{4, 6, 12\}$. For other values of κ it is not clear how membership in G_q may be determined more efficiently than computing a full exponentiation modulo p.

Now consider how the test for membership in G_q can be removed by forcing G_q to *equal* the larger group G for which arithmetic is implemented, i.e. G_q is no longer a proper subgroup. This is clearly not possible if $G = \mathbb{F}_{p^k}^*$ for odd p (including $\mathbb{Z}_p^*$), since the order $p^k - 1$ of $\mathbb{F}_{p^k}^*$ is always an even number in this case. It is possible for $G = \mathbb{F}_{2^p}^*$, when $q = 2^p - 1$ is a Mersenne prime. It is not known if there are infinitely many Mersenne primes, and there exist for practical security parameters only two or three useful values of p, but for practical purposes this may be an alternative.

Another alternative is to use an elliptic curve group of prime order. This alternative requires that it is verified that an element is contained on the appropriate curve.

Remark 5.1. In some implementations of arithmetic for elliptic curves the implementation does not depend on all parameters of the curve. Thus even if the curve itself is of prime order, i.e. the group G_q is the curve itself, the implementation may operate perfectly well on inputs from other curves. It is reasonable to expect that this allows attacks similar to the ones presented here. We do not investigate this further here.

The General Case

The ideas in the remainder of this chapter are not restricted to any particular type of group, but for concreteness we continue to use $\mathbb{Z}_p^*$ as a generic example. To verify that $a \in G_q$ given that $a \in \mathbb{Z}_p^*$ is costly in general. The best way we are aware of is to verify that $a^q = 1$ by computing a full exponentiation in $\mathbb{Z}_p^*$.

If $a \in \mathbb{Z}_p^*$, we may write $a = a'\zeta$ for some $a' \in G_q$ and $\zeta \in G_\kappa$. The following simple observations are useful.

1. We have $a^\kappa = (a')^\kappa \zeta^\kappa$, and since the order of ζ divides κ by definition, we have $\zeta^\kappa = 1$, and $a^\kappa = (a')^\kappa \in G_q$. It is also very easy to compute a^κ, since this corresponds to at most $\frac{3}{2}\log_2 \kappa$ multiplications in G_q. We call such exponentiations κ-exponentiations. Similarly, multiplication by κ in $\mathbb{Z}_q$ is not a full multiplication, and we call this a κ-multiplication.

2. Let $\theta \in \mathbb{Z}_q$ have the property that $\gcd(\theta, \kappa) = 1$ (when θ and κ are viewed as integers). We have $a^\theta = (a')^\theta \zeta^\theta$. Clearly $(a')^\theta \in G_q$, but $\zeta^\theta \notin G_q$, since the order of ζ divides κ and $\gcd(\theta, \kappa) = 1$. Thus if $\gcd(\theta, \kappa) = 1$, then $a \in \mathbb{Z}_p^*\backslash G_q$ implies that $a^\theta \in \mathbb{Z}_p^*\backslash G_q$.

The first observation gives an efficient map $\mathbb{Z}_p^* \to G_q$. The second observation shows that for any $a \in \mathbb{Z}_p^*$ and $r \in \mathbb{Z}_q$ there is a unique efficiently computable $\theta(r) \in \mathbb{Z}_q$ such that if $a \notin G_q$ then $a^{\theta(r)} \notin G_q$.

These observations may be used to modify protocols slightly such that very few explicit verifications of membership in G_q are needed. It seems impossible to give a single detailed description of exactly how to proceed for each individual protocol, but it is possible to give a generic recipe.

> RECIPE: Consider the elements of a protocol that are required to belong to G_q. Find a small subset of these such that if membership in G_q is explicitly verified for these, the protocol may be modified using the observations above to ensure that the rest of the elements also belong to G_q.

Depending on the protocol, different methods may be required, but the basic idea is the same. Below we give some concrete examples of how this is done.

5.4 Application to Proofs of Knowledge of Logarithms

In this section we apply the above observations to proofs of knowledge of discrete logarithms. For concreteness we consider a specific proof of knowledge, but the ideas are easily adapted to the wide variety of similar proofs of knowledge in the literature. In this section we let $h : \{0,1\}^* \to \mathbb{Z}_q$.

A Proof of Knowledge

Let $a, A, b, B \in G_q$, where $A = a^\gamma$, and $B = b^\gamma$ for some $\gamma \in \mathbb{Z}_q$ known by the prover. We review the standard non-interactive zero-knowledge proof of knowledge in the random oracle model of a $\gamma \in \mathbb{Z}_q$ such that $\log_a A = \log_b B = \gamma$.

1. The prover chooses $\delta \in \mathbb{Z}_q$ randomly and computes $\alpha = a^\delta$, $\beta = b^\delta$, $\theta = h(\alpha, \beta, A, B, a, b)$, and $d = \gamma\theta + \delta$. The proof consists of the tuple (α, β, d).
2. The verifier computes $\theta = h(\alpha, \beta, A, B, a, b)$ and verifies that $A^\theta \alpha = a^d$, and $B^\theta \beta = b^d$.

An equivalent variant of this that gives a shorter proof is to let (θ, d) be the proof and let the verifier check that $\theta = h(a^d/A^\theta, b^d/B^\theta, A, B, a, b)$, but for increased readability we use the variant described above. A proof of the following proposition can be found in the literature, e.g. [11, 60].

Proposition 5.2. *Let $a, b, A, B \in G_q$. Then the above is a zero-knowledge proof of knowledge in the random oracle model of a $\gamma \in \mathbb{Z}_q$ such that* $\log_a A = \log_b B = \gamma$.

An Adversarial Strategy

Remember that $\mathbb{Z}_p^* = G_q \times G_\kappa$, and let $1 \neq \zeta \in G_\kappa$ have order ξ. We consider two scenarios:

1. A is replaced by ζA, and
2. a is replaced by ζa.

The prover is malicious and claims that it knows a $\gamma \in \mathbb{Z}_q$ such that $\log_a A = \log_b B = \gamma$. The adversarial strategy is very simple. Repeatedly, i.e. at most a polynomial number of times in $\log_2 p$, simulate the honest prover on the corrupted joint input a, b, A, B, until it outputs a proof that is deemed valid by the verifier. It is clear that if the prover succeeds, the verifier is convinced.

What the adversary hopes for in the scenarios is that $\zeta^\theta = 1$ and $\zeta^\delta = \zeta^d$ respectively. If this happens it is easy to see that the proof produced by the simulated prover is deemed valid.

Lemma 5.3. *Each iteration succeeds with probability at least $\frac{1}{\xi} + O(1/q)$, and the adversarial strategy fails with negligible probability.*

Proof. Consider Scenario 1, where $A = \zeta a^\gamma$. The proof (α, β, d) is deemed valid if $A^\theta \alpha = \zeta^\theta a^{\gamma\theta} a^\delta = a^d$. This happens if $\zeta^\theta = 1$, i.e. with probability $\frac{1}{\xi} + O(1/q)$. Similarly, for Scenario 2, the proof is deemed valid if $\zeta^\delta = \zeta^d$. This also happens with probability $\frac{1}{\xi} + O(1/q)$, since δ and d are independently distributed.

From independence of the iterations follows that the strategy fails with negligible probability.

The reader may note that if we instead consider the public coin zero-knowledge proof of knowledge corresponding to the above protocol, the probability of success for the adversary equals the probability of success in a single iteration in the random oracle model setting.

We stress that we do not claim that Proposition 5.2 is false. So how is it possible that the prover can fool the verifier with non-negligible probability? The answer is simple. Proposition 5.2 requires that $a, b, A, B \in G_q$, and when this is not the case the proposition is no longer guaranteed to hold.

To verify that $a, b, A, B, \in G_q$ before each invocation of the protocol of Section 5.4 would be costly, and many applications do ensure, or can be modified to ensure, that $a, b \in G_q$ prior to the invocation of the protocol. Thus an efficient proof of knowledge under the weaker assumption that $a, b \in G_q$ and $A, B \in \mathbb{Z}_p^*$ is useful. In the next section we describe such a protocol.

How to Avoid Some Explicit Verifications

In this section we assume that the verifier is somehow convinced that $a, b \in G_q$, i.e. the calling context ensures this. We apply the generic method of Section 5.3 to modify the proof of knowledge of Section 5.4, such that it stays a proof of knowledge under the weaker assumption that $a, b \in G_q$ and $A, B \in \mathbb{Z}_p^*$.

Since we identified $[q] = \{0, \ldots, q-1\}$ and $\mathbb{Z}_q$, we may interpret $\theta \in \mathbb{Z}_q$ as an integer, denote by θ^* the unique integer such that $\theta = \theta^* \prod_{i=1}^s p_i$, where $p_i \nmid \theta^*$ for $i = 1, \ldots, s$, and then view θ^* as an element in $\mathbb{Z}_q$ (recall that $\kappa = \prod_{i=1}^s p_i$). The modified proof of knowledge is defined as follows:

1. The prover chooses $\delta \in \mathbb{Z}_q$ randomly and computes $\alpha = a^\delta$, $\beta = b^\delta$, $\theta = h(\alpha, \beta, A, B, a, b)$, and $d = \gamma\theta^* + \kappa\delta$. The proof consists of the tuple (α, β, d).

2. The verifier first computes $\theta = h(\alpha, \beta, A, B, a, b)$. Then it verifies that the equalities $A^{\theta^*}\alpha^\kappa = a^d$, and $B^{\theta^*}\beta^\kappa = b^d$ hold.

Recall that κ is bounded by a constant. This implies that the prover and the verifier may compute θ^* from θ by repeated trial division with the prime factors $p_1, \ldots, p_s$ of κ. It also implies that the exponentiations α^κ and β^κ can be performed quickly.

Proposition 5.4. *Let $a, b \in G_q$, and $A, B \in \mathbb{Z}_p^*$. Then the above is a zero-knowledge proof of knowledge in the random oracle model of a $\gamma \in \mathbb{Z}_q$ such that $\log_a A = \log_b B = \gamma$.*

Proof. First we prove that a proof is not valid if either A or B is in $\mathbb{Z}_p^* \backslash G_q$. We may by the structure of $\mathbb{Z}_p^*$ write $A = A'\zeta$ for some $A' \in G_q$ and $1 \neq \zeta \in G_\kappa$. This implies that $A^{\theta^*} = (A')^{\theta^*}\zeta^{\theta^*}$. We have $(A')^{\theta^*} \in G_q$, and since $\gcd(\theta^*, \kappa) = 1$ by construction we have $1 \neq \zeta^{\theta^*} \in G_\kappa$. Thus if $A \in \mathbb{Z}_p^* \backslash G_q$ then $A^{\theta^*} \in \mathbb{Z}_p^* \backslash G_q$. Similarly, since $\alpha \in \mathbb{Z}_p^*$ we may write $\alpha = \alpha'\zeta$ for some $\alpha' \in G_q$ and $\zeta \in G_\kappa$. This implies that $\alpha^\kappa = (\alpha')^\kappa\zeta^\kappa = (\alpha')^\kappa \in G_q$. Intuitively we may view this as if the prover and verifier jointly generated a uniformly distributed element $\alpha^\kappa \in G_q$. It is clear that $a^d \in G_q$. Thus if $A \in \mathbb{Z}_p^* \backslash G_q$ we have that $A^{\theta^*}\alpha^\kappa \in \mathbb{Z}_p^* \backslash G_q$, which clearly can not equal $a^d \in G_q$ in $\mathbb{Z}_p^*$. The proof for B is similar.

The extractor of the above protocol is almost identical to the extractor of the original protocol. The only difference is that when $\alpha, \beta \in G_q$ and $\theta_0, \theta_1 \in \mathbb{Z}_q$ are found such that the prover outputs valid proofs (α, β, d_0) and (α, β, d_1) given θ_0 or θ_1 as answer to the question $(\alpha, \beta, A, B, a, b)$ the extractor solves the equation system $\{d_b = \gamma\theta_b^* + \kappa\delta\}_{b \in \{0,1\}}$ in the unknowns γ and δ.

The simulator is almost identical to the simulator of the original protocol. Since κ is a generator of the additive group $\mathbb{Z}_q$, and δ is uniformly distributed in $\mathbb{Z}_q = [q]$, d is uniformly distributed in $\mathbb{Z}_q$. Thus the simulator chooses $d, \theta \in \mathbb{Z}_q$ uniformly at random and defines $\alpha = (a^d/A^{\theta^*})^{1/\kappa}$, and $\beta = (b^d/B^{\theta^*})^{1/\kappa}$. It follows that the transcript (α, β, d) is correctly distributed.

There are obvious variations of the above for proofs of knowledge involving more complex relations between logarithms.

5.5 Application to El Gamal Based Mix-Nets

In this section we describe how our ideas can be applied to break the formal robustness of many El Gamal based mix-nets.

The Generic Structure of El Gamal Based Mix-Nets

It is assumed that the mix-servers $M_1, \ldots, M_k$ have set up a distributed El Gamal cryptosystem, where M_j is somehow given a random $x_j \in \mathbb{Z}_q$, and $y_l = g^{x_l}$ is made public for $l = 1, \ldots, k$. We define a joint public key $y = \prod_{j=1}^k y_j$, with corresponding private key $x = \sum_{j=1}^k x_j$. It is also assumed that x_j is shared verifiably, secretly and robustly to all M_l for $l \neq j$. How this is done is of no importance to us here, but it allows the robust elimination of any mix-server identified as a cheater.

The following is a generic description of an El Gamal based mix-net (recall that $F_{(g,y)}$ denotes El Gamal re-encryption).

Protocol 5.5 (Generic Mix-Net).

1. Each sender S_i computes $(u_i, v_i) = E_y(m_i)$, and a proof of knowledge ϕ_i of m_i, and writes the pair $((u_i, v_i), \phi_i)$ on the bulletin board. Tsionis and Yung [65] show that $((u_i, v_i), \phi_i)$ is a non-malleable [18] cryptotext in the random oracle model.

2. Let L_0 be the list of cryptotexts (u_i, v_i) for which ϕ_i is valid, and assume that the length of L_0 is N. For $j = 1, \ldots, k$, M_j does the following:

   a) It chooses $R_j \in \mathbb{Z}_q^N$ and $\pi_j \in \Sigma_N$ randomly, computes the list $L_j = F_{(g,y)}(\pi_j L_{j-1}, R_j)$, and writes L_j on the bulletin board.
   b) It gives a zero-knowledge proof of knowledge of π_j and R_j such that $L_j = F_{(g,y)}(\pi_j L_{j-1}, R_j)$.

3. Each mix-server M_j computes $\Lambda_j = P_{x_j}(L_k)$, writes Λ_j on the bulletin board, and gives a zero-knowledge proof of knowledge of x_j such that $y_j = g^{x_j}$ and $\Lambda_j = P_{x_j}(L_k)$.

4. Each mix-server M_j computes the component-wise product $\Lambda = \prod_{j=1}^k \Lambda_j$ and the list of permuted cleartexts $D_P(\Lambda, L_k)$.

If some mix-server M_j produces a corrupt proof of knowledge, its private key x_j is reconstructed using the secret sharing scheme and its computations are performed openly.

An Attack for El Gamal Based Mix-Nets

Most mix-nets, not only those that fit the generic description above, end with a joint decryption step similar to Step 3 above. In several mix-nets in the literature the mix-servers neglect to verify that all components of Λ_j belong to G_q.

Let $L_k = \{(u_{k,i}, v_{k,i})\}_{i=1}^N$. Then the above implies that the strategy of Section 5.4 can be used by a corrupted mix-server to publish a list $\Lambda_j = \{\lambda_{j,i}\}_{i=1}^N$ such that $\lambda_{j,i} = \zeta u_{k,i}^{-x_j}$, where $1 \neq \zeta \in G_\kappa$, instead of $\lambda_{j,i} = u_i^{-x_j}$ and produce a proof of knowledge deemed valid by all other mix-servers. This means that m_i is replaced by ζm_i in the output without detection and the robustness of the mix-net is broken.

This attack is applicable to many El Gamal based mix-nets in the literature, e.g. [34, 40, 49, 38, 59, 52, 1, 50], but there are also mix-nets that are not vulnerable to this attack, e.g. [15]. Although some authors may be aware of this attack, we think that verifications should be an integral part of a protocol or it should be clearly stated that explicit verifications must take place at *every* call to a subprotocol.

5.6 Application to “An Efficient Scheme for Proving a Shuffle”

Consider the generic structure of an El Gamal based mix-net as presented in Section 5.5. The second step of the re-encryption and permutation stage states that the active mix-server M_j should prove in zero-knowledge that it performed the re-encryption correctly.

Furukawa and Sako [25] claim to provide exactly this. Unfortunately, their protocol is not zero-knowledge. Apparently the authors are aware that the proof of the zero-knowledge property is flawed [25]. In previous versions of the work of this section we were not aware of this. Still it is an example of a more complicated protocol that allows an adversarial strategy similar to the one in Section 5.4.

Given L_{j-1} and L_j the protocol is meant to prove knowledge of R_j and π_j such that $F_{(g,y)}(\pi_j L_{j-1}, R_j)$. We have a closer look at their proof. Write

$$L_{j-1} = \{(u_i, v_i)\}, \quad L_j = \{(u'_i, v'_i)\}, \quad \text{and} \quad \pi = \pi_j \ .$$

We also denote the prover M_j by P and any verifier, e.g. M_l for $l \neq j$, by V. In this section we denote by h a random oracle $h : \{0,1\}^* \to \mathbb{Z}_q^N$. Following [25] we define the permutation matrix $A = (A_{ij})$ by:

$$A_{ij} = \begin{cases} 1 & \text{if } \pi(i) = j \\ 0 & \text{otherwise} \end{cases} ,$$

and let $\tilde{g}, \tilde{g}_1, \ldots, \tilde{g}_N \in G_q$ be a set of system-wide uniformly and independently generated random elements, i.e. no party knows any non-trivial relation among these elements. The $\tilde{g}, \tilde{g}_i$ elements are an integral part of the Furukawa-Sako protocol.

The protocol is very complicated. Readers that are not familiar with the protocol can safely ignore the details of how the protocol works and consider the overall structure. On the other hand we choose to follow the notation of Furukawa and Sako to allow readers that are familiar with the construction to recognize the details.

The Furukawa-Sako Construction

The list $g, y, \tilde{g}, \{\tilde{g}_i\}, \{(u_i, v_i)\}, \{(u'_i, v'_i)\}$ is the joint input to the prover P and the verifier V, and $\{r_i\}, \pi$ is the additional input to P. $\pi \in \Sigma_N$.

Protocol 5.6 (Furukawa-Sako).

1. P chooses $\sigma, \rho, \tau, \vartheta, \vartheta_i, \lambda, \lambda_i \in \mathbb{Z}_q$ uniformly and independently at random.

2. P computes:

$$\begin{aligned}
&t = g^{\tau}, \quad \nu = g^{\rho}, \quad \omega = g^{\sigma}, \quad \mu = g^{\lambda}, \quad \mu_i = g^{\lambda_i} \\
&\tilde{g}'_i = \tilde{g}^{r_i} \prod_{j=1}^{N} \tilde{g}_j^{A_{ji}}, \quad \tilde{g}' = \tilde{g}^{\vartheta} \prod_{j=1}^{N} \tilde{g}_j^{\vartheta_j}, \quad u' = g^{\vartheta} \prod_{j=1}^{N} u_j^{\vartheta_j}, \quad v' = y^{\vartheta} \prod_{j=1}^{N} v_j^{\vartheta_j} \\
&\dot{\nu}_i = g^{\sum_{j=1}^N 3\vartheta_j^2 A_{ji} + \rho r_i}, \quad \dot{\nu} = g^{\sum_{j=1}^N 3\vartheta_j^3 + \tau\lambda + \rho\vartheta}, \\
&\dot{\omega}_i = g^{\sum_{j=1}^N 2\vartheta_j A_{ji} + \sigma r_i}, \quad \dot{\omega} = g^{\sum_{j=1}^N \vartheta_j^2 + \sigma\vartheta}, \quad t_i = g^{\sum_{j=1}^N 3\vartheta_j A_{ji} + \tau\lambda_i}
\end{aligned}$$

3. P computes a challenge:

$$\begin{aligned}
(c_1, \ldots, c_N) \quad = \quad & h\big(t, \nu, \omega, \mu, \{\mu_i\}, \{\tilde{g}'_i\}, \tilde{g}', u', v', \{t_i\}, \{\dot{\nu}_i\}, \dot{\nu}, \{\dot{\omega}_i\}, \dot{\omega}, \\
& g, y, \tilde{g}, \{\tilde{g}_i\}, \{(u_i, v_i)\}, \{(u'_i, v'_i)\}\big) \ .
\end{aligned}$$

4. P computes $s = \vartheta + \sum_{j=1}^N r_j c_j$, $s_i = \vartheta_i + \sum_{j=1}^N A_{ij} c_j$, and $\lambda' = \lambda + \sum_{j=1}^N \lambda_j c_j^2$. The proof consists of the following tuple:

$$(t, \nu, \omega, \mu, \{\mu_i\}, \{\tilde{g}'_i\}, \tilde{g}', u', v', \{t_i\}, \{\dot{\nu}_i\}, \dot{\nu}, \{\dot{\omega}_i\}, \dot{\omega}, s, \{s_i\}, \lambda') \ .$$

5. V computes

$$\begin{aligned}
(c_1, \ldots, c_N) \quad = \quad & h\big(t, \nu, \omega, \mu, \{\mu_i\}, \{\tilde{g}'_i\}, \tilde{g}', u', v', \{t_i\}, \{\dot{\nu}_i\}, \dot{\nu}, \{\dot{\omega}_i\}, \dot{\omega}, \\
& g, y, \tilde{g}, \{\tilde{g}_i\}, \{(u_i, v_i)\}, \{(u'_i, v'_i)\}\big) \ ,
\end{aligned}$$

and verifies that:

$$\begin{aligned}
&g^s \prod_{j=1}^N u_j^{s_j} = u' \prod_{j=1}^N u_j'^{c_j}, \quad y^s \prod_{j=1}^N v_j^{s_j} = v' \prod_{j=1}^N v_j'^{c_j}, \qquad (5.1) \\
&\tilde{g}^s \prod_{j=1}^N \tilde{g}_j^{s_j} = \tilde{g}' \prod_{j=1}^N \tilde{g}_j'^{c_j}, \quad g^{\lambda'} = \mu \prod_{j=1}^N \mu_j^{c_j^2}, \\
&\omega^s g^{\sum_{j=1}^N (s_j^2 - c_j^2)} = \dot{\omega} \prod_{j=1}^N \dot{\omega}_j^{c_j}, \quad \text{and} \quad t^{\lambda'} \nu^s g^{\sum_{j=1}^N (s_j^3 - c_j^3)} = \dot{\nu} \prod_{j=1}^N \dot{\nu}_j^{c_j} t_j^{c_j^2} \ .
\end{aligned}$$

Furukawa and Sako [25], present an interactive protocol corresponding to the above, but their intention [27] is that the protocol is made non-interactive using the Fiat-Shamir heuristic.

An Adversarial Strategy

In this section we describe an adversarial strategy for Protocol 5.6 above. We stress that we do not break the protocol, i.e. our adversarial strategy does not exploit the fact that the protocol is not zero-knowledge. Similarly as in Section 5.4 we instead exploit how this proof of knowledge may be applied.

Let ζ be an element in G_κ of order ξ, and let $\{(u_i, v_i)\}$, and $\{(u'_i, v'_i)\}$ be as defined in the previous section. We consider three scenarios, where $l \in [N]$

1. $v'_{\pi^{-1}(l)}$ is replaced by $\zeta^\iota v'_{\pi^{-1}(l)}$ for some ι,
2. v_l is replaced by ζv_l, and
3. a combination of 1 and 2.

In each scenario the strategy convinces an honest verifier that the prover knows $r_i \in \mathbb{Z}_q$ and a permutation π such that $\{(u'_i, v'_i)\} = \{(g^{r_i} u_{\pi^{-1}(i)}, g^{r_i} v_{\pi^{-1}(i)})\}$, despite that this is not even possible in Scenario 1 and 2.

The strategy is very simple. Repeatedly, i.e. at most a polynomial number of times in $\log_2 p$, run the first 4 steps of the Furukawa-Sako protocol of Section 5.6 on the corrupted input $g, y, \{\tilde{g}_i\}, \{(u_i, v_i)\}, \{(u'_i, v'_i)\}, \{r_i\}, \pi$ until it gives a proof that is deemed valid by the verifier. Then output the proof. It may seem that the strategy always succeeds in Scenario 3, but for similar reasons as in Section 5.4 this is not true.

Lemma 5.7. *Let a and b be uniformly distributed in $[q]$, and write $c = a + b$ and $c = c' + m_c q$, where $m_c \in \{0, 1\}$ and $0 \leq c' < q$. Then we have*
$\Pr[c' = c \pmod q] = \frac{1}{2} + \frac{1}{2q}$.

Proof. If $c < q$ we have $c' = c = a+b$, and there are $\frac{q^2+q}{2}$ such pairs (a, b). Otherwise we have $c = c' + q \neq c' \pmod \xi$, since $q \neq 0 \pmod \xi$. Thus, the probability is $\frac{1}{2} + \frac{1}{2q}$.

Lemma 5.8. *Each iteration in Scenario 1 and Scenario 2 succeeds with probability at least $\frac{1}{\xi} + O(1/q)$. Each iteration in Scenario 3 succeeds with probability at least $\frac{1}{2}$. In all scenarios the adversarial strategy fails with negligible probability.*

Proof. Consider Scenario 1. The only equation involving $v_{\pi^{-1}(l)}$ in the verification procedure is Equation (5.1). Thus it suffices to consider this equation. By construction we may rewrite Equation (5.1) as

$$y^s \prod_{j=1}^{N} v_j^{s_j} = \zeta^{c_{\pi^{-1}(l)}} v' \prod_{j=1}^{N} v_j'^{c_j} \quad .$$

Thus, if $\zeta^{c_{\pi^{-1}(l)}} = 1$, Equation (5.1) holds, and this happens at least with probability $\frac{1}{\xi} + O(1/q)$. The analysis for Scenario 2 is similar.

For Scenario 3 we note that by construction we may rewrite Equation (5.1) as

$$\zeta^{s_l} y^s \prod_{j=1}^{N} v_j^{s_j} = \zeta^{c_{\pi^{-1}(l)}} \zeta^{\vartheta_l} v' \prod_{j=1}^{N} v_j'^{c_j} \quad .$$

Thus, if $\zeta^{s_l} = \zeta^{c_{\pi^{-1}(l)}} \zeta^{\vartheta_l}$, Equation (5.1) holds. This event occurs exactly when $s_l = c_{\pi^{-1}(l)} + \vartheta_l \bmod \xi$, and by definition there is an m such that $s_l + mq = c_{\pi^{-1}(l)} + \vartheta_l$. We apply lemma 5.7 to conclude that the probability of this event is at least $\frac{1}{2}$.

From independence follows that the adversarial strategy fails with negligible probability.

Numerous variants of the the above three scenarios may be considered, e.g. several elements may be multiplied by ζ, or different $\zeta_i \in G_\kappa$ may be used for different elements.

An Attack for the Generic Mix-Net Using Furukawa-Sako Proofs

In this section we describe an attack for the mix-net defined in Section 5.5, where the the Furukawa-Sako construction is used to prove the correctness of a re-encryption. The attack is similar to the attack in Section 4.2 of Chapter 4.

The goal of the adversary is to break the privacy of any particular sender S_z.

Let $1 \neq \zeta \in G_\kappa$. The adversary corrupts the first and last mix-servers M_1 and M_k, and as many other arbitrarily chosen mix-servers as possible. The execution then proceeds as follows.

1. M_1 simulates the first step of an honest mix-server, but it does not write L_1 on the bulletin board. Instead it forms the list:

$$L_1' = \{(u_{1,1}, v_{1,1}), \ldots, (u_{1,\pi_1(z)}, \zeta v_{1,\pi_1(z)}), \ldots, (u_{1,N}, v_{1,N})\}$$

   and writes L_1' on the bulletin board, i.e. it multiplies the second component of the El Gamal pair of S_z by ζ. Then it uses the adversarial strategy of Section 5.6 on the corrupt input $g, y, \{\tilde{g}_i\}, L_0, L_1', \{r_{1,i}\}, \pi_1$ to construct a proof deemed valid.

2. Each honest mix-server executes its re-encryption and permutation step honestly.

3. Each corrupt mix-server except M_1 and M_k first simulates the first step of an honest mix-server. Then it uses the adversarial strategy of Section 5.6 to construct a proof that is deemed valid.

4. M_k simulates the first step of an honest mix-server, but it does not write L_k on the bulletin board. Instead it finds an l such that $v_{k,l}^q \neq 1$. Finally it forms the corrupted list

$$L_k' = \{(u_{k,1}, v_{k,1}), \ldots, (u_{k,l}, \zeta^{-1} v_{k,l}), \ldots, (u_{k,N}, v_{k,N})\}$$

and writes L'_k on the bulletin board. Then it uses the adversarial strategy of Section 5.6 on the corrupt input $g, y, \{\tilde{g}_i\}, L_{k-1}, L'_k, \{r_{k,i}\}, \pi_k$ to construct a proof deemed valid by any verifier.

5. All mix-servers, including M_1 and M_k, jointly decrypt the El Gamal pairs of the list L'_k, and publish the result $\{m'_1, \ldots, m'_N\}$, i.e. a permutation of $\{m_1, \ldots, m_N\}$.

6. The adversary concludes that S_z was the sender of m'_l.

Proposition 5.9. *Let k' be the number of honest mix-servers. Then the attack succeeds with probability at least $2^{-k'}$.*

Proof. We must show that the attack is not detected, and that m'_l indeed is the message sent by S_z with high probability.

From Lemma 5.8 follows that the proofs of knowledge produced by the corrupt mix-servers, including M_1 and M_k, are deemed valid by any verifier with overwhelming probability. Honest mix-servers on the other hand essentially computes a single iteration of the adversarial strategy in Scenario 3 defined in Section 5.6. From Lemma 5.8 follows that the proof produced by an honest mix-server is valid with probability at least $\frac{1}{2}$. From independence follows that the attack is detected with probability at most $2^{-k'}$.

Since all mix-servers except M_1 and M_k behave honestly in the first step, there is a unique $l_j \in \{1, \ldots, N\}$ for each $j = 1, \ldots, k$ such that $v_{j,l_j} \notin G_q$, and by construction we have $l = \pi_1^{-1}(z)$. Thus the adversary identifies the message m'_l sent by S_z correctly.

The proposition says that we can break the privacy of S_z with probability $2^{-k'}$. In many applications k, and thereby k', is relatively small. Thus the attack is quite practical in terms of success probability. On the other hand, if the adversarial strategy fails some honest mix-server produces a corrupt proof, and is accused of cheating. It is likely that the honest mix-server accused of cheating performs an investigation to find out how it was "framed", and discovers the fact that its inputs are not in G_q. If this is done the corrupted mix-servers are eventually identified.

Furukawa et al. do perform verifications of inputs in later work [26], and the protocol described there is not vulnerable to the above attack.

In earlier versions of this work we showed that our recipe of Section 5.3 can be used to modify Protocol 5.6 such that it need not verify its input, but since this protocol is not zero-knowledge as claimed we omit this here. Instead we show how the generic mix-net, Protocol 5.5, can be modified to avoid most explicit verifications.

5.7 A Generic Mix-Net that Avoids Verifications

The attack of Section 5.6 exploits that the mix-servers do not verify that all components of their input are members of G_q in Protocol 5.5 and Protocol 5.6. Since it is costly to verify all inputs explicitly in every step we would like to avoid it. We assume that each mix-server explicitly verifies that $g, y \in G_q$.

Given a list L_j of elements from G_q we denote by L_j^κ the element-wise exponentiation with κ. Similarly, if R_j is a list of elements from $\mathbb{Z}_q$ we denote by κR_j the element-wise product with κ. A modified generic mix-net that avoids explicit verifications follows below.

Protocol 5.10 (Generic Mix-Net).

1. Each sender S_i computes $(u_i, v_i) = E_y(m_i)$, and a proof of knowledge ϕ_i of m_i, and writes the pair $((u_i, v_i), \phi_i)$ on the bulletin board.
2. Let L_0 be the list of cryptotexts (u_i, v_i) for which ϕ_i is valid, and assume that the length of L_0 is N. For $j = 1, \ldots, k$, M_j does the following:
   a) It chooses $R_j \in \mathbb{Z}_q^N$ and $\pi_j \in \Sigma_N$ randomly, computes the list $L_j = F_{(g,y)}(\pi_j L_{j-1}^\kappa, R_j)$, and writes L_j on the bulletin board.
   b) It gives a zero-knowledge proof of knowledge of π_j and κR_j such that $L_j^\kappa = F_{(g,y)}(\pi_j L_{j-1}^{\kappa^2}, \kappa R_j)$.
3. Each mix-server M_j computes $\Lambda_j = P_{x_j}(L_k^\kappa)$, and writes Λ_j on the bulletin board, and gives a zero-knowledge proof of knowledge of x_j such that $y_j = g^{x_j}$ and $\Lambda_j = P_{x_j}(L_k^\kappa)$.
4. Each mix-server M_j computes the component-wise product $\Lambda = \prod_{j=1}^k \Lambda_j$ and the list of permuted cleartexts $D_P(\Lambda^\kappa, L_k^{\kappa^2})^{1/\kappa^{k+2}}$.

If some mix-server M_j produces a corrupt proof of knowledge, its private key x_j is reconstructed using the secret sharing scheme and its computations are performed openly.

Each mix-server computes some additional κ-exponentiations to ensure that its inputs belongs to G_q, but since κ is small this can be done efficiently. Thus the main additional cost is the N full exponentiations required in the final step. This cost can be moved to the sender by requiring that a sender encrypts $m_i^{1/\kappa^{k+2}}$ to send the message m_i.

5.8 Further Applications

We expect that natural adaptations of the adversarial strategy for the proofs of knowledge of logarithms can be used to break other protocols. In particular it seems possible to give a purely theoretical attack for the mix-net of Neff [50] similar in spirit to the attack of Section 5.6.

On the other hand we also expect that the methods we outline in Section 5.3 can be used to counter such attacks efficiently.

Chapter 6

On the Malleability of the El Gamal Cryptosystem

The homomorphic property of the El Gamal cryptosystem is useful in the construction of efficient protocols. It is believed that only a small class of transformations of cryptotexts are feasible to compute. In the program of showing that these are the only computable transformations we rule out a large set of natural transformations.

Several efficient cryptographic protocols are based on the El Gamal cryptosystem. The reasons for this are mainly the algebraic simplicity of the idea, and the homomorphic property it possesses. The latter property makes the El Gamal system malleable, i.e. given $c = E(m)$ it is feasible to compute $c' = E(f(m))$, for some non-trivial function f.

It is commonly conjectured that the El Gamal cryptosystem is malleable only for a small class of simple functions, and this is sometimes used implicitly in arguments about the security of protocols. Thus it is an important problem to characterize the malleability of the El Gamal cryptosystem. We take a first step in this direction.

We formalize the problem, and discuss why restrictions of the problem are necessary. Then we show that the only transformations that can be computed perfectly are those of a well known particularly simple type. Furthermore we give two examples that show that possible future results are not as strong as we may think. Finally we rule out a large set of natural transformations from being computable.

6.1 The Problem

For any message $m \in G_q$ there exists a unique element $m_e \in \mathbb{Z}_q$ such that $m = g^{m_e}$. Thus any El Gamal encryption $(g^r, y^r m)$ can be written $(g^r, y^r g^{m_e})$. The latter notation, is sometimes more natural and we use both conventions.

There is a small and well know class of transformations of cryptotexts, used in many protocols, that we summarize in an observation.

Observation 6.1. Set $\phi(r) = b_1 r + b_0$ and $\psi(m_e) = b_1 m_e + h_0$. Then the map:

$$(g^r, y^r g^{m_e}) \mapsto (g^{\phi(r)}, y^{\phi(r)} g^{\psi(m_e)})$$

is feasible to compute.

Some authors use the term homomorphic cryptosystem, since these transformations can be formulated as group homomorphisms.

It is natural to ask what other transformations can or can not be computed "under encryption". For simplicity we use the non-uniform computational model, i.e. feasible transformations are transformations that can be computed by a deterministic non-uniform polynomial size circuit family.

Given $y = g^x$, each pair $(u, v) \in G_q \times G_q$ can be uniquely represented on the form $(u, v) = (g^r, y^r g^{m_e})$. This implies that for each function $f : G_q \times G_q \to G_q \times G_q$, and $y \in G_q$, there are unique functions $\phi_y, \psi_y : \mathbb{Z}_q \times \mathbb{Z}_q \to \mathbb{Z}_q$, such that:

$$f(u, v) = f(g^r, y^r g^{m_e}) = (g^{\phi_y(r,m_e)}, y^{\phi_y(r,m_e)} g^{\psi_y(r,m_e)}) \ .$$

Most general functions f are not what we intuitively would consider "transformations computed under encryption", and it seems difficult to prove anything useful if we consider any function f a transformation of cryptotexts.

Our approach is therefore to require that a transformation is given by a fixed pair (ϕ, ψ) of deterministic functions $\phi, \psi : \mathbb{Z}_q \times \mathbb{Z}_q \to \mathbb{Z}_q$ and parameterized by y, i.e. we define a map $(y, \phi, \psi) : G_q \times G_q \to G_q \times G_q$ for each y by the following:

$$(y, \phi, \psi) : (g^r, y^r g^{m_e}) \mapsto (g^{\phi(r,m_e)}, y^{\phi(r,m_e)} g^{\psi(r,m_e)}) \ .$$

Such transformations act uniformly for all y, i.e. given $(u_i, v_i) = (g^r, y_i^r g^{m_e})$ for $i = 1, 2$ we have $(y_i, \phi, \psi)(u_i, v_i) = (g^{\phi(r,m_e)}, y_i^{\phi(r,m_e)} g^{\psi(r,m_e)})$.

Our method can not be applied to general uniform transformations, and we are forced to further restrict the problem. We require that ϕ depends only on r, and that ψ depends only on m_e. Thus we study the special problem posed as follows:

Problem 6.2. Given $\phi, \psi : \mathbb{Z}_q \to \mathbb{Z}_q$, let $(y, \phi, \psi)(g^r, y^r g^{m_e}) = (g^{\phi(r)}, y^{\phi(r)} g^{\psi(m_e)})$. For which ϕ and ψ is the transformation (y, ϕ, ψ) feasible to compute?

6.2 Our Results

We exhibit two propositions. The first shows that only transformations of the type described in Observation 6.1 can be computed perfectly. Then we give two examples that show that strong results can not be hoped for. Finally we give a proposition, which may have some practical significance. It identifies a set of functions ψ such that the map (y, ϕ, ψ) is hard to compute for every ϕ.

Some Preparation

The hypothesis of the propositions differ only slightly depending on which case is considered. To avoid duplication of the hypothesis, and for increased clarity we give it here. Let $G_q = \{G_{q_n}\}$ be a family of groups such that $|G_{q_n}| = q_n$, where q_n is a prime number such that $\lceil \log_2 q_n \rceil = n$, and assume that DDH holds in G_q. Let $g = \{g_n\}$ be a generator of G_q.

Definition 6.3.

1. Let $X = \{X_n\}$ be a family of random variables, where X_n is u.i.d. in $\mathbb{Z}_{q_n}$, and let $Y = \{Y_n\}$, where $Y_n = g^{X_n}$.

2. Let $R = \{R_n\}$ be a family of random variables, where R_n is u.i.d. in $\mathbb{Z}_{q_n}$.

3. Let $M = \{M_n\}$ be a family of random variables on G_{q_n}, and define the induced family $(U, V) = \{(U_n, V_n)\}$ of random variables by setting $(U_n, V_n) = E_{Y_n}(M_n, R_n)$.

4. Let $\phi = \{\phi_n\}$ and $\psi = \{\psi_n\}$ be families of functions over $\mathbb{Z}_q$, i.e. $\phi_n, \psi_n : \mathbb{Z}_{q_n} \to \mathbb{Z}_{q_n}$. Define for each family $y = \{y_n\} \in G_q$ a family of maps $(y, \phi, \psi) = \{(y_n, \phi_n, \psi_n)\}$, where:

$$\begin{aligned}(y_n, \phi_n, \psi_n) &: G_{q_n} \times G_{q_n} \to G_{q_n} \times G_{q_n} \\ (y_n, \phi_n, \psi_n) &: (g_n^r, y_n^r g_n^{m_e}) \mapsto (g_n^{\phi_n(r)}, y_n^{\phi_n(r)} g_n^{\psi_n(m_e)}) \ .\end{aligned}$$

Definitions of M, ϕ, and ψ are given separately in each proposition. The following definition, first given by Goldwasser and Micali [31] define what should be considered randomly guessing the output of a knowledge function.

Definition 6.4. Let $M = \{M_n\}$ be a family of random variables, where the outcomes of M_n are in G_{q_n}, and let $f = \{f_n\}$ be a family of functions $f_n : G_{q_n} \to G_{q_n}$. We define:

$$p_n(f, M) = \max_{v \in G_{q_n}} \Pr[M_n \in f_n^{-1}(v)] \ .$$

The probability $p_n(f, M)$ is the maximum probability of any algorithm to guess $f_n(M_n)$ using no information on the outcome of M_n except its distribution.

Since El Gamal is semantically secure [31, 65] we have that under the DDH-assumption with arbitrary $f = \{f_n\}$, that $\forall A \in \text{PC}$, $\forall c$, $\exists n_0$ such that for $n > n_0$ it holds that:

$$\Pr[A(Y, (U, V)) = f(M)] < p_n(f, M) + \frac{1}{n^c} \ .$$

The Perfect Case

The following proposition says that if we require a circuit family to succeed with probability 1 in computing the map (y, ϕ, ψ) the only possible maps are those where ψ is linear.

Proposition 6.5. *Let G_q, X, Y, M, (U, V), ϕ and ψ be as in Definition 6.3, let M be arbitrarily distributed in G_q, and assume that $\psi_n(x)$ is non-linear for infinitely many n.*

Then $\forall A \in \mathrm{PC}$, $\exists n_0$ such that $\forall n > n_0$:

$$\Pr[A(Y, (U, V)) = (Y, \phi, \psi)(U, V)] < 1 \ .$$

Proof. The proof is by contradiction. Assume that A, ϕ, and ψ as above show the proposition false for indices n in some infinite index set $\mathcal{N}$. Then $\psi_1(x) = \psi(1+x) - \psi(x)$ is not constant. Let g^{m_0} and g^{m_1} be two messages such that $\psi_1(m_0) \neq \psi_1(m_1)$. Let A' be the circuit family that given a public key y and an encryption (u, v) of the message g^{m_b} computes $(u_0, v_0) = A(y, (u, v))$ and $(u_0, v_1) = A(y, (u, vg))$, and returns b when $v_1/v_0 = g^{\psi_1(m_b)}$.

Clearly A' breaks the polynomial indistinguishability, and thus the semantic security of the El Gamal cryptosystem.

Two Examples of Possible Approximations

A reasonable goal is to bound the probability for any adversary to compute (y, ϕ, ψ) for any choice of ϕ.

We now present two examples that show that we should not hope for general strong results. Both examples assume that G_q, X, Y, M, (U, V), ϕ and ψ are as in Definition 6.3, and that M is u.i.d. .

Example 6.6. Let ψ be arbitrary but fixed and let w maximize $\Pr[M \in \psi^{-1}(w)]$. Let A be the circuit family that computes $r' = h(u)$, where $h : G_q \to \mathbb{Z}_q$, and then outputs $(g^{r'}, y^{r'} g^w)$.

Clearly $\Pr[A(Y, (U, V)) = (Y, \phi, \psi)(U, V)] = p_n(\psi, M)$, where $\phi(r) = h(g^r)$. The example shows that for every ψ there is a non-trivial ϕ such that the map (y, ϕ, ψ) can be computed with probability at least $p_n(\psi, M)$.

Thus the best general result under Definition 6.3 we could hope for at this point is to show that $\forall A \in \mathrm{PC}$, $\forall c > 0$, $\exists n_0 > 0$, such that for $n > n_0$:

$$\Pr[A(Y, (U, V)) = (Y, \phi, \psi)(U, V)] < p_n(\psi, M) + \frac{1}{n^c} \ ,$$

but no such result exists as the next example shows.

Example 6.7. Let $c > 0$ be fixed and define $B_n = \{x \in \mathbb{Z}_{q_n} : 0 \leq x \leq \frac{q_n}{n^c}\}$, $B = \{B_n\}$. Define $\psi_n(x) = x + 1$ if $x \in B_n$, and $\psi_n(x) = x^2$ otherwise, and set $\phi = \mathrm{id}$. Let A be the circuit family that assumes that the input $(u, v) = (g^r, y^r g^{m_e})$ satisfies $m_e \in B$, and simply outputs (u, vg).

We have $|\psi^{-1}(x)| \leq 3$ for all $x \in \mathbb{Z}_q$, which implies $p_n(\psi, M) \leq \frac{3}{q_n}$, but still A computes (y, ϕ, ψ) with probability $1/n^c$ for a fixed c. Thus the example shows that we can sometimes compute a transformation with much greater probability than $p_n(\psi, M)$, i.e. the probability of guessing $\psi(m_e)$.

Intuitively the problem seems to be that our ability to compute transformations from the class described in Observation 6.1 changes what should be considered guessing.

A Class of Hard Transformations

We now exhibit a class of ψ that are hard in the sense that the map (y, ϕ, ψ) is hard to compute for all ϕ.

The idea of Proposition 6.11 below is that given input $(y, (u, v))$ and an oracle A for computing a transformation (y, ϕ, ψ) we can ask A several different but related questions. If A answers our questions correctly we are able to compute some derived knowledge function f of the cleartext.

Let $\psi = \{\psi_n\}$ be a family of functions, $\psi_n : \mathbb{Z}_{q_n} \to \mathbb{Z}_{q_n}$, and let $s \in \mathbb{Z}_q$. Denote by ψ_s the function given by $\psi_s(x) = \psi(x + s) - \psi(x)$. We prove below that a ψ that satisfies the following definition has the desired property.

Definition 6.8. Let $\psi = \{\psi_n\}$ be a family of functions, $\psi_n : \mathbb{Z}_{q_n} \to \mathbb{Z}_{q_n}$, let $M = g^{M_e}$, where M_e is a random variable in $\mathbb{Z}_q$, and let S be u.i.d. in $\mathbb{Z}_q$.

If $\forall c > 0$, $\exists n_0 > 0$ such that $\forall n > n_0$ we have:

$$\Pr[p_n(\psi_S, M) < \frac{1}{n^c}] > 1 - \frac{1}{n^c} \ ,$$

then we say that ψ is *strongly non-linear* with respect to M.

The following definition may seem more natural to some readers.

Definition 6.9. Let $\psi = \{\psi_n\}$ be a family of functions, $\psi_n : \mathbb{Z}_{q_n} \to \mathbb{Z}_{q_n}$, let M_e and S be random variables in $\mathbb{Z}_q$, where S is u.i.d. .

If $\forall a \in \mathbb{Z}_q$, $\forall c > 0$, $\exists n_0$ such that $\forall n > n_0$ we have:

$$\Pr[\psi(M_e + S) - \psi(M_e) = \psi(S) + a] < \frac{1}{n^c} \ ,$$

then we say that ψ is *strongly non-linear (variant)* with respect to M_e.

Unfortunately it may capture a larger class than Definition 6.8 as Lemma 6.10 below shows, and we can not prove Proposition 6.11 for all ψ satisfying this definition.

The essential difference between the two definitions is that in the second a is fixed, and does not depend on s, whereas in the first $p_n(\psi_s, M)$ is maximized for each s independently. Note that if we fix $S = s$ in the second definition there is always an a such that the resulting conditioned probability equals $p_n(\psi_s, M)$, but in general a depends on s.

Lemma 6.10. *Strongly non-linear implies strongly non-linear (variant).*

Proof. Set $J(S) = p_n(\psi_S, M)$. Then $\forall c > 0$, $\exists n_0$ such that $\forall n > n_0$:

$$\begin{aligned}
&\Pr[\psi(M_e + S) - \psi(M_e) = \psi(S) + a] \\
&= \sum_{s \in \mathbb{Z}_q} \Pr[S = s] \Pr[\psi(M_e + s) - \psi(M_e) = \psi(s) + a] \\
&\leq \sum_{s \in \mathbb{Z}_q} \Pr[S = s] J(s) = \mathrm{E}[J(S)] \\
&= \Pr[J(S) < \frac{1}{n^c}] \mathrm{E}[J(S) | J(S) < \frac{1}{n^c}] + \Pr[J(S) \geq \frac{1}{n^c}] \mathrm{E}[J(S) | J(S) \geq \frac{1}{n^c}] \\
&< 1 \cdot \frac{1}{n^c} + \frac{1}{n^c} \cdot 1 = \frac{2}{n^c} \quad .
\end{aligned}$$

The Proposition

Informally the proposition below says that if ψ is strongly non-linear, then (y, ϕ, ψ) is hard to compute for all ϕ.

Proposition 6.11. *Let G_q, X, Y, M, (U, V), ϕ and ψ be as in Definition 6.3, let M be u.i.d. in G_q, and assume that ψ is strongly non-linear with respect to M.*

Then $\forall A \in \mathrm{PC}$, $\forall c > 0$, $\exists n_0 > 0$, such that for $n > n_0$:

$$\Pr[A(Y, (U, V)) = (Y, \phi, \psi)(U, V)] < \frac{1}{n^c} \quad .$$

Proof. The proof is by contradiction. Assume A, $c > 0$, ϕ, and ψ, as above shows the proposition false for indices n in some infinite index set $\mathcal{N}$. Define a function f_s for each $s \in \mathbb{Z}_q$ by $f_s(g^{m_e}) = g^{\psi_s(m_e)}$.

We describe a probabilistic circuit family A' that uses A to compute the knowledge function f_s with notable probability. This breaks the semantic security of the El Gamal cryptosystem, if $p_n(f_s, M)$ is negligible. Given input $(y, (u, v))$, where $(u, v) = (g^r, y^r m) \in G_q \times G_q$, A' does the following:

1. It randomly chooses $s \in \mathbb{Z}_q$.
2. It uses A to compute $(u_0, v_0) = A(y, (u, v))$ and $(u_1, v_1) = A(y, (u, vg^s))$
3. It returns $\frac{v_1}{v_0}$.

Let $S = \{S_n\}$ be a u.i.d. random variable over $\mathbb{Z}_q$, and let H_0 denote the event that $A(Y, (U, V)) = (Y, \phi, \psi)(U, V)$, and H_1 denote the event that $A(Y, (U, Vg^S)) = (Y, \phi, \psi)(U, Vg^S)$.

If the events H_0 and H_1 take place we have $\frac{v_1}{v_0} = f_S(M)$ by definition of the algorithm.

We see that $((U, V)|R = r)$ and $((U, Vg^S)|R = r)$ are independent variables. Since R is u.i.d. we have:

$$\begin{aligned}
\Pr[H_0 \wedge H_1] &= \sum_{r \in \mathbb{Z}_q} \Pr[R = r] \Pr[H_0 \wedge H_1 | R = r] \\
&= \sum_{r \in \mathbb{Z}_q} \Pr[R = r] \Pr[H_0 | R = r]^2 \\
&\geq \left(\sum_{r \in \mathbb{Z}_q} \Pr[R = r] \Pr[H_0 | R = r] \right)^2 \\
&= \Pr[H_0]^2 \geq \frac{1}{n^{2c}}
\end{aligned}$$

where the inequality is implied by the convexity of the function $h(x) = x^2$ and Jensen's Inequality.

We are only interested in outcomes s of S such that $p_n(\psi_s, M) = p_n(f_s, M)$ is negligible (in particular $s \neq 0$). Let W denote the event that S has this property. By assumption the probability of $\overline{W}$ is negligible and we have:

$$\Pr[W \wedge A'(Y, (U, V)) = f_S(M)] \geq \frac{1}{2n^{2c}} \quad .$$

The inequality implies that there exists for each $n \in \mathcal{N}$ an outcome s_n of S_n such that the inequality still holds. Let $A'' = \{A''_n\}$ be the circuit family that is identical to A' except that A''_n uses this fixed s_n instead of choosing it randomly. We set $s = \{s_n\}$ and $f_s = \{f_{s_n}\}$, and conclude that A'' has the property:

$$\Pr[A''(Y, (U, V)) = f_s(M)] \geq \frac{1}{2n^{2c}} \quad ,$$

for $n \in \mathcal{N}$. Semantic security of the El Gamal cryptosystem implies that $\forall c' > 0$, $\exists n_0$ such that for $n > n_0$ holds:

$$\Pr[A''(Y, (U, V)) = f_s(M)] < p_n(f_s, M) + \frac{1}{n^{c'}} \quad .$$

Since f_s was constructed such that $p_n(f_s, M)$ is negligible we have reached a contradiction.

The proposition can be slightly generalized by considering distributions of the messages that are only almost uniform on its support when the support is sufficiently large. To keep this note simple we omit this analysis.

We proceed by defining a special subclass of the strongly non-linear functions that is particularly simple, and may be important in applications.

Definition 6.12. Let $\psi = \{\psi_n\}$ be a family of functions, $\psi_n : \mathbb{Z}_{q_n} \to \mathbb{Z}_{q_n}$. We say that ψ has *low degree* if $\forall c > 0$, $\exists n_0$ such that for $n > n_0$ it holds that:

$$\frac{\deg \psi_n}{q_n} < \frac{1}{n^c} \ .$$

A simple example of a family $\psi = \{\psi_n\}$ that satisfies the above definition is where $\psi_n(x) = p(x)$ for some fixed polynomial $p(x)$ for all n.

We have the following corollary almost immediately from the proposition.

Corollary 6.13. *Let G_q, X, Y, M, (U, V), ϕ and ψ be as in Definition 6.3, let M be u.i.d. in G_q, and assume that ψ has low degree and that $\deg \psi_n \leq 1$ for at most finitely many n.*

Then $\forall A \in \mathrm{PC}$, $\forall c > 0$, $\exists n_0 > 0$, such that for $n > n_0$:

$$\Pr[A(Y, (U, V)) = (Y, \phi, \psi)(U, V)] < \frac{1}{n^c} \ .$$

Proof. It suffices to show that if ψ has low degree and $\deg \psi_n \leq 1$ for finitely many n then ψ is strongly non-linear. For $s \neq 0$ and large enough n we have $\deg \psi_s > 0$ and $\deg \psi_s = \deg \psi - 1$. This implies that when $s \neq 0$ we have $p_n(\psi_s, M) = \frac{\max |\psi_s^{-1}(v)|}{q_n} \leq \frac{\max |\psi^{-1}(v)|}{q_n} \leq \frac{\deg \psi}{q_n}$, which is negligible since ψ has low degree.

6.3 Future Work

It seems impossible to prove anything useful about general malleability of the El Gamal cryptosystem as discussed in Section 6.1. Instead we have formalized what we consider a reasonably restricted problem.

Under these restrictions we have exhibited a class of transformations that are not feasible to compute, when the message distribution is uniform. We have also given examples that indicate that the best possible results are not as strong as one may think.

It is an open problem to characterize further classes of transformations. A natural generalization is to consider lists of cryptotexts and consider the difficulty of computing transformations on such lists.

Another interesting line of research is to investigate the malleability properties of El Gamal in concrete groups, e.g. the multiplicative group of integers modulo a prime, or an elliptic curve group.

Chapter 7

Algorithms for the Cubic and Quartic Residue Characters

The problem of quadratic residuosity can be described as follows. Given a prime p and an integer a determine if there exists a solution to the equation $x^2 = a \pmod{p}$, i.e. determine if a is a square in $\mathbb{Z}_p^*$. The Legendre symbol $(\frac{a}{p})$ is defined to be equal to 0 if $p \mid a$, equal to 1 if there is a solution to the equation, and equal to -1 if there is no solution to the equation. For a non-prime n with decomposition $n = p_1 \cdot \ldots \cdot p_s$ into primes the Jacobi symbol $(\frac{a}{n})$ is defined in terms of the Legendre symbol by $(\frac{a}{n}) = \prod_{j=1}^{s} (\frac{a}{p_i})$. In contrast to the Legendre symbol, the Jacobi symbol may equal one without there being a solution to $x^2 = a \pmod{n}$.

Shallit and Sorenson [62] has devised an efficient algorithm for computing the Jacobi symbol. Their algorithm runs in quadratic time in the bit size of the inputs, and is based on the same idea as the binary algorithm for the greatest common divisor introduced by Stein [64].

In this chapter we consider the problem of computing the cubic and quartic residue characters in the ring $\mathbb{Z}[\omega]$, for a primitive third root of unity ω, and in the ring $\mathbb{Z}[i]$ respectively. These problems are natural generalizations of the problem of computing the Jacobi symbol. We show that Shallit and Sorenson's algorithm can be generalized to handle this case.

Our motivation to investigate this problem stems from the results of Chapter 5, where we illustrate why an efficient test for membership in a subgroup G_q of prime order q of $\mathbb{Z}_p^*$ for some prime $p = \kappa q + 1$ is useful. Andy Neff pointed us to the efficient algorithm for computing the Jacobi symbol.

Damgård and Skjovbjerg Frandsen [14] have independently of us described algorithms similar to the algorithms in this chapter. However, their approach to the key problem is different from ours. Although both our algorithm and the algorithm of Damgård and Skjovbjerg Frandsen run in quadratic time in the input length, the constant factor of their algorithm seems smaller. Below we compare their approach to ours and show how both ideas may be combined.

This chapter is joint work with Anders Holst. The author of this thesis constructed the algorithms using the standard norm, but the special norms presented within were discovered during discussions with Anders Holst.

7.1 Definitions and Useful Results

We need a number of results on the rings $\mathbb{Z}[\omega]$ and $\mathbb{Z}[i]$. Most of these results are at least a hundred years old and can be found in several textbooks. We follow Chapters 1 and 9 of the classical text of Ireland and Rosen [37], and take the liberty to generalize the notion of "primary" and some of the results as outlined in the exercises of Chapter 9 of [37]. The only result we need that is not proven here or given as an exercise in [37] is Theorem 7.20, "Supplement to the Biquadratic Reciprocity Law". One possible source is an old paper in facsimile, Dintzl [17], online at ERAM [20].

It can be shown that both $\mathbb{Z}[\omega]$ and $\mathbb{Z}[i]$ are Euclidean domains, and that every Euclidean domain is a Principal Ideal Domain (PID). In PID:s the notions of irreducible and prime elements coincide. Thus the definitions below of prime (or irreducible) elements makes sense.

Definitions and Results on $\mathbb{Z}[\omega]$

Define the ring $D = \mathbb{Z}[\omega]$, where $\omega = -\frac{1}{2} + \frac{\sqrt{-3}}{2}$. The norm N is defined by $N\alpha = \alpha\overline{\alpha} = a^2 - ab + b^2$, for an element $\alpha = a + b\omega$ in D.

Proposition 7.1. *$\alpha \in D$ is a unit iff $N\alpha = 1$, and the units in D are 1, -1, ω, ω^2, $-\omega$, and $-\omega^2$.*

Definition 7.2. A non-unit $\alpha \in D$ is prime (or irreducible) if $\alpha \neq 0$ and $\alpha \mid \beta\gamma$ implies that $\alpha \mid \beta$ or $\alpha \mid \gamma$.

Proposition 7.3. *$1 - \omega$ is prime and $3 = -\omega^2(1-\omega)^2$.*

Definition 7.4 (Primary). $\alpha \in D$ is *primary* if $\alpha = 2 \pmod 3$.

Lemma 7.5. *Let $\alpha = a + b\omega \in D$. Then we have that $\omega\alpha = -b + (a-b)\omega$, $\omega^2\alpha = (b-a) - a\omega$, $-\alpha = -a - b\omega$, $-\omega\alpha = b + (b-a)\omega$, and $-\omega^2\alpha = (a-b) + a\omega$.*

Proof. Follows from the elementary properties of complex numbers.

Proposition 7.6. *Let $\alpha \in D$ be a non-unit, where $(1-\omega) \nmid \alpha$. Then there exists a unit γ such that $\gamma\alpha$ is primary.*

Proof. Suppose that the proposition is false and that $\alpha = a + b\omega \pmod 3$ is a counter-example. We must have $\alpha \neq 0 \pmod 3$ since $(1-\omega) \mid 3$ and $(1-\omega) \nmid \alpha$.

Furthermore, since $2 + \omega = (1-\omega)(1+\omega)$ and $1 + 2\omega = -1(2+\omega) \pmod 3$ we have $(1-\omega) \mid (2+\omega)$ and $(1-\omega) \mid (1+2\omega)$ which implies that we must have

$\alpha \neq 1+2\omega \pmod 3$ and $\alpha \neq 2+\omega \pmod 3$. We must have $b \neq 0 \pmod 3$ since otherwise $a \in \{1,2\} \pmod 3$, i.e. either α or $-\alpha$ would be primary. Similarly we must have $a \neq 0$ since otherwise $\omega^2\alpha \in \{1,2\} \pmod 3$, i.e either $\omega^2\alpha$ or $-\omega^2\alpha$ would be primary. Finally we must have $a \neq b \pmod 3$ since otherwise $\omega\alpha = a\omega - b(1+\omega) = -b \in \{1,2\} \pmod 3$, i.e. either $\omega\alpha$ or $-\omega\alpha$ would be primary.

Definition 7.7 (Cubic Residue Character). Let $\pi \in D$ be prime. If $N\pi \neq 3$ the *cubic residue character* of α modulo π is defined by

1. $\chi_\pi(\alpha) = 0$ if $\pi \mid \alpha$.
2. $\chi_\pi(\alpha) = \alpha^{(N\pi-1)/3} \pmod \pi$, with $\chi_\pi(\alpha)$ equal to 1, ω, or ω^2.

For a non-prime β the cubic residue character is defined $\chi_\beta(\alpha) = \prod_{l=1}^{t} \chi_{\pi_l}(\alpha)$, where $\beta = \pm \prod_{l=1}^{t} \pi_t$ is the prime decomposition of β.

Note that $\chi_\pi(\alpha)$ is not the residue class modulo π to which $\alpha^{(N\pi-1)/3}$ belongs. It is the unique representative of this residue class from the set of complex numbers $\{1, \omega, \omega^2\}$.

Proposition 7.8. *Let π be primary and $\alpha \in D$.*

1. *$\chi_\pi(\alpha) = 1$ iff $x^3 = \alpha \pmod \pi$ is solvable.*
2. *$\chi_\pi(\alpha\beta) = \chi_\pi(\alpha)\chi_\pi(\beta)$.*
3. *If $\alpha = \beta \pmod \pi$ then $\chi_\pi(\alpha) = \chi_\pi(\beta)$.*

Theorem 7.9 (The Law of Cubic Reciprocity). *Let $\alpha, \beta \in D$ be primary and relatively prime. Then $\chi_\alpha(\beta) = \chi_\beta(\alpha)$.*

Theorem 7.10 (Supplement to the Cubic Reciprocity Law). *Let $\alpha = (3m-1) + 3n\omega$ be primary. Then $\chi_\alpha(\omega) = \omega^{m+n}$ and $\chi_\alpha(1-\omega) = \omega^{2n}$.*

Definitions and Results on $\mathbb{Z}[i]$

Define the ring $E = \mathbb{Z}[i]$, where $i = \sqrt{-1}$. The norm N is defined by $N\alpha = \alpha\overline{\alpha} = a^2 + b^2$, for an element $\alpha = a + bi$ in E.

Proposition 7.11. *α is a unit iff $N\alpha = 1$, and the units in E are 1, -1, i, and $-i$.*

Definition 7.12. A non-unit $\alpha \in E$ is prime (or irreducible) if $\alpha \neq 0$ and $\alpha \mid \beta\gamma$ implies that $\alpha \mid \beta$ or $\alpha \mid \gamma$.

Lemma 7.13. *$1 + i$ is irreducible and $2 = -i(1+i)^2$.*

Definition 7.14 (Primary). $\alpha \in E$ is *primary* if $\alpha = 1 \pmod{(1+i)^3}$.

Lemma 7.15. *Let $\alpha = a + bi \in E$. Then we have that $i\alpha = -b + ai$, $-\alpha = -a - bi$, $-i\alpha = b - ai$.*

Proof. Follows from the elementary properties of complex numbers.

Lemma 7.16. *Let $\alpha \in E$ be a non-unit, $(1+i) \nmid \alpha$. then there is a unique unit γ such that $\gamma\alpha$ is primary.*

To distinguish between the cubic residue character and the quartic residue character we denote the latter by ψ instead of the usual generic notation χ.

Definition 7.17 (Quartic Residue Character). Let $\pi \in E$ be prime. If $N\pi \neq 3$ the *quartic residue character* of α modulo π is defined by

1. $\psi_\pi(\alpha) = 0$ if $\pi \mid \alpha$.
2. $\psi_\pi(\alpha) = \alpha^{(N\pi-1)/4} \pmod{\pi}$, with $\psi_\pi(\alpha)$ equal to -1, 1, i, or $-i$.

For a non-prime β the quartic residue character is defined $\psi_\beta(\alpha) = \prod_{l=1}^{t} \psi_{\pi_l}(\alpha)$, where $\beta = \prod_{l=1}^{t} \pi_t$ is the prime decomposition of β.

Note that $\psi_\pi(\alpha)$ is not the residue class modulo π to which $\alpha^{(N\pi-1)/4}$ belongs. It is the unique representative of this residue class from the set of complex numbers $\{\pm 1, \pm i\}$.

Proposition 7.18. *Let π be primary and $\alpha \in E$.*

1. *$\psi_\pi(\alpha) = 1$ iff $x^4 = \alpha \pmod{\pi}$ is solvable.*
2. *$\psi_\pi(\alpha\beta) = \psi_\pi(\alpha)\psi_\pi(\beta)$.*
3. *If $\alpha = \beta \pmod{\pi}$ then $\psi_\pi(\alpha) = \psi_\pi(\beta)$.*

Theorem 7.19 (The Law of Biquadratic Reciprocity). *Let $\alpha = a_0 + a_1 i$ and $\beta = b_0 + b_1 i$ be primary and relatively prime elements in E. Then $\psi_\alpha(\beta) = \psi_\beta(\alpha)(-1)^{(a_0-1)(b_0-1)/4}$.*

Theorem 7.20 (Supplement to the Biquadratic Reciprocity Law). *Let $\alpha = a + bi$ be primary. Then $\psi_\alpha(i) = i^{(a^2+b^2-1)/4}$ and $\psi_\alpha(1+i) = i^{(a-b^2-b-1)/4}$.*

7.2 The Algorithms

Shallit and Sorenson [62] presents a binary algorithm for computing quadratic residuosity based on the same idea as the binary gcd algorithm given by Stein [64]. A detailed description and analysis of the latter algorithm can be found in Knuth [43]. The algorithms we propose can be seen as a step by step translation of Shallit and Sorenson's algorithm to the cubic and quartic setting. Our goal is to find an algorithm with quadratic running time in the length of the inputs like the original.

The key observation used in the original algorithm translates in a natural way to the new setting. Originally, if a and b are positive odd integers we know that $2 \mid (a-b)$. In the new setting, if α and β are primary elements in D (or E) then $3 \mid (\alpha - \beta)$ (or $(1+i)^3 \mid (\alpha - \beta)$).

On the other hand there are also some differences listed below.

- In $\mathbb{Z}$, if we subtract a positive odd integer from a positive odd integer with larger norm (absolute value) and remove all factors of two, we end up with another *positive odd* number. This allows direct application of the quadratic reciprocity theorem.

  In D (or E), if we subtract a primary element from a primary element with larger norm and divide away all $(1-\omega)$:s (or $(1+i)$:s), we may not end up with a *primary* number. The problem with non-primary elements is that the cubic (or biquadratic) reciprocity theorem is only valid for *primary* elements. This problem is solved by application of Proposition 7.6 (or Proposition 7.16), which guarantees that we can find a primary associate to any element that is not divisible by $(1-\omega)$ (or $(1+i)$).

- In $\mathbb{Z}$, if we subtract a positive integer b from a larger positive integer a, we have that $|a-b| < \max\{|a|, |b|\}$, i.e. all recursive calls are made with smaller parameters than was given as input.

  In D (or E) this is not the case, i.e. if α and β are primary elements such that $N\alpha > N\beta$, then it is only guaranteed that $N(\alpha - \beta) \leq 4 \max\{N\alpha, N\beta\}$. The problem with this is that the norm of the parameters to recursive calls may grow, i.e. it is no longer obvious that the algorithm halts. This is solved by use of the key observation described above. It turns out that the divisibility property ensures that the new algorithms halt.

- The original algorithm uses the standard norm in $\mathbb{Z}$, i.e. the absolute value. This is very fast.

  In the new setting the standard norm for D and E can be used, but it results in relatively slow algorithms. The problem is that the standard norm is too computationally expensive, since it requires performing multiplications in $\mathbb{Z}$. Instead we introduce other measures of the size of elements that behave almost like the standard norm, but that can be computed in linear time.

- The original algorithm is devised for inputs in binary radix. This is good since division by 2 may be replaced by a left-shift.

  Binary radix works fine also for inputs from E, i.e. given an element $a+bi \in E$, we assume that a and b are given in binary radix. On the other hand, for an element $a + b\omega \in D$ it is more natural to represent a and b in ternary radix. If input is given in another radix we first convert it to ternary radix.

Descriptions of the Algorithms

In this section we give descriptions of the two algorithms using the standard norm. In Section 7.4 we replace the standard norm with a measure of the size of elements for which the computational cost is lower. We use a description format similar to the description of the algorithm for computing the Jacobi symbol given in [45]. The algorithm for the quartic character is slightly more complicated, since the biquadratic reciprocity law is more complicated than the cubic reciprocity law.

Algorithm 7.21 (Cubic Character).
CUBIC(α, β)
INPUT: $\alpha, \beta \in D$, where β is primary and $N\beta \neq 3$.
OUTPUT: The cubic character $\chi_\beta(\alpha)$ of α modulo β.

```
1. If $\alpha = 0$ return 0.
2. Write $\alpha = 3^e\alpha'$, where $3 \nmid \alpha'$, and set $s_1 = \chi_\beta(3^e)$.
3. Write $\alpha' = (1-\omega)^d\alpha''$, where $d \in \{0,1\}$, and set $s_2 = \chi_\beta(1-\omega)^d$.
4. If $\alpha''$ is a unit, return $s_1 \cdot s_2 \cdot \chi_\beta(\alpha'')$.
5. Find a unit $\gamma$ such that $\gamma\alpha''$ is primary, and set $s_3 = \chi_\beta(\gamma^{-1})$.
6. If $N\alpha'' > N\beta$, then set $\alpha''' = \gamma\alpha''$ and $\beta' = \beta$, and otherwise set $\alpha''' = \beta$ and
   $\beta' = \gamma\alpha''$.
7. Return $s_1 \cdot s_2 \cdot s_3 \cdot$ CUBIC$(\alpha''' - \beta', \beta')$.
```

Algorithm 7.22 (Quartic Character).
QUARTIC(α, β)
INPUT: $\alpha, \beta \in E$, where β is primary and $N\beta \neq 2$.
OUTPUT: The quartic character $\psi_\beta(\alpha)$ of α modulo β.

```
1. If $\alpha = 0$ return 0.
2. Write $\alpha = 2^e\alpha'$, where $2 \nmid \alpha'$, and set $s_1 = \psi_\beta(2^e)$.
3. Write $\alpha' = (1+i)^d\alpha''$, where $d \in \{0,1\}$, and set $s_2 = \psi_\beta(1+i)^d$.
4. If $\alpha''$ is a unit, return $s_1 \cdot s_2 \cdot \psi_\beta(\alpha'')$.
5. Find a unit $\gamma$ such that $\gamma\alpha''$ is primary, and set $s_3 = \psi_\beta(\gamma^{-1})$.
6. If $N\alpha'' > N\beta$, then set $\alpha''' = \gamma\alpha''$, $\beta' = \beta$ and $s_4 = 1$. Otherwise set $\alpha''' = \beta$,
   $\beta' = \gamma\alpha''$, and $s_4 = (-1)^{(a_0-1)(b_0-1)/4}$, where $\alpha''' = a_0 + a_1 i$ and $\beta' = b_0 + b_1 i$.
7. Return $s_1 \cdot s_2 \cdot s_3 \cdot s_4 \cdot$ QUARTIC$(\alpha''' - \beta', \beta')$.
```

As described above the algorithms are not really complete, since we have not described how to compute the s_i's or how to find γ such that $\gamma\alpha''$ is primary. In the proof of Proposition 7.32 below this is described in detail.

7.3 Correctness

Next we prove that if the algorithms halt they output the correct results. That this is the case is intuitively clear from Section 7.1 and Section 7.1. The reader may worry about the apparent illegal use of Theorem 7.9 (or Theorem 7.19) in Step 6 when the inputs are not relatively prime, but as is shown below this will not pose a problem.

Lemma 7.23. *If the algorithm* CUBIC *halts on input α and β in D it outputs $\chi_\beta(\alpha)$, the cubic character of α modulo β.*

Proof. The algorithm clearly outputs the correct result if $\alpha = 0$. Suppose now that $\alpha \neq 0$. After Step 3 the algorithm has defined α'' such that $\alpha = 3^e(1-\omega)^d\alpha''$, where $e \in \mathbb{N}$ and $d \in \{0, 1\}$.

From Proposition 7.8 we have $\chi_\beta(\alpha) = \chi_\beta(3)^e\chi_\beta(1-\omega)^d\chi_\beta(\alpha'')$, and from the definition of the algorithm we have $s_1s_2\chi_\beta(\alpha'') = \chi_\beta(3)^e\chi_\beta(1-\omega)^d\chi_\beta(\alpha'')$. The element α'' can not be a unit if α and β are not relatively prime. Thus the output is at least correct if α'' is a unit.

Now assume that α'' is not a unit. Since there by Theorem 7.6 exists a unit γ such that $\gamma\alpha''$ is primary and $1 = \chi_\beta(\gamma)\chi_\beta(\gamma^{-1})$ the output is correct if $s_3 \cdot \mathrm{CUBIC}(\alpha''' - \beta', \beta') = \chi_\beta(\alpha'')$.

There are two cases:

1. If $N\alpha'' > N\beta$ we have $\mathrm{CUBIC}(\alpha''' - \beta', \beta') = \mathrm{CUBIC}(\gamma\alpha'' - \beta, \beta)$. We also have $\chi_\beta(\gamma\alpha'') = \chi_\beta(\gamma\alpha'' - \beta)$ by Proposition 7.8. Thus the output is correct if $\mathrm{CUBIC}(\gamma\alpha'' - \beta, \beta) = \chi_\beta(\gamma\alpha'' - \beta)$.
2. If $N\alpha'' \leq N\beta$ then we have $\mathrm{CUBIC}(\alpha''' - \beta', \beta') = \mathrm{CUBIC}(\beta - \gamma\alpha'', \gamma\alpha'')$. There are two sub-cases:
   a) If $\gamma\alpha''$ and β are relatively prime then the application of Theorem 7.9 in Step 6 is valid, i.e. $\chi_\beta(\gamma\alpha'') = \chi_{\gamma\alpha''}(\beta)$. From Proposition 7.8 we also have $\chi_{\gamma\alpha''}(\beta) = \chi_{\gamma\alpha''}(\beta - \gamma\alpha'')$. Thus the output is correct if the output of $\mathrm{CUBIC}(\beta - \gamma\alpha'', \gamma\alpha'')$ is correct, i.e. equal to $\chi_{\gamma\alpha''}(\beta)$.
   b) If $\gamma\alpha''$ and β are not relatively prime then the output is correct if $\mathrm{CUBIC}(\beta - \gamma\alpha'', \gamma\alpha'') = 0$.
   In this case it may seem that the application of Theorem 7.9 is not valid. Fortunately $\beta - \gamma\alpha''$ and $\gamma\alpha''$ are not relatively prime, since $\gamma\alpha''$ and β are not relatively prime. This implies that the output is correct if the output of $\mathrm{CUBIC}(\beta - \gamma\alpha'', \gamma\alpha'')$ is correct, i.e. equal to 0. Thus it does not matter that we apply the reciprocity theorem for relatively non-prime elements!

Finally note that in each recursive call the second parameter β' is a primary element and hence a valid parameter.

To summarize, the output is correct if the recursive call $\mathrm{CUBIC}(\alpha''' - \beta', \beta')$ gives correct output. Thus if the algorithm halts it outputs the correct result.

Lemma 7.24. *If the algorithm* QUARTIC *halts on input α and β in E it outputs $\psi_\beta(\alpha)$, the quartic character of α modulo β.*

Proof. The proof is very similar to the proof of Lemma 7.23. For clarity we give a full proof.

The algorithm clearly outputs the correct result if $\alpha = 0$. Suppose now that $\alpha \neq 0$. After Step 3 the algorithm has defined α'' such that $\alpha = 2^e(1+i)^d\alpha''$, where $e \in \mathbb{N}$ and $d \in \{0,1\}$.

From Proposition 7.18 we have $\psi_\beta(\alpha) = \psi_\beta(2)^e\psi_\beta(1+i)^d\psi_\beta(\alpha'')$, and from the definition of the algorithm we have $s_1s_2\psi_\beta(\alpha'') = \psi_\beta(2)^e\psi_\beta(1+i)^d\psi_\beta(\alpha'')$. The element α'' can not be a unit if α and β are not relatively prime. Thus the output is at least correct if α'' is a unit.

Now assume that α'' is not a unit. Since there by Theorem 7.16 exists a unit γ such that $\gamma\alpha''$ is primary and $1 = \psi_\beta(\gamma)\psi_\beta(\gamma^{-1})$ the output is correct if $s_3 \cdot \mathrm{QUARTIC}(\alpha''' - \beta', \beta') = \psi_\beta(\alpha'')$.

There are two cases:

1. If $N\alpha'' > N\beta$ we have $\mathrm{QUARTIC}(\alpha''' - \beta', \beta') = \mathrm{QUARTIC}(\gamma\alpha'' - \beta, \beta)$. We also have $\psi_\beta(\gamma\alpha'') = \psi_\beta(\gamma\alpha'' - \beta)$ by Proposition 7.18. Thus the output is correct if $\mathrm{QUARTIC}(\gamma\alpha'' - \beta, \beta) = \psi_\beta(\gamma\alpha'' - \beta)$.

2. If $N\alpha'' \leq N\beta$ then we have $\mathrm{QUARTIC}(\alpha''' - \beta', \beta') = \mathrm{QUARTIC}(\beta - \gamma\alpha'', \gamma\alpha'')$. There are two sub-cases:

   a) If $\gamma\alpha''$ and β are relatively prime then the application of Theorem 7.19 in Step 6 is valid, i.e. $\psi_\beta(\gamma\alpha'') = (-1)^{(a_0-1)(b_0-1)/4}\psi_{\gamma\alpha''}(\beta)$, where $\beta = a_0 + a_1 i$ and $\gamma\alpha'' = b_0 + b_1 i$. From Proposition 7.18 we also have $\psi_{\gamma\alpha''}(\beta) = \psi_{\gamma\alpha''}(\beta - \gamma\alpha'')$. Since $s_4 = (-1)^{(a_0-1)(b_0-1)/4}$ the output is correct if the output of $\mathrm{QUARTIC}(\beta - \gamma\alpha'', \gamma\alpha'')$ is correct, i.e. equal to $\psi_{\gamma\alpha'}(\beta)$.

   b) If $\gamma\alpha''$ and β are not relatively prime then the output is correct if $\mathrm{QUARTIC}(\beta - \gamma\alpha'', \gamma\alpha'') = 0$.

      In this case it may seem that the application of Theorem 7.19 is not valid. Fortunately $\beta - \gamma\alpha''$ and $\gamma\alpha''$ are not relatively prime, since $\gamma\alpha''$ and β are not relatively prime. This implies that the output is correct if the output of $\mathrm{QUARTIC}(\beta - \gamma\alpha'', \gamma\alpha'')$ is correct, i.e. equal to 0.

Finally note that in each recursive call the second parameter β' is a primary element and hence a valid parameter.

To summarize, the output is correct if the recursive call $\mathrm{QUARTIC}(\alpha''' - \beta', \beta')$ gives correct output. Thus if the algorithm halts it outputs the correct result.

We have proved that if the algorithms halt the output result is correct, but we have not yet proved that the algorithms halt. The difference of two primary elements may be larger than its terms. This is illustrated in the next lemma. We must show that the size of the elements decreases despite this.

Lemma 7.25. *Let $\alpha, \beta \in D$ (or E) then $N(\alpha - \beta) \leq 4 \max\{N\alpha, N\beta\}$.*

Proof. The norms in D and E are really identical, but given in different coordinates, i.e. if $\alpha = a + b\omega$, then we may change coordinates and write $\alpha = A + Bi$, in which case $N\alpha = A^2 + B^2$.

Note that $N\alpha = |\alpha|^2$, where $|\alpha| = \sqrt{a^2 + b^2}$ is the standard complex modulus for $\alpha = a + bi$. The triangle inequality implies that $|\alpha - \beta| \leq 2 \max\{|\alpha|, |\beta|\}$. Squaring both sides gives the lemma.

Fortunately we know that the difference between two primary elements in D (or E) is divisible by 3 (or by $(1+i)^3$). The next lemma uses this fact and guarantees that the size of elements decreases despite the phenomenon described above.

Each call to CUBIC makes at most one recursive call. Given any pair of inputs α_0 and β_0, we may denote the parameters to the (only) recursive call made in CUBIC(α_0, β_0) by α_1 and β_1. Similarly we denote the parameters in the recursive call made in CUBIC(α_1, β_1) by α_2 and β_2, and so on. Thus any two parameters α_0 and β_0 gives rise to a sequence $S_{\text{CUBIC}}(\alpha_0, \beta_0) = ((\alpha_0, \beta_0), (\alpha_1, \beta_1), (\alpha_2, \beta_2), \ldots)$. We use the corresponding notation in the quartic case. We also use the notational convention introduced in the description of the algorithms to denote the variables in the i:th invocation of one of the algorithms, e.g. α_i'', γ_i etc.

We show that this sequence is never infinite, i.e. that the algorithms halt.

Lemma 7.26. *Given any valid inputs α_0 and β_0 to* CUBIC *and any (α_i, β_i) and $(\alpha_{i+2}, \beta_{i+2})$ in $S_{\text{CUBIC}}(\alpha_0, \beta_0)$ we have*

$$\max\{N(\gamma_{i+2}\alpha_{i+2}''), N\beta_{i+2}\} \leq \frac{4}{9} \max\{N(\gamma_i \alpha_i''), N\beta_i\} \ .$$

Given any valid inputs α_0 and β_0 to QUARTIC *and any (α_i, β_i) and $(\alpha_{i+2}, \beta_{i+2})$ in $S_{\text{QUARTIC}}(\alpha_0, \beta_0)$ we have*

$$\max\{N(\gamma_{i+2}\alpha_{i+2}''), N\beta_{i+2}\} \leq \frac{1}{2} \max\{N(\gamma_i \alpha_i''), N\beta_i\} \ .$$

Proof. From Lemma 7.25 we have that $N\alpha_{i+1} \leq 4 \max\{N(\gamma_i\alpha_i''), N\beta_i\}$. Furthermore we know that β_i is primary, and by construction $\gamma_i\alpha_i''$ is also primary.

In the cubic case this implies that $3 \mid \alpha_{i+1} = \alpha_i''' - \beta_i'$ and in the quartic case it implies that $(1+i)^3 \mid \alpha_{i+1} = \alpha_i''' - \beta_i'$.

We treat the cubic and quartic case jointly. In the cubic case we set $d = 9$ since $N(3) = 9$ and in the quartic case we set $d = 8$ since $N(1+i)^3 = 8$.

The observations above imply that $N(\gamma_{i+1}\alpha_{i+1}'') \leq \frac{1}{d} N\alpha_{i+1}$. This gives us $N(\gamma_{i+1}\alpha_{i+1}'') \leq \frac{4}{d} \max\{N(\gamma_i\alpha_i''), N\beta_i\}$ for all i. By construction we have $N\beta_{i+1} = \min\{N(\gamma_i\alpha_i''), N\beta_i\}$ for all i. Thus we have

$$\begin{aligned}
N(\gamma_{i+2}\alpha_{i+2}'') &\leq \frac{4}{d}\max\{N(\gamma_{i+1}\alpha_{i+1}''), N\beta_{i+1}\} \leq \frac{4}{d}\max\{N(\gamma_i\alpha_i''), N\beta_i\}, \ \text{and} \\
N\beta_{i+2} &= \min\{N(\gamma_{i+1}\alpha_{i+1}''), N\beta_{i+1}\} \leq N(\gamma_{i+1}\alpha_{i+1}'') \\
&\leq \frac{4}{d}\max\{N(\gamma_i\alpha_i''), N\beta_i\} \ ,
\end{aligned}$$

which concludes the proof.

In both algorithms the norm of the maximal parameter may decrease more, but the above is what is guaranteed in two successive recursive calls.

Proposition 7.27. *The algorithms* CUBIC *and* QUARTIC *are correct.*

Proof. From Lemma 7.23 and Lemma 7.24 we know that the algorithms give correct output if they halt.

Suppose that one of the algorithms does not halt on input α_0 and β_0. Then Lemma 7.26 implies that for large enough i, we can make $\max\{N\alpha_i, N\beta_i\}$ arbitrarily close to zero. On the other hand the norm of any element in D or E is a non-negative *integer*. Thus there must be an index such that $\max\{N\alpha_i, N\beta_i\} = 0$. $N\alpha_i = 0$ implies that $\alpha_i = 0$ and the definitions of the algorithms, i.e. Step 1, imply that they halt on inputs α_i and β_i when $\alpha_i = 0$.

At this point we have described algorithms and proved their correctness, but unfortunately they are relatively slow, i.e. they do not run in quadratic time in the size of the inputs like the Shallit-Sorenson algorithm for the Jacobi symbol. The problem is that it is too expensive to compute the norm in each step of the algorithm, since it requires computing multiplications in $\mathbb{Z}$, and it is not known how to multiply in linear time.

7.4 Alternative Measures of the Size of Elements

In this section we replace the computationally expensive norm N with a measure M of the "size" of numbers that is very easy to compute. The key idea is to realize that we do not need a norm over D (or E), but only something that behaves sufficiently close to a norm to serve the purposes of the algorithms. Note that the norm N, is in fact a $\mathbb{Q}(i)$-norm. When we think of the field $\mathbb{Q}(i)$ as a two dimensional vector space over $\mathbb{Q}$ we write $\mathbb{Q}^2$. It turns out that the measure M we are looking for, in both cases can be defined as a special $\mathbb{Q}^2$-norm.

The difference between a $\mathbb{Q}(i)$ norm and a $\mathbb{Q}^2$-norm is not that great. In fact the algorithms do not exploit all of the properties special to the $\mathbb{Q}(i)$-norm N. Careful examination shows that the following special properties are used.

1. N is invariant under multiplication by units. This ensures that α'' does not grow when multiplied with a unit γ in Step 5 of the respective algorithm.

2. We have $N(\alpha-\beta) \leq 4\max\{N\alpha, N\beta\}$ for both the cubic and quartic case, and $N(\alpha/3) \leq \frac{1}{9}N\alpha$ in the cubic case and $N(\alpha/(1+i)^3) \leq \frac{1}{8}N\alpha$ in the quartic case. In the cubic case this ensures that the elements get smaller in each recursive call.

3. For any $\alpha \in D$ we have $N\alpha \leq N((1-\omega)\alpha)$, and for any $\alpha \in E$ we have $N\alpha < N((1+i)\alpha)$. This ensures that the norm of an element can not grow in Step 3.

This means that any $\mathbb{Q}^2$-norm that satisfies the above additional special properties can replace the standard $\mathbb{Q}(i)$-norm N in the algorithms. Next we define such $\mathbb{Q}^2$-norms for the rings D and E, and show how these satisfies properties similar to the above.

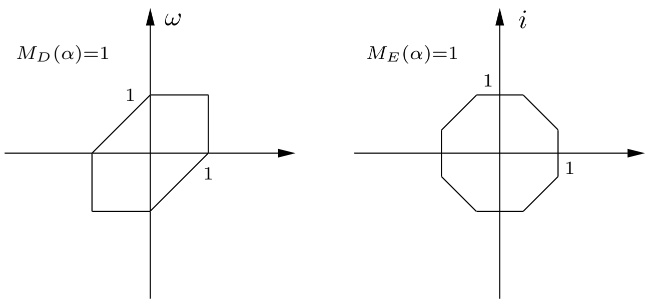

Figure 7.1: The left curve is the equidistance curve $M_D(\alpha) = 1$ in the coordinates 1 and ω. The right curve is the equidistance curve $M_E(\alpha) = 1$ in the coordinates 1 and i.

The Size of Elements in D.

The $\mathbb{Q}^2$-norm M_D we use to measure the size of elements in D instead of N is defined as follows for $\alpha = a + b\omega$.

Definition 7.28 (Simple D-Measure). $M_D\alpha = \max\{|a-b|, |a|, |b|\}$.

It is easy to see that the function M_D satisfies the inequalities $M_D(\alpha - \beta) \leq 2\max\{M_D\alpha, M_D\beta\}$ and $M_D(\alpha/3) \leq \frac{1}{3}M_D(\alpha)$. We also have have from Lemma 7.5 that M_D is invariant under multiplication by units. If $\alpha' = (1-\omega)\alpha$, where $\alpha = a+b\omega$, we have $\alpha' = (a+b)+(2b-a)\omega$. It follows that $M_D\alpha = \max\{|a-b|, |a|, |b|\}$, and $M_D\alpha' = \max\{|a+b|, |2a-b|, |2b-a|\}$. If a and b have opposite signs we have $M_D\alpha = |a-b| \leq |2a-b| \leq M_D\alpha'$. Otherwise we have $M_D\alpha = \max\{|a|, |b|\} \leq |a+b| \leq M_D\alpha'$. We have $M_D(\alpha) \leq M_D\alpha'$, which means that an element never grows in Step 3. Thus M_D satisfies all the required properties.

A more illuminating derivation of the property of invariance under multiplication by units follows. We may change coordinates and write $A+Bi = (a-\frac{1}{2}b)+\frac{\sqrt{3}}{2}i$. The norm N expressed in the new coordinates is A^2+B^2. Thus if $A'+B'i$ is an associate to α, then (A', B') satisfies the equation $x^2+y^2 = A^2+B^2$. Furthermore, multiplication by a unit in D corresponds either to mirroring over the y-axis, or rotation by a multiple of 120 degrees. Thus if we pick as equidistance curves the regular hexagon we get a $\mathbb{Q}^2$-norm that is invariant under multiplication by units. If we start with a regular hexagon in the new coordinates and change coordinates back to the original coordinates the hexagon is mapped to an equidistance curve of the $\mathbb{Q}^2$-norm M_D (see Figure 7.1 for an illustration of an equidistance curve of M_D).

We write CUBIC* for the algorithm CUBIC where each use of the norm N is replaced by M_D.

The Size of Elements in E.

The $\mathbb{Q}^2$-norm M_E we use to measure the size of elements in E instead of N is defined as follows for $\alpha = a + bi$.

Definition 7.29 (Simple E-Measure). $M_E\alpha = \max\{5|a-b|, 5|a+b|, 7|a|, 7|b|\}$.

It is easy to see that the function M_E satisfies the inequality $M_E(\alpha - \beta) \leq 2\max\{M_E\alpha, M_E\beta\}$ and from Lemma 7.15 we see that M_E is invariant under multiplication by units.

Since multiplication by the units $\{\pm 1, \pm i\}$ corresponds to rotation by 90 degrees or mirroring diagonally it may appear that we could have defined M_E to be the max-norm $M'_E\alpha = \max\{|a|, |b|\}$. Indeed this norm is invariant under multiplication by units in E, but it does not give us the inequality we need for the elements to shrink. Note that $M'_E((1+i)^3\alpha) = 2\max\{|a+b|, |a-b|\}$, which if $b = 0$ equals $2|a|$. The problem is that in the proof of Lemma 7.26 we need that $M_E\alpha < \frac{1}{2}M_E((1+i)^3\alpha)$.

Since $M_E(\alpha/2) = \frac{1}{2}M_E(\alpha)$ and $(1+i)^3 = 2i(1+i)$ it suffices to show that $M_E\alpha < M_E((1+i)\alpha)$.

We have that $M_E\alpha = \max\{5|a-b|, 5|a+b|, 7|a|, 7|b|\}$, and $M_E((1+i)\alpha) = \max\{10|a|, 10|b|, 7|a-b|, 7|a+b|\}$, from which it follows that $M_E\alpha \leq \frac{5}{7}M_E\alpha'$. Thus M_E satisfies all the required properties (see Figure 7.1 for an illustration of an equidistance curve of M_E).

We write QUARTIC* for the algorithm QUARTIC where each use of the norm N is replaced by M_E.

Consequences

The following corollary follows from the proof of Lemma 7.26 and the fact that both M_D and M_E satisfies the three special properties listed above.

Corollary 7.30. *Given any valid inputs α_0 and β_0 to* CUBIC* *and any (α_i, β_i) and $(\alpha_{i+2}, \beta_{i+2})$ in $S_{\text{CUBIC}^*}(\alpha_0, \beta_0)$ we have*

$$\max\{M_D(\gamma_{i+2}\alpha''_{i+2}), M_D\beta_{i+2}\} \leq \frac{2}{3}\max\{M_D(\gamma_i\alpha''_i), M_D\beta_i\} \ .$$

Given any valid inputs α_0 and β_0 to QUARTIC* *and any (α_i, β_i) and $(\alpha_{i+2}, \beta_{i+2})$ in $S_{\text{QUARTIC}^*}(\alpha_0, \beta_0)$ we have*

$$\max\{M_E(\gamma_{i+2}\alpha''_{i+2}), M_E\beta_{i+2}\} \leq \frac{5}{7}\max\{M_E(\gamma_i\alpha''_i), M_E\beta_i\} \ .$$

Note that the proof of Proposition 7.27 only assumes that the elements decrease by a constant factor in size. Thus we immediately have the following.

Corollary 7.31. *The algorithms* CUBIC* *and* QUARTIC* *are correct.*

Complexity

We now analyze the complexity of the algorithms CUBIC* and QUARTIC*. For simplicity we assume that the cost of adding, subtracting, shifting and comparing two integers of size s is given by $\log_2 s$.

Proposition 7.32. *The complexity of* CUBIC* *is* $O(35 \log_2 \max\{M_D\alpha, M_D\beta\})$ *and the complexity of* QUARTIC* *is* $O(88 \log_2 \max\{M_E\alpha, M_E\beta\})$. *Both algorithms run in quadratic time in the bit-size of the input.*

Proof. First we consider the amount of work performed by the algorithms in a recursive call with inputs α and β such that $m = \max\{M_D\alpha, M_D\beta\}$ (or $m = \max\{M_E\alpha, M_E\beta\}$).

In general all explicit computations of characters, i.e. s_1, s_2, s_3 and s_4, and also $\chi_\beta(\alpha'')$ is done using Theorem 7.10 (or Theorem 7.20). Thus these computations are performed in constant time.

Step 1 can clearly be done in constant time.

Step 2 is done by left shifting a and b for $\alpha = a + b\omega$ (or $\alpha = a + bi$).

Step 3 is done as follows. In the cubic case, if $(1-\omega) \mid \alpha'$ we have $3 \mid -\omega^2(1-\omega)\alpha'$, and $-\omega^2(1-\omega)\alpha' = (2a-b) + (a+b)\omega$ for $\alpha' = a + b\omega$. Thus checking if Step 3 should be performed can be done in constant time by checking if $2a - b = a + b = 0 \pmod 3$. The actual division is carried out by performing 2 additions and a subtraction. In the quartic case, if $(1+i) \mid \alpha'$ we have $2 \mid -i(1+i)\alpha'$, and $-i(1+i)\alpha' = (a+b) + (b-a)i$. Thus checking if Step 3 should be performed can be done in constant time by checking if $a + b = b - a = 0 \pmod 2$. The actual division is carried out by performing one addition and a subtraction. Heuristically one would expect that the additional division is carried out half of the invocations.

Step 4 can be done in constant time by explicit checking.

Step 5 is done as follows. In the cubic case Lemma 7.5 is applied to conclude that this step can be done by performing a single subtraction. Checking if this step should be carried out can be done in constant time similarly as Step 3. Heuristically one would expect that a subtraction is necessary half of the invocations. In the quartic case it follows from Lemma 7.15 that it can be performed in constant time.

Step 6 is done as follows. In the cubic case M_D can be computed using at most one subtraction and one comparison, since $|a-b| \geq \max\{|a|,|b|\}$ precisely when a and b have opposite signs. Thus at most two subtractions and three comparisons is necessary to determine if $M_D\alpha'' > M_D\beta$. Using similar tricks in the quartic case, and the facts that $5 = 2^2 + 1$ and $7 = 2^3 - 1$, M_E can be computed using at most 6 subtractions or additions and 3 comparisons. Thus at most 12 subtractions or additions and 7 comparisons are necessary to determine if $M_D\alpha'' > M_D\beta$.

Step 7 requires computing a single subtraction before executing the recursive call.

To summarize, CUBIC* performs at most 2 shifts, 7 additions and 3 comparisons of m-size integers, and QUARTIC* performs at most 2 shifts, 15 additions and 7 comparisons of m-size integers.

Next we determine the total complexity of the algorithms. We set $c = 12$ (or $c = 24$) and $d = 3/2$ (or $d = 7/5$), to consider both the cubic and quartic case simultaneously.

From Corollary 7.30 we conclude that there can be at most $2t$ recursive calls, where we define $t = \log_d m = \log_2 m / \log_2 d$. In every other recursive call the maximal norm of the inputs decreases by a factor of $1/d$. We sum the work performed in between every other recursive call and conclude that the total complexity is bounded by:

$$\begin{aligned}\sum_{l=0}^{t-1} O(2c\log_2\left((1/d)^l m\right)) &= O\left(2c\sum_{l=0}^{t-1}(\log_2 m - l\log_2 d)\right)\\ &= O(c\,(2/\log_2 d - \log_2 d)\log_2^2 m)\ .\end{aligned}$$

7.5 How to Combine the Ideas

Damgård and Skjovbjerg Frandsen [14] solves the norm problem differently than we do here. They show that the standard norm N may be approximated, within a factor 8/9. Their approximation can be computed using a single subtraction, i.e. it can be computed in linear time.

As shown above our special measures of the size of elements are also computed in linear time, but we need several additions and subtractions. On the other hand our decreasing factor is slightly lower than theirs, but it does not cancel the effect of having more additions and subtractions. Thus their algorithms are faster than ours. We think that the simplicity of our solution still has some merit, particularly in the cubic case.

Interestingly, we can also apply their idea to our specialized measures of the size of elements. This gives approximations of our specialized measures that can be computed in constant time.

The following is a variant of Lemma 2 in [14].

Lemma 7.33. *Given $\alpha \in D$ (or $\alpha \in E$) an approximation $\tilde{M}_D\alpha$ to $M_D\alpha$ (or $\tilde{M}_E\alpha$ to $M_E\alpha$) such that*

$$(1-\frac{1}{3^\delta})M_D\alpha < \tilde{M}_D\alpha \le M_D\alpha \quad \left(\textit{or } (1-\frac{1}{2^\delta})M_E\alpha < \tilde{M}_E\alpha \le M_E\alpha\right)$$

can be computed in time $O(\delta)$ (i.e. constant time in the input size).

Proof. In the cubic case we set $\alpha = a + b\omega$. For an integer $a = \sum_{j=0}^{k_a} a_j 3^j$ we define $t_\delta(a) = \sum_{j=k_a-\delta}^{k} a_j 3^j$, i.e. $t_\delta(a)$ is a but with precision truncated to $\delta + 1$ trits. If a and b have opposite signs, we define $\tilde{M}_D\alpha = |t_\delta(a) - t_\delta(b)|$, and otherwise we set $\tilde{M}_D\alpha = \max\{|t_\delta(a)|, |t_\delta(b)|\}$. In the first case we have that $M_D\alpha - \tilde{M}_D\alpha = |\sum_{j=0}^{k_a-\delta-1} a_j 3^j| + |\sum_{j=0}^{k_b-\delta-1} b_j 3^j| < 3^{\max\{k_a,k_b\}-\delta} \leq 3^{-\delta} M_D\alpha$. In the second case we have $M_D\alpha - \tilde{M}_D\alpha = |\sum_{j=0}^{k_a-\delta-1} a_j 3^j| < 3^{k_a-\delta} \leq 3^{-\delta} M_D\alpha$ (or similarly with a replaced by b).

In the quartic case we set $\alpha = a + bi$. For an integer $a = \sum_{j=0}^{k_a} a_j 2^j$ we define $t_\delta(a) = \sum_{j=k_a-\delta}^{k} a_j 2^j$, i.e. $t_\delta(a)$ is a but with precision truncated to $\delta + 1$ bits. We define $\tilde{M}_E\alpha = \max\{5(|t_{\delta+1}(a)| + |t_{\delta+1}(b)|), 7|t_{\delta+1}(a)|, 7|t_{\delta+1}(b)|\}$. Without loss we assume that $|a| > |b|$. If $5|a + b| \geq \max\{7|a|, 7|b|\}$ then we have $M_E\alpha - \tilde{M}_E\alpha < 5 \cdot 2^{\max\{k_a,k_b\}-\delta} \leq 2^{-\delta} M_E\alpha$. Otherwise we have $M_E\alpha - \tilde{M}_E\alpha < 7 \cdot 2^{k_a-\delta} \leq 2^{-\delta} M_E\alpha$.

The approximations are always lower bounds and the claims follow.

Denote by CUBIC** and QUARTIC** the algorithms that uses $\tilde{M}_D$ and $\tilde{M}_E$ instead of M_D and M_E respectively. For concreteness we set $\delta = 7$, but this value should be optimized experimentally in an actual implementation. Although our specialized norms M_D and M_E are computed in linear time the comparison in Step 6 is still the most expensive step in each invocation. The lemma above allows us to decrease this cost. The comparison can now be done in constant time, but we may have execute more recursive calls.

Proposition 7.34. *The algorithms* CUBIC** *and* QUARTIC** *are correct. The complexity of* CUBIC$^{*}*$ *is* $O(18 \log_2 \max\{M_D\alpha, M_D\beta\})$ *and the complexity of* QUARTIC$^{*}*$ *is* $O(19 \log_2 \max\{M_E\alpha, M_E\beta\})$.

Proof. The only change to Corollary 7.30 is that the guaranteed value of d in two successive calls is multiplied by a factor $(1 - \frac{1}{3^\delta})$ (or $(1 - \frac{1}{2^\delta})$). This implies that the algorithms halt, which by Lemma 7.23 and Lemma 7.24 implies that they are correct.

The analysis of the complexity is only changed in that Step 6 is done in constant time and the shrinkage factor is changed. Thus the claim follows.

The combined algorithms are faster by a constant factor than the original algorithms by Damgård and Skjovbjerg Frandsen [14], since their computation of the norm requires a subtraction whereas we compute the approximated special measures in constant time. However, this comparison is somewhat unfair since they do not optimize their algorithms aggressively, and it probably possible to compute a variant of their approximative norm in constant time.

7.6 On the Problem of Verifying Membership in G_q

Let G_q be the unique subgroup of prime order q of $\mathbb{Z}_p^*$, where $p = \kappa q + 1$ is prime for some relatively small integer κ. The problems of computing the κ-power character modulo p of an element and verifying membership in G_q are intimately connected. In the quadratic case the two problems are even equivalent. However, in the general case it does not suffice to determine the κ-power character modulo p to check for membership in G_q. It is not clear how the κ-power character for arbitrary κ can be exploited to determine membership in G_q quickly, but for $\kappa = 3, 4$ the relation is as follows [37].

Suppose that $\kappa = 3$. The ring $\mathbb{Z}[\omega]$ is a unique factorization domain, and p factors as $\pi\overline{\pi}$ for some irreducible $\pi \in \mathbb{Z}[\omega]$. Then $\chi_\pi(a) = 1$ if and only if $a \in G_q$.

Suppose that $\kappa = 4$. The ring $\mathbb{Z}[i]$ is a unique factorization domain, and p factors as $\pi\overline{\pi}$ for some irreducible $\pi \in \mathbb{Z}[i]$. Then $\psi_\pi(a) = 1$ if and only if $a \in G_q$.

Thus to determine if $a \in G_q$ for $\kappa \in \{2, 3, 4\}$ we need only factor p and run either the Shallit-Sorensen algorithm or our algorithms above. Furthermore, if $\kappa = 6$ or 12 the Chinese remainder theorem implies that $a \in G_q$ if and only if $(\frac{a}{p}) = 1$ and $\chi_\pi(a) = 1$, or $\chi_\pi(a) = 1$ and $\psi_\pi(a) = 1$ respectively.

For a discussion on how these elementary facts can be combined with precomputations to check membership in G_q for $\kappa \in \{2, 4, 6, 12\}$ efficiently we refer the reader to Damgård and Skjovbjerg Frandsen [14].

7.7 Future Work

Our algorithms are natural generalizations of Shallit and Sorenson's algorithm [62], which is based on the binary gcd algorithm given by Stein [64]. An interesting research problem is to characterize the values of l for which Shallit and Sorenson's algorithm can be generalized to compute l-power residue characters efficiently in the ring of integers D_l of the cyclotomic field $\mathbb{Q}(\zeta_l)$ for a primitive l:th root of unity ζ_l.

It is not trivial to generalize Shallit and Sorensons's idea to compute the l:th power character in D_l. One problem is that, in general, the key observation, i.e. that the difference between two primary elements is divisible by 3 (or $(1+i)^3$), does not generalize naturally. This follows from the general definition of primary. A non-unit element $\alpha \in D_l$ is primary if α is relatively prime to l, and $\alpha = n \mod (1 - \zeta_l)^2$ for some $n \in \mathbb{Z}$. There is also the problem of finding explicit reciprocity and supplemental laws that can be used efficiently.

Chapter 8

Conclusions

We present an efficient mix-net in Chapter 2, however, no proofs of the security of the construction are given. The protocol is based on the combination of the notion of "repetition robustness", introduced by Jakobsson [38], and "double encryption", introduced here. An important future research question is to prove the security of this protocol. To do this it seems necessary to develop non-malleability assumptions and investigate their soundness.

In Chapter 3 we prove that the security of a mix-server is equivalent to the security of the underlying cryptosystem. The correctness of this result is intuitively clear, but no proof has formerly been given in the literature. An important future question is to formulate a useful definition of security for a mix-net as a whole. We are currently working on this.

The attacks we present in Section 4 illustrate the importance of details in the construction of cryptographic protocols. Some of our attacks extend attacks of Pfitzmann [58, 57], Desmedt and Kurosawa [15], and Lim and Lee [44], but we also provide an attack that seems unrelated to any previous attacks. It is important that existing and future mix-net constructions, in particular hybrid mix-nets are verified to be resistant to our attacks.

In Chapter 5 we consider the malicious use of invalid inputs to protocols. Lim and Lee [44] give a key recovery attack based on the malicious use of invalid inputs and propose a counter-measure. We extend their attack and show that their proposed counter-measure does not suffice for all protocols. We also describe how costly verifications of inputs can be avoided.

The results of Chapter 6 give a class of transformations that can not be computed under El Gamal encryption. Hence we take a first step in the program of investigating the malleability properties of El Gamal. An interesting research question is to investigate how the malleability properties of the El Gamal cryptosystem changes depending on the underlying group.

The algorithms presented in Chapter 7 generalize the binary Jacobi symbol algorithm of Shallit and Sorenson [62]. An interesting future research question is if there are generalizations of these algorithms to compute higher power residue characters.

Bibliography

[1] M. Abe, *Universally Verifiable mix-net with Verification Work Independent of the Number of Mix-centers*, Eurocrypt '98, pp. 437-447, LNCS 1403, 1998.

[2] Abe, Imai, *Flaws in Some Robust Optimistic Mix-Nets*, Proc. 8th Australasian Conference on Information Security and Privacy (ACISP '03), pp. 39-50, LNCS 2727, 2003.

[3] J. Benaloh, M. Yung, *Distributing the power of a government to enhance the privacy of voters*, Proc. 5th ACM Symposium on Principles of Distributed Computing (PODC '86), pp. 52-62, 1986.

[4] J. Benaloh, D. Tuinstra, *Receipt-free secret-ballot elections*, Proc. 26th ACM Symposium on the Theory of Computing (STOC '94), pp. 544-553, 1994.

[5] M. Ben-Or, S. Goldwasser, and A. Wigderson, *Completeness theorems for non-cryptographic fault-tolerant distributed computation*, Proc. 20th ACM Symposium on the Theory of Computing (STOC '88), pp. 1-10, 1988.

[6] O. Baudron, P.A. Fouque, D. Pointcheval, G. Poupard and J. Stern, *Practical Multi-Candidate Election Scheme*, Proc. 20th ACM Symposium on Principles of Distributed Computing (PODC '01), pp. 274-283, 2001.

[7] Dan Boneh, *The Decision Diffie-Hellman Problem*, Proc. 3rd Algorithmic Number Theory Symposium, pp. 48-63, LNCS 1423, 1998.

[8] D. Chaum, *Untraceable Electronic Mail, Return Addresses and Digital Pseudonyms*, Communications of the ACM (CACM '81), Vol. 24, No. 2, pp. 84-88, 1981.

[9] D. Chaum, C. Créepeau, and I. Damgård, *Multiparty unconditionally secure protocols*, Proc. 20th ACM Symposium on the Theory of Computing (STOC '88), pp. 11-19, 1988.

[10] J. Cohen, M. Fischer, *A robust and verifiable cryptographically secure election scheme*, Proc. 28th IEEE Symposium on the Foundations of Computer Science (FOCS '85), pp. 372-382, 1985.

[11] R. Cramer, R. Gennaro, and B. Schoenmakers, *A secure and optimally efficient multi-authority election scheme*, Eurocrypt '97, pp. 103-118, LNCS 1233, 1997.

[12] R. Cramer, V. Shoup, *A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack*, Crypto '98, pp. 13-25, LNCS 1462, 1998.

[13] Ivan Damgard and Mads Jurik, *A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System*, Proc. Public Key Cryptography (PKC '01), pp. 119-136, LNCS 1992, 2001.

[14] I. Damgård, G. Skjovbjerg Frandsen, *Efficient Algorithms for gcd and Cubic Residuosity in the Ring of Eisenstein Integers*, BRICS Technical Report, ISSN 0909-0878, BRICS RS 03-8, 2003.

[15] Y. Desmedt, K. Kurosawa, *How to break a practical MIX and design a new one*, Eurocrypt 2000, pp. 557-572, LNCS 1807, 2000.

[16] W. Diffie and M. E. Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory, vol. 22, no. 6, pp. 644-654, 1976.

[17] E. Dintzl, *Über den zweiten Ergänzungssatz des biquadratischen Reciprocitätsgesetzes*, Monatshefte für Mathematik und Physik Vol. 10, pp. 88-96, 1899. (can be found at ERAM [20])

[18] D. Dolev, C. Dwork, M. Naor, *Non-Malleable Cryptography*, Proc. 23rd Symposium on Theory of Computing (STOC '91), pp. 542-552, 1991.

[19] T. El Gamal, *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, IEEE Transactions on Information Theory, Vol. 31, No. 4, pp. 469-472, 1985.

[20] Electronic Research Archive for Mathematics, `http://www.emis.de/MATH/JFM/JFM.html`.

[21] P. Feldman, *A practical scheme for non-interactive verifiable secret sharing*, Proc. 28th IEEE Symposium on the Foundations of Computer Science (FOCS '87), pages 427-438, 1987.

[22] FIPS 180-1, *Secure hash standard*, Federal Information Processing Standards Publication 180-1, U.S. Department of Commerce/N.I.S.T., National Technical Information Service, Springfield, Virginia, April 17, 1995.

[23] FIPS 180-1, *Secure hash standard*, Federal Information Processing Standards Publication 180-2, U.S. Department of Commerce/N.I.S.T., National Technical Information Service, Springfield, Virginia, August 1, 2002.

[24] A. Fujioka, T. Okamoto and K. Ohta, *A practical secret voting scheme for large scale elections*, Auscrypt '92, LNCS 718, pp. 244-251, 1992.

[25] J. Furukawa, K. Sako, *An efficient scheme for proving a shuffle*, Crypto 2001, LNCS 2139, pp. 368-387, 2001.

[26] J. Furukawa, H. Miyauchi, K. Mori, S. Obana, K. Sako, *An Implementation of a Universally Verifiable Electronic Voting Scheme based on Shuffling*, Proc. Financial Crypto (FC '02), LNCS 2357, pp. 16-30, 2003.

[27] J. Furukawa, *Personal communication*, email, 13-25 february, 2003.

[28] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, *Secure Distrubuted Key Generation for Discrete-Log Based Cryptosystems*, Eurocrypt '99, LNCS 1592, pp. 295-310, 1999.

[29] O. Goldreich, *Secure Multi-Party Computation*, Manuscript, `http://www.wisdom.weizmann.ac.il/~oded/pp.html`, 2002.

[30] O. Goldreich, S. Micali and A. Wigderson, *How to Play any Mental Game*, 19th Symposium on Theory of Computing (STOC '87), pp. 218-229, 1987.

[31] S. Goldwasser, S. Micali, *Probabilistic Encryption*, Journal of Computer and System Sciences (JCSS), Vol. 28, No. 2, pp. 270-299, 1984.

[32] S. Goldwasser, S. Micali, and C. Rackoff, The knowledge complexity of interactive proof systems, SIAM Journal on Computing, vol. 18, pp. 186-208, 1989.

[33] N. Groth, *A Verifiable Secret Shuffle of Homomorphic Encryptions*, Proc. Public Key Cryptography (PKC '03), pp. 145-160, LNCS 2567, 2003.

[34] Golle, Zhong, Boneh, Jakobsson, Juels, *Optimistic Mixing for Exit-Polls*, Asiacrypt 2002, LNCS, 2002.

[35] Golle, Zhong, Boneh, Jakobsson, Juels, *Private Communication*, 16 October 2002.

[36] M. Hirt, K. Sako, *Efficient Reciept-Free Voting Based on Homomorphic Encryption*, Eurocrypt 2000, LNCS 1807, pp. 539-556, 2000.

[37] K. Ireland, M. Rosen, *A Classical Introduction to Modern Number Theory*, 2nd edition 5:th printing, Springer-Verlag 1998, ISBN 0-387-97329-5.

[38] M. Jakobsson, *A Practical Mix*, Eurocrypt '98, LNCS 1403, pp. 448-461, 1998.

[39] M. Jakobsson, D. M'Raihi, *Mix-based Electronic Payments*, Proc. 5th Workshop on Selected Areas in Cryptography (SAC '98), LNCS 1556, pp. 157-173, 1998.

[40] M. Jakobsson, *Flash Mixing*, Proc. 18th Symposium on Principles of Distributed Computing (PODC '98), pp. 83-89, 1998.

[41] M. Jakobsson, A. Juels, *Millimix: Mixing in small batches*, DIMACS Techical report 99-33, June 1999.

[42] M. Jakobsson, A. Juels, *An optimally robust hybrid mix network*, Proc. 20th Symposium on Principles of Distributed Computing (PODC '01), pp. 284-292, 2001.

[43] D. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms* 3rd edition, Addison Wesley Longman 1997, ISBN 0-201-89684-2.

[44] C. H. Lim, P. J. Lee, *A Key Recovery Attack on Discrete Log-based Schemes Using a Prime Order Subgroup*, Crypto '97, LNCS 1294, pp. 249-263, 1997.

[45] A. Menezes, P. van Oorshot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, ISBN 0-8493-8523-7, 1997.

[46] R. Merkle, *Secure communication over insecure channels*, Communications of the ACM, vol. 21, no. 4, pp. 294-299, 1978.

[47] S. Micali, C. Rackoff, B. Sloan, *The notion of security for probabilistic cryptosystems*, SIAM Journal of Computing, Vol. 17, No. 2, pp. 412-426, 1988.

[48] M. Michels, P. Horster, *Some remarks on a reciept-free and universally verifiable Mix-type voting scheme*, Asiacrypt '96, LNCS 1163, pp. 125-132, 1996.

[49] M. Mitomo, K. Kurosawa, *Attack for Flash MIX*, Asiacrypt 2000, LNCS 1976, pp. 192-204, 2000.

[50] A. Neff, *A verifiable secret shuffle and its application to E-Voting*, Proc. 8th ACM Conference on Computer and Communications Security (CCS '01), pp. 116-125, 2001.

[51] V. Niemi, A. Renvall, *How to prevent buying of votes in computer elections*, Asiacrypt'94, LNCS 917, pp. 164-170, 1994.

[52] W. Ogata, K. Kurosawa, K. Sako, K. Takatani, *Fault Tolerant Anonymous Channel*, Information and Communications Security - ICICS '97, LNCS 1334, pp. 440-444, 1997.

[53] P. Paillier, *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, Eurocrypt '99, LNCS 1592, pp. 223-238, 1999.

[54] C. Park, K. Itoh, K. Kurosawa, *Efficient Anonymous Channel and All/Nothing Election Scheme*, Eurocrypt '93, LNCS 765, pp. 248-259, 1994.

[55] T. Pedersen, *A threshold cryptosystem without a trusted party*, Eurocrypt '91, LNCS 547, pp. 522-526, 1991.

[56] B. Pfitzmann, M. Waidner, *Composition and Integrity Preservation of Secure Reactive Systems*, Proc. 7th ACM Conference on Computer and Communications Security (CCS '00), pp. 245-254, 2000.

[57] B. Pfitzmann, *Breaking an Efficient Anonymous Channel*, Eurocrypt '94, LNCS 950, pp. 332-340, 1995.

[58] B. Pfitzmann, A. Pfitzmann, *How to break the direct RSA-implementation of mixes*, Eurocrypt '89, LNCS 434, pp. 373-381, 1990.

[59] K. Sako, J. Killian, *Reciept-free Mix-Type Voting Scheme*, Eurocrypt '95, LNCS 921, pp. 393-403, 1995.

[60] C. Schnorr, *Efficient signature generation by smart cards*, Journal of Cryptology, No. 4, pp. 161-174, 1991.

[61] C. Schnorr, M. Jakobsson, *Security of Signed El Gamal Encryption*, Asiacrypt 2000, LNCS 1976, pp. 73-89, 2000.

[62] J. Shallit, J. Sorenson, *A binary algorithm for the Jacobi symbol*, ACM SIGSAM Bulletin, 27 (1), pp. 4-11, 1993.

[63] S. Singh, The Code Book, Fourth Estate, London, 1999.

[64] J. Stein, *Computational problems associated with Racah algebra*, Journal of Computational Physics No. 1, pp. 397-405, 1969.

[65] Yiannis Tsiounis, Moti Yung, *On the Security of El Gamal based Encryption*, Proc. Public Key Cryptography (PKC '98), LNCS 1431, pp. 117-134, 1998.

[66] D. Wagner *A Generalized Birthday Problem*, Crypto 2002, LNCS 2442, pp. 288-304, 2002.

[67] D. Wikström, *The Security of a Mix-Center Based on a Semantically Secure Cryptosystem*, Indocrypt 2002, LNCS 2551, pp. 368-381, 2002.

[68] D. Wikström, *A Note on the Malleability of the El Gamal Cryptosystem*, Indocrypt 2002, LNCS 2551, pp. 176-184, 2002.

[69] D. Wikström, *Five Practical Attacks for "Optimistic Mixing for Exit-Polls"*, Proc. 10th Workshop on Selected Areas in Cryptography (SAC '03), to appear in the LNCS-series, 2004.

[70] D. Wikström, *An Efficient Mix-Net*, Swedish Institute of Computer Science (SICS) Technical Report T2002:21, ISSN 1100-3154, SICS-T-2002/21-SE, 2002, `http://www.sics.se`, (An implementation was demonstrated during the Conference of the Swedish Research Institute for Information Technology (SITI), feb 7, 2002).

[71] D. Wikström, *Elements in $\mathbb{Z}_p^*\backslash G_q$ are Dangerous*, Swedish Institute of Computer Science (SICS) Technical Report: T2003:05, ISSN: 1100-3154, ISRN: SICS-T-2003/05-SE, 27 february, 2003, `http://www.sics.se`.

[72] D. Wikström, A. Holst, *Algorithms for the Cubic and Quartic Residue Characters*, Manuscript, may 2003.

[73] A. C. Yao, *Protocols for secure computations*, Proc. 23rd IEEE Symposium on the Foundations of Computer Science (FOCS '82), pp. 160-164, IEEE Press, 1982.