Group Blind Digital Signatures: Theory and Applications

by

Zulfikar Amin Ramzan

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Electrical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1999
[June 1999]

Author ....
Department of Electrical Engineering and Computer Science
May 18th, 1999

Certified by.......................................
Ronald L. Rivest
Webster Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by ....

Chairman, Departmental Committee on Graduate Students

Group Blind Digital Signatures: Theory and Applications

by

Zulfikar Amin Ramzan

Submitted to the Department of Electrical Engineering and Computer Science
on May 18th, 1999, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Electrical Engineering

Abstract

In this thesis we introduce a new cryptographic construct called a Group Blind Digital Signature. This construct combines the already existing notions of a Group Digital Signature and a Blind Digital Signature. A group blind signature allows individual members of a possibly large group to digitally sign a message on behalf of the entire group in a cryptographically secure manner. In addition to being hard to forge, the resulting digital signatures are anonymous and unlinkable, and only a pre-specified group manager can determine the identity of the signer. Finally, the signatures have a blindness property, so if the signer later sees a message he has signed, he will not be able to determine when or for whom he signed it. Group Blind Digital Signatures are useful for various aspects of electronic commerce. In particular, through the use of such signatures we can design protocols for secure distributed electronic banking, and secure online voting with multiple voting centers. In this thesis we show, for the first time, how to construct such signatures based on number-theoretic assumptions. In addition, we discuss the novel implications of our construction to Electronic Cash and Electronic Voting scenarios. Many of the results in this thesis are joint work with Anna Lysyanskaya and a preliminary version of the results were published in the proceedings of *The International Conference on Financial Cryptography* [24].

Thesis Supervisor: Ronald L. Rivest
Title: Webster Professor of Electrical Engineering and Computer Science

Acknowledgments

First and foremost, I would like to thank my advisor Ron Rivest. Ron was an invaluable source of help when we were conducting the research that eventually evolved into this thesis. He helped us shape our ideas into a coherent body of work, and spent a considerable amount of time making sure that we had thought through our ideas thoroughly. This thesis has also benefited greatly from Ron's comments. He caught errors in previous versions of this thesis which I don't think I would ever have noticed even if I had reread the document thoroughly a hundred times. I take full responsibility, of course, for any errors that still crept into the final version. I would also like to acknowledge Anna Lysyanskaya. The main results in this thesis were published in the proceedings of *The International Conference on Financial Cryptography 1998* in an extended abstract written by Anna and myself. This thesis aims to provide a more thorough and lucid exposition of the ideas in the original paper. Additionally, I would like to thank the following people for helpful discussions and for providing me with useful references: Giuseppe Ateniese, Jan Camenisch, Jacque Traore, and Gene Tsudik. Finally, I would like to acknowledge the support of Darpa grant #DABT63-96-C-0018, and a National Science Foundation Graduate Fellowship.

Contents

Chapter 1

Introduction

This thesis introduces the notion of a *Group Blind Digital Signature* and provides a construction which enables us to realize this notion. A Group Blind Digital Signature enables members of a given group to digitally sign documents on behalf of an entire group in a manner that meets a number of necessary security requirements. This first chapter starts by giving the reader a high-level picture of the field of cryptography and explains where Group Blind Digital Signatures fit into this picture. In the next two sections, high-level descriptions of Group Signatures and Blind Signatures are given. The subsequent section gives a high-level overview of Group Blind Digital Signatures – such signatures combine the notions of Group Signatures and Blind Signatures. Finally, we conclude this first chapter by providing a summary of the mathematical notation used throughout this thesis as well as a road map of the remaining chapters.

1.1 High-Level Overview of Cryptography

The fundamental goal in the field of cryptography is to enable two parties to engage in some form of secure communications over a possibly insecure channel. Security could have many possible definitions. One important measure of security is to require that an eavesdropper listening in on a conversation between two parties should not be able to determine the contents of that conversation. This can be achieved by a cryptographic primitive known as *encryption*. The process of encryption applies a transformation to a message (also called the *cleartext* or *plaintext*) to get what is called the *ciphertext*. A sender can then encrypt his messages and send the corresponding ciphertext to the recipient. The ciphertext should be produced in such a manner that an eavesdropper will be incapable of determining the original plaintext from the ciphertext. Moreover, it should be possible for the intended recipient of the message to perform an inverse transformation, called *decryption*, to retrieve the original plaintext from the ciphertext. Both the encryption and decryption processes should be efficient and secure.

In the early days of cryptography, systems were designed with the notion of a *secret key*. Here, the two parties who wished to communicate would have to somehow pre-agree on a special key that would help determine the encryption and decryption process. This key would have to be agreed upon well in advance, and in a very secure manner. Additionally, this key would have to be kept absolutely secret, and would be known *only* by the two parties communicating. The two parties can hopefully use this secret key to perform secure communications. The process of trying to agree on a secret key itself can be rather difficult, so this raised the question of whether or not the processes of secure encryption and decryption could occur without the prior exchange of a secret key.

In their path-breaking paper, Diffie and Hellman [16] gave a solution to this problem. Their solution enabled two parties to securely agree on a secret key over a possibly insecure channel without requiring any form of prior communication between the communicating parties. The construction they gave was based on an a certain number-theoretic assumption (the intractability of computing the discrete logarithm); although no one has been able to break the security of their construction, no one has been able to prove that it is completely secure either.

In addition to the solution to this key-agreement problem, Diffie and Hellman outlined a methodology, called *Public-Key Cryptography*, which could be used for key agreement and had applications to other cryptographic problems. The idea in Public-Key Cryptography is to use two distinct keys: a *public key* for encryption, and a *private key* for decryption. The keys should be designed in such a way that determining the private key from the public key should be infeasible. Diffie and Hellman did not provide concrete constructions for how this concept of public-key cryptography could be implemented in practice. It was not until the fundamental work of Rivest, Shamir, and Adleman [33] that the first public-key cryptosystem was realized. Like the system of Diffie and Hellman, it was also based on a certain number-theoretic assumption (the intractability of computing $e - th$ roots modulo composites of a certain form).

The concept of public-key cryptography gave rise to a new and quite remarkable notion: *the digital signature.* The digital signature is the electronic analog of the traditional handwritten signature. The purpose of the digital signature is to enable a person to "digitally" sign some type of electronic document. One would like for these digital signatures to have the same properties as traditional signatures: they should be easy to produce, easy to check, and yet difficult to forge. By using the private key to sign, and the public key to verify, this notion was achieved. As time passed, several other realizations of public-key cryptosystems and digital signatures were proposed; see for example the papers of ElGamal and Schnorr [17, 36]. Once people understood these techniques, they tried to use them in designing more complex signature protocols which were geared toward more complex tasks. This thesis presents such a protocol: the Group Blind Digital Signature. This type of signature combines two notions which previously existed in the literature: the Group Digital Signature [3, 1, 2, 6, 8, 9, 14, 15] and the Blind Digital Signature [12, 13, 11, 21, 30]. These group blind digital signatures are useful for applications such as electronic cash and online voting. The central ideas in this thesis first appeared in a paper by Lysyanskaya and Ramzan [24].

We now give a high level description of blind digital signatures, group digital signatures, and group blind digital signatures. Thereafter, we give some of the basic preliminaries, assumptions, and notation used in this thesis, and we conclude this chapter with an outline of the rest of this thesis.

1.2 Blind Digital Signatures

The concept of a Blind Digital Signature was introduced by Chaum [12] to enable spender anonymity in Electronic Cash systems. Such signatures require that a signer be able to sign a document without knowing its contents. Moreover, should the signer ever see the document/signature pair, he should not be able to determine when or for whom he signed it (even though he can verify that the signature is indeed valid). This intuitively corresponds to signing a document with your eyes closed. If you happen to see the document and signature later on, you can indeed verify that the signature is yours, but you will probably have great difficulty in recollecting when or for whom you signed the original document. At first this concept seems a little strange – why would you want to sign something without seeing it? It turns out that, when applied properly, this notion has some very nice applications in situations where anonymity is a big issue. Two such applications are online voting and electronic cash. When you submit an online vote, you might like for that vote to be anonymous so no one can tell whom you voted for. Similarly with electronic cash, you might not want someone else to know who you are when you spend it. This is similar to normal paper cash – when you make a purchase, the vendor more or less has no idea who you are, but he can probably tell whether the money you gave him is legitimate. Specifically, in this electronic cash scenario, a document corresponds to an electronic coin or note, and the signer represents a bank. The spender retains anonymity in any transaction that involves electronic coins if they are blindly signed. We discuss blind digital signatures and their applications in more detail later in this thesis.

1.3 Group Digital Signatures

In a group digital signature scheme, members of a given group are allowed to digitally sign a document on behalf of the entire group. In addition, the signatures can be verified using a single *group public key*. Also, once a document is signed, no one, except a designated group manager, can determine which particular group member actually signed it. Companies can use group signatures to validate price lists, press releases, or digital contracts; customers would only need to know a single company public key to verify signatures. Companies are then capable of concealing their internal structure, while still being able to determine which employee signed a given document. Group Digital Signatures were first introduced and implemented by Chaum and van Heyst [14]. Some subsequent work on Group Signatures was done by Chen and Pedersen [15], Camenisch [6], Camenisch and Stadler [9], Ateniese and Tsudik [3].

1.4 Group Blind Digital Signatures

Group Blind Digital Signatures combine the properties of group signatures and blind signatures. They are useful in many of the settings where Blind Signatures are used. The first ever construction of a group blind digital signature scheme was given by Lysyanskaya and Ramzan [24], and the work therein is the central focus of this thesis. Some applications of group blind digital signatures include the design of electronic cash systems involving multiple banks, and online voting systems involving multiple voting servers. Such systems are more realistic than the standard single bank models and single voting server models.

1.5 Preliminaries, Assumptions, and Notation

In this section we describe some of the notation and assumptions used throughout this thesis.

1.5.1 Basic Notation

We employ the following basic notation:

- We use "," to denote the concatenation of two (binary) strings, or of binary representations of integers and group elements.

- For a finite set A, "$a \in_R A$" means that a is chosen uniformly at random from A.

- For an integer l, we let $\{0,1\}^l$ denote the set of all binary strings of length l.

- For an integer n, $\mathcal{Z}_n$ denotes the ring of integers modulo n.

- For an integer n, $\mathcal{Z}_n^*$ denotes the multiplicative group modulo n.

- For two integers n and m, we write $n|m$ if n is a divisor of m.

- For two integers n and m, we denote by (n,m) the greatest common divisor of n and m. If $(n,m) = 1$ then we say n and m are *relatively prime.*

- For an integer n, we denote by $\phi(n)$ the number of positive integers less than n that are relatively prime to n. For example, if $n = 10$, then $\phi(n) = 4$ since the numbers $1, 3, 7, 9$ are relatively prime to 10.

- For a group G, and an element $g \in G$, we denote by $\langle g \rangle$ the subgroup generated by g; i.e., $\{g^0, g^1, g^2, \ldots\}$. We write $G = \langle g \rangle$ if g generates the entire group G.

- For a binary string c we let $c[i]$ denote the $i - th$ most significant bit of c. For example, in the string 01011, $c[1] = 0$, $c[2] = 1$, $c[3] = 0$, $c[4] = 1$, and $c[5] = 1$.

1.5.2 Use of a Hash Function

We assume the existence of an ideal hash function $\mathcal{H}$. We assume this ideal hash function has the following properties:

- $\mathcal{H}$ is collision resistant. In other words, it is hard to find elements x, y where $x \neq y$ but $\mathcal{H}(x) = \mathcal{H}(y)$

- $\mathcal{H}$ hides all partial information about its input. In other words, if we are given a y such that $y = \mathcal{H}(x)$ then it is infeasible for us to gain any information about x other than that $y = \mathcal{H}(x)$.

Sometimes we write $\mathcal{H}_l(x)$ to denote the first l bits of $\mathcal{H}(x)$. When it is clear from context, we omit writing the l. In practice, one could replace $\mathcal{H}$ by an appropriately modified version of SHA-1 [28] or MD5 [32] which are believed to possess the types of properties mentioned above [35]. We require these assumptions to prove security of our scheme in the random oracle model [30, 5].

1.6 Organization of this Thesis

CHAPTER 2: We give a more detailed exposition on blind digital signatures. We first give a general overview and history, and then we detail two popular blind signature schemes. Subsequently, we discuss applications of blind signatures to electronic cash and online voting.

CHAPTER 3: This chapter focuses on group digital signatures. We discuss the formal definitions of group signatures, and specify the security requirements, the allowed operations, and the efficiency criteria. Next we discuss the relevant history and previous work on group signature schemes. Finally, we present a group signature scheme due to Camenisch and Stadler [9]. Their construction is important because our group blind digital signature scheme uses it as a building block. The original scheme of Camenisch and Stadler is vulnerable to a certain type of attack. We discuss how this attack works, and how to modify the scheme to make it more secure.

CHAPTER 4: We now begin discussing the notion of a group blind digital signature – though we do not give an actual construction just yet. We provide a formal model which consists of the security requirements of group blind digital signatures, as well as the procedures one can perform on them. This closely resembles the group signature model. Finally, we give detailed protocols for electronic cash, and online voting applications.

CHAPTER 5: This chapter introduces the notion of a blind signature of knowledge. Such signatures of knowledge are the fundamental building blocks of our group blind signature scheme. We give blind signatures of knowledge for three number-theoretic primitives which are used in our group blind digital signature scheme.

CHAPTER 6: Finally, we give our group blind digital signature scheme. The scheme modifies the group signature scheme of Camenisch and Stadler [9] by adding the blinding property. We conclude this chapter by analyzing the security and efficiency of our scheme.

Chapter 2

Blind Digital Signatures

In this chapter, we discuss the notion of a blind digital signature. This is a variant on the traditional digital signature, and it has applications in making digital cash and online voting secure and anonymous. The chapter is organized as follows. In the first section, we give a history of the traditional digital signature and the blind digital signature. We discuss the relevant security requirements as well. In the next section, we discuss Chaum's digital signature scheme [12], which is a modification of the RSA signature scheme [33]. The subsequent section discusses the Blind Schnorr Signature Scheme. This signature scheme is important because some of the ideas presented therein were used by Camenisch and Stadler in their Group Digital Signature Scheme [9], and subsequently by us in our Group Blind Digital Signature Scheme [24]. The next two sections discuss how to apply blind digital signatures to electronic cash and online voting respectively. The final section presents basic extensions to these original protocols which make them more secure.

2.1 Overview

Digital Signature Schemes enable people to electronically "sign" their documents in a secure and efficient manner. In other words, it is difficult to forge the signatures, yet verifying the validity of the digital signature is easy. The notion of a digital signature was originally defined in the path-breaking paper of Diffie and Hellman [16], and the first construction based on a number-theoretic assumption was given by Rivest, Shamir and Adleman [33]. The formal definitions of security for digital signatures were first outlined by Goldwasser, Micali, and Rivest [20]. They discussed the notion of an *existential adaptive chosen-message attack* which is the strongest form of possible attack one could imagine on a digital signature. Furthermore, they also presented a digital signature scheme that was secure against this type of attack; their scheme was based on the existence of Claw-Free permutations – which exist if factoring is hard. Subsequently, schemes with this level of security were constructed based on the existence of trapdoor permutations [4], one-way permutations [26], and finally on the existence of general one-way functions [34].

An interesting variant on the basic digital signature is the *blind digital signature.* The concept of a Blind Digital Signature was introduced by Chaum [12] to enable spender anonymity in Electronic Cash systems. Such signatures require that a signer be able to sign a document without knowing its contents. Moreover, should the signer ever see the document/signature pair, he should not be able to determine when or for whom he signed it (even though he can verify that the signature is indeed valid). Intuitively, this corresponds to signing a document with your eyes closed. In the electronic cash scenario, a document corresponds to an electronic coin, and the signer represents a bank. The spender retains anonymity in any transaction that involves electronic coins since they are blindly signed. We give more formal protocols later in this chapter. Blind Signatures have been looked at extensively in the literature [12, 11, 30, 21], and they are used in several electronic cash schemes.

A natural concern with cryptographic schemes is showing that they are secure. The two most common approaches seen in the literature are the complexity-based proof of security [4, 16, 20, 21, 26, 34], and the random-oracle-model based proofs of security [5, 18, 30]. We elaborate on these two notions:

1. **Complexity-Based Proofs:** The complexity based proof approach was first used in the seminal paper of Diffie and Hellman [16]. The idea is to start by making a well-defined hardness assumption; for example, the intractability of factoring, or determining a discrete logarithm modulo a prime. Then, you can

show security of a cryptographic primitive by showing that if you can successfully attack the security of this primitive, then this attack can be converted into a method for violating your hardness assumption. In other words, you want to be able to make statements of the form: "If anyone can break my cryptosystem, then their attack can be converted into a technique for factoring efficiently." Perhaps the most general hardness assumption one can make is that one-way functions exist. Intuitively, this says that if there are *any* intractable problems of a certain very general form, then they can be used as the basis of a cryptosystem I am designing.

2. **Random Oracle Model Based Proofs:** In many cases when it is difficult to attain a complexity-theoretic based proof of security, one opts for a random oracle model based proof. This approach was used in several papers [5, 18, 30]. The idea here is to assume that some cryptographic primitive such as SHA [27, 28] behaves like a random function. Then, you should prove that your system is secure under this assumption. This form of proof is acceptable, but much less preferable to the complexity-based approach. The Group Blind Digital Signature Scheme we will later present is also proven secure under this model.

The idea of proving security of a blind digital signature scheme, in a strict complexity-theoretic sense, was studied in a paper by Juels, Luby, and Ostrovsky [21]. They formalized a complexity-theoretic notion of security for Blind Digital Signatures, and provided a general technique for constructing secure Blind Digital Signatures from any one-way trapdoor permutation. This result was fundamentally of great importance, but unfortunately the construction is very complex, and therefore is not suitable for practical use.

Pointcheval and Stern [30] presented blind variants of various digital signature schemes. The signature schemes they addressed included those of Okamoto [29] and Schnorr [36]. The proofs of security in these schemes required various number-theoretic conjectures and were given in the Random Oracle Model.

Having presented some of the history of blind digital signatures, we now move on to discuss two well-known blind signature schemes. Later we show how to use these schemes in electronic cash applications.

2.2 Chaum's Blind Digital Signature Scheme

We present a Blind Digital Signature Scheme due to Chaum [12]. It is the first blind signature scheme to have been presented in the cryptographic literature. The scheme is based on the RSA signature scheme [33], which we now present.

2.2.1 The Original RSA Signature Scheme

Let $n = pq$ where p and q are two large primes and let e be chosen such that $(e, \phi(n)) = 1$. Moreover, let d be such that $de = 1 \bmod \phi(n)$. We assume that the signer's public key is (n, e), and the private key is (p, q, d). It is proven that obtaining a private key from the public key is very difficult unless you know the factorization of n [33]. Finally let $\mathcal{H}$ be a collision-resistant hash function.

Definition 1 *For a given message m, a pair (m, s) is a valid RSA signature if:*

$$s^e = \mathcal{H}(m) \pmod n$$

The signature pair can be computed easily by a signer who knows the message and secret key (p, q, d) as follows:

$$s = \mathcal{H}(m)^d \pmod n$$

It is easy to verify that (m, s) is valid by checking if the following equality holds:

$$s^e = \mathcal{H}(m) \pmod n$$

where equality follows because $de = 1 \pmod{\phi(n)}$. The hash function $\mathcal{H}$ is used to enhance security and efficiency. Having examined the original RSA scheme, we now show how to convert this scheme into a blind signature scheme. This construction is due to Chaum [12].

2.2.2 Blinding the RSA Signature Scheme

In this context, we suppose that Bob requires Alice's signature on some document. Moreover, Bob wants it to be the case that Alice does not know the contents of this document; at first this might seem a little strange, but we later show how this construct can be used in electronic cash and online voting systems. Bob and Alice now engage in the following protocol.

Blind Signature Protocol for Message M

Bob Round 1

1. Bob wants a message M to be blindly signed by Alice. He informs Alice of this.
2. Bob picks $r \in_R \mathcal{Z}_n^*$ and computes the following "blinded" message $\hat{M}$:

   $$\hat{M} = \mathcal{H}(M) \cdot r^e \pmod{n}$$

   where n and e are taken from Alice's public key.
3. Bob sends $\hat{M}$ to Alice.

Alice Round 2

1. Alice takes $\hat{M}$ and computes the following signature on it:

   $$\sigma(\hat{M}) = \hat{M}^d \pmod{n}$$

   Observe that

   $$\hat{M}^d \pmod{n} = (\mathcal{H}(\mathcal{M})r^e)^d \pmod{n} = \mathcal{H}(\mathcal{M})^d \cdot r \pmod{n}$$

2. Alice sends $\sigma(\hat{M})$ to Bob.

Bob Round 3

1. Bob takes the signature $\sigma(\hat{M})$ given by Alice on the blinded message $\hat{M}$ and "extracts" an appropriate signature for M:

$$\sigma(M) = \sigma(\hat{M})/r \pmod{n}.$$

2. The pair $(M, \sigma(M))$ now represents a valid message / signature pair under Alice's public key. This follows because as we observed earlier,

$$\sigma(\hat{M}) = \mathcal{H}(M)^d \cdot r \pmod{n}.$$

The most important thing to observe about this protocol is that Alice has issued the signature without ever seeing the actual message. This happens because the *blinding factor* r^e was multiplied to the message, and as a result the final message just looked like a random element of $\mathcal{Z}_n^*$ to Alice. Later, after the signature is issued, and Bob divides out by r (to "remove" the blinding factor), the resulting message / signature pair is unrecognizable to Alice. In fact, if Alice later sees this message, she can easily verify that the signature is indeed hers, but she will be severely limited in accurately determining when for whom she signed the message. At best she may be able to make a random guess from among the signatures she has issued, but she cannot do any better.

2.3 Blind Schnorr Digital Signature Scheme

We give another blind signature scheme based on the scheme of Schnorr [36]. This scheme is based on the intractability of the discrete logarithm problem, and is secure in the random oracle model. We start by giving the original Schnorr Signature Scheme, and then we show how to blind it. The blinded scheme first appeared in a paper by Okamoto [29].

We let G be a subgroup of $\mathcal{Z}_n^*$ of order q, for some value n and some prime q. We choose a generator $g \in G$ that makes computing discrete logarithms in G difficult. We let $x \neq 0$ be the secret key of the signer, and $y = g^x$ be his public key. Finally, let $\mathcal{H}$ be a collision-resistant hash function whose domain is $\{0,1\}^*$ and whose range is $\mathcal{Z}_q$.

Definition 2 *For a message $m \in \{0,1\}^*$ a pair (c, s) is said to be a valid Schnorr signature on m if it satisfies the following verification equation:*

$$c = \mathcal{H}(m, g^s y^c)$$

where $(m, g^s y^c)$ refers to the concatenation of m and $g^s y^c$.

Observe that the value c occurs on both the left hand and right hand sides of the equation. Given that $\mathcal{H}$ is collision resistant, it appears to be quite difficult to construct such a valid signature. It turns out that one can do it if he happens to know the discrete logarithm of y to the base g. A valid Schnorr signature (c, s) on a message m can be generated by a signer (who knows x) as follows:

1. Choose $r \in_R \mathcal{Z}_q$.
2. Let $c = \mathcal{H}(m, g^r)$.
3. Then, choosing $s = r - cx \pmod q$ creates a valid Schnorr signature.

This works because:

$$g^s y^c = g^{r-cx}(g^x)^c = g^r \pmod n$$

Hence, $\mathcal{H}(m, g^s y^c) = \mathcal{H}(m, g^r) = c$. It turns out that the Schnorr signature scheme can be made blind.

2.3.1 Blinding the Original Schnorr Signature Protocol

We present a protocol for obtaining blind Schnorr signatures. This protocol was originally given in a slightly different format by Okamoto [29]. The protocol is more complicated than the blind RSA protocol, and it requires more rounds of interaction between the signer and the recipient. The protocol is as follows.

Blind Schnorr Signature Protocol

(Signer's secret key is x, and his public key is $y = g^x \pmod{n}$)

(The recipient wants to have message m blindly signed.)

Signer Round 1:

1. Pick $\hat{r} \in_R \mathcal{Z}_q$
2. Set $\hat{t} = g^{\hat{r}} \pmod{n}$ and send $\hat{t}$ to the recipient.

Recipient Round 2:

1. Pick $\gamma, \delta \in_R \mathcal{Z}_q$
2. Set $t = \hat{t} g^{\gamma} y^{\delta} \pmod{n}$
3. Set $c = \mathcal{H}(m, t)$
4. Set $\hat{c} = c - \delta \pmod{q}$ and send it to the signer.

Signer Round 3:

1. Set $\hat{s} = \hat{r} - \hat{c}x \pmod{q}$ and send it to the recipient.

Recipient Round 4:

1. Set $s = \hat{s} + \gamma \pmod{q}$.

The signature is now (c, s). It is not hard to see why this signature is blind. The signer never gets to see any information about either c or s because these values are *blinded* by the random blinding factors δ and γ respectively. Furthermore, the signature is valid:

$$g^s y^c = g^{\hat{s}+\gamma} y^{\hat{c}+\delta} = g^{\hat{r}-\hat{c}x+\gamma+\hat{c}x} y^{\delta} = \hat{t} g^{\gamma} y^{\delta} = t \pmod{n}$$

which means $c = \mathcal{H}(m, t) = \mathcal{H}(m, g^s y^c)$.

2.4 Applications to Digital Cash

We now outline how to apply the blind digital signature idea to digital cash. We give a basic protocol due to Chaum [12] for a digital cash transaction. We describe a very simple version of the protocol primarily to give the main ideas. As a result, there are some minor flaws in it, but standard techniques exist to remedy these flaws. There are several more comprehensive works on electronic cash which discuss this, as well as other well-known electronic cash schemes [23, 38]. In the protocol we present, we assume that there is a person Alice who wants to buy a cryptography textbook, which costs $50 from the fictitious online vendor Online Crypto Books. Furthermore, Alice and Online Crypto Books both use the same bank, which we call the Bank. The entire transaction protocol is broken up into three stages: Withdrawal, Spending, and Deposit.

Basic Digital Cash Transaction Protocol

Withdrawal

1. Alice creates a piece of digital currency C. This currency consists of a string of bits which specify certain information such as a large random "serial number" and the dollar amount (which is $50 in this case).

2. Alice now takes C, and gets the Bank to blindly sign it; in other words, Alice and the Bank engage in a blind signature protocol at the end of which Alice possesses a valid signature on the currency.

3. Upon successful completion of the protocol, the Bank deducts $50 from Alice's bank account.

Spending

1. Alice requests a copy of the cryptography textbook she wants from Online Crypto Books. Alice takes this valid $50 currency C, along with the bank's signature on that currency, and gives it to Online Crypto Books.

2. Online Crypto Books checks that C is valid by verifying the signature on it with the Bank's public key. If the signature is invalid, Online Crypto Books immediately ends the protocol.

Deposit

1. Online Crypto Books now takes the currency C, and the Bank's signature on C, and gives it to the Bank.

2. The Bank verifies that the signature on the currency is indeed valid; i.e. that the currency was indeed signed by the Bank. If it is valid, the bank needs to check that the coin has not yet been spent. We outline a method for performing this check later in this chapter. If the signature on the coin is valid and the currency has not yet been spent, then the Bank credits Online Crypto Books's bank account by \$50.

3. If all checks out, then Online Crypto Books gives Alice the cryptography textbook she requested.

Thus we have completed a transaction. Observe that Alice's identity remains anonymous to both the Bank and to Online Crypto Books. This happens because a blind signature was issued on the currency. Therefore, once Alice has the signed currency, the vendor certainly will not be able to tell that Alice is the owner of the currency. Moreover, when the vendor gives the money to the Bank for deposit, the Bank will not be able to link the currency it just received to Alice because it blindly signed the message, and therefore it had no idea what the contents were.

There are a few potential security breaches in this system. For example, what is to prevent Alice from giving her bank a message of the form: "I am signing over 10 Billion Dollars to Alice?" Also, since electronic money is easy to copy (it is just a string of bits!) what is to prevent Alice from using the same piece of Digital Currency later? One dangerous aspect of having anonymous electronic cash is that once a transaction is complete, there is absolutely no way of tracking a particular customer since her identity is completely anonymous. Fortunately, there are some standard techniques for addressing these issues. We discuss those techniques in a later section.

2.5 Applications to Online Voting

We give an application of the traditional blind signature protocol to electronic voting. We now suppose that we have two entities: A voter Alice, and a Central Tabulating Facility (CTF). We assume that the votes are of the form "Yes" or "No." The protocol we give is adapted from the one presented by Schneier [35]. The voting protocol is divided into two stages: registration and voting.

Online Voting Protocol

Registration

1. Alice creates two electronic ballots B_1 and B_2 – these consists of a serial number, and some other relevant information to voting. In addition, B_1 contains the vote "Yes" and B_2 contains the vote "No."

2. Alice then takes the ballots B_1, B_2 and blinds them. It sends the blinded versions to the CTF.

3. The CTF checks its database to make sure that Alice has not voted before. If this is the case, then it signs the blinded ballots and it gives them back to Alice.

Voting

1. Alice unblinds the messages, and now has two sets of valid votes signed by the CTF.

2. Alice picks which vote ("Yes" or "No") she wants to choose, along with the CTF's signature on that vote, and encrypts it with the CTF's public key.

3. Alice sends in her vote.

4. The CTF decrypts the votes, checks that the signature is valid, and checks its database to make sure that the serial number on the vote has not been used before (this prevents Alice from trying to vote more than once). If this checks out then the CTF tabulates the vote, saves the serial number and records it in the database. At the end of the election, the CTF publishes the results of the election, as well as every serial number and its associated vote.

2.6 Some Extensions to the Basic Protocols

We address the problems relevant to the digital cash scheme and give standard techniques for making our protocols more secure. These techniques directly apply to the voting scheme as well. Perhaps the foremost concern is how one can prevent Alice from giving the bank a completely fraudulent document to sign. For example, what is to prevent Alice from giving the bank the message "Please give Alice $100 Million Dollars?" After all, since the bank applies a *blind signature* it has no way of knowing what the document contents are. There are two standard solutions to this problem:

1. The Bank can use different public keys for different dollar amounts. Hence if Alice wants a $100 piece of currency, then she must get the bank to sign it with a public key that corresponds to this amount. As a result, Alice will not be able to trick anyone into believing that a coin is worth any more than it actually is.

2. Alice and the Bank can execute a *cut and choose* protocol. Here, Alice first prepares some number, say 9, coins $C_1, \ldots, C_9$. Each coin should be identical, the only difference between them should be their serial numbers. Then, Alice blinds each of these coins, and sends them to the Bank. The bank picks all but one of the coins, and tells Alice to reveal the *blinding factor* for those coins. That is, it asks Alice for the appropriate information so it can unblind these coins. After Alice reveals the blinding factors for these eight coins, the bank unblinds the coins, and examines them. If they are all of the correct form, then the Bank applies a blind signature to the remaining coin, and sends it to Alice. The bank has high assurance that the remaining coin is of the proper form – because if Alice included even a single improper coin among the first nine, then she would get caught with probability at least 8/9.

The next major security issue to address is how one can keep Alice from possibly *double spending* her electronic currency C. After all, C is just a string of bits, and can easily be duplicated. There are two issues at play here: preventing Alice from double spending a coin and being able detect if Alice is double spending a coin. We discuss the issue of detection. Hopefully if the penalty is high and it is easy to detect double spending, Alice may be discouraged from trying to engage in this type of activity. There is a standard solution to detecting a double spent coin. We discuss this solution here, and refer the reader to Wayner's text on electronic cash [38] for a more thorough discussion on other solutions to this problem.

We can have the Bank maintain some type of universal list of already spent coins – perhaps this list can contain all the serial numbers. Then during every transaction, after receiving a coin from the vendor, the Bank checks to see if the coin has been spent already. If it has, then the Bank informs the vendor of this, and the vendor refuses to give the merchandise to the customer Alice. This scheme is accurate, but expensive since for every single transaction, the Vendor must go to the bank and have it verify the coin before giving the merchandise to the user – the vendor must do this because if Alice double spends, and does not get caught immediately, then she will never be caught since her identity is completely anonymous. This is costly. An additional drawback is that this scheme presumes that the vendor knows who it is dealing with. If Alice is caught double-spending, but is communicating with the vendor through an anonymous channel, then she escapes. Finally, another disadvantage of the scheme is that it is conceivable that Alice could spend her coin C, and then respend it fast enough before the list is updated. Such a security breach has to be prevented.

Chapter 3

Group Digital Signatures

3.1 Introduction

The Group Digital Signature extends the traditional Digital Signature concept to a multi-party setting. In a group digital signature scheme, members of a given group are allowed to sign on behalf of the entire group. In addition, the signatures can be verified using a single *group public key*. Also, once a document is signed, no one, except for a designated group manager, should be able to determine which particular group member actually signed it. Group signatures should also be designed so that no group member can forge another member's signature on a given document.

For example, suppose you are the CEO of a company and you want some of your individual employees to validate price lists, press releases, or digital contracts on behalf of the entire company. In this case, you can set up a group signature scheme, and act as the *group manager*. Then your employees can sign or validate various documents on behalf of the entire company. By using this approach, you will conceal your company's internal structure, and your customers would only have to know a single company public key to verify the signatures on your documents. Moreover, only you can determine which employee signed which document. Additionally, the signatures would satisfy various security requirements of interest. Among other things, the signatures would be unforgeable, and it would be next to impossible for one of your employees to "fake" the signature of another one of your employees. In fact, it would also be impossible for the group manager to fake the signatures of the individual group members. This is just one simple application of group digital signatures. The group blind digital signature scheme given in this thesis sheds light on new applications of group digital signatures to electronic commerce protocols.

This chapter is organized as follows: In the next three sections, we discuss the formal specifications of group signature schemes. This consists of the possible procedures allowed by group signatures, as well as the necessary security requirements of group signatures. In the fourth section, we discuss what it means for a group signature to be efficient. The following section contains a summary of the previous work done on group signatures. In section six, we present the original version of a group signature scheme due to Camenisch and Stadler [9]. The final section gives a "quasi-attack" on the original scheme; this attack is mainly of theoretical importance since the scheme can easily be modified to prevent it. We close this section by giving a variant on the original Camenisch and Stadler scheme that is more secure.

3.2 Procedures Allowed by Group Digital Signatures

A group digital signature scheme consists of the following five procedures:

`Setup` a probabilistic algorithm that generates the group's public key $\mathcal{Y}$ and a secret administration key $\mathcal{S}$ for the group manager.

`Join` an interactive protocol between the group manager and the new group member Bob that produces Bob's secret key x, and his membership certificate A.

`Sign` an interactive protocol between group member Bob and an external user Alice which takes as input a message m from Alice, and a secret key x from Bob, and produces a signature s on m.

`Verify` an algorithm which on input $(m, s, \mathcal{Y})$, determines if s is a valid signature for the message m with respect to the group public key $\mathcal{Y}$.

`Open` an algorithm which, on input $(m, s, \mathcal{S})$ determines the identity of the group member who issued the signature s on the message m.

3.3 Security Requirements for Group Digital Signatures

A Group Digital Signature must satisfy the following security requirements.

1. **Unforgeability:** Only group members can issue valid signatures on behalf of the entire group; i.e. only group members can issue signatures that are verifiable by the group public key.

2. **Conditional Signer Anonymity:** Anyone can easily check that a message / signature pair was signed by some group member, but *only* the group manager can determine which member issued the signature. There is also a slightly different model which one might want to consider in which the signer's anonymity is always retained, even with respect to the group manager, but this is *not* the model we consider in this thesis.

3. **Undeniable Signer Identity:** The group manager can always determine the identity of the group member who issued a valid signature. Moreover, he can also prove to some other entity (such as a judge) which member signed a given document without compromising that particular group member's anonymity in previous or future messages he may sign.

4. **Unlinkability:** Determining if two different signatures were computed by the same group member is computationally infeasible for everyone but the group manager.

5. **Security Against Framing Attacks:** No subset of group members (perhaps including the group manager) can sign a message on behalf of another group member. That is, if the `Open` procedure is invoked on the message, it should

not specify the name of another group member not belonging to the original subset.

6. **Coalition Resistance:** No subset of group members (perhaps including the group manager) should be able to collude and generate valid group signatures that are untraceable. In particular, we want to prevent attacks in which a coalition of group members get together, pool their information, and generate signatures which are approved by the `Verify` procedure, but for which the `Open` procedure fails to reveal any group member.

3.4 Efficiency of Group Signatures

The following parameters are of interest when evaluating the efficiency of a particular group signature scheme.

- The size (number of bits) of the group public key $\mathcal{Y}$.
- The size (number of bits) of an actual group signature on a message.
- The efficiency of the `Sign`, `Verify`, `Setup`, `Open`, and `Join` protocols.

3.5 Previous Work

Group Digital Signatures were first introduced and implemented by Chaum and van Heyst [14]. This paper proposed four different schemes. Three of these schemes required the group manager to contact each group member in order to find out who signed a particular message. These three schemes provided a computational anonymity guarantee; that is, the signer's anonymity was assured under some assumption on the hardness of a particular problem. The fourth scheme guaranteed anonymity in an information-theoretic sense. In this last scheme, and in one of the previous schemes, it is impossible to add a new group member once the scheme has been set up.

Two additional Group Signature schemes were suggested by Chen and Pedersen [15]. One of their schemes provides information-theoretic anonymity while the other provides computational anonymity. Unfortunately, their schemes suffers from the drawback that the group manager is capable of signing messages that appear to come from other group members.

Another paper on group signatures was written by Camenisch [6]. This paper proposes a group signature scheme that is more efficient than the previous two. The scheme does not have the drawback that the group manager can falsely accuse a member of having signed a message and the scheme enables the addition of new group members after the initial setup of the group. This particular scheme provides computational anonymity.

One of the best known group signature schemes is that of Camenisch and Stadler [9]. Their scheme meets all the above security criteria, and it has the property that the size of the group public key $\mathcal{Y}$ is *constant* in the size of the group. In fact, the public key given in their scheme remains the same throughout the lifetime of the group. Moreover, the Sign, Verify, Setup and Join protocols all operate in time that is independent of the group size. This property makes the scheme very nice for large groups. Unfortunately, the original scheme of Camenisch and Stadler is vulnerable to a coalition quasi-attack; we outline this attack later in this section. Fortunately, there is a simple modification to the original scheme that makes it much more secure. We chose this particular scheme as the building block for our Group Blind Digital Signatures. It is the first published scheme to have all the desired properties, and since it has not been broken yet, it has withstood some test of time.

Some subsequent work on Group Digital Signatures was done by Ateniese and Tsudik [3]. We developed a blinded variant of this scheme using slightly different techniques than the ones presented in this thesis [31]. Unfortunately, there are coalition attacks on this group signature scheme as well as against another scheme developed by Ateniese and Tsudik [1]. These attacks were discovered by Traore [37].

3.6 The Original Camenisch and Stadler Group Signature Scheme

We now describe the original group digital signature scheme of Camenisch and Stadler [9] (which we refer to as CS97 for notational convenience).

3.6.1 Preliminaries

Let G be a cyclic group of order n generated by some $g \in G$ (hence $G = \langle g \rangle$). Let $a \in \mathcal{Z}_n^*$. Our scheme will rely on the hardness of computing the following number-theoretic primitives.

Definition 3 *The discrete logarithm of $y \in G$ to the base g is the* smallest *non-negative integer x satisfying*

$$g^x = y$$

It is a widely held assumption that G, g, and n can be chosen such that the discrete logarithm problem is infeasible to solve.

Definition 4 *The double discrete logarithm of $y \in G$ to the bases g and a is the smallest non-negative integer x satisfying:*

$$g^{(a^x)} = y,$$

if such an x exists.

We conjecture that G, g, n, and a can be chosen such that the double discrete logarithm problem is infeasible to solve.

Definition 5 *An e-th root of the discrete logarithm of $y \in G$ to the base g is an integer x satisfying*

$$g^{(x^e)} = y \ ,$$

if such an x exists.

If the factorization of n is unknown, computing e-th roots in $\mathcal{Z}_n^*$ is assumed to be infeasible [33].

We also assume the existence of an ideal hash function $\mathcal{H}$: $\{0,1\}^* \mapsto \{0,1\}^k$ satisfying the following properties ($\mathcal{H}_l$ denotes the first l bits of $\mathcal{H}$):

1. For a specified parameter l, $\mathcal{H}_l(x)$ contains an equal number of 0's and 1's.
2. $\mathcal{H}_l$ is collision-resistant.
3. $\mathcal{H}$ hides all partial information about its input.

The last two properties are fairly standard. Proving security under the assumption that hash functions satisfying these two properties exist has been termed the Random Oracle Model [5]. The first property is not standard, but we can construct a hash function having all three properties assuming the existence of a hash function satisfying just the last two. In particular, let $\mathcal{H}$ be a hash function satisfying the last two properties. Consider

$$\mathcal{H}'(x) = \mathcal{H}(x) \circ \overline{\mathcal{H}(x)}$$

where $\overline{\mathcal{H}(x)}$ is the bitwise complement of $\mathcal{H}(x)$ (i.e. all 0's are flipped to 1's and all 1's are flipped to 0's) and $\circ$ denotes the concatenation operator. Then $\mathcal{H}'(x)$ has an equal number of 0's and 1's. Moreover, since $\mathcal{H}$ satisfies the last two properties, it is easy to see that $\mathcal{H}'$ satisfies them as well.

3.6.2 Signatures of Knowledge

A signature of knowledge is a construct in which a signer can use knowledge of some secret information (such as the discrete logarithm of a value y to the base g where $\langle g \rangle = G$) in order to digitally sign a message. This differs from the notion of a proof of knowledge, which is a way for one person to convince another person that he knows some fact without actually revealing that fact. In a signature of knowledge, the signer ties his knowledge of a secret to the message being signed. A signature of knowledge is used both for the purpose of signing a message and proving knowledge of a particular secret. Signatures of knowledge were used by Camenisch and Stadler [9]. Their construction is based on the Schnorr signature scheme [36] to prove knowledge. All the signatures of knowledge proposed by Camenisch and Stadler [9] can be proved secure in the random oracle model and their interactive versions are zero-knowledge.

One of the fundamental contributions of this thesis is to construct blinded variants of these signatures of knowledge. The proofs of security that work for the CS97 scheme [9] also work for our signatures of knowledge. We describe our blind signatures of knowledge in a later chapter.

The first signature of knowledge we consider is the signature of knowledge of the discrete logarithm of a given $y \in G$ to a given base g ($\langle g \rangle = G$). This signature of knowledge was originally derived from the Schnorr Signature scheme [36].

Definition 6 *An $(l+1)$-tuple $(c, s_1, \ldots, s_l) \in \{0,1\}^l \times \mathcal{Z}_n^l$ satisfying*

$$c = \mathcal{H}_l(m, y, g, g^{s_1} y^{c[1]}, g^{s_2} y^{c[2]}, \ldots, g^{s_l} y^{c[l]})$$

where $c[i]$ is the i-th leftmost bit of c, is a signature of knowledge of the discrete logarithm of $y \in G$ to the base g on a message m, with respect to security parameter l, denoted

$$SKLOG_l[\alpha \mid y = g^{\alpha}](m).$$

In general we use Greek letters to represent values whose knowledge will be proven by the signer, and Roman letters to denote values that are known to both the signer and the user.

If the signer does not know the discrete logarithm of y to the base g then it is infeasible for him to construct the $l+1$ tuple $(c, s_1, \ldots, s_l)$ satisfying the above equation. We can think of the above definition as an interactive protocol in which the $c[i]$'s represent challenges and the hash function $\mathcal{H}$ serves to remove the interaction. If the prover knows x such that $x = log_g(y)$, he computes $r_1, r_2, \ldots, r_l \in_R \mathcal{Z}_n$, plugs $m, y, g, g^{r_1}, g^{r_2}, \ldots, g^{r_l}$ to hash function $\mathcal{H}$ to obtain random challenge c, and obtains $s_1, s_2, \ldots, s_l$ by setting

$$s_i = \begin{cases} r_i & \text{if } c[i] = 0 \\ r_i - x \pmod{n} & \text{otherwise} \end{cases} \tag{3.1}$$

Another useful way to think of signatures of knowledge is as a methodology for converting interactive challenge/response protocols into non-interactive protocols in which the challenge bits are derived from applying a random function to a given message. Thus, the message is tied in with the protocol, and the challenge bits are truly random.

We now present two other signatures of knowledge. These signatures of knowledge are based on the double discrete logarithm problem, and the root of the discrete logarithm problem respectively.

Definition 7 *A signature of knowledge of a double discrete logarithm of y to the bases g and a, on message m, with security parameter l denoted $SKLOGLOG_l[\alpha \mid y = g^{(a^\alpha)}](m)$, is an $(l+1)$-tuple $(c, s_1, \ldots, s_l) \in \{0,1\}^l \times \mathcal{Z}^l$ satisfying the equation*

$$c = \mathcal{H}_l(m,y,g,a,P_1,\ldots,P_l), \text{ where } P_i = \begin{cases} g^{(a^{s_i})} & \text{if } c[i] = 0 \\ y^{(a^{s_i})} & \text{otherwise} \end{cases}$$

We often drop the subscript l from the notation $SKLOGLOG_l$ when the value is clear from the context. It can be shown, in the random oracle model, that a $SKLOGLOG_l[\alpha \mid y = g^{(a^\alpha)}](m)$ can be computed only if a double discrete logarithm $x \in \mathcal{Z}_n$ of the group element $y \in G$ to the bases $g \in G$ and $a \in \mathcal{Z}_n^*$ is known (where $G = \langle g \rangle$ and $|G| = n$). One does not necessarily know the order of $a \in \mathcal{Z}_n^*$, but it is easy to find λ such that $|\mathcal{Z}_n^*| \le 2^\lambda - 1$. Knowing x, λ, and a $\mu > 0$ compute the signature as follows (we use μ to make sure that the distribution of r_i is indistinguishable from the distribution of $r_i - x$, for a secret $x \le 2^\lambda - 1$ - thus we should choose μ to be large):

1. For $1 \le i \le l$, generate random $2^\lambda \le r_i \le 2^{\lambda+\mu} - 1$.

2. Set $P_i = g^{(a^{r_i})}$ and compute $c = \mathcal{H}_l(m,y,g,a,P_1,\ldots,P_l)$.

3. Set $s_i = \begin{cases} r_i & \text{if } c[i] = 0 \\ r_i - x & \text{otherwise} \end{cases}$

Definition 8 *A signature of knowledge of an e-th root of the discrete logarithm of y to the base g, on message m, denoted $SKROOTLOG_l[\alpha \mid y = g^{(\alpha^e)}](m)$, is an $(l+1)$-tuple $(c, s_1, \ldots, s_l) \in \{0,1\}^l \times \mathcal{Z}_n^{*l}$ satisfying the following equation:*

$$c = \mathcal{H}_l(m,y,g,e,P_1,\ldots,P_l), \ \ where P_i = \begin{cases} g^{(s_i^e)} & if\ c[i] = 0 \\ y^{(s_i^e)} & otherwise \end{cases}$$

It can be shown that such a signature can only be computed if the e-th root of the discrete logarithm x of y to the base g is known; where $x \in \mathcal{Z}_n^*$, $y \in G$, $|G| = n$, $\langle g \rangle = G$, $e \in \mathcal{Z}_n$. When x is known, construct $SKROOTLOG_l[\alpha \mid y = g^{(\alpha^e)}](m)$ as follows:

1. For $1 \le i \le l$, generate random $r_i \in \mathcal{Z}_n^*$.

2. Set $P_i = g^{(r_i^e)}$ and compute $c = \mathcal{H}_l(m,y,g,e,P_1,\ldots,P_l)$.

3. Set $s_i = \begin{cases} r_i & \text{if } c[i] = 0 \\ r_i/x \pmod n & \text{otherwise} \end{cases}$

Having discussed the necessary building blocks, we now give the actual construction in the Camenisch and Stadler group signature scheme.

3.6.3 The CS97 Construction

Camenisch and Stadler [9] described two schemes in their paper. The second is slightly more efficient, but less secure. We concentrate on making their first scheme blind, but our techniques apply to both.

Setup

To set up the group signature scheme, as in CS97 [9], the group manager chooses a security parameter l and computes the following values:

- An RSA Public Key (n,e) and Private Key d, where the length of n is at least $2l$ bits. We require that $n = pq$ where $p = 2P + 1$, $q = 2Q + 1$, and p, q, P, Q are all primes. In practice, one should choose p and q to be *significantly* longer than l bits each in order to ensure that factoring the modulus n is infeasible.

- A cyclic group $G = \langle g \rangle$ of order n for which computing discrete logarithms is hard. In particular, we can choose G to be a cyclic subgroup of $\mathcal{Z}_{p_2}^*$ where p_2 is a prime and $n|(p_2 - 1)$.

- An element $a \in \mathcal{Z}_n^*$ where a has large multiplicative order modulo all the prime factors of n.

- An upper bound λ on the length of the secret keys and a constant $\mu > 1$. Again, larger values of μ result in a more secure system.

- We also assume the existence of a public-key infrastructure (PKI) – anyone wishing to join the group must apriori be a part of this PKI. All members of the PKI have public and private keys associated with this PKI.

The group's public key is $\mathcal{Y} = (n, e, G, g, a, \lambda, \mu)$.

Join

As in the CS97 scheme [9], if Alice wants to join the group she executes the following steps:

- She picks a *secret key* $x \in_R \{0, 1, \ldots, 2^\lambda - 1\}$.

- She calculates $y = a^x \pmod{n}$.

- She computes a *membership key* $z = g^y$, and

- Alice signs z with her secret key associated with the PKI. This forces Alice to tie her identity with the membership key z, and thus she is, in some sense, "committed" to it.

To obtain a *membership certificate*, she sends (y,z) along with her signature on z to the group manager and proves to him that she knows x (without actually revealing x) using the signature of knowledge of discrete logarithm. If the group manager is convinced that Alice knows x he gives her a *membership certificate*:

$$v \equiv (y+1)^d \pmod{n}.$$

It is an additional security assumption due to Camenisch and Stadler [9] that computing v without factoring n is hard.

Signing Messages

To sign a given message $m \in \{0,1\}^*$, Alice computes the following values:

- $\tilde{g} := g^r$ for $r \in_R \mathcal{Z}_n$
- $\tilde{z} := \tilde{g}^y \ (= z^r)$
- $V_1 := SKLOGLOG_l[\alpha : \tilde{z} = \tilde{g}^{a^\alpha}](m)$
- $V_2 := SKROOTLOG_l[\beta : \tilde{z}\tilde{g} = \tilde{g}^{\beta^e}]$

  The signature on the message m consists of $(\tilde{g}, \tilde{z}, V_1, V_2)$.

Verify

To verify the signature s on a given message m with respect to group public key $\mathcal{Y}$, a user simply has to do the following.

- Verify that V_1 is a valid signature of knowledge of the double discrete logarithm.
- Verify that V_2 is a valid signature of knowledge of the root of the discrete logarithm.

We now explain why this signature of knowledge really proves that signer is a valid group member. To start with, V_1 proves that $\tilde{z}$ is of the form $\tilde{g}^{a^\alpha}$. Thus, we are assured that $\tilde{z}\tilde{g}$ must be of the form:

$$\tilde{z}\tilde{g} = \tilde{g}^{a^\alpha + 1}$$

where α is some integer which is known to the signer. Now, V_2 proves that the signer knows the $e - th$ root of the logarithm of $\tilde{z}\tilde{g}$ – which means she knows the $e - th$ root of $(a^\alpha + 1)$. Now, putting this together, we have that the signer knows α and $(a^\alpha + 1)^d$. This corresponds to a secret key, and a membership certificate with respect to that secret key. And by assumption it is impossible to compute a pair $(\alpha$, $(a^\alpha + 1)^d)$ without talking to the group manager (who knows the factorization of n).

Open

Given a signature $(\hat{g}, \hat{z}, V_1, V_2)$ for a message m, the group manager can determine the signer by testing if $\hat{g}^{y_P} = \hat{z}$ for every group member P (where $y_P = \log_g z_P$ and z_P is P's membership key). Thus the running time of this scheme is linear in the size of the group.

The group manager can prove the identity of the signer to someone else (e.g. a judge in case of a dispute) without giving away y_P using the signer's membership key z_P, the signer's commitment to z_P, and a non-interactive proof that $\log_g z = \log_{\hat{g}} \hat{z}$. Since it is considered difficult to test if $\log_{\hat{g}} \hat{z} = \log_{\hat{g}'} \hat{z}'$, members' signatures are anonymous and unlinkable.

3.7 A More Secure Variant of the Camenisch and Stadler Scheme

Unfortunately, there is a minor problem with the original scheme presented by Camenisch and Stadler. In this section we describe a coalition quasi-attack on the CS97 scheme. This attack, which is due to Ateniese and Tsudik [2], shows how several group members might be able to collude and generate valid but untraceable group signatures. We use the term "quasi-attack" since we can easily modify the CS97 scheme to prevent the attack; we present this modification later in this section. The modification is motivated by a comment in the paper of Ateniese and Tsudik [2], but has not previously appeared explicitly in the literature. We stress that even with this modification, there is no guarantee that the scheme is still secure. The security does, however, rest on a well-defined number-theoretic conjecture. One thing to note is that there are no schemes in the literature which are guaranteed to be secure against coalition resistance. The only scheme which "guaranteed" coalition resistance was the one given by Ateniese and Tsudik [1], however, this scheme was subsequently broken by Traore [37].

3.7.1 A Coalition Quasi-Attack Against CS97

We describe a coalition quasi-attack on the Camenisch and Stadler scheme. This attack is due to Ateniese and Tsudik [1]. Fortunately, the scheme can easily be modified to prevent it. Thus the attack does not have any significant practical value; it does however have some theoretical value since it might point out a potential flaw in the scheme. A *coalition* attack happens when some collection of group members (possibly including the group manager) collude and combine their secret membership keys in such a manner that they can generate a valid, yet untraceable group signature for a particular message. These signatures are untraceable in the sense that the Open procedure will fail to identify a particular group member when given this message and signature pair as input. A potentially more dangerous attack is one where some coalition can get together and fraudulently generate a signature that appears to be from some other member of the group. In other words, the Open procedure would identify that particular member when given this message and signature pair as input. Fortunately, no framing attack of this nature is known for the CS97 scheme.

The coalition resistance in the Camenisch and Stadler scheme rests on the following number-theoretic conjecture:

Conjecture 1 *Suppose (n, e) is a valid RSA public key, and let $a \in \mathcal{Z}_n$ be an element of large multiplicative order modulo the prime factors of n – such that computing* *discrete logarithms to the base a is infeasible. Suppose also, that you are given access to an oracle which for any chosen value y produces a value v satisfying the following equation:*

$$v^e \equiv 1 + a^y \pmod{n}.$$

If the factorization of n is unknown, then it is hard to compute another pair (y', v') of integers that satisfy this equation. Here y' must differ from any y which was previously given as a query to the oracle.

Unfortunately, this conjecture is false. We do stress, however, that even if the conjecture is false, the scheme might still be secure, albeit the falsity of the conjecture gives evidence of a weakness in the scheme. We show how three individuals (call them Alice, Greg, and Joe) can launch a successful coalition quasi-attack against the Camenisch and Stadler scheme. We term this a quasi-attack because it turns out to be very easy to fix; however, it points to what might be a potential weakness in the scheme. Consider the following attack:

1. Alice joins the group with a secret key y and gets back the membership certificate $A = (a^y + 1)^d \pmod{n}$ from the group manager. Observe that $A = a^{yd}(1 + a^{-y})^d \pmod{n}$.

2. Greg joins the group with secret key $-y$ and obtains the membership certificate $G = (a^{-y} + 1)^d \pmod{n}$.

3. Joe joins the group with the secret key $y \cdot c$, where c is chosen at random, and he gets back $J = (a^{cy} + 1)^d \mod n$ as his membership certificate.

4. Now, the three parties pool their values to get a new value C:

$$C := J \cdot (A \cdot G^{-1})^{-c} \pmod{n}.$$

It turns out that this new value C is a valid group membership certificate corresponding to the user secret key $-cy$. This follows from the following:

$$
\begin{aligned}
J \cdot (A \cdot G^{-1})^{-c} &= J \cdot (a^{yd}(1 + a^{-y})^{d} \cdot (1 + a^{-y})^{-d})^{-c} \pmod{n} && (3.2)\\
&= J \cdot a^{-cdy} \pmod{n} && (3.3)\\
&= (a^{cy} + 1)^{d} \cdot a^{-cdy} \pmod{n} && (3.4)\\
&= (a^{-cy} + 1)^{d} \pmod{n}. && (3.5)
\end{aligned}
$$

And $(a^{-cy} + 1)^{d} \pmod{n}$ is a valid group membership certificate for the secret key $-cy$. Thus three group members have gotten together, have generated a new secret key, and have generated a membership certificate corresponding to that secret key. Moreover, the group manager doesn't know about this new key, so the group members can sign documents on "behalf" of the group under this secret key, and at the end *no one* will be able to trace this information back to them.

We can modify the Camenisch and Stadler scheme to make this quasi-attack *impossible* to mount. We describe this modification in the next section. We stress that even though we give a method to prevent this attack, the scheme may still be vulnerable to other, perhaps more clever, coalition attacks. We can, however, give a precise number-theoretic conjecture on which to base the security of the system.

3.7.2 Modifying the Join Protocol to Make Coalition Attacks More Difficult

We give a minor modification to the join protocol in CS97 which prevents the previous attack. The idea is that the previous attack required that the malicious colluding group members get together *before* joining the group in order to perform the attack, and the attack rested heavily on their being able to pick their secret-key values. In the new protocol we describe, the Group Manager *forces* the joiner to pick a secret key that is random and independent from previous keys. Since the previous attack relied on members picking their secret keys in some kind of a pre-specified format, as opposed to randomly, the attack is prevented.

More Secure Join Protocol

Alice Round 1

1. Alice picks a value $x \in_R \{0, 1, \ldots, 2^\lambda - 1\}$.

2. She calculates $y' = a^x \pmod{n}$ and sends this value y' to the Group Manager.

Group Manager Round 2

1. The group manager picks a value $\gamma \in_R \{0, 1, \ldots, 2^\lambda - 1\}$, and sends this value to Alice.

Alice Round 3

1. Alice first checks if $x + \gamma \in \{0, 1, \ldots, 2^\lambda - 1\}$. If it is not, Alice and the group manager repeat the protocol from the beginning.

2. Alice computes $y = a^{x+\gamma} \pmod{n}$ and she sends this value to the group manager. Alice must also convince the group manager that $x + \gamma$ lies in the range $\{0, 1, \ldots, 2^\lambda - 1\}$. This can be accomplished via cut and choose. There are several more efficient methods for doing this which do not reveal information about $x + \gamma$; for example, see the paper of Camenisch and Michels [8], or the paper of Chan, Frankel and Tsiounis [10] and the references therein.

Group Manager Round 4

1. The group manager verifies that $y = y' \cdot a^\gamma \pmod{n}$– this condition should hold if Alice computed y according to the protocol.

Alice Round 5

1. Alice computes a *membership key* $z = g^y$.

2. Alice commits to z (by signing it).

3. To obtain a *membership certificate*, she sends (y,z) along with her commitment to z to the group manager and proves to him that she knows $x + \gamma$ (without actually revealing $x + \gamma$) using the signature of knowledge of the discrete logarithm.

Group Manager Round 6

1. If the group manager is convinced that Alice knows x he gives her a *membership certificate*:

$$v \equiv (y+1)^d \ (mod\ n)$$

This concludes the protocol. The idea behind the protocol is to randomize the process of picking the secret membership key. The key is now random because it is of the form $x + \gamma$ where γ was chosen randomly by the group manager. The rest of the protocol is basically the same as the original protocol.

We now state an alternate conjecture on which to rest the security of this modified scheme. If this conjecture is true, then the scheme is secure. The converse does not necessarily hold; i.e., even if a counterexample to the conjecture is obtained, the scheme may still be secure.

Conjecture 2 *Suppose (n, e) is a valid RSA public key, and let $a \in \mathcal{Z}_n$ be an element of large multiplicative order modulo the prime factors of n such that computing discrete logarithms to the base a is infeasible. Suppose also, that you are given access to an oracle which provides randomly chosen pairs (y, v) satisfying*

$$v^e \equiv 1 + a^y \ (\mathrm{mod}\ n).$$

If the factorization of n is unknown, then it is hard to compute another pair (y', v') of integers that satisfy this equation. Here y' must differ from any y which was previously supplied by the oracle.

Chapter 4

The Group Blind Digital Signature Model

4.1 Introduction

In this chapter we give more formal definitions of what constitutes a Group Blind Digital Signature. In the next section, we give the security requirements of group blind digital signatures. In section three we discuss the procedures allowed by group blind digital signatures. One will notice that the procedures and security requirements of group blind digital signatures are similar to those of regular group signatures. The final section addresses applications of group blind digital signatures to online distributed banking, offline distributed banking, and online voting.

4.2 Security Requirements for Group Blind Digital Signature

The security requirements of a Group Blind Digital signature are very similar to those of a Group Digital Signature. The only addition is that we require the blindness property in the signature. Here are the security requirements:

1. **Blindness of Signatures:** The signer is unable to view the messages he signs. Moreover, the signer should have no recollection of having signed a particular document even though he (or anyone else for that matter) can verify that the signature is indeed valid.

2. **Unforgeability:** Only group members can issue valid signatures on behalf of the entire group; i.e. only group members can issue signatures that are verifiable by the group public key.

3. **Conditional Signer Anonymity:** Given a message / signature pair anyone can easily check that the signature is valid, and that the message was signed by some particular group member, but *only* the group manager can determine which specific member issued the signature. One can consider an alternate model in which the signer's anonymity is retained, even with respect to the group manager, but this is *not* the model we consider in this thesis.

4. **Undeniable Signer Identity:** The group manager can always determine the identity of the group member who issued a valid signature. Moreover, he can also prove to some other entity (such as a judge) which member signed a given document without compromising that particular group member's anonymity in previous or future messages he may sign.

5. **Unlinkability:** Determining if two different signatures were computed by the same group member is computationally infeasible.

6. **Security Against Framing Attacks:** No subset of group members (perhaps including the group manager) can sign a message on behalf of another group member. That is, if the `Open` procedure is invoked on the message, it should not specify the name of another group member not belonging to the original subset.

7. **Coalition Resistance:** Any subset of group members (*not* including the group manager) should be incapable of generating valid group signatures that are

untraceable. In particular, we want to prevent attacks in which a coalition of group members get together, pool their information, and generate signatures which are approved by the `Verify` procedure, but for which the `Open` procedure fails to reveal any group member.

4.3 Procedures Allowed by Group Blind Digital Signatures

The procedures given in our Group Blind Digital Signature model are identical to those given in the original group signature model. We list them here again for completeness.

`Setup` a probabilistic algorithm that generates the group's public key $\mathcal{Y}$ and a secret administration key $\mathcal{S}$ for the group manager.

`Join` an interactive protocol between the group manager and the new group member Bob that produces Bob's secret key x, and his membership certificate A.

`Sign` an interactive protocol between group member Bob and an external user Alice, which on input a message m from Alice and Bob's secret key x produces a signature s on m that satisfies the properties above.

`Verify` an algorithm which on input $(m, s, \mathcal{Y})$, determines if s is a valid signature for the message m with respect to the group public key $\mathcal{Y}$.

`Open` an algorithm which, on input $(s, \mathcal{S})$ determines the identity of the group member who issued the signature s.

4.4 Applications of Group Blind Digital Signatures

We now give applications of Group Blind Digital Signatures to electronic cash. From the previous exposition on Blind Digital Signatures, it should be readily apparent how to apply Group Blind Digital Signatures to Electronic Cash. Here we outline a slightly different electronic banking model, and give protocols to achieve the desired properties in this model.

4.4.1 Distributed Electronic Banking

Consider a scheme in which there is a large group of banks, monitored by the country's Central Bank (e.g. the US Treasury or the Federal Reserve), where each bank can dispense electronic cash. We would like such a scheme to have the following properties:

1. No bank should be able to trace any e-cash it issues. If the bank issues an electronic coin to a customer, and if it later happens to see that coin again (after it was spent), it should not be capable of determining which particular customer spent that coin. In addition, if the customer happens to spend several electronic coins, and the bank sees those coins, it should not be able to determine that those were spent by the same customer. Therefore, just as with paper money, people can spend their e-cash in a completely anonymous fashion.

2. A vendor only needs to invoke a single universal verification procedure, based on the group public key, to ensure the validity of the bank's signature on any e-cash he receives. This procedure works regardless of which bank issued the e-cash. This makes the vendor's task much easier since he only needs to know the single group public key. We note that even if the signature on the coin is valid, it still might be possible that the coin is not valid for spending; e.g. if the coin has already been spent, then the signature is valid, even though the coin should not be spent.

3. There is a single public key for the entire group of banks. The size of this public key is independent of the number of banks. Moreover, the public key should not be modified if more banks join the group. Thus, the scheme is still practical even if there are a large number of participating banks.

4. Given a valid piece of e-cash only the Central Bank can tell which bank in

the group issued it. No vendor can even determine the bank from which the customer got her e-cash even though the vendor can easily check that the e-cash is valid. This restriction gives an extra layer of anonymity since we conceal both the spender's identity and the identity of the bank she uses.

5. No subset of banks (perhaps even including the Central Bank) can issue e-cash on behalf of another bank; i.e. no subset of banks or any other entities, can "frame" another bank.

6. Any coalition of banks (*not* including the Central Bank) should be unable to construct valid looking electronic coins that are untraceable. In other words, no coalition should be able to construct coins such that the Central Bank is incapable of associating that coin to any particular bank in the group.

In this chapter we show how to implement such a scheme with the desired properties using Group Blind Digital Signatures. Many previous electronic cash schemes focus on a model in which a single bank distributes all the e-cash. In real life, one would like for more than one bank to be able to dispense electronic money. Our scheme is unique since it considers scalability as a criterion in the design of electronic cash systems. Moreover, the scheme is conceptually novel since we conceal the bank's identity in addition to the spender's. We outline the protocols necessary for achieving this scheme in the next section.

4.4.2 Applying Group Blind Digital Signatures to Distributed Electronic Banking

We describe the necessary protocols for achieving the distributed banking scheme we envisioned earlier. The parties involved in our protocol are: Alice, Bob, BankA, and BankB. Here Alice is a customer who happens to use BankA. Alice wishes to purchase some particular item from a vendor Bob. BankB is Bob's bank.

Setup

The Banks in our scheme form a group, and their group manager will be some type of central bank; for example the US Treasury or the Federal Reserve. If other Banks wish to join the group later, they may do so rather easily. Each Bank performs the join protocol with the group manager, and is then capable of signing documents on behalf of the entire group of banks.

Withdrawal

To withdraw money, the following steps are performed.

1. Alice creates an electronic coin C. This coin consists of some serial number, as well as some other information pertaining to currency, such as value, etc,.
2. Alice then asks BankA to apply a Group Blind Digital Signature to C.
3. BankA applies the signature to C, and withdraws the appropriate amount of money from Alice's account. Alice now possesses a coin C along with a valid group signature on that coin C.

Spending

To spend money, the following steps are performed.

1. Alice gives the coin C along with the bank's signature on C to the vendor Bob.
2. Bob checks whether or not the bank's signature on C is authentic. This can be done easily via the *Group Public Key*.
3. If all checks out, Bob gives the coin to his bank for deposit, and awaits the bank's response.

Deposit

To deposit a coin, the following steps must be performed.

1. Bob takes the coin C, along with BankA's signature on the coin, and gives it to BankB.

2. BankB looks at the coin, and verifies its validity by checking the signature on the coin. Note that BankB *does not* need to know who BankA is in order to perform this check. This can simply be done by using the group public key.

3. BankB then checks whether this coin has already been spent; the banks may need to keep some global list of already spent coins to accomplish this double-spending check.

4. If all checks out, then BankB credits Bob's account with the appropriate amount.

5. Bob can give Alice her merchandise, if he has not done so already.

4.4.3 Offline Electronic Cash Scheme

The one problem with the scheme outlined above is that it is *online*. That is, the vendor Bob must clear the coin with his bank *before* giving the merchandise to the customer. If the cash transaction is small, it is impractical for the vendor to engage in a protocol with the bank every time a transaction is made. However, for purely anonymous cash in the framework presented thus far, it turns out that we *need* to have an online scheme for security reasons. For example, if Alice is trying to spend the same coin twice, and if we fail to catch her right away (by talking to the bank), then we have no hope of catching her once the transaction is complete because her identity is completely anonymous. There are several possible remedies to this problem.

One possible solution is to have what is called an *anonymity-revoking trustee.* This person is a trusted third party who can revoke the identity of the spender in case any kind of potential fraud is detected. However, we would like to have a trustee whose role is minimal. That is, the trustee should be involved when anonymity revocation needs to take place, and perhaps when the user opens his bank account, but that should be extent of the trustee's involvement. It certainly would be impractical if the trustee were involved in every transaction. We now present a scheme which incorporates such a trustee. We can then have an offline electronic cash scheme; this enables the vendor to collect all electronic coins, and deposit them with his bank in a single shot. A trustee based electronic cash scheme was presented by Camenisch. Maurer, and Stadler [7]. Their scheme used trustees who were involved only when it came time for anonymity revocation, and at no other time. Their scheme only involved a single bank model. Our scheme, on the other hand, works in a multiple bank model, but the trustees are also involved whenever each user opens his account. The idea is to have all the consumers form a group, and have the anonymity-revoking trustee be their group manager. Customers will append their group signatures to electronic coins during the spending phase, and as a result their identity will be encoded in the transaction. Furthermore, only the trustee will be able to determine the identity of the customer. We formally give the protocol – it is broken up into five phases: setup, withdrawal, spending, depositing, and anonymity revocation.

Setup

The Banks in our scheme form a group, and their group manager will be some type of central bank; for example the US Treasury or the Federal Reserve. If other Banks wish to join the group later, they may do so rather easily. Each Bank performs the join protocol with the group manager, and is then capable of signing documents on behalf of the entire group of banks. Additionally, we set up a group for customers. Whenever a customer opens her bank account, she must join the customer group as well. The anonymity revoking-trustee is the group manager for this customer group. Finally, we assume that all vendors are part of a Public-Key infrastructure (PKI). That is, all vendors have public and private keys associated with this PKI.

Withdrawal

To withdraw money, we perform the same protocol as in the original online case.

Spending

To spend money, the following steps are performed.

1. Alice gives the coin C, and the bank's signature on C, to the vendor Bob.

2. Bob first checks that the bank's signature on C is valid; this can be done easily using the *Bank's Group Public Key.* If it is valid, he executes the following sequence of steps:

   (a) Bob generates a random sequence of bits (called a *nonce*).

   (b) He creates a message consisting of his name or identity, the nonce, and the current time.

   (c) Bob digitally signs this message with his secret key associated with the PKI.

   (d) Bob sends the message and signature to Alice.

3. Alice first checks that the signature on Bob's message is valid, and that the time entered in the message is recent. Alice constructs a her own message which consists of the concatenation of the coin C, the bank's signature on C, and the message she just received from Bob along with his signature on that message. She applies her own group signature to her newly formed message, and sends the message and signature to Bob.

4. Bob first checks that Alice has given him a message of the correct form, and that his original message and signature are a part of the message Alice gives him. Bob now checks that Alice's group signature on the message is valid. This ensures him that the trustee will be able to revoke Alice's anonymity should there be any reason to do so.

5. If all checks out, Bob gives Alice her merchandise.

We incorporate the nonce and Bob's signature into this protocol to prevent the following transferability type of attack. Suppose that in the above protocol, we only required Alice to apply her group signature on C without the nonce or extra message; furthermore, suppose Alice herself was a vendor. Alice can then engage in two different transaction protocols concurrently. In the first protocol, she acts as a customer, and Bob is the vendor. In the second protocol, she acts as a vendor and Charles is the customer. Now, if we did not have Bob sign his message to Alice, then Alice could simply take any responses she receives from Bob, and give them to Charles – that is she could "pretend" to be Bob when talking with Charles. Conversely, she can take any responses she receives from Charles and give them to Bob. In this case, she would be pretending to be Charles when talking with Bob. The upshot is that at the end of such a transaction, Alice would have, in effect, taken Charles's coin, and given it directly to Bob. Hence her identity was never involved in the protocol and she is in no way tied to the coin! Thus, if the anonymity-revocation mechanism were ever applied to the electronic coin, Charles's name would be the only one ever revealed, and Alice may be able to get away scot-free with malicious behavior.

Now, since we incorporate the vendor's digital signature, if Alice tries to pass Bob's exact responses to Charles, then Charles can check the signature on the message, and determine whether the message is from Alice. If on the other hand Alice attempts to remove Bob's signature, and append her own, then the resulting signed electronic coin she receives from Charles will have her signature. If she tries giving this same cash back to Bob, then Bob will immediately notice that the signature on the nonce has been changed.

Perhaps a minor drawback of the above scheme is that we force the identity of the vendor to be known, which need not have been the case in the previous schemes. On the other hand, it is not clear that maintaining vendor anonymity is terribly useful. Although the vendor may not mind that the customer is anonymous, given that he can check that the electronic coin is legitimate, the buyer may want to know the identity of the vendor in case the merchandise turns out to be bad. While there may be exceptions and special cases, it does not seem like vendor anonymity is something with which we should be terribly concerned.

Deposit

At the end of some specified period, Bob can deposit all the coins he has collected. To deposit a coin, the following steps must be performed.

1. Bob takes the coin C, BankA's signature on the coin, along with Alice's signature on both, and gives it to BankB.

2. BankB looks at the coin, and verifies its validity by checking the signatures on the coin. Note that BankB *does not* need to know who BankA is in order to perform this check. This can simply be done by using the group public keys for the bank group and the customer group.

3. BankB now checks to see whether the coin has been spent already. This task could be done by having all the banks maintain a global list of coins that have already been spent. If the coin has already been spent, then the trustee must be called upon to perform anonymity revocation. We discuss precisely how this anonymity revocation is done shortly.

4. If all checks out, then BankB credits Bob's account with the appropriate amount.

5. Bob can give Alice her merchandise, if he has not done so already.

Anonymity Revocation

If there is some reason to believe that a coin is fraudulent, then the identity of the spender can be determined as follows.

1. The coin C, the bank's signature on C, and the customer's signature on both are given to the trustee.

2. Since the trustee is the group manager of all the consumers, he can use the `Open` algorithm to determine the identity of the original signer.

4.4.4 Online Voting

We now present a distributed online voting scheme using group blind digital signatures. This scheme is similar to the voting scheme based on blind digital signatures. We remark that the scheme given here was chosen because it is simple and easy to present. Our techniques can possibly be applied to other, perhaps more complex and secure, online voting protocols; for example, the protocol of Fujioka, Okamoto, and Ohta [19].

Suppose that we have three entities: A voter Alice, a local registration facility (LRF), and vote submission facility (VSF). Moreover, there are many such LRF's – and each voter can only register with some particular one; intuitively this corresponds to a person who registers to vote at a local voting center. Finally, these LRFs form a group, and are managed by some central facility. We assume that the votes are of the form "Yes" or "No". The protocol we give is adapted from the one given in Schneier's text [35]. The voting protocol is divided into two stages: registration and voting.

Online Voting Protocol

Registration

1. Alice creates two electronic ballots B_1 and B_2 – these consists of a serial number, and some other relevant information to voting. In addition, B_1 contains the vote "Yes" and B_2 contains the vote "No."

2. Alice then takes the ballots B_1, B_2 and blinds them. It sends the blinded versions to the LRF. We assume Alice is only allowed to use a particular LRF (within some local area).

3. The LRF checks its database to make sure that Alice has not voted before. If this is the case, then it signs the blinded ballots and it gives them back to Alice.

Voting

1. Alice unblinds the messages, and now has two sets of valid votes signed by the LRF.

2. Alice now picks which vote ("Yes" or "No") she wants to choose, along with the LRF's signature on that vote, and encrypts it with the VSF's public key.

3. Alice sends in her vote to the VSF.

4. The VSF decrypts the votes, checks that the signature is valid by using the group public key for the LRFs, and checks its database to make sure that the identification number on the vote has not been used before (this prevents Alice from trying to vote more than once). If this checks out then the VSF tabulates the vote, saves the serial number and records it in the database. At the end of the election, the VSF publishes the results of the election, as well as every serial number and its associated vote.

Chapter 5

Blind Signatures of Knowledge

5.1 Introduction

We describe how to blind the signature of knowledge protocols given earlier. We use these blind signatures of knowledge as the fundamental building blocks of our Group Blind Digital Signature Scheme. The fundamental technical contributions of this thesis are contained within this chapter. We start by giving a blinded variant on the Signature of Knowledge of the Discrete Logarithm. This variant enables the user and signer to engage in a protocol such that at the end, the user obtains a valid signature on the message m, and the signer will be unable to tie the message or the signature with the signing process. In other words, even if the signer sees the message and signature later on, he will gain no information about when, or for whom, he signed it. In the subsequent sections we give blind signature of knowledge protocols for the double discrete logarithm, and for the root of the discrete logarithm.

5.2 Blind Signature of Knowledge of the Discrete Logarithm

We give a blind version of the signature of knowledge of the discrete logarithm (SKLOG). Recall that we let G be a cyclic group of order n generated by some $g \in G$

(hence $G = \langle g \rangle$). We defined the signature of knowledge of the discrete logarithm as follows:

Definition 9 *An $(l+1)$-tuple $(c, s_1, \ldots, s_l) \in \{0,1\}^l \times \mathcal{Z}_n^l$ satisfying*

$$c = \mathcal{H}_l(m, y, g, g^{s_1} y^{c[1]}, g^{s_2} y^{c[2]}, \ldots, g^{s_l} y^{c[l]})$$

is a signature of knowledge of the discrete logarithm of $y \in G$ to the base g on a message m, with respect to security parameter l, denoted

$$SKLOG_l[\alpha \mid y = g^{\alpha}](m).$$

Now, our goal is to make this a *blind signature*. How can we do this? In order to have a blind signature, two things must be accomplished.

- The signer should not be able to determine when or for whom he signed the original message.
- The signer should not be able to recognize the signature on the original message, though he can verify that the signature is indeed valid.

Before giving our final protocol, we first provide a protocol that does not work, and then modify it so it does. We start with a protocol that just hides the message, but fails to hide the signature. In this protocol, we assume that the signer's secret key is $x \in \mathcal{Z}_n$ and his public key is $y = g^x$.

User Round 0: User wants message m signed and sends a sign request to the signer.

Signer Round 1:

1. Choose random $r_1, \ldots, r_l \in_R \mathcal{Z}_n$.
2. Set $z_j = g^{r_j}$, and send the z_j's to the user.

User Round 2:

1. Set $c = \mathcal{H}_l(m, y, z_1, \ldots, z_l)$, and send c to the signer.

Signer Round 3:

1. The signer sets $s_j = r_j - c[j]x_j$, $1 \leq j \leq j$. (recall that $c[i]$ denotes the ith most significant bit of c.)

We now prove that $(c, s_1, \ldots, s_l)$ is a valid signature of knowledge of the discrete logarithm on message m.

Lemma 1 *The tuple* $(c, s_1, \ldots, s_l) = SKLOG_l[\alpha \mid y = g^\alpha](m)$

Proof:

$$\begin{aligned}
&\mathcal{H}_l(m, y, g, g^{s_1}y^{c[1]}, \ldots, g^{s_l}y^{c[l]}) && (5.1)\\
&= \mathcal{H}_l(m, y, g, g^{s_1+c[i]x}, \ldots, g^{s_l+c_i[x]}) && (5.2)\\
&= \mathcal{H}_l(m, y, g, g^{r_1}, \ldots, g^{r_l}) && (5.3)\\
&= \mathcal{H}_l(m, y, z_1, \ldots, z_l) && (5.4)\\
&= c. && (5.5)
\end{aligned}$$

$\mathcal{QED}$

In the protocol just given, the message is hidden from the signer, but the signer can still recognize the signature because he gets to see both c and $s_1, \ldots, s_l$. Let us consider what needs to be done to make this protocol truly blind. To start with, not only does the message have to be hidden, but so does all of the signer's input into the protocol. We can start by blinding the z_j's – this can be accomplished by multiplying by a random blinding factor g^{a_j} (where a_j is chosen randomly). Next, we need to blind the challenge bits c. Since the outputs of the hash function $\mathcal{H}$ always contains an equal number of 0's and 1's, it follows that randomly permuting the order of the individual bits of c results in completely blinding c. In order to be able to reconstruct a valid signature after blinding c, we need to perform the following steps:

1. Scramble the order of the z_j's by applying a permutation σ to the indices.

2. Apply the hash function $\mathcal{H}$ to the scrambled version of these z_j's and get a challenge c.

3. Apply σ^{-1} to the individual bits of c to get a blinded challenge.

4. Have the signer respond to the blinded challenge, and adjust his responses to take into account that the order of the z_j's was scrambled, and that a_j's were used to blind them.

Consider the following interactive protocol between a signer and a user that incorporates these ideas:

User Round 0: User wants message m signed and sends a sign request to the signer.

Signer Round 1:

1. Obtain $\{r_i \in_R \mathcal{Z}_n\}$, $1 \leq i \leq l$.
2. Set $P_i := g^{r_i}$, $1 \leq i \leq l$.
3. Send $\{P_i\}$ to the user, thus commiting to the $\{P_i\text{'s}\}$.

User Round 2:

1. Obtain a random permutation $\sigma : \{1, \ldots, l\} \mapsto \{1, \ldots, l\}$ and set $Q_i := P_{\sigma(i)}$, $1 \leq i \leq l$. (σ will be used to blind the result of $\mathcal{H}$.)
2. Obtain random $a_1, \ldots, a_l$, and set $R_i := Q_i g^{a_i}$ $1 \leq i \leq l$. ($\{a_i\}$ are used to blind the z_i's which are inputs to $\mathcal{H}$.)
3. Calculate $c := \mathcal{H}_l(m, y, g, R_1, \ldots, R_l)$.
4. Calculate c' such that $c'[i] = c[\sigma^{-1}(i)]$. ($c'$ is the blinded challenge.)
5. Send c' to the signer.

Signer Round 3:

1. Using secret x (recall $x = \log_g y$) compute for $1 \leq i \leq l$,

$$t_i = \begin{cases} r_i & \text{if } c'[i] = 0 \\ r_i - x (\text{mod } n) & \text{otherwise} \end{cases}$$

2. Send $\{t_i\}$ to the user.

User Round 4:

1. Verify that $P_i = g^{t_i} y^{c'[i]}$.
2. Compute $s_i := t_{\sigma(i)} + a_i \pmod{n}$.
3. Output $(c, s_1, \ldots, s_l)$. This constitutes the signature on m.

We can show that this protocol is correct and that the resulting signature is blind.

Lemma 2 (Correctness) *The protocol given above produces a valid signature of knowledge. That is,*

$$(c, s_1, \ldots, s_l) = SKLOG_l[\alpha \mid y = g^\alpha](m).$$

Proof: Observe that c was set according to the following equation:

$$c = \mathcal{H}_l(m, y, g, R_1, \ldots, R_l).$$

Now, consider R_i:

$$R_i = Q_i g^{a_i} = P_{\sigma(i)} g^{a_i} = g^{t_{\sigma(i)}} y^{c'[\sigma(i)]} g^{a_i} = g^{s_i - a_i} g^{a_i} y^{c'[\sigma(i)]} = g^{s_i} y^{c'[\sigma(i)]} = g^{s_i} y^{c[i]}.$$

Therefore:

$$c = \mathcal{H}_l(m, y, g, g^{s_1} y^{c[1]}, \ldots, g^{s_l} y^{c[l]}).$$

And by definition, this constitutes a valid signature.

$\mathcal{QED}$

Lemma 3 (Blindness) *In the random oracle model, the protocol described above produces a blind signature, that is,* $(c, s_1, \ldots, s_l)$ *is a signature on* m *that cannot be linked to the signer's view of the protocol.*

Proof: First, we assume that $\mathcal{H}$ is a random oracle. That is, it hides all partial information about its input. We also assume that $\mathcal{H}_l(x)$ has an equal number of 0s and 1s. As we argued earlier, this is not a unrealistic assumption. Let us compare the resulting signature with the signer's view of the protocol. The final signature is $(c, s_1, \ldots s_l)$. First let us consider the s_i. These are $t_{\sigma(i)}$ plus some blinding factor a_i. Observe that the signer gets no information about the a_i because the only information he receives from the user is c' and c' is the output of the random oracle $\mathcal{H}$. Therefore, the signer has no information about a_i, hence the values $s_i = t_{\sigma(i)} + a_i$ look completely random. Moreover, they are independent of each other. In addition, the value c will look completely random to the signer. This follows because the signer does not know the random permutation σ which was applied to c'. Moreover, c' hides any information about σ because c' was the output of a random oracle. Therefore, the output values $(c, s_1, \ldots, s_l)$ are completely random, and are independent of the values seen by the signer. Hence the signer will be unable to recognize them. $\mathcal{QED}$

Also note that since in the proposed protocol the signer does not provide more information than in the regular proof of knowledge, our protocol is zero-knowledge. We now give blind signature protocols for the variants of the discrete logarithm problem.

5.3 Blind Signature of Knowledge of the Double Discrete Logarithm

Let us consider the double discrete logarithm problem. Recall the definition of a signature of knowledge of the double discrete logarithm:

Definition 10 *A signature of knowledge of a double discrete logarithm of y to the bases g and a, on message m, with security parameter l denoted*

$$SKLOGLOG_l[\alpha \mid y = g^{(a^\alpha)}](m)$$

is an $(l+1)$-tuple

$$(c, s_1, \ldots, s_l) \in \{0,1\}^l \times \mathcal{Z}^l$$

satisfying the equation

$$c = \mathcal{H}_l(m,y,g,a,P_1,\ldots,P_l), \ \textit{where } P_i = \begin{cases} g^{(a^{s_i})} & \textit{if } c[i]=0 \\ y^{(a^{s_i})} & \textit{otherwise} \end{cases}$$

To obtain a blind $SKLOGLOG_l[\alpha \mid y = g^{(a^\alpha)}](m)$, we can use very similar techniques to the ones above. Here is a blind protocol for $SKLOGLOG$:

User Round 0: User wants message m signed and sends a sign request to the signer.

Signer Round 1:

1. For $1 \le i \le l$, generate random $2^\lambda \le r_i \le 2^{\lambda+\mu} - 1$.
2. Set $P_i := g^{(a^{r_i})}$.
3. Send $\{P_i\}$ to the user, committing to them.

User Round 2:

1. Obtain a random permutation $\sigma : \{1,\ldots,l\} \mapsto \{1,\ldots,l\}$ and set $Q_i := P_{\sigma(i)}$.
2. For $1 \le i \le l$, generate random $2^{\lambda+\mu} \le b_i \le 2^{\lambda+2\mu} - 1$, and set $R_i := Q_i^{(a^{b_i})}$.
3. Calculate $c := \mathcal{H}_l(m,y,g,R_1,\ldots,R_l)$.
4. Calculate c' such that $c'[i] = c[\sigma^{-1}(i)]$.
5. Send c' to the signer.

Signer Round 3:

1. Compute, for $1 \le i \le l$,

$$t_i = \begin{cases} r_i & \text{if } c'[i]=0 \\ r_i - x (\text{mod } n) & \text{otherwise} \end{cases}$$

2. Send $\{t_i\}$ to the user.

User Round 4:

1. Verify that

$$P_i = \begin{cases} g^{(a^{t_i})} & \text{if } c'[i]=0 \\ y^{(a^{t_i})} & \text{otherwise} \end{cases}$$

2. Compute $s_i := t_{\sigma(i)} + a_i$, $1 \leq i \leq l$.

3. Output $(c, s_1, \ldots, s_l)$. This constitutes the signature on m.

We now prove that our protocol is both correct, and produces a blind signature.

Lemma 4 (Correctness) *The protocol given above produces a valid signature of knowledge. That is,*

$$(c, s_1, \ldots, s_l) = SKLOGLOG_l[\alpha \mid y = g^{a^\alpha}](m).$$

Proof: The tuple $(c, s_1, \ldots, s_l)$ is a signature on m, because

$$c = \mathcal{H}_l(m,y,g,a,R_1,\ldots,R_l)$$

where

$$R_i = Q_i^{(a^{a_i})} = P_{\sigma(i)}^{(a^{a_i})} = \begin{cases} g^{(a^{t_{\sigma(i)}})(a^{a_i})} = g^{(a^{s_i})} & \text{if } c'[\sigma(i)] = c[i] = 0 \\ y^{(a^{t_{\sigma(i)}})(a^{a_i})} = y^{(a^{s_i})} & \text{if } c'[\sigma(i)] = c[i] = 1 \end{cases}$$

which satisfies the definition of $SKLOGLOG_l$.

Lemma 5 (Blindness) *In the random oracle model, the protocol described above produces a blind signature, that is, $(c, s_1, \ldots, s_l)$ is a signature on m that cannot be linked to the signer's view of the protocol*

Proof: The proof is almost identical to the proof in the $SKLOG$ case, so we omit the details. In particular, the signature $(c, s_1, \ldots, s_l)$ cannot be linked to the signer's view of the protocol, because σ blinds c and the $a_1, \ldots, a_l$ blind $s_1, \ldots, s_l$.

5.4 Blind Signature of Knowledge of the Root of Discrete Logarithm

We now give a protocol which gives a blind signature of knowledge of the root of the discrete logarithm. Recall the definition of $SKROOTLOG$:

Definition 11 *A signature of knowledge of an e-th root of the discrete logarithm of y to the base g, on message m, denoted* $SKROOTLOG_l[\alpha \mid y = g^{(\alpha^e)}](m)$ *, is an* $(l+1)$*-tuple* $(c, s_1, \ldots, s_l) \in \{0,1\}^l \times \mathcal{Z}_n^{*l}$ *satisfying the following equation:*

$$c = \mathcal{H}_l(m,y,g,e,P_1,\ldots,P_l), \text{ where } P_i = \begin{cases} g^{(s_i^e)} & \text{if } c[i]=0 \\ y^{(s_i^e)} & \text{otherwise} \end{cases}$$

Using similar techniques to the ones mentioned above, we obtained the following blind protocol for $SKROOTLOG$:

User Round 0: Want message m signed. Send a sign request to the signer.

Signer Round 1:

1. For $1 \le i \le l$, generate random $r_i \in \mathcal{Z}_n^*$.
2. Set $P_i := g^{(r_i^e)}$.
3. Send $\{P_i\}$ to the user, thus committing to them.

User Round 2:

1. Obtain a random permutation $\sigma : \{1,\ldots,l\} \mapsto \{1,\ldots,l\}$ and set $Q_i := P_{\sigma(i)}$.
2. For $1 \le i \le l$, generate random $a_i \in \mathcal{Z}_n^*$ and set $R_i := Q_i^{(a_i^e)}$.
3. Calculate $c := \mathcal{H}_l(m,y,g,R_1,\ldots,R_l)$.
4. Calculate c' such that $c'[i] = c[\sigma^{-1}(i)]$.
5. Send c' to the signer.

Signer Round 3:

1. Compute, for $1 \le i \le l$,

$$t_i = \begin{cases} r_i & \text{if } c'[i] = 0 \\ r_i/x \pmod{n} & \text{otherwise} \end{cases}$$

2. Send $\{t_i\}$ to the user.

User Round 4:

1. Verify that
$$P_i = \begin{cases} g^{(t_i^e)} & \text{if } c'[i] = 0 \\ y^{(t_i^e)} & \text{otherwise} \end{cases}$$
2. Compute $s_i := t_{\sigma(i)} * a_i$, $1 \leq i \leq l$.
3. Output $(c, s_1, \ldots, s_l)$. This is the signature on m.

We now prove that our protocol is correct and produces a blind signature.

Lemma 6 (Correctness) *The protocol given above produces a valid signature of knowledge. That is,*

$$(c, s_1, \ldots, s_l) = SKROOTLOG_l[\alpha \mid y = g^{(\alpha^e)}](m).$$

Proof: The tuple $(c, s_1, \ldots, s_l)$ is a signature on m, because

$$c = \mathcal{H}_l(m, y, g, e, R_1, \ldots, R_l)$$

where

$$\begin{aligned} R_i &= Q^{(a_i^e)} && (5.6) \\ &= P_{\sigma(i)}^{(a_i^e)} && (5.7) \\ &= \left(g^{r^e_{\sigma(i)}}\right)^{(a_i^e)} && (5.8) \\ &= g^{(t_{\sigma(i)} x^{c'[\sigma(i)]} a_i)^e} && (5.9) \\ &= g^{(s_i x^{c[i]})^e}. && (5.10) \end{aligned}$$

Which satisfies the definition of $SKROOTLOG_l$ $\mathcal{QED}$

Lemma 7 (Blindness) *In the random oracle model, the protocol described above produces a blind signature, that is, $(c, s_1, \ldots, s_l)$ is a signature on m that cannot be linked to the signer's view of the protocol*

Proof: The proof is almost identical to the proof in the $SKLOG$ case, so we omit the details. In particular the signature $(c, s_1, \ldots, s_l)$ cannot be linked to the signer's view of the protocol, because σ blinds c and the $a_1, \ldots, a_l$ blind $s_1, \ldots, s_l$. $\mathcal{QED}$

Having provided blinded variants of the $SKLOG$, $SKLOGLOG$, and $SKROOTLOG$ protocols, we can now proceed to use them in building a group blind digital signature scheme.

Chapter 6

Our Group Blind Digital Signature Scheme

6.1 Introduction

Having discussed all the relevant building blocks, we are now in a position to give the details of our group blind digital signature scheme. Our scheme involves taking the scheme of Camenisch and Stadler [9] and modifying it so that it has the blindness property as well. We primarily focus our efforts on the `Sign` protocol in their scheme, and show how to modify it so it is blind. Moreover, we were able to make our modifications in such a manner that the rest of the scheme stays the same. This chapter is organized as follows. In the next section, we present the details of our scheme, and show how to modify the `Sign` protocol of Camenisch and Stadler. In the subsequent section, we analyze our scheme. We prove that the scheme is both secure and blind. Finally we talk about efficiency, and some possible extensions.

6.2 Our Construction

Our construction only involves modifying the `Sign` protocol of Camenisch and Stadler's scheme [9]. We make use of our blind signatures of knowledge as the fundamental building blocks of our scheme.

6.2.1 Sign and Verify

Unlike the CS97 scheme [9], our signature construction protocol is blind.

Modified Sign Protocol:

The user makes a sign request on a message m, and the signer responds in the following manner:

1. Obtain $q \in_R \mathcal{Z}_n^*$ and set
$$\begin{aligned} \tilde{g} &:= g^q \text{ and} \\ \tilde{z} &:= \tilde{g}^y. \end{aligned} \tag{6.1}$$

2. Obtain random
$$2^\lambda \leq u_i \leq 2^{\lambda+\mu} - 1, \text{ for } 1 \leq i \leq l$$
and set
$$P_i^{SKLOGLOG} := \tilde{g}^{(a^{u_i})}, \text{ for } 1 \leq i \leq l. \tag{6.2}$$

3. Obtain random $v_i \in \mathcal{Z}_n^*$, for $1 \leq i \leq l$ and set
$$P_i^{SKROOTLOG} := \tilde{g}^{(v_i^e)}, \text{ for } 1 \leq i \leq l. \tag{6.3}$$

4. Send $(\tilde{g}, \tilde{z}, \{P_i^{SKLOGLOG}\}, \{P_i^{SKROOTLOG}\})$ to the user.

In turn, the user does the following:

1. Obtains $b \in_R \{1 \ldots 2^\lambda - 1\}$, and $f \in_R \mathcal{Z}_n^*$, and sets $w := (af)^{eb} \pmod{n}$.

2. Set
$$\begin{aligned} \hat{g} &:= \tilde{g}^w, \\ \hat{z} &:= \tilde{z}^w, \\ \hat{P}_i^{SKLOGLOG} &:= (P_i^{SKLOGLOG})^w, \text{ and} \\ \hat{P}_i^{SKROOTLOG} &:= (P_i^{SKROOTLOG})^w \end{aligned} \tag{6.4}$$

3. Execute rounds 2, 3, and 4 of the blind $SKLOGLOG$ protocol, and take $\{\hat{P}_i^{SKLOGLOG}\}$ as the commitment values. Adjust the responses $\{t_i^{SKLOGLOG}\}$ by adding eb.

4. Execute rounds 2, 3, and 4 of the blind $SKROOTLOG$ protocol, and take $\{\hat{P}_i^{SKROOTLOG}\}$ as the commitment values. Adjust the responses $\{t_i^{SKROOTLOG}\}$ by multiplying by $(af)^b$.

We have now obtained:

$$\begin{aligned} V_1 &= SKLOGLOG_l[\alpha \mid \hat{z} = \hat{g}^{a^\alpha}](m) \text{ and} \\ V_2 &= SKROOTLOG_l[\beta \mid \hat{z}\hat{g} = \hat{g}^{\beta^e}](m) \end{aligned} \tag{6.5}$$

The resulting signature on the message m consists of $(\hat{g}, \hat{z}, V_1, V_2)$ and can be verified by checking correctness of V_1 and V_2.

6.2.2 Analysis of Our Construction

Unforgeability

Lemma 8 *In the random oracle model, the signature produced by the above protocol is not forgeable. Specifically, only a valid group member could have issued this signature.*

Proof: We show that you must be a group member in order to issue a valid signature. This holds because, in the random oracle model, V_1 proves that the signer knows a membership key and V_2 proves that the signer knows a membership certificate corresponding to this key. That is, V_1 shows that

$$\hat{z} = \hat{g}^{a^\alpha}$$

and therefore:

$$\hat{z}\hat{g} = \hat{g}^{(a^\alpha+1)} \tag{6.6}$$

for an integer α that the signer knows. On the other hand, V_2 proves that

$$(a^\alpha + 1) = \beta^e \tag{6.7}$$

for some β that the signer knows. Under our hardness assumption on the forgeability of membership certificates, this can only happen if the signer is a group member, α is his secret key, and β is his membership certificate. $\mathcal{QED}$

Blindness of the Resulting Signature

Lemma 9 *In the random oracle model, the resulting signature produced by the protocol above*

$$(\hat{\mathrm{g}}, \hat{\mathrm{z}}, V_1, V_2)$$

is blind. It is a signature on m that cannot be linked to the signer's view of the protocol.

Proof: The signer's input into the protocol has been blinded by blinding $\tilde{g}$ and $\tilde{z}$ into random $\hat{g}$ and $\hat{z}$ by the *random* blinding factor w. Moreover, the two signatures of knowledge that were constructed are blind; the values in contained in these signatures are uniformly distributed among all possible valid signatures, and the values contained are *independent* of $\tilde{g}$ and $\tilde{z}$. Therefore, the resulting signature $(\hat{g}, \hat{z}, V_1, V_2)$ cannot be linked to the signer's view of the protocol. $\mathcal{QED}$

Efficiency of the Proposed Scheme

The proposed scheme is asymptotically as efficient as the basic group signature scheme proposed by Camenisch and Stadler. When the security parameter l is treated as a fixed constant, all of the described operations, except `Open`, take constant time. The `Open` protocol takes time linear in the size of the group. All signatures take constant space, and the communication complexity per signature is constant. The computational, space, and communication complexities are linear in the security parameter l.

Security of the Proposed Scheme

Our scheme is as secure as Camenisch and Stadler's Basic Group Signature Scheme [9]. The security of the scheme is based on the assumptions that the discrete logarithm, double discrete logarithm, and the roots of discrete logarithm problems are hard. In addition, it is based on the security of the Schnorr [36] and the RSA [33] signature schemes and on the additional assumption due to Camenisch and Stadler [9] that computing membership certificates is hard. Finally, the security of our scheme is proved under the random oracle model.

Improvements on the Efficiency

Camenisch and Stadler [9] also proposed a group signature scheme in which the computational complexity of the `Open` algorithm is constant in the size of the group. This scheme can also be extended to a group blind signature scheme using our techniques. We chose to discuss this alternate scheme in the appendix since it is less secure and is known to be broken for certain values of its parameters [9].

Appendix A

Another Group Blind Digital Signature Scheme

In their original paper, Camenisch and Stadler [9] presented two group digital signature schemes. Developing blinded variants for the first of these two schemes was the central focus of this thesis. In this appendix, we develop blinded variants for the second of these two schemes. This alternate scheme has the advantage that the `Open` procedure takes only time *constant* in the size of the group, whereas in the first scheme the `Open` procedure required linear time. Unfortunately, this second scheme has uncertain security, and is known to be broken for certain values of the parameters [9]. For this reason, we relegated the discussion of this scheme to the appendix.

This appendix is organized as follows. In the next section we present the signature of Knowledge of Representation, and give a blinded variant. In section 2, we discuss the Signature of Knowledge of Roots of Logarithms of Part of Representation. Finally, in section 3, we present the alternate group blind digital signature scheme.

A.1 Signature of Knowledge of Representation

Definition 12 *A signature of the knowledge of representation of $y_1, \ldots, y_w$ to the bases $g_1, \ldots, g_v$, with security parameter l, on the message m, denoted as:*

$$SKREP_l[(\alpha_1, \ldots, \alpha_u) \mid (y_1 = \prod_{j=1}^{q_1} g_{b_{1j}}^{\alpha_{e_{1j}}}) \wedge \ldots \wedge (y_w = \prod_{j=1}^{q_w} g_{b_{wj}}^{\alpha_{e_{wj}}})](m)$$

where the indices $e_{ij} \in \{1, \ldots, u\}$ refer to the elements $\alpha_1, \ldots, \alpha_u$ and the indices $b_{ij} \in \{1, \ldots, u\}$ refer to the base elements $g_1, \ldots, g_v$, consists of a $(u \times l + 1)$-tuple $(c, s_{1,1}, \ldots, s_{l,u}) \in \{0,1\}^l \times \mathcal{Z}_n^{l \times u}$ satisfying the equation

$$\begin{aligned} c = \mathcal{H}_l(\ & m, y_1, \ldots, y_w, g_1, \ldots, g_v, \{\{e_{ij}, b_{ij}\}_{j=1}^{q_i}\}_{i=1}^{w}, \\ & y_1^{c[1]} \textstyle\prod_{j=1}^{q_1} g_{b_{1j}}^{s_{1e_{1j}}}, \ldots, y_1^{c[1]} \prod_{j=1}^{q_w} g_{b_{wj}}^{s_{1e_{wj}}}) \\ & , \ldots, \\ & y_1^{c[l]} \textstyle\prod_{j=1}^{q_1} g_{b_{1j}}^{s_{le_{1j}}}, \ldots, y_1^{c[l]} \prod_{j=1}^{q_w} g_{b_{wj}}^{s_{le_{wj}}}). \end{aligned}$$

In order to be comprehensible, this definition requires an example. Suppose we want to obtain a signature on m that is a signature of knowledge that we know $\alpha_1, \alpha_2, \alpha_3$ such that, given y_1, y_2, y_3 and g_1, g_2, g_3, the following holds:

$$\begin{aligned} y_1 &= g_1^{\alpha_1} g_1^{\alpha_2} g_2^{\alpha_1} g_3^{\alpha_2} g_3^{\alpha_3} \\ y_2 &= g_1^{\alpha_2} g_2^{\alpha_2} g_2^{\alpha_3} g_3^{\alpha_1} g_3^{\alpha_3} \\ y_3 &= g_1^{\alpha_3} g_2^{\alpha_1} g_3^{\alpha_2} g_3^{\alpha_3}. \end{aligned}$$

We do this by constructing

$$\begin{aligned} SKREP_l[(\alpha_1, \alpha_2, \alpha_3) \mid & \\ y_1 &= g_1^{\alpha_1} g_1^{\alpha_2} g_2^{\alpha_1} g_3^{\alpha_2} g_3^{\alpha_3} \wedge \\ y_2 &= g_1^{\alpha_2} g_2^{\alpha_2} g_2^{\alpha_3} g_3^{\alpha_1} g_3^{\alpha_3} \wedge \\ y_3 &= g_1^{\alpha_3} g_2^{\alpha_1} g_3^{\alpha_2} g_3^{\alpha_3}](m), \end{aligned}$$

that is, by producing $(c, s_{1,1}, s_{1,2}, s_{1,3}, \ldots, s_{l,1}, s_{l,2}, s_{l,3})$ such that

$$\begin{aligned} c = \mathcal{H}_l \quad & (m,y_1,y_2,y_3,g_1,g_2,g_3, \\ & 1,1,2,1,1,2,2,3,3,3,2,1,2,2,3,2,1,3,3,3,3,1,1,2,2,3, \\ & y_1^{c[1]} g_1^{s_1} g_1^{s_2} g_2^{s_1} g_3^{s_2} g_3^{s_3}, y_2^{c[1]} g_1^{s_2} g_2^{s_2} g_2^{s_3} g_3^{s_1} g_3^{s_3}, y_3^{c[1]} g_1^{s_3} g_2^{s_1} g_3^{s_2} g_3^{s_3} \\ & ,\ldots, \\ & y_1^{c[l]} g_1^{s_1} g_1^{s_2} g_2^{s_1} g_3^{s_2} g_3^{s_3}, y_2^{c[l]} g_1^{s_2} g_2^{s_2} g_2^{s_3} g_3^{s_1} g_3^{s_3}, y_3^{c[l]} g_1^{s_3} g_2^{s_1} g_3^{s_2} g_3^{s_3}). \end{aligned}$$

SKREP can be constructed as follows: the signer generates $r_{ij} \in_R \mathcal{Z}_n$, $1 \leq i \leq l$, $1 \leq j \leq u$, and computes

$$\begin{aligned} c = \mathcal{H}_l(\quad & m,y_1,\ldots,y_w,g_1,\ldots,g_v,\{\{e_{ij}, b_{ij}\}_{j=1}^{q_i}\}_{i=1}^{w}, \\ & \textstyle\prod_{j=1}^{q_1} g_{b_{1j}}^{r_{1e_{1j}}}, \ldots, \prod_{j=1}^{q_w} g_{b_{wj}}^{r_{1e_{wj}}}) \\ & ,\ldots, \\ & \textstyle\prod_{j=1}^{q_1} g_{b_{1j}}^{r_{le_{1j}}}, \ldots, \prod_{j=1}^{q_w} g_{b_{wj}}^{r_{le_{wj}}}) \end{aligned}$$

and then sets $s_{ij} := r_{ij} - c[i]\alpha_j \pmod{n}$, $1 \leq i \leq l$, $1 \leq j \leq u$.

A blind signature of knowledge of representation can be constructed by generalizing the protocol for blinding *SKLOG*:

User Round 0: User wants message m signed and sends a sign request to the signer.

Signer Round 1:

1. Obtain random $r_{1,1}, \ldots, r_{l,u}$.
2. Set $P_{ij} := \prod_{f=1}^{q_j} g_{b_{jf}}^{r_{ie_{jf}}}$.
3. Send $\{P_{ij}\}$ to the user, thus commiting to the $\{P_{ij}\text{'s}\}$.

User Round 2:

1. Obtain a random permutation $\sigma : \{1, \ldots, l\} \mapsto \{1, \ldots, l\}$ and set $Q_{ij} := P_{\sigma(i)j}$, $1 \leq i \leq l$. (σ will be used to blind the result of $\mathcal{H}$.)
2. Obtain random $a_{1,1}, \ldots, a_{l,u}$, and set $R_{ij} := Q_{ij} \prod_{f=1}^{q_j} g_{b_{jf}}^{a_{ie_{jf}}}$.
3. Calculate $c := \mathcal{H}_{\updownarrow}(m,y_1,\ldots,y_w,g_1,\ldots,g_v,indices,R_{1,1},\ldots,R_{l,u})$
4. Calculate c' such that $c'[i] = c[\sigma^{-1}(i)]$.

5. Send c' to the signer.

Signer Round 3:

1. Using secret $x_1, \ldots, x_u$, compute for $1 \le i \le l$, $1 \le j \le u$,

$$t_{ij} = \begin{cases} r_{ij} & \text{if } c'[i] = 0 \\ r_{ij} - \alpha_j \pmod{n} & \text{otherwise} \end{cases}$$

2. Send $\{t_{ij}\}$ to the user.

User Round 4:

1. Verify that $P_{ij} := y_j^{c[i]} \prod_{f=1}^{q_j} g_{b_{jf}}^{t_{ie_{jf}}}$.
2. Compute $s_{ij} := t_{\sigma(i)j} + a_{ij} \pmod{n}$.
3. Output $(c, s_{1,1}, \ldots, s_{l,u})$.

A.2 Signature of Knowledge of Roots of Logarithms of Part of Representation

This is another signature of knowledge based on *SKREP* discussed above. Consider the following definition:

Definition 13 *An signature of knowledge of the e-th root of the g-part of a representation of v to the bases h and g, with respect to security parameter l, denoted*

$$SKROOTREP_l[(\alpha, \beta) \mid v = h^{\alpha} g^{\beta^e}](m)$$

consists of an $(e-1)$*-tuple* $(v_1, \ldots, v_{e-1}) \in G^{e-1}$ *and of a signature of knowledge*

$$\begin{aligned} U = SKREP_l[(\gamma_1, \ldots, \gamma_e, \delta) \mid & \\ v_1 &= h^{\gamma_1} g^{\delta}, v_2 = h^{\gamma_2} v_1^{\delta}, \\ &\vdots \\ v_{e-1} &= h^{\gamma_{e-1}} v_{e-2}^{\delta}, \\ v &= h^{\gamma_e} v_{e-1}^{\delta}](m). \end{aligned}$$

For a small e, a signature like this is efficient to implement. Why would the verifier be convinced that the prover knows (α, β), if he knows $(\gamma_1, \ldots, \gamma_e, \delta)$? Because

$$v = h^{\gamma_e}(h^{\gamma_{e-1}}(\ldots h^{\gamma_2}(h^{\gamma_1} g_\delta)^\delta \ldots)^\delta)^\delta = h^{\gamma_e + \gamma_{e-1}\delta + \ldots + \gamma_2 \delta^{e-2} + \gamma_1 \delta^{e-1}} g^{\delta^e} = h^\alpha g^{\beta^e}.$$

If (α, β) is known, such signature is computed as follows:

1. For $1 \le i \le e-1$, generate $r_i \in_R \mathcal{Z}_n^*$ and set $v_i := h^{r_i} g^{x^i}$ and make them available to the verifier. Note that by doing so no information is leaked because $\{v_i\}$ are random group elements.

2. Compute signature of knowledge U by setting, for $1 \le i \le e-1$, $\gamma_i := r_i$ and computing γ_e such that

$$\gamma_e + \gamma_{e-1}\delta + \ldots + \gamma_2 \delta^{e-2} + \gamma_1 \delta^{e-1} = \alpha.$$

This signature of knowledge can be made blind also through similar techniques to those used for blinding $SKREP_l$.

A.3 Another Group Blind Signature Scheme

We present an alternate group blind digital signature scheme based on the second scheme proposed by Camenisch and Stadler [9] The scheme has the advantage that the time complexity of Open is constant. Unfortunately, the security of this new scheme is not clear; for this reason we relegated the discussion of this scheme to the appendix.

A.3.1 Setup

The group manager computes the following values:

- An RSA modulus n and two public exponents $e_1, e_2 > 1$, such that they are relatively prime to $\phi(n)$.

- Two integers $f_1, f_2 > 1$ for which e_1-th and e_2-th roots cannot be computed without knowing the factorization of n.
- A cyclic group $G = \langle g \rangle$ of order n in which computing discrete logarithms is infeasible.
- An element $h \in G$ for which the discrete logarithm to the base g must not be known.
- The group manager's randomly chosen secret key $\rho \in \mathcal{Z}_n$.
- $y_R = h^\rho$.

The group's public key consists of $\mathcal{Y} = (n, e_1, e_2, f_1, f_2, G, g, h, y_R)$. The group's private key is the factorization of n and the value of ρ.

A.3.2 Join

To become a group member, Alice computes her membership key as follows:

- $x \in_R \mathcal{Z}_n^*$
- $y = x^{e_1} \pmod{n}$
- $z = g^y$

A certificate in this scheme is of the form

$$v = (f_1 y + f_2)^{1/e_2} \pmod{n}.$$

We omit the details of the protocol since they are the same as in the original Camenisch and Stadler paper [9].

A.3.3 Sign and Verify

To obtain a group blind signature, the user and the signer go through an interactive protocol to generate a blinded signature that consists of:

- $\tilde{z} = h^r g^y$ for $r \in_R \mathcal{Z}_n^*$ (in the end, r is known to neither the signer, nor the user, because the signer only knows the random element he generated, and the user only knows the random element used to blind that random element. We omit the details of blinding here because they are exactly the same as in the blind signatures we described throughout the paper.)
- $d = y_R^r$
- $V_1 = SKROOTREP[(\alpha, \beta) \mid \tilde{z} = h^\alpha g^{\beta^{e_1}}](m)$
- $V_2 = SKROOTREP[(\gamma, \delta) \mid \tilde{z}^{f_1} g^{f_2} = h^\gamma g^{\delta^{e_2}}](m)$
- $V_3 = SKREP[(\epsilon, \zeta) \mid d = y_R^\epsilon, \tilde{z} = h^\epsilon g^\zeta](m)$

The signature V_1 proves that the signer knows the signer knows a representation of (α, β^{e_1}) of $\tilde{z}$ to the bases g and h and that she knows the e_1-th root of the $g - part$ of this representation (technically, none of the two parties constructing the signature knows all about this representation, but together they know it). The signature V_2 proves the knowledge of a membership certificate corresponding to β. The signature V_3 is a modified ElGamal encryption of g^y encrypted using the group manager's secret key and enables the group manager to open signatures. See the original paper of Camenisch and Stadler [9] for a more detailed description of this group signature.

A.3.4 Open

When the group manager wants to open a signature $(\tilde{z}, d, V_1, V_2.V_3)$ on message m, he computes $\hat{z} = \tilde{z}/d^{1/\rho}$ which corresponds to the signer's membership key z. To prove that z is indeed encrypted in $\tilde{z}$ and d, the group manager computes $SKREP[\alpha \mid \tilde{z} = zd^\alpha, h = y_R^\alpha]('')$ which he can do because $1/\alpha$ corresponds to his administration key.

Bibliography

[1] G. Ateniese and G. Tsudik. A coalition-resistant group signature scheme. Technical Report, In Submission, October 1998.

[2] G. Ateniese and G. Tsudik. Some open issues and new directions in group signature. Proceedings of the International Conference on Financial Cryptography '99 (To Appear), November 1998.

[3] G. Ateniese and G. Tsudik. Group signatures a' la carte. In *ACM Symposium on Discrete Algorithms*, January 1999.

[4] Mihir Bellare and Silvio Micali. How to sign given any trapdoor function. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 32–42, Chicago, 1988. ACM.

[5] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73, Fairfax, 1993. ACM.

[6] Jan Camenisch. Efficient and generalized group signatures. In *Proc. EUROCRYPT 97*, pages 465–479. Springer-Verlag, 1997. Lecture Notes in Computer Science No. 1233.

[7] Jan Camenisch, Ueli Maurer, and Markus Stadler. Digital payment systems with passive anonymity-revoking trustees. *Journal of Computer Security*, 5(1), 1997.

[8] Jan Camenisch and Markus Michels. A group signature scheme with improved efficiency. In *Advances in Cryptology-ASIACRYPT '98*, pages 160–174. Springer-Verlag, 1998. Lecture Notes in Computer Science No. 1514.

[9] Jan Camenisch and Markus Stadler. Efficient group signatures for large groups. In *Proc. CRYPTO 97*, pages 410–424. Springer-Verlag, 1997. Lecture Notes in Computer Science No. 1294.

[10] Agnes Chan, Yair Frankel, and Yiannis Tsiounis. Easy come – easy go divisible cash. In *Proc. EUROCRYPT 98*. Springer-Verlag, 1998. Lecture Notes in Computer Science No. 1403.

[11] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In S. Goldwasser, editor, *Proc. CRYPTO 88*, pages 319–327. Springer-Verlag, 1988. Lecture Notes in Computer Science No. 403.

[12] David Chaum. Blind signatures for untraceable payments. In R. L. Rivest, A. Sherman, and D. Chaum, editors, *Proc. CRYPTO 82*, pages 199–203, New York, 1983. Plenum Press.

[13] David Chaum. Blind signature system. In D. Chaum, editor, *Proc. CRYPTO 83*, pages 153–153, New York, 1984. Plenum Press.

[14] David Chaum and Eugène van Heyst. Group signatures. In *Proc. EUROCRYPT 91*, pages 257–265. Springer-Verlag, 1991. Lecture Notes in Computer Science No. 547.

[15] L. Chen and T. P. Pedersen. New group signature schemes (extended abstract). In *Proc. EUROCRYPT 94*, pages 171–181. Springer-Verlag, 1994. Lecture Notes in Computer Science No. 547.

[16] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, IT-22:644–654, November 1976.

[17] T. ElGamal. A public key cryptosystem and signature scheme based on discrete logarithms. In *Proc. CRYPTO 84*, 1984.

[18] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A.M. Odlyzko, editor, *Proc. CRYPTO 86*, pages 186–194. Springer-Verlag, 1987. Lecture Notes in Computer Science No. 263.

[19] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In J. Seberry and J. Pieprzyk, editors, *Advances in Cryptology - AUSCRYPT '92*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer-Verlag, 1992.

[20] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17(2):281–308, April 1988.

[21] A. Juels, M. Luby, and R. Ostrovsky. Security of blind digital signatures. In *Proc. CRYPTO 97*, Lecture Notes in Computer Science, pages 150–164. Springer-Verlag, 1997. Lecture Notes in Computer Science No. 1294.

[22] J. Killian and E. Petrank. Identity escrow. In *Proc. CRYPTO 98*, 1998.

[23] Laurie Law, Susan Sabett, and Jerry Solinas. How to make a mint: the cryptography of anonymous electronic cash. National Security Agency, Office of Information Security Research and Technology, Cryptology Division, June 1996.

[24] A. Lysyanskaya and Z. Ramzan. Group blind digital signatures: A scalable solution to electronic cash. In *Proceedings of the International Conference on Financial Cryptography*, 1998.

[25] Gary L. Miller. Riemann's hypothesis and tests for primality. *Journal of Computer and System Sciences*, 13(2):300–317, December 1976.

[26] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proc. 21st ACM Symp. on Theory of Computing*, pages 33–43, Seattle, 1989. ACM.

[27] National Institute of Standards and Technology (NIST). *FIPS Publication 46-1: Data Encryption Standard*, January 22, 1988. Originally issued by National Bureau of Standards.

[28] National Institute of Standards and Technology (NIST). *FIPS Publication 180: Secure Hash Standard (SHS)*, May 11, 1993.

[29] T. Okamoto. Provable secure and practical identification schemes and corresponding signature signature schemes. In *CRYPTO92*, pages 31–53. SPVER, 1992. Lecture Notes in Computer Science No. 740.

[30] David Pointcheval and Jacques Stern. Provably secure blind signature schemes. In M.Y. Rhee and K. Kim, editors, *Advances in Cryptology-ASIACRYPT '96*, pages 252–265. Springer-Verlag, 1996. Lecture Notes in Computer Science No. 1163.

[31] Zulfikar Ramzan. Group blind signatures a' la carte. Unpublished Manuscript, September 1998.

[32] Ronald L. Rivest. The MD5 message-digest algorithm. Internet Request for Comments, April 1992. RFC 1321.

[33] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[34] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proc. 22nd ACM Symp. on Theory of Computing*, pages 387–394, Baltimore, Maryland, 1990. ACM.

[35] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* John Wiley & Sons, New York, 1993.

[36] C. P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *Proc. CRYPTO 89*, pages 239–252. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 435.

[37] Jacque Traore. Weaknesses of the "coalition resistant" group signature scheme. Unpublished Manuscript, November 1998.

[38] Peter Wayner. *Digital Cash: Commerce on the Net.* Academic Press, 1996.