Deniable Ring Signatures

by

Eitan Reich

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

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2007
[June 2007]

Author .... [illegible]
Department of Electrical Engineering and Computer Science
May 8, 2007

Certified by .... [illegible]
Silvio Micali
Professor
Thesis Supervisor

Accepted by .... [illegible]
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Deniable Ring Signatures

by

Eitan Reich

Submitted to the Department of Electrical Engineering and Computer Science
on May 8, 2007, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

Ring Signatures were developed by Rivest, Shamir and Tauman, in a paper titled *How to Leak a Secret*, as a cryptographically secure way to authenticate messages with respect to ad-hoc groups while still maintaining the signer's anonymity. While their initial scheme assumed the existence of random oracles, in 2005 a scheme was developed that does not use random oracles and meets the strongest security definitions known in the literature. We argue that this scheme is not *deniable*, meaning if someone signs a message with respect to a ring of possible signers, and at a later time the secret keys of all of the possible signers are confiscated (including the author), then the author's anonymity is no longer guaranteed. We propose a modification to the scheme that guarantees anonymity even in this situation, using a scheme that depends on ring signature users generating keys that do not distinguish them from other users who did not intend to participate in ring signature schemes, so that our scheme can truly be called a *deniable* ring signature scheme.

Thesis Supervisor: Silvio Micali
Title: Professor

Acknowledgments

I would like to thank my advisor, Silvio Micali, for his insightful ideas and guidance through all the stages of research. I also thank Tony Eng and Masayuki Abe for introducing me to cryptography research. My friends, Igor Ginzburg, Noam Yemini, Koorosh Elihu and Michael Star, have been a wonderful source of conversation and assistance in proofreading my thesis. Finally I would like to thank my parents for their constant support and always encouraging me to pursue my goals.

Contents

Chapter 1

Introduction

Ring signature schemes, first developed by Rivest, Shamir and Tauman [13, 14], allow a user to sign a message anonymously with respect to a ring (essentially a group or list) of possible signers as long as that list includes the actual signer. The basic security requirements for such a scheme are unforgeability and anonymity. Unforgeability means that someone must be in the ring to produce a ring signature for that ring and anonymity essentially means that the ring signature gives no more information about the signer's identity other than the fact that it is one of the members of the ring. Ring signatures differ from the seemingly related notion of group signatures in several main respects. Group signatures [4] include a central authority that can reveal the identity of a signer and the group of possible signers is fixed in advance. Ring signature schemes are much more flexible because there is no such central authority, and the members included in a ring of possible signers can be chosen at the time the message is signed and therefore need not be fixed. This means that someone that had no intention of ever participating in ring signature schemes may still be included in a ring signature as a possible signer. This flexibility leads to a variety of applications for ring signature schemes.

The canonical application for ring signature schemes is for whistle blowing, as stated in the title of the first work in this area, "How to leak a secret" [13]. For example, a member of the board of directors of a company may wish to make known the illegal activities going on in the company without revealing their own identity but still proving that the message came from a valid source. This board member could use ring signatures to sign their message with respect to a ring of possible signers that includes all of the board members. Another interesting application of ring signatures is for designated verifier e-mail. If a sender signs their e-mail message with respect to the ring of two users including the sender and receiver, then the receiver will be convinced that the message came from the claimed sender, but they will not be able to prove this to anyone else because the receiver could have produced this signature themselves. One final application we will mention here is electronic voting [11]. A vote can be a ring signature with respect to the ring of eligible voters so that votes can be made public and counted publicly without revealing the votes of individual people. The property of linkability is augmented onto basic ring signature schemes to ensure that people can't vote twice.

The original ring signature scheme in [13] elegantly uses public-key encryption to achieve anonymity and unforgeability. Any user who publishes a public key of the type used in the scheme can therefore be included in a ring signature even if they never anticipated participating in ring signatures. This flexibility was extended in [1] where it was shown that ring signature schemes could be based on public encryption keys even if the keys are not all of the same type. The main drawback of these schemes were their dependence on random oracles, a sort of idealized hash function that is used in theoretical work but is assumed to be replaced by a hash function in practice. This dependence is seen as a drawback since it was shown in [12] that the random oracle model is not secure, in the sense that there exist cryptosystems that are secure using random oracles, but are not secure whenever these oracles are replaced by real world hash functions. This motivates the recent results of [2] that first demonstrate the existence of ring signature schemes without random oracles, which are the basis for the work in this thesis.

In addition to developing a ring signature scheme that does not use random oracles, [2] give the strongest security requirements for unforgeability and anonymity known in the literature and show that their scheme meets these requirements. The two strongest anonymity requirements they propose are anonymity against attribution attacks and anonymity against full key exposure. The first of these requirements guarantees anonymity even if the adversary is given the secret keys (and randomness that generated these keys) for all but one of the possible signers in the ring. Anonymity against full key exposure is the stronger of the two requirements, guaranteeing anonymity even if the adversary knows the secret keys and randomness for *all* of the members of the ring. While the scheme presented in [2] meets the weaker of these two requirements, they propose a slight variation on their scheme that meets the stronger requirement, by having users generate encryption keys for which they themselves don't even know the secret key. We argue that this variation is inconsistent with the ad-hoc nature of ring signatures, because the other users included in the ring may actually know their own secret keys and therefore the real signer can be distinguished from the other members of the ring by being the only member who can not produce their secret key. We argue that in order for ring signatures to be deniable, all of the keys used by the signer must be indistinguishable from the keys of the other members of the ring, even in the situation where the secret keys are exposed. We build on the scheme of [2] to produce a scheme without random oracles that is anonymous with respect to full key exposure and depends on the use of keys that will not distinguish ring signature participants from those users who had no intention of participating in ring signature schemes.

Chapter 2

Preliminaries

In this section we present some of the notation and basic definitions we will be using throughout the paper. We will introduce some of the notation and definitions to describe probabilistic algorithms and some of the definitions needed to concisely describe the assumptions and results we will be working with.

2.1 Notation

Since we will often be working with probabilistic algorithms, we define $A(x)$, where A is a probabilistic algorithm and x is an input to that algorithm, as the distribution that assigns to every string y the probability that algorithm A on input x produces y. We will use $A(x;\omega)$ to refer to the deterministic computation of A on input x with ω as its random coins. When A is both probabilistic and runs in polynomial time, we will sometimes refer to it as a PPT (Probabilistic Polynomial Time algorithm). In general if S is a probability space, we may write $x \leftarrow S$ to mean that x is chosen at random from S.

When we say a function f is *negligible*, we mean that for every polynomial $p(n)$, there exists some n_0 such that $f(n) < \frac{1}{p(n)}$ for all $n > n_0$. In such a case we may also write $f(n) = \mathsf{negl}(n)$. Similarly we say that a function f is *non-negligible* if there exists some polynomial $p(n)$ such that $f(n) > \frac{1}{p(n)}$ for infinitely many n.

2.2 One-way Functions

Most of the results in modern cryptography are based on either assumptions about the hardness of some problems (such as number-theoretic problems) or standard assumptions such as the existence of one-way or trapdoor one-way functions or permutations. To summarize, a one-way function is a function that is easy to compute but hard to invert, while a trapdoor one-way function is a one-way function that is easy to invert if one knows some secret trapdoor information. We define one-way functions here:

Definition 1 (One-way Functions) *A* one-way function *is a function f that satisfies the following two properties:*

1. *f can be computed in polynomial time.*
2. *f is hard to invert, meaning for any non-uniform PPT A, for any auxiliary input z_n,*

$$Pr[x \leftarrow \{0,1\}^n; y \leftarrow A(f(x), z_n); f(x) = f(y)] = \mathsf{negl}(n)$$

.

2.3 Computational Indistinguishibility

Many of the results we will address in this paper are framed in terms of computational indistinguishibility, which is essentially a way to express the difficulty of distinguishing between two ensembles of probability distributions. For example, we will obtain pseudorandom numbers from one-way functions and express our results in the form of computational indistinguishibility to say that no efficient algorithm could tell the difference between pseudorandom numbers and truly random numbers. In general we define computational indistinguishibility in terms of an adversarial game:

Definition 2 (Computational Indistinguishibility) *Two ensembles of probability distributions,* $\{A_x\}_{x\in I}$ *and* $\{B_x\}_{x\in I}$ *are said to be* Computationally Indistinguishible *if for any non-uniform PPT D, for all* $x \in I \cap \{0,1\}^n$*, and any auxiliary input* z_n*,*

$$|Pr[y \leftarrow A(x); D(x,y,z_n) = 1] - Pr[y \leftarrow B(x); D(x,y,z_n) = 1]| = \mathit{negl}(n)$$

In such a case we may use the shorthand $\{A_x\} \approx \{B_x\}$.

Chapter 3

Previous Work

In this section we present the basic cryptographic primitives and constructions that will be used to construct the deniable ring signature scheme in the next section. The results we present are all framed in terms of adversarial games and the assumptions used are all stated in terms of generic complexity based assumptions, such as the existence of one way functions, or number theoretic assumptions, such as the difficulty of distinguishing quadratic residues from quadratic non-residues.

The first primitive we present, Pseudorandom Number Generators, provides the basis for the main innovation in our scheme. The next three sections, about Public-key Encryption, Digital Signatures, and Zero Knowledge Proofs, introduce primitives that are used in the Ring Signature scheme in [2], which is also the last subsection and the basis for our construction in the next section. For the basic primitives (the first four sections) we simply provide the definitions and theorems we will use later, while for the section presenting the scheme in [2] we provide more detail because we will be using the main ideas from it in our own construction rather than simply applying results.

3.1 Pseudorandom Generators

The goal of pseudorandom generators is to take random strings and expand them into longer strings that appear to be random. The result that we will use is that the existence of one-way functions implies the existence of pseudorandom generators. We first define pseudorandom generators as they were defined in [3, 7], as functions that stretch random strings of length n to strings of length polynomial in n that are indistinguishable from random:

Definition 3 (Pseudorandom Generator) *Let $f_n : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ be a polynomial time function ensemble for some polynomial $l(n) > n$. $\{f_n\}_{n \in \mathbb{N}}$ is a* Pseudorandom Generator *if the probability ensemble of f_n on random strings of length n is computationally indistinguishable from the probability ensemble of random strings of length $l(n)$.*

It was originally shown by Blum and Micali [3] that pseudorandom generators can be constructed from functions f that have a hard-core predicate b, meaning $b(x)$ can be computed easily given x but can not be computed given only $f(x)$. These pseudorandom generators can expand the random input by any polynomial amount by cleverly repeating the application of the hard-core predicate to obtain extra bits. Goldreich and Levin [7] then showed that any one-way function can be turned into a function with a hard-core predicate that can be used for pseudorandom generators as in [3]. This gives us the following theorem:

Theorem 4 *If One-way Functions exist, then there exist Pseudorandom Generators for any polynomial $l(n)$.*

3.2 Public-key Encryption

To construct ring signature schemes as in [2] we will be using public-key encryption schemes with the property that no feasible adversary can distinguish between encryptions of two different messages chosen by the adversary. This property, known as polynomial security against chosen plaintext attacks, has been shown to be equivalent to semantic security, which was presented by Goldwasser and Micali [8], and essentially means that no feasible adversary can obtain any partial information about a message given only an encryption of that message. While there exist encryption schemes that meet stronger security definitions, such as security against chosen ciphertext attacks, the requirements from [8] will be sufficient for constructing our ring signature schemes. We first define a probabilistic public-key encryption scheme:

Definition 5 (Public-key Encryption Scheme) *A* Public-key Encryption Scheme *is a triplet of PPTs (G, E, D) where*

- *G is a key generator that takes input 1^k, where k is the security parameter, and returns (PK, SK) where PK is the public key and SK is the secret key.*
- *E is an encryption algorithm that takes as input a message m and the public key PK of the recipient of the message, and returns a ciphertext $c = E_{PK}(m)$.*
- *D is a decryption algorithm that takes as input a ciphertext c and the secret key SK of the user and returns the original message $m = D_{SK}(c)$.*
- *For all key pairs (PK, SK) that G might produce, and for all messages m, $Pr[D_{SK}(E_{PK}(m)) = m] = 1$.*

We will now formally define polynomial security against chosen plaintext attacks in terms of an adversarial game. The idea is that the adversary, given the public key, will produce two messages that it claims to be able to distinguish between if given an encryption of one of them. After receiving an encryption of one of the messages, it will guess which message was encrypted and we will assert the probability that the adversary is correct is no more than negligibly far from $\frac{1}{2}$:

Definition 6 (Chosen Plaintext Security) *A public-key encryption scheme* (G, E, D) *is polynomially secure against* Chosen Plaintext *attacks if for any PPT A the probability* $p(n)$ *that A succeeds at the following game satisfies* $|p(n) - \frac{1}{2}| = \mathit{negl}(n)$*:*

1. *Keys* $(PK, SK) \leftarrow G(1^n)$ *are generated randomly and PK is given to A.*

2. *A returns two messages,* m_0 *and* m_1*.*

3. *A bit b is chosen at random and A is given* $c \leftarrow E_{PK}(m_b)$*.*

4. *A returns* $b' \in \{0, 1\}$ *and succeeds if* $b' = b$*.*

Since this security definition is equivalent to semantic security, we may use this definition when describing schemes that we assume to be semantically secure. Several semantically secure encryption systems have been developed under various assumptions, such as the number theoretic Quadratic Residuosity Assumption. We will use these results to construct ring signature schemes, and state the theorem to be used here:

Theorem 7 *Under the Quadratic Residuosity Assumption, there exist public-key encryption schemes that are semantically secure.*

3.3 Digital Signatures

In addition to public-key encryption, the ring signature scheme of [2] also utilizes digital signatures. However, unlike with encryption, the signature schemes they assume use the strongest security definitions in the literature. In particular they use signature schemes for which any adversary, even if given access to an oracle to produce sample signatures, has no more than a negligible chance of producing a forgery of a message for which it did not already query the oracle. This security requirement is known as existential unforgeability under adaptive chosen-message attacks. Before defining any security notions, we first define digital signature schemes as we did for public-key encryption schemes earlier:

Definition 8 (Digital Signature Scheme) *A* Digital Signature Scheme *is a triplet of PPTs (G, S, V) where*

- *G is a key generator that takes input 1^k, where k is the security parameter, and returns $(\mathsf{PK}, \mathsf{SK})$ where PK is the public key and SK is the secret key.*
- *S is a signing algorithm that takes as input a message m and the secret key SK of the user, and returns a signature $\sigma = S_{SK}(m)$.*
- *V is a verification algorithm that takes as input a signature σ and the public key PK of the claimed signer and returns a bit $b = V_{PK}(m, \sigma)$ where $b = 1$ (respectively 0) if the signature is valid (respectively invalid).*
- *For all key pairs $(\mathsf{PK}, \mathsf{SK})$ that G might produce, and for all messages m, $Pr[V_{PK}(m, S_{SK}(m)) = 1] = 1$.*

We will now define the security requirement we will use for signature schemes in this paper in terms of an adversarial game as we did for public-key encryption. The idea is that the adversary will be given a public key and access to a signing oracle and is then challenged to produce a forgery of any message for which he did not already query the oracle. The requirement is that the adversary must succeed with no more than negligible probability:

Definition 9 (Chosen Message Security) *A digital signature scheme (G, S, V) is secure against* existential forgery under adaptive chosen-message attacks *if for any PPT A the probability $p(n)$ that A succeeds at the following game satisfies $p(n) = \mathsf{negl}(n)$:*

1. *Keys $(\mathsf{PK}, \mathsf{SK}) \leftarrow G(1^n)$ are generated randomly and PK is given to A.*

2. *A is given access to a signing oracle SO that it can adaptively query with queries of the form $SO(m)$, which return a signature $\sigma \leftarrow S_{SK}(m)$.*

3. *A returns a signature σ for a message m and succeeds if m was never queried to the oracle and $V_{PK}(m, \sigma) = 1$.*

It was first shown by Goldwasser, Micali and Rivest [10] that there exist digital signature schemes that meet this strongest security definition under the number theoretic assumption that factoring is hard. It was later shown by Rompel [15] that the assumption can be weakened to just the existence of one-way functions, giving us the following result which we will use to construct ring signatures:

Theorem 10 *The existence of one-way functions implies the existence of Digital Signature schemes that are secure against existential forgery under adaptive chosen-message attacks.*

3.4 Zero Knowledge and ZAPs

The final cryptographic primitive we will present before constructing the ring signature scheme of [2] is Zero Knowledge Proofs, of which ZAPs are a particular flavor. Zero Knowledge Proofs were first proposed by Goldwasser, Micali and Rackoff [9] as an interactive proof technique whereby a prover could prove a statement to a verifier without giving away any more information other than the fact that the statement is true. For example, a prover could prove that a graph is 3-colorable given a witness, such as a 3-coloring of the graph, without giving away the witness or even divulging any information about the witness. Informally, the requirements for a protocol to be a Zero Knowledge Proof System are the following:

1. **Completeness:** The prover should be able to prove any true statement.

2. **Soundness:** It should be very hard for any prover to prove any false statement.

3. **Zero Knowledge:** No verifier should be able to obtain any information from interacting with the prover other than what it could have obtained on its own.

Feige and Shamir [6] introduced a weaker requirement called *witness indistinguishibility* to replace the zero knowledge requirement. Witness indistinguishibility allows information to leak about the witness but asserts that if there are several witnesses for a statement proven, then the verifier does not learn any information about which witness the prover used to construct the proof. To give a rough idea of why this weaker requirement will suffice for ring signatures, we will see that zero knowledge proofs will be used to hide the identity of the signer of a ring signature. Since the possible witnesses correspond to the possible signers (i.e. the ring members), we need not hide all information; we just need to give no information about which witness was used.

The ring signature scheme we will look at in the next section is heavily dependent on ZAPs, which are a kind of 2-round witness indistinguishable protocol developed by Dwork and Naor [5] that can be used to prove statements of the form $x \in L$ for any language L in NP. Associated with any NP language L is a polynomially bounded witness relation R_L such that $x \in L \iff \exists w, (x, w) \in R_L$. The idea of ZAPs is that the verifier sends a first message r and the prover responds with a proof π, which it can produce because it has access to a witness w for the statement x. These two rounds of communication complete the protocol, and witness indistinguishibility guarantees that the verifier can not tell which witness was used from all of the w such that $(x, w) \in R_L$. It is even shown that the first message can be fixed and used to prove many different statements, so that someone can post their first message as a kind of public key that people can look up in a directory when they want to prove a statement to that person. The formal definition of a ZAP is given below:

Definition 11 (ZAP) *A ZAP is a triplet (P, V, l) where P is a PPT prover, V is a polynomial time deterministic verifier, and $l(n)$ is a polynomial satisfying the three following conditions:*

1. ***Completeness:*** *For all* $(x, w) \in R_L$ *and all* $r \in \{0,1\}^{l(n)}$,

$$Pr[\pi \leftarrow P_r(x, w); V_r(x, \pi) = 1] = 1$$

2. ***Adaptive Soundness:*** *With high probability there is no* $x \notin L$ *for which there is a valid proof:*

$$Pr[r \leftarrow \{0,1\}^{l(n)}; \exists(x, \pi), x \notin L, V_r(x, \pi) = 1] = \textit{negl}(n)$$

3. ***Witness Indistinguishibility:*** *For any* $x \in L$ *and pair of witnesses* w_0 *and* w_1 *such that* $(x, w_0) \in R_L$ *and* $(x, w_1) \in R_L$*, the pair of ensembles of probability distributions* $\{P_{r_n}(x, w_0)\}$ *and* $\{P_{r_n}(x, w_1)\}$ *indexed over* n*, where* $r_n \leftarrow \{0,1\}^{l(n)}$*, are computationally indistinguishable.*

The ring signature scheme in the next section as well as the modified scheme presented in this paper both use the following result from [5] to construct ZAPs for particular NP languages related to ring signatures:

Theorem 12 (ZAPs) *There exist ZAPs for any language* $L \in NP$.

3.5 Ring Signatures without Random Oracles

The first ring signature scheme that was not dependent on random oracles was introduced by Bender, Katz and Morselli [2], in a paper that also introduced stronger security requirements. The requirement we will focus on in this paper is anonymity, since this is the area we will show improvement. The main scheme presented in [2] does not meet the strongest anonymity requirement they propose. While they propose a modification to meet the strongest anonymity definition, we will argue that this modification is inconsistent with the aims of ring signatures because it eliminates deniability. In this section we will first present the definitions of a ring signature scheme and the security definitions from [2], then we will present their main scheme, and finally we will review their modification and argue why a different type of modification is necessary to achieve the strongest anonymity requirement.

3.5.1 Definitions

We first provide a formal definition of a ring signature scheme. A ring signature scheme has a similar formal structure to an ordinary signature scheme except that to create a signature, in addition to using a secret key, one also uses the public keys of the other users that are to be included in the ring.

Definition 13 (Ring Signature Scheme) *A* Ring Signature Scheme *is a triplet of PPTs (G, S, V) where*

- *G is a key generator that takes input 1^n, where n is the security parameter, and returns (PK, SK) where PK is the public key and SK is the secret key.*
- *S is a signing algorithm that takes as input a message m, a ring $R = (PK_1, , PK_l)$ of public keys, a user i^* and their secret key SK_{i^*}, and returns a signature $\sigma = S_{i^*, SK_{i^*}}(m, R)$. We assume here that (SK_{i^*}, PK_{i^*}) is a valid key pair generated by G and $|R| \geq 2$.*
- *V is a verification algorithm that takes as input a signature σ and a ring $R = (PK_1, ..., PK_l)$ of public keys and returns a bit $b = V_R(m, \sigma)$ where $b = 1$ (respectively 0) if the signature is valid (respectively invalid).*
- *For any polynomial l in n, and any set of key pairs $\{(PK_i, SK_i)\}_{i=1}^{l}$ that $G(1^n)$ might produce, and for any message m and user $i^* \in [1, l]$, $Pr[V_R(m, S_{i^*, SK_{i^*}}(m, R)) = 1] = 1$, where $R = (PK_1, ..., PK_l)$.*

One thing to note here is that because of the ad-hoc nature of ring signatures, and the fact that we will want to sign messages with respect to rings that include users that never planned on participating in ring signature schemes, the key generator algorithm must perform a function that most people already perform on their own without intending to do ring signatures. For example, the public keys could be public encryption keys or public keys for digital signature schemes, or some combination thereof, because it is expected that most people create and publish such keys regardless of the existence of ring signature schemes that utilize them. On the other hand, a key generator algorithm for ring signatures that requires users to generate public keys that are not ubiquitous, such as a specially designed key made with ring signatures in mind, would not be of much use because then one would not be able to sign messages with respect to rings that include people who were not also interested in doing ring signatures.

We now give the strongest unforgeability definition given in [2], which they refer to as unforgeability with respect to insider corruption. The idea behind the adversarial game in this definition is that the adversary is given a set of validly generated public keys, and an oracle in which it can make queries with respect to a ring of its choice (even rings including adversarially generated keys). The adversary is even given the power to choose the author in the oracle queries, provided the specified author does not correspond to one of the adversarially generated keys. The requirement is that the adversary succeeds in this game by producing a forgery, i.e. a signature for a message and ring it never queried before, with negligible probability.

Definition 14 (Unforgeability) *A ring signature scheme (G, S, V) is* unforgeable *if for any PPT A and any polynomial k the probability $p(n)$ that A succeeds at the following game satisfies $p(n) = \mathsf{negl}(n)$:*

1. *Keys $\{(PK_i, SK_i)\}_{i=1}^{k(n)} \leftarrow G(1^n)$ are generated randomly and the set of public keys $S = \{PK_i\}_{i=1}^{k(n)}$ is given to A.*

2. *A is given access to a signing oracle SO that it can adaptively query with queries of the form $SO(i^*, m, R)$, which return a signature $\sigma \leftarrow S_{SK_{i^*}}(m, R)$, under the*

condition that $PK_{i^*} \in R$.

3. *A is also given access to a corruption oracle CO that it can query with queries of the form $CO(i^*)$, which returns the secret key SK_{i^*}.*

4. *A returns a signature σ for a message m and a ring S^* and succeeds if SO was never queried with message m and ring S^*, $S^* \subset S - C$ (where C is the set of corrupted users) and $V_{S^*}(m, \sigma) = 1$.*

We will now see the two strongest anonymity definitions presented in [2], anonymity against attribution attacks and anonymity with respect to full key exposure. The definitions are very similar and differ only in a single detail related to the powers granted to the adversary. In both adversarial games, the adversary is given a signing oracle as in the unforgeability definition, and is supposed to produce a sample message m, a ring R, and two possible signers i_0 and i_1 in R. The adversary is then given a ring signature of m with respect to R, signed by one of the two authors, and the adversary is supposed to guess which author was used. In both anonymity definitions, along with the challenge signature, the adversary is given the randomness to generate the keys for several users in the ring R. The difference between the two definitions is that in the weaker definition the adversary is given the randomness for all but one of the users, while in the stronger definition the adversary is given the randomness to compute the keys for *all* users. The two definitions are presented together because they only differ in one place:

Definition 15 (Anonymity against attribution attacks / full-key exposure) *A ring signature scheme (G, S, V) is* anonymous against attribution attacks *if for any PPT A and polynomial l, the probability $p(n)$ that A succeeds at the following game satisfies $|p(n) - \frac{1}{2}| = \mathsf{negl}(n)$:*

1. *Keys $\{(PK_i, SK_i)\}_{i=1}^{l(n)} \leftarrow G(1^n)$ are generated randomly and the set of public keys $S = \{PK_i\}_{i=1}^{l(n)}$ is given to A. A is given access to a signing oracle SO that*

it can adaptively query with queries of the form $SO(i^*, m, R)$, which return a signature $\sigma \leftarrow S_{SK_{i^*}}(m, R)$, under the condition that $PK_{i^*} \in R$.

2. *A outputs a message m, a ring R, and the indices of two users i_0 and i_1 such that PK_{i_0} and PK_{i_1} are both in R.*

3. *A random bit b is selected and A is given a signature $\sigma \leftarrow S_{SK_{i_b}}(m, R)$ as well as $\{\omega_i\}_{i \neq i_0}$.*

4. *A outputs a bit b' and succeeds if $b' = b$.*

If in step 3 the adversary is also given the randomness ω_{i_0} then we say that the scheme is anonymous against full key exposure.

3.5.2 Scheme from [2]

The main scheme from [2] meets the strongest unforgeability requirement and the weaker of the two anonymity requirements presented in the last section. The scheme assumes the existence of a semantically secure encryption scheme (G^E, E, D), a digital signature scheme (G^S, S^S, V^S) and a ZAP (P, V^Z, l) for a particular NP language that we will define. The idea of the scheme is that for a user to sign a message m, with respect to a ring R containing that user, the user first signs the message with their own ordinary signature, then encrypts this signature using the public encryption keys of all the users in the ring, and then proves using the ZAP that the encryption is an encryption of a signature of m using the secret signing keys of one of the members of the ring. Therefore a public key PK for a ring signature scheme really consists of a public encryption key, a public signature key, and a ZAP message corresponding to the first message sent by the verifier, while a secret key SK for a ring signature scheme consists of a secret signature key to produce the ordinary signature.

To illustrate the scheme more clearly, we will begin by defining what we mean when we say the user encrypts using the public encryption keys of all the users in a ring. We define an encryption $E_{R_E}(\sigma)$ of a message σ with respect to the ring of public encryption keys $R_E = \{\mathsf{PK}_i^E\}_{i=1}^k$ as follows: First select $k-1$ random strings $s_1, ..., s_{k-1} \in \{0,1\}^{|\sigma|}$ of the same size as σ. We then define the cipher text C as:

$$C = E_{R_E}(\sigma) = \left(E_{PK_1^E}(s_1), E_{PK_2^E}(s_2), ..., E_{PK_{k-1}^E}(s_{k-1}), E_{PK_k^E}(\sigma \oplus \bigoplus_{i=1}^{k-1} s_i) \right)$$

It can be shown that this larger encryption scheme is also semantically secure as long as the secret keys of at least one of the users in the ring R_E is kept secret.

We can now describe in more detail how the ring signature scheme works. After the user i^* in a ring of k users creates an ordinary signature σ, they create a ciphertext $C_{i^*} = E_{R_E}(\sigma)$. Additionally the user creates $k-1$ more ciphertexts $\{C_i = E_{R_E}(0)\}_{i \neq i^*}$. There are now k ciphertexts $C_1, ..., C_k$, and the author of the ring signature, user i^*, has encrypted their signature into the i^*th ciphertext, and encrypted 0 into the rest. The semantic security of the encryption scheme hides which ciphertext contains an encryption of a signature. Now the user can prove, using the ZAP, that there exists some i between 1 and k such that the ith ciphertext has the signature of the ith user. This corresponds to proving membership in the following NP language, where R_S is defined similarly to R_E except for public signing keys rather than public encryption keys:

$$L = \{(m, R_S, R_E, \{C_i\}_{i=1}^k) : \exists \sigma, \omega, s.t. \bigvee_{i=1}^{k} (E_{R_E}(\sigma; \omega) = C_i \wedge V_{PK_i^S}^S(m, \sigma) = 1)\}$$

The language is obviously in NP because there is only a single existential quantifier, the variables are all polynomial size in n, and the statement can be checked in polynomial time because the PPT E_{R_E} becomes deterministic once the randomness ω is provided and the signature verification algorithm V is deterministic polynomial time to begin with.

We are now ready to define the ring signature scheme (G, S, V) formally:

- **Key Generator** $G(1^n)$:

  1. Generate signing key pair $(\mathsf{PK}^S, \mathsf{SK}^S) \leftarrow G^S(1^n)$.
  2. Generate encryption key pair $(\mathsf{PK}^E, \mathsf{SK}^E) \leftarrow G^E(1^n)$ (we will ignore the secret encryption key for ring signature purposes).
  3. Choose an initial ZAP message $r \leftarrow \{0,1\}^{l(n)}$.
  4. Output as the public key $\mathsf{PK} = (\mathsf{PK}^S, \mathsf{PK}^E, r)$ and as the private key $\mathsf{SK} = \mathsf{SK}^S$.

- **Signing Algorithm** $S_{i^*,SK_{i^*}}(m, R)$:

  1. Parse $R = (\mathsf{PK}_1, ..., \mathsf{PK}_k)$ and each $\mathsf{PK}_i = (\mathsf{PK}_i^S, \mathsf{PK}_i^E, r_i)$ and $\mathsf{SK}_{i^*} = \mathsf{SK}_{i^*}^S$. Set $R_E = (\mathsf{PK}_1^E, ..., \mathsf{PK}_k^E)$ and $R_S = (\mathsf{PK}_1^S, ..., \mathsf{PK}_k^S)$.
  2. Set $M = m|\mathsf{PK}_1|...|\mathsf{PK}_k$ where "|" denotes string concatenation and compute the digital signature $\sigma = S^S_{SK_{i^*}}(M)$.
  3. Compute ciphertext $C_{i^*} = E_{R_E}(\sigma)$ and record the random coins used in ω_{i^*}.
  4. Compute ciphertexts $C_i = E_{R_E}(0)$ for each $i \neq i^*$.
  5. Set statement $x = (M, R_S, R_E, \{C_i\}_{i=1}^k)$, and witness $w = (\sigma_{i^*}, \omega_{i^*})$ and use the ZAP with the first message r_1 from the lexicographically first public key to produce the proof $\pi \leftarrow P_{r_1}(x, w)$ of the statement $x \in L$.
  6. Output the signature $\sigma = (\{C_i\}_{i=1}^k, \pi)$.

- **Verification Algorithm** $V_R(m, \sigma)$:

  1. Parse $R = (\mathsf{PK}_1, ..., \mathsf{PK}_k)$ and each $\mathsf{PK}_i = (\mathsf{PK}_i^S, \mathsf{PK}_i^E, r_i)$ and $\sigma = (\{C_i\}_{i=1}^k, \pi)$. Set $R_E = (\mathsf{PK}_1^E, ..., \mathsf{PK}_k^E)$, $R_S = (\mathsf{PK}_1^S, ..., \mathsf{PK}_k^S)$, and $M = m|\mathsf{PK}_1|...|\mathsf{PK}_k$.
  2. Set statement $x = (M, R_S, R_E, \{C_i\}_{i=1}^k)$.
  3. Verify the proof π of $x \in L$ signature and output $V^Z_{r_1}(x, \pi)$.

This ring signature scheme can be shown to achieve the highest unforgeability guarantee. However it only achieves anonymity with respect to attribution attacks because if all of the secret keys are exposed, then the encryption scheme E_{R_E} can be decrypted and the ordinary signature will be seen in the clear beneath one of the ciphertexts, thereby giving away the signer's identity. If all but one of the secret keys are exposed, however, the signer's identity will still be kept secret because the encryption will still be unbroken, giving us the following theorem from [2]:

Theorem 16 *If (G^E, E, D) is a semantically secure encryption scheme, (G^S, S^S, V^S) is a signature scheme that is existentially unforgeable against adaptive chosen message attacks, and (P, V^Z, l) is a ZAP for the language L, then the ring signature scheme defined above is unforgeable and anonymous against attribution attacks.*

3.5.3 Achieving the strongest anonymity guarantee

The reason why the ring signature scheme from [2] does not achieve the strongest anonymity guarantee is because the encryption scheme they propose is not secure if all of the secret keys are exposed. To solve this problem they propose a modification to their scheme in which an oblivious encryption key generator is used, rather than a standard encryption key generator. If an encryption system has an oblivious generator, this means that a user can generate a public encryption key for which they do not know the secret key. Oblivious key generators also allow the user to expose the randomness used to generate the public key and still know that it will be infeasible for an adversary to distinguish their public keys from keys for which the secret key is known. This modification to the scheme achieves anonymity against full key exposure because now we can reveal the randomness used to generate all of the encryption keys, and the adversary will still not be able to decrypt the ciphertexts because they will not be given the decryption keys.

There are several reasons why this modification, while it does satisfy the strongest anonymity definition, is unsatisfactory. The first reason is that this would require a user to publish encryption keys for which they do not know the secret key, and pass this off as a regular encryption key that they will use to receive secure communication. Using an oblivious generator eliminates the ability to actually *use* the key for encryption because the messages can not be decrypted. While the definition of ring signature schemes does not explicitly require that the primary functionality of the primitives that ring signatures are built on be preserved, this is certainly a weakness of this scheme.

A more significant weakness of the scheme under this modification is that the adversarial game in the strongest anonymity definition no longer models the real world situation accurately if such keys are used. In the adversarial game, both users that the adversary is trying to distinguish between as author have had their keys generated specifically for ring signatures, so that the encryption keys were generated without any known private key. The corresponding situation in the real world is that all members of a ring have had their secret encryption keys exposed, either by coercion or other means, except for two members, both of whom generated their keys obliviously (i.e. without a private key) because they both intended to participate in ring signatures. Even under coercion, these two users can claim inability to produce their private keys because their public keys were indeed produced obliviously, though in the process they will be forced to admit their intention to participate in ring signature schemes. While in this situation, anonymity would be preserved, the adversarial game does not capture another plausible situation in which a user signs a ring signature with respect to a ring of users all of whom (except for the signer) have the private encryption keys to go along with their public keys. In this situation, all the users' keys will be exposed and the signer will be the only user who claims inability to produce secret keys. While even in this case the ciphertexts can not be decrypted, anonymity is no longer preserved because the ring member who can not produce their private keys can no longer deny their involvement with ring signatures while everyone else can.

What we really seek is a scheme that meets the strongest anonymity requirement and is also built on cryptographic primitives that are identical to those of users who had no intention of participating in ring signatures. In such a scheme, even when all the members of a ring are coerced to give up the randomness used to generate their keys, ring signature participants still have deniability, or the ability to deny participation in ring signature schemes. An oblivious key generator has deniability until the point at which the randomness and secret keys are exposed, at which point users who did not use the oblivious key generator will give up their secret keys and thereby distinguish themselves from those who intended to participate in ring signatures. In the next section we will modify the ring signature scheme from [2] to produce a scheme that meets the strongest security definitions from [2] and is also deniable.

Chapter 4

Deniable Ring Signature Scheme

We will modify the scheme in [2] so that it achieves the strongest security guarantees while also achieving *deniability.* By deniability we mean that the scheme only uses keys such that when the secret keys and randomness used to generate the keys are exposed, the user who constructed their keys to participate in ring signatures can claim that they never intended to participate in ring signatures and only generated the keys to use them for their default functions, such as secure communication or message authentication for encryption and digital signatures respectively. We stated in the previous section that a way to ensure deniability is to have the key generator for a ring signature scheme generate public and private keys that are identical to those generated by people who never anticipated participating in ring signature schemes. The main scheme in [2] uses keys for public key encryption, digital signatures, and ZAPs, all of which have a public key infrastructure and for which it can be assumed that people will want this functionality independent of the fact that they can be used for ring signatures. We will build on this scheme by using the same three cryptographic primitives.

In order to construct a ring signature scheme based on the scheme from [2] that is anonymous even if all of the secret encryption keys are divulged we will have the signer avoid encrypting their ordinary signature in E_{R_E}. Instead, the signer will encrypt a string of zeroes the same length as an ordinary signature. When the secret keys are divulged and the cipher texts are decrypted, the underlying message 0 will be revealed behind each cipher text, thereby preserving anonymity. The problem here is that now the signer can not prove using the ZAP that one of the cipher texts is an encryption of someone's signature, because in fact the statement is false. We will have to change the language L for the ZAP so that the signer can still create a valid signature (preserving completeness) while simultaneously ensuring that only someone in the ring could prove membership of $x \in L$ (preserving unforgeability).

The trick behind our scheme is that when we generate our digital signature keys, we will do so pseudorandomly rather than randomly. We will explain what this means more formally in the next section, but just to give an idea of why the scheme works, we note that this allows us to prove using the ZAP that "either the cipher texts C_i contain an encryption of the signature of someone in the ring, OR someone in the ring had their digital signature keys generated pseudorandomly rather than randomly." The idea is that only someone who participated in the pseudorandom generation of their signature keys would be able to prove that someone's keys were generated pseudorandomly, thereby proving that if someone could create this ZAP proof then they must be in the ring. Anonymity is still preserved because the ZAP is witness hiding and therefore does not give away whose keys were generated pseudorandomly, and this can not be inferred by just looking at the keys because random and pseudorandom are indistinguishable. This indistinguishability between random and pseudorandom also allows us to achieve deniability because even if someone's keys were generated pseudorandomly in order to create ring signatures, they can deny this fact even after their secret keys are divulged and no feasible adversary would be able to tell whether they were lying because they can not tell whether their keys are random or pseudorandom. We have achieved deniability by requiring that the ring signature scheme be based on keys that are *indistinguishable* from, rather than *identical* to the keys that people generate for their own use of these public key cryptographic primitives.

In the following sections we will define and prove the security of pseudorandom key generation schemes, formally define our modified ring signature scheme, and prove that our scheme satisfies the strongest security definitions from [2].

4.1 Pseudorandom Key Generation

When a user of digital signature schemes generates their keys via $G^S(1^n)$, the generator algorithm flips some random coins $\omega \in_R \{0,1\}^{q(n)}$ for some polynomial q and then computes the deterministic function $G^S(1^n; \omega)$. If instead of using $q(n)$ random bits, we used only half as many random bits, but used a pseudorandom number generator to expand our string to the appropriate length, we will produce keys that are computationally indistinguishable from the keys we would have produced if we had picked all $q(n)$ bits randomly. Furthermore, the digital signature scheme will be just as secure if we use these pseudorandomly generated keys as it would have been if we generated the keys randomly.

We now formally define a pseudorandom key generator $\hat{G}^S$ for a digital signature scheme (G^S, S^S, V^S), given a pseudorandom number generator f. Let f_k be a pseudorandom generator as in Definition 3, with $l(k) = 2k$ (meaning it stretches by a factor of 2). Given $\omega \in \{0,1\}^{q(n)/2}$, we define $\hat{\omega} = f_{q(n)/2}(\omega)$. We note that $|\hat{\omega}| = q(n)$ and the ensembles of distributions $\{\omega \leftarrow \{0,1\}^{q(n)}\}$ and $\{\omega \leftarrow \{0,1\}^{q(n)/2} : \hat{\omega} = f_{q(n)/2}(\omega)\}$ are computationally indistinguishable by the definition of a pseudorandom generator.

Now that we have defined a pseudorandom key generator $\hat{G}^S$, we will prove two lemmas that will allow us to replace G^S with this new generator and still be able to use the keys for digital signatures, as well as for ring signatures. The first lemma implies that if we were to use $\hat{G}^S$ to generate signature keys rather than G^S, then the ring signature scheme built upon this would still be deniable because no feasible adversary could tell which generator we used, even if we divulge both the secret key and the pseudorandom string (which we will try and pass off as random) used to generate these keys.

Lemma 17 *Given a PPT digital signature key generator G^S that uses a polynomial $q(n)$ bits of randomness, and a pseudorandom generator f_k with $l(k) = 2k$, the following two ensembles of probability distributions are computationally indistinguishable, where $\hat{G}^S$ is defined as above:*

$$\{\omega \leftarrow \{0,1\}^{q(n)}; (PK^S, SK^S) = G^S(1^n; \omega) : (\omega, PK^S, SK^S)\}$$

$$\{\omega \leftarrow \{0,1\}^{q(n)/2}; (PK^S, SK^S) = \hat{G}^S(1^n; \omega) : (f_{q(n)/2}(\omega), PK^S, SK^S)\}$$

Proof Suppose there existed some non-uniform PPT adversary A that could distinguish between the two ensembles for some auxiliary information z_n with non-negligible probability. We can obtain a contradiction by constructing an A' that distinguishes between the two ensembles of distributions $\{\omega\}$ and $\{\hat{\omega}\}$ defined above, using the same auxiliary input z_n. Given a string ω from one of the two distributions, we simply run $G(1^n; \omega) = (\mathsf{PK}^S, \mathsf{SK}^S)$ and then run A on input $(\omega, \mathsf{PK}^S, \mathsf{SK}^S)$, with auxiliary input z_n, and output whatever A outputs. We see that we will distinguish with the same non-negligible probability as A because if our input is from the distribution $\{\omega\}$ then our input to A will be from the first distribution, while if our input is from the distribution $\{\hat{\omega}\}$ then our input to A will be from the second distribution. ■

This next lemma implies that if we generate our keys using $\hat{G}^S$ instead of G^S, we will preserve the functionality of the original digital signature scheme, even preserving achievement of the strongest security guarantees. It is trivial to see that $(\hat{G}^S, S^S, V^S)$ is a digital signature scheme because completeness holds with probability 1 for the original scheme and the keys that we generate with $\hat{G}^S$ are a subset of the keys generated by G^S. We just need to show that the unforgeability condition holds:

Lemma 18 *Given a digital signature scheme (G^S, S^S, V^S) that is secure against existential forgery under adaptive chosen-message attacks, the digital signature scheme*

$(\hat{G}^S, S^S, V^S)$ where $\hat{G}^S$ is defined as above, is also secure against existential forgery under adaptive chosen-message attacks.

Proof We can show that the existence of an adversary A that can produce a forgery will allow us to distinguish between the two distributions of keys that were proven computationally indistinguishable in the last lemma. Given $(\omega, \mathsf{PK}^S, \mathsf{SK}^S)$ from the first distribution, we know from the fact that (G^S, S^S, V^S) is existentially unforgeable, that adversary A will produce a forgery with negligible probability if we play the adversarial game using those keys. On the other hand, given $(\omega, \mathsf{PK}^S, \mathsf{SK}^S)$ from the second distribution, by assumption A will be able to produce a forgery with non-negligible probability because this is the distribution of keys generated via the scheme $(\hat{G}^S, S^S, V^S)$ that we are assuming A can break. Therefore if our adversary A' for distinguishing the two distributions simply outputs 1 if and only if A produces a forgery, A' will be distinguishing between the two distributions, contradicting the previous lemma. ■

4.2 Scheme Implementation

The implementation of our deniable ring signature scheme differs from that of [2] in three main respects: **1)** Signature keys are generated pseudorandomly rather than randomly, **2)** we do not produce an ordinary signature and then encrypt it but rather just encrypt a string of zeroes, and **3)** we use the ZAP prove membership in a slightly different NP language that allows for witnesses that attest to the pseudorandomness of a ring members signature keys.

We begin by defining the NP language $\hat{L}$ that we will be proving membership in. The difference between $\hat{L}$ and L is that in $\hat{L}$ we take every clause from L and "or" it with the statement that a ring member's digital signature keys were generated pseudorandomly, i.e. there exist random coins ω and a secret signing key SK^S such that the pseudorandom key generator $\hat{G}^S$ defined in the previous section, when given random coins ω, produces the correct secret and public signing key.

$$\hat{L} = \Big\{(m, R_S, R_E, \{C_i\}_{i=1}^k) : \exists \sigma, \omega_1, \omega_2, \mathsf{SK}^S, s.t.$$

$$\bigvee_{i=1}^{k} (E_{R_E}(\sigma; \omega_1) = C_i \wedge V^S_{PK_i^S}(m, \sigma) = 1) \vee \hat{G}^S(1^n; \omega_2) = (\mathsf{PK}_i^S, \mathsf{SK}^S)\Big\}$$

We can see that this language is in NP just as L was because the only difference is an extra "or" statement with each clause, two more polynomial size variables after the existential quantifier, and the computation of a PPT function $\hat{G}^S$ which is deterministic once the randomness ω_2 is provided. We are now ready to define the modified ring signature scheme.

- **Key Generator** $G(1^n)$**:**

  1. Generate signing key pair $(\mathsf{PK}^S, \mathsf{SK}^S) \leftarrow \hat{G}^S(1^n)$ and record random coins ω^S from $\hat{G}^S$.
  2. Generate encryption key pair $(\mathsf{PK}^E, \mathsf{SK}^E) \leftarrow G^E(1^n)$ (we will ignore the secret encryption key for ring signature purposes).
  3. Choose an initial ZAP message $r \leftarrow \{0,1\}^{l(n)}$.
  4. Output as the public key $\mathsf{PK} = (\mathsf{PK}^S, \mathsf{PK}^E, r)$ and as the private key $\mathsf{SK} = (\omega^S, \mathsf{SK}^S)$.

- **Signing Algorithm** $S_{i^*, SK_{i^*}}(m, R)$**:**

  1. Parse $R = (\mathsf{PK}_1, , \mathsf{PK}_k)$ and each $\mathsf{PK}_i = (\mathsf{PK}_i^S, \mathsf{PK}_i^E, r_i)$ and $\mathsf{SK}_{i^*} = (\omega_{i^*}^S, \mathsf{SK}_{i^*}^S)$. Set $R_E = (\mathsf{PK}_1^E, ..., \mathsf{PK}_k^E)$ and $R_S = (\mathsf{PK}_1^S, ..., \mathsf{PK}_k^S)$.
  2. Set $M = m|\mathsf{PK}_1|...|\mathsf{PK}_k$ where $|$ denotes string concatenation.
  3. Compute cipher texts $C_i = E_{R_E}(0)$ for each $i \in [1, k]$.

4. Set statement $x = (M, R_S, R_E, \{C_i\}_{i=1}^k)$, and witness $w = (0, 0, \omega_{i^*}^S, \mathsf{SK}_{i^*}^S)$ and use the ZAP with the first message r_1 from the lexicographically first public key to produce the proof $\pi = P_{r_1}(x, w)$ of the statement $x \in \hat{L}$.

5. Output the signature $\sigma = (\{C_i\}_{i=1}^k, \pi)$.

- **Verification Algorithm $V_R(m, \sigma)$:**

1. Parse $R = (\mathsf{PK}_1, ..., \mathsf{PK}_k)$ and each $\mathsf{PK}_i = (\mathsf{PK}_i^S, \mathsf{PK}_i^E, r_i)$ and $\sigma = (\{C_i\}_{i=1}^k, \pi)$. Set $R_E = (\mathsf{PK}_1^E, ..., \mathsf{PK}_k^E)$, $R_S = (\mathsf{PK}_1^S, ..., \mathsf{PK}_k^S)$, and $M = m|\mathsf{PK}_1|...|\mathsf{PK}_k$.

2. Set statement $x = (M, R_S, R_E, \{C_i\}_{i=1}^k)$.

3. Verify the proof π of $x \in \hat{L}$ and output $V_{r_1}^Z(x, \pi)$.

To verify the completeness of this ring signature scheme, we can see that a signer that generated their digital signature keys pseudorandomly, via $\hat{G}^S$, as specified in G, will produce a proof π using their ZAP with witness $w = (0, 0, \omega, \mathsf{SK}^S)$ that satisfies the NP relation (x, w) for $\hat{L}$. By the completeness of the ZAP, the proof will be verified with probability 1 and hence the signature will be verified with probability 1.

4.3 Proofs of Security

Before we prove that our ring signature scheme achieves the strongest security guarantees proposed by [2], we make an observation about what it means to hand over to the adversary the randomness used to generate keys. The reason we generated keys pseudorandomly was so that when our secret keys and randomness are exposed, we could pass off this pseudorandom string as our random string and noone would notice and therefore suspect that we had planned to do ring signatures anymore than anyone else. Therefore it would not be unreasonable to weaken the anonymity requirement from [2] so that instead of the randomness from our ring signature generator G being divulged, we can choose to divulge some concocted randomness that we pass off as the randomness used to generate all of the underlying primitives, since we do not admit to running any ring signature generator G anyways. However, the anonymity game in the definition entails at least two honest users, where honest means they both had their keys generated for ring signatures. Since there are at least two users who admit to participating in ring signatures, we can have both users submit the *actual* randomness used to generate their ring signature keys and still show that the adversary can not distinguish between the two possible signers in this case. We therefore prove that we achieve the strongest guarantees from [2], but admit that the definition could be weakened.

Theorem 19 *If (G^E, E, D) is a semantically secure encryption scheme, (G^S, S^S, V^S) is a signature scheme that is existentially unforgeable against adaptive chosen message attacks, $\hat{G}^S$ is a pseudorandom key generator for G^S, and (P, V^Z, l) is a ZAP for the language $\hat{L}$, then our ring signature scheme defined above is unforgeable and anonymous against full key exposure.*

Proof We begin by proving anonymity and will then prove unforgeability.

4.3.1 Anonymity

We will show that if there exists an adversary A that can win the adversarial game in Definition 15 with some non-negligible probability $\frac{1}{p(n)}$ above $\frac{1}{2}$ for some polynomial p and infinitely many n, then we will use A to construct an adversary A' that can distinguish between witnesses in a ZAP proof for the language $\hat{L}$ associated with our ring signature scheme. A' will begin by generating keys $\{(\mathsf{PK}_i, \mathsf{SK}_i)\}_{i=1}^{l(n)} \leftarrow G(1^n)$ and giving the set of public keys $S = \{\mathsf{PK}_i\}_{i=1}^{l(n)}$ to A.

When A makes a query of the form $SO(i^*, m, R)$ with $\mathsf{PK}_{i^*} \in R$, we respond by giving A a signature $\sigma \leftarrow S_{SK_{i^*}}(m, R)$ which we can do because we have the secret keys SK_{i^*} for all $i^* \in [1, l]$. Eventually A will output a message m, a ring R, and the indices of two users i_0 and i_1 such that PK_{i_0} and PK_{i_1} are both in R. We will begin constructing a signature for message m with respect to ring R as specified in S, which we can do without committing to a signer i^* until we have to construct the witness for the ZAP proof. So far we will have constructed cipher texts $C_i = E_{R_E}(0)$ for each $i \in [1, |R|]$ and we will also have the statement $x = (M, R_E, R_S, \{C_i\}_{i=0}^{|R|})$, for which we need to provide a proof that $x \in L$. We note that at this point, we have two valid witnesses that we could use: $w_0 = (0, 0, \omega_{i_0}^S, \mathsf{SK}_{i_0}^S)$ and $w_1 = (0, 0, \omega_{i_1}^S, \mathsf{SK}_{i_1}^S)$ where $\omega_{i_0}^S$ and $\omega_{i_1}^S$ are the random coins used in the pseudorandom signature key generation algorithm $\hat{G}^S$ and stored in SK_{i_0} and SK_{i_1} respectively.

Our adversary A' then submits the statement x and the two witnesses w_0 and w_1 for statement x as a challenge to try and distinguish between proofs using the different witnesses. A' is then given back a proof π of x which uses one of the two witnesses. This proof π allows us to complete the challenge signature $\sigma = (\{C_i\}_{i=1}^{|R|}, \pi\}$ to give back to adversary A. We also give A the randomness $\{\omega_i\}_{i=0}^{l}$ used to produce all the keys in G. We note that if A' was given a proof using witness w_0, then the distribution of signatures given to A will follow the distribution of signatures using i_0 as author, while if A' is given a proof using witness w_1, then the distribution of signatures given to A will follow the distribution of signatures that use i_1 as author. Furthermore the randomness $\{\omega_i\}_{i=0}^{l}$ that we give A is the same regardless of which witness was used. Therefore with probability $\frac{1}{2} + \frac{1}{p(n)}$, A will output a bit b that correctly corresponds to the witness that was used. So if A' simply outputs whatever bit A outputs, then the probability that A' guesses the correct witness is also $\frac{1}{2} + \frac{1}{p(n)}$. This contradicts the witness indistinguishability of the ZAP and therefore completes our proof of anonymity.

4.3.2 Unforgeability

The basic idea behind our proof for unforgeability is that if there exists an adversary that produces forgeries with non-negligible probability, we will use this adversary to extract a forgery for the digital signature scheme of one of the members of the ring, contradicting the unforgeability of the digital signature scheme. We note that if an adversary is producing forgeries, this means that the ZAP proofs produced in the signatures are valid, implying one of the two following events: there is an ordinary signature belonging to one of the ring members encrypted in one of the cipher texts or one of the ring members had their digital signature keys generated pseudorandomly. We would like to force the adversary to produce proofs using the first type of witness, so that we can extract a forgery. This is tough to do as long as all the ring members' signature keys are in fact generated pseudorandomly. To get around this, we create hybrid experiments in which we switch the signature keys of all of the users from pseudorandom to random, counting on the fact that the adversary can not tell the difference. The main challenge is in how we respond to oracle queries during the hybrid experiments.

To formalize how we may behave in response to oracle queries in the different hybrid experiments, we first define two alternative versions of the signing algorithm S: S_1 and S_2. Like the original signing algorithm S, S_1 assumes that the author i^* had their digital signature keys generated pseudorandomly, and the algorithm has access to a witness ω that generated these pseudorandom keys. Signing algorithm S_1 operates just like S except that in creating the cipher texts C_i, instead of encrypting 0 in all the cipher texts, S_1 encrypts a digital signature $\sigma_{i^*}(M)$ in the i^*th cipher text, as in the scheme from [2]. This opens up the opportunity for a different witness, namely $w = (\sigma_{i^*}, \omega_{i^*}, 0, 0)$ (where ω_{i^*} is the randomness from the encryption algorithm E_{R_E}), to be used for the ZAP proof, rather than the witness attesting to the pseudorandomness of $\mathsf{SK}^S_{i^*}$. However, we still define S_1 use the same witness as S.

Signing algorithm S_2 operates just like S_1 except that we use the alternative witness w attesting to the fact that an ordinary signature of i^* has been encrypted in one of the cipher texts. We note that in this signing algorithm we are no longer dependent on the pseudorandomness of i^*'s digital signature keys and so this signing algorithm would produce a valid signature (meaning it would pass verification by V)

even if the digital signature keys were generated randomly.

We first define an alternate adversarial game, in which the keys are all generated randomly and queries are all answered using signing algorithm S_2 and show that the existence of a successful adversary in this game contradicts the unforgeability of the underlying digital signature scheme. Then we will show, using hybrid experiments, that the existence of a successful adversary in this alternate adversarial game is equivalent to the existence of a successful adversary for the original adversarial game.

Our new adversarial game will be similar to the original game except that we alter the key generation algorithm G and the signing algorithm S. We alter S so that instead of producing signature keys via $\hat{G}^S$, it produces these keys using G^S. We also alter G to store the secret encryption keys produced by G^E. We also replace the signing algorithm S with the alternate signing algorithm S_2 that does not depend on pseudorandom keys. Suppose there exists some PPT adversary A that can produce a forgery in this altered game with non-negligible probability $\frac{1}{p(n)}$ for some polynomial p. We construct an adversary A' that forges the underlying digital signature scheme with non-negligible probability $\frac{1}{p(n)l(n)}$ where $l(n)$ is the polynomial number of ring signature keys given to A.

We know by the soundness of the ZAP that with all but negligible probability, A will not be able to produce proofs for false theorems. We also know that because the digital signature keys were generated randomly, via G^S, rather than pseudorandomly, the probability that there exists some random coins ω so that $\hat{G}^S(1^n;\omega)$ produces one of these keys is negligibly small. Therefore with all but a negligible probability, when A produces a forgery, there is in fact an ordinary signature of one of the ring members encrypted in one of the cipher texts. Our strategy will be to randomly guess in advance, out of the $l(n)$ possible signers, which signer will be used for the ordinary signature produced in the forgery produced by A. We can then create an adversary A' to forge the digital signature scheme. A' will be given a public signing key, PK^S and adaptive access to an oracle for digital signatures. A' will produce $l(n)$ ring signature keys, as the adversarial game mandates, except that for one randomly chosen user $\hat{\imath}$, we will insert PK^S as their public signing key, and not generate a secret signing key for them.

A' can give these keys to A and respond to the oracle and corruption queries as follows: If A requests a signature for a message m with respect to a ring R and a signer $i^* \neq \hat{\imath}$, then we can produce the signature honestly using S_2 because we know the secret key $\mathsf{SK}^S_{i^*}$. If A asks for a signature of a message m for a ring R with $\hat{\imath}$ as the signer, we compute S_2 except for the part where we produce the ordinary signature. For this we take the corresponding $M = m|\mathsf{PK}_{i_1}|...|\mathsf{PK}_{i_{|R|}}$, and submit it to our digital signature oracle, and proceed with the computation of S_2 using the σ that was returned by the oracle. If A corrupts a user $i \neq \hat{\imath}$, we provide their secret key SK_i which we know. If A corrupts user $\hat{\imath}$, we simply abort. Eventually A will output a forgery for a message m and a ring R that it did not query before. With probability negligibly close to $\frac{1}{l(n)}$ of the times that A successfully produces a forgery, this forgery will contain a digital signature σ' of $\hat{\imath}$ of message M corresponding to m on ring R, and we will not have aborted in this run because A could not have corrupted any of the users in the ring R. Since A did not query m with ring R before, the corresponding M could not have been queried by A' to its digital signature oracle. Therefore the A' can decrypt the cipher texts of the signature produced by A and extract the ordinary signature σ' which it can then output as its forgery.

Since A outputs a forgery with non-negligible probability, and A' will be able to extract a digital signature forgery a non-negligible fraction of those times, we have contradicted the unforgeability of the digital signature scheme.

Now we will show that given a PPT adversary A that succeeds with non-negligible probability in the original adversarial game, A also succeeds with non-negligible probability at this second adversarial game. Call our interaction with A via the second adversarial game experiment E_a and our interaction with A via the original adversarial game experiment E_b. We will show that E_a and E_b are computationally indistinguishable using a series of hybrid experiments. We will use the polynomially long sequence of hybrid experiments ($E_a = E_0^0, E_0^1, ..., E_0^{l(n)} = E_1^0, E_1^1, ..., E_1^{q(n)} = E_2^0, E_2^1, ..., E_2^{q(n)} = E_b$), where $q(n)$ is the number of queries adversary A makes to the signing oracle. We define the experiment E_0^i to be the experiment in which the first i signature keys out of the l keys given to A are generated pseudorandomly, while the rest are generated randomly, and all oracle queries are still answered with signing algorithm S_2 as in the second adversarial game. We define E_1^i to be the experiment in which all keys are generated pseudorandomly, as in adversarial game 2, the first i out of $q(n)$ oracle queries are answered with signing algorithm S_1, and the rest are answered with S_2. Finally we define E_2^i to be the experiment in which all keys are generated pseudorandomly, the first i oracle queries are answered with the original signing algorithm S, and the rest are answered with S_1. It can be verified by definition that the equalities written in the sequence of experiments are true. If we use the notation that $\approx$ denotes computational indistinguishability, then we need to show the following three statements: $E_0^i \approx E_0^{i+1}, E_1^i \approx E_1^{i+1}, E_2^i \approx E_2^{i+1}$.

To show the first computational indistinguishability result, we note that the only difference between the two experiments is that one extra digital signature key that is given to the adversary has been generated randomly rather than pseudorandomly. If this difference triggers a difference in behavior in A (such as successfully producing forgeries in one case and not the other), then we can exploit this difference to distinguish pseudorandom keys from random keys, contradicting Lemma 17. To do this, we generate the first i keys pseudorandomly, as both experiments mandate, and we take the digital signature keys for user $i + 1$ as a challenge, being either random or pseudorandom. We generate the remaining keys randomly, also as both experiments mandate. Now we can see that if our challenge keys were generated randomly, we will be running E_0^i, while if they are pseudorandom we will be running E_0^{i+1}. Since A distinguishing between those two experiments would allow us to distinguish between the two distributions, we have shown that $E_0^i \approx E_0^{i+1}$.

For the second pair of experiments, E_1^i and E_1^{i+1}, we note that the only difference is that in the second experiment, the $(i+1)$st oracle query was answered using S_1 rather than S_2. Since the only difference between the two signing algorithms is the witness used in the ZAP proof, we can use a distinguisher between these two experiments to contradict the witness indistinguishability of the ZAP. This proves that $E_1^i \approx E_1^{i+1}$.

For the last pair of experiments, we note that the only difference is that in the second experiment, the $(i+1)$st oracle query is answered with signing algorithm S rather than S_1. Since the only difference between these two signing algorithms is that in S we compute an encryption of 0 while in S_1 we compute an encryption of a signature, the semantic security of the encryption scheme implies that these two experiments are indistinguishable. Otherwise we could use A to distinguish between encryptions of the two messages, 0 and σ. More specifically, we need to take as challenge an encryption scheme and plant this encryption scheme in the adversarial game with A in such a way that for the $i+1$st query, the challenge encryption scheme is used. This means we need to guess randomly one of the users, $\hat{i}$ that will be specified in the ring of the $i+1$st query and set their encryption public keys to be the challenge keys. Then when the $i+1$st query comes along, we can construct two possible messages to encrypt in $\hat{i}$'s part of the encryption E_{R_E}: one message that forces the entire message encrypted to add up to 0, and another message that causes the total message to add up to an ordinary signature signed by the specified author i^*. These two messages can be submitted as the challenge and the returned cipher text can be inserted into the ring signature so that in one case the resulting signature follows the distribution from S and in the other case it follows the distribution of S_1. Since we will guess a user in the ring with probability at least $\frac{2}{l(n)}$ and in this situation A will distinguish between the two distributions with non-negligible probability, this allows us to distinguish between encryptions of the two messages with non-negligible probability, thereby proving that $E_2^i \approx E_2^{i+1}$.

These three computational indistinguishability results can be combined to show that $E_a \approx E_b$. This is really iterating over a polynomial number of computational indistinguishability results, which we can do because a negligible difference multiplied by a polynomial is still negligible. Since the two end experiments are computationally indistinguishable, this shows that the PPT adversary that produces a forgery in our ring signature scheme contradicts the unforgeability of the underlying digital signature scheme, thereby proving that our scheme is unforgeable. ■

Chapter 5

Future Work and Conclusion

We have demonstrated the existence of a ring signature scheme, without random oracles, that achieves the strongest security definitions in the literature, and does so in such a way that does not depend on primitives that might distinguish a ring signature user from someone who never intended to participate in ring signatures and had the same primitives for other purposes. One weakness of our scheme is that instead of having users generate signature keys according to the same distribution as ordinary digital signature users do, we mandate that to participate in ring signatures, one must generate their keys according to a distribution that is different, albeit indistinguishable, from this generic distribution. This is a weakness in the sense that it prevents someone from producing ring signatures if they did not plan on producing ring signatures at the time they generated their keys. While this could be overcome by changing one's keys, it must still be seen as a weakness and a scheme that allowed someone to convert to a ring signature user without any key generation or replacement would be ideal.

One criticism that might be leveled at both the scheme presented here and the scheme in [2] is the dependence on ZAPs. While ZAPs could function as a general purpose tool because they are not designed specifically for ring signatures and it is understandable that many people might publish ZAP keys just so people could prove theorems to them, they are not ubiquitous in the way that public key encryption and digital signature keys are at the moment. An improvement might therefore be to construct a scheme that meets the strongest definitions but without depending on ZAPs.

Another issue to consider is a stronger anonymity definition in which a signer is protected against being framed, in the sense that if they are the only honest user in a ring for which they produced a signature, we would at least want the adversary to not to be able to prove to some separate authority that the signer was indeed the author of the message, even when all secret keys are exposed. We postulate here, without formal proof, that the scheme we present in this paper achieves this guarantee because this would essentially amount to an adversary proving that the signing keys of the author are pseudorandom (which is hard to show without the original randomness) and that all of the keys belonging to the adversary were *not* pseudorandom. Any polynomial time judge that might accept proofs of this form could be fooled by altering the keys of the adversary to be pseudorandom and setting the adversary as the author. In this case the judge would be accepting false theorems and therefore could not be considered an authority to accept such proofs.

It is also foreseeable that the unforgeability requirement could be strengthened by having the adversary's corruption oracle return the randomness that produced the secret keys in addition to just the secret keys. While the unforgeability of the scheme presented here, as well as that of the modified scheme in [2], seem to depend on the fact that the adversary does not have access to this randomness, it is unclear why this should be the case and whether a scheme could be constructed that meets the stronger variation of this unforgeability requirement.

One final direction for future work is to find applications for ring signatures that meet the strongest security requirements. The flexibility of ring signature schemes to include arbitrary groups of users, whether or not they planned on participating in ring signatures, has already produced numerous applications, some of which we have mentioned here, and promises to lead to many more.

Bibliography

[1] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 1-out-of-n signatures from a variety of keys. In *ASIACRYPT '02: Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security*, pages 415–432, London, UK, 2002. Springer-Verlag.

[2] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. Cryptology ePrint Archive, Report 2005/304, 2005.

[3] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984.

[4] David Chaum and Eugène van Heyst. Group signatures. In D.W. Davies, editor, *Advances in Cryptology Eurocrypt 91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer-Verlag, 1991.

[5] Cynthia Dwork and Moni Naor. Zaps and their applications. In *IEEE Symposium on Foundations of Computer Science*, pages 283–293, 2000.

[6] U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *STOC '90: Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 416–426, New York, NY, USA, 1990. ACM Press.

[7] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 25–32, New York, NY, USA, 1989. ACM Press.

[8] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences 28*, pages 270–299, April 1984.

[9] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 291–304, New York, NY, USA, 1985. ACM Press.

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

[11] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups. Cryptology ePrint Archive, Report 2004/027, 2004.

[12] Shai Halevi Ran Canetti, Oded Goldreich. The random oracle methodology, revisited. Cryptology ePrint Archive, Report 1998/011, 1998.

[13] R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In C. Boyd, editor, *Proceedings of Asiacrypt 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–65. Springer-Verlag, Dec. 2001.

[14] R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret: Theory and applications of ring signatures. In A. Rosenberg O. Goldreich and A. Selman, editors, *Theoretical Computer Science Essays in Memory of Shimon Even*, volume 3985 of *Lecture Notes in Computer Science*, pages 164–86. Springer-Verlag, 2005.

[15] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC '90: Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 387–394, New York, NY, USA, 1990. ACM Press.