An Exploration of Group and Ring Signatures

Sarah Meiklejohn

February 4, 2011

Abstract

Group signatures are a modern cryptographic primitive that allow a member of a specific group (e.g., "the White House staff" or "employees of Corporation X that publish press releases") to sign messages on behalf of the group as a whole; i.e., without revealing their individual identities and thus providing them with a certain degree of anonymity and privacy. They have quite a number of potential applications (and in fact have been incorporated into the latest version of the Trusted Platform Module, or TPM), and are still an active area of research. In this work, we explore group signatures and the many variations that have been considered since their original introduction in 1991 by Chaum and van Heyst. We furthermore discuss the basic primitives used to realize group signatures and their numerous definitions of security; we then outline a generic group signature construction (due to Bellare, Micciancio, and Warinschi) that achieves the strongest notion of security and give intuition for how it does this.

As a complement to group signatures we also consider *ring signatures*, in which users can enjoy anonymity properties similar to those of group signatures but can form their groups in an ad-hoc manner; i.e., without any setup or consent required from the other members of these "ad-hoc groups," or rings. We again consider the different notions of security for ring signatures, as well as the different variations on the original concept introduced by Rivest, Shamir, and Tauman in 2001. We then outline a ring signature construction (due to Shacham and Waters) that achieves the strongest notion of security, and conclude with a discussion of open problems for both group and ring signatures.

1 Introduction

Group signatures, originally introduced by Chaum and van Heyst [42], allow members of a specified group to sign messages on behalf of the entire group; i.e., without revealing their individual identities but still guaranteeing that they are in fact a member of the right group. Members of the group thus enjoy *anonymity*, meaning their identities are hard to recover given just a signature they have created. This primitive has quite a number of applications, and is in fact currently deployed in two real-world settings. The first of these applications is *anonymous attestation*, in which a server wants to authenticate a trusted platform running on a user's laptop remotely, but the user would like to preserve his privacy by revealing to the server only that he is in fact a valid user of the trusted platform but not who he is. Direct Anonymous Attestation (DAA) [29], which accomplishes exactly this, is built on top of a variant of a group signature (the original group signature is due to Ateniese et al. [5] and the variant to Brickell, Camenisch, and Chen [29]), and has in fact been adopted by the Trusted Computing Group in the newest version (version 1.2) of their Trusted Platform Module (TPM) [89]. In the second application (first described by Boneh, Boyen, and Shacham [20]), group signatures can be used [1] in a Vehicle Safety Communications (VSC) system [66] to preserve the privacy of its users. In these systems, cars are embedded with dedicated short-range transmitters, which then allow a car

[1] While this approach can be used, in practice some approximation of group signatures is used instead.

to communicate with all the other cars within some small radius in case of emergency; e.g., to let them know that it needs to brake abruptly or perform some similar maneuver. Using a group signature, where the members of the group are all cars equipped to send these types of messages, drivers can protect their privacy by not revealing their exact speed and location when transmitting these safety messages.

More generally, group signatures are useful in any application in which the importance of a signature lies in the fact that it came from the group as a whole, and there is thus no benefit in revealing which specific group member formed the signature. As in any privacy-preserving application, we would like to maintain the anonymity of the users as much as possible, but still need to be sure that there is a mechanism in place for misbehaving members of a group to get caught (as otherwise dishonest or malicious group members could take advantage of their anonymity and cheat with impunity). For example, if the group consists of members of a certain corporation responsible for putting out press releases, a disgruntled employee who publishes and signs off on a false or damaging press release should be able to be identified and (presumably) fired or otherwise penalized. This property is referred to as *traceability*; it is important to note that only a party with a specific piece of information should be able to perform the tracing (in the above example, perhaps the employee's boss), as otherwise the anonymity property discussed above would be violated. We might also say that we would like this authority to use its tracing power only in the case that it is required; e.g., if a press release contains certain keywords or in general some specified policy is not followed, as otherwise it can simply trace messages at will. While this issue is often not addressed in the group signature literature (typically the tracing party is always assumed to be trusted), the notion of "contractual anonymity" was introduced by Schwartz, Brumley, and McCune in 2010 [86] to deal with exactly this problem; essentially, it says that if users obey the policy then they enjoy unconditional anonymity, while if they don't follow the rules they are subject to exposure (but still only to the tracer).

Outside of these two basic properties, anonymity and traceability, there are many variations within group signatures and many additional properties that we may consider. In a typical group signature scheme, a trusted *group master* defines the group of users and issues secret keys to the members; it additionally publishes the public key for the group. The group master may also take on the responsibility of performing the tracing operation, although this functionality may also be split between two authorities; note that this separation has the desirable property that the tracer does not necessarily have all the secret information available to the group master (in fact, in practice it has quite a lot less). Additionally, a *group manager* may be used in place of the group master; the manager differs from the master in that it will interact with the users to help issue their secret keys, but it will not learn them and so only the user will have access to his own secret key. The members of the groups may also be *static*, meaning they are predefined at the start, or *dynamic*, meaning group members can be added throughout the evolution of the group. Finally, schemes that support *revocation* allow the group master/manager to revoke the keys of misbehaving members, meaning they can no longer sign messages on behalf of the group. We discuss all these possible extensions in more detail in Section 4.

Even with a group signature scheme that supports dynamic addition and revocation of users, there is still one fundamental shortcoming with respect to signing flexibility: At the time that a message is signed, the members of the group are in fact fixed and static; i.e., groups cannot be formed on an ad-hoc, signature-by-signature basis. To fill this gap, Rivest, Shamir, and Tauman proposed *ring signatures* in 2001 [82], in which the signer of a message can specify a "ring" at the time of signing. The signature then provides the same anonymity property as a group signature; namely that a recipient of this signature will learn that it was signed by a member of this ad-hoc ring, but not which particular member was responsible for the signing. Rather than the somewhat cumbersome setup required by group signatures, ring signatures assume only that each member has a public key published for a standard digital signature scheme; in some ring signature schemes, participants can even have keys

for different signature schemes (e.g., in the scheme of Rivest et al., any signature scheme based on trapdoor one-way permutations will suffice).

In addition to the lack of any required setup, ring signatures differ from group signatures in another fundamental way: The anonymity of signers is provided *unconditionally*, as there is no tracing authority. Furthermore, individual users have more fine-grained control over their own anonymity, as they can pick the other members of their ring each time they sign a message (as opposed to group signatures, in which users have no control over the other members of the group). The potential applications of ring signatures are thus slightly different from those of group signatures. Recall that with group signatures, our goal was to preserve privacy in settings in which there is simply no benefit or need to reveal individual identities. With ring signatures, on the other hand, we would like to preserve privacy in settings in which it is actively undesirable for a signer's identity to be revealed; for example, in the canonical example of Rivest, Shamir, and Tauman (and in fact the title of their paper), ring signatures can be used to leak a secret, in which the user signing the message is not acting on behalf of any organization but would still like to guarantee anonymity. We will see an example of a ring signature (due to Shacham and Waters [87]) in Section 7.

2 Cryptographic Background

Group and ring signatures are quite advanced primitives; as such, they are often built on a variety of more basic cryptographic primitives. In this section, we provide brief descriptions of primitives that are commonly used (and we will in fact see used for group signatures in Section 6) and their design.

2.1 Public-key encryption

Public-key encryption (along with all of public-key cryptography) was originally introduced in the 1970s [81, 75] as a way to address a fundamental drawback in symmetric-key encryption; namely, the fact that two parties wishing to send messages to each other would have to first come up with a way to share a secret key between them. Since then, public-key encryption has remained one of the most well-studied primitives in cryptography [55, 49, 79, 88, 1, 30, 21, 22, 43, 65].

Formally, a public-key encryption scheme consists of three algorithms: a randomized KeyGen algorithm, a randomized Enc algorithm, and a deterministic Dec algorithm. To generate keys, a user will run the KeyGen algorithm (on input some security parameter 1^k) to get a public encryption key pk and a secret decryption key sk; this public key can then be put in a registry somewhere and associated with the user. If someone else would like to encrypt some message m for this user, she can look up his public key and compute $c \leftarrow \mathsf{Enc}(pk, m)$, where we call c the ciphertext. Given this ciphertext, the user can then decrypt it using his secret key to get back $m = \mathsf{Dec}(sk, c)$.

In terms of security, there are two main notions we can consider for encryption: *IND-CPA security*, which is short for INDistinguishability against Chosen Plaintext Attack, and *IND-CCA security*, which is short for INDistinguishability against Chosen Ciphertext Attack. Informally, the former of these says that an adversary, given a challenge ciphertext on one of two messages (where these messages are chosen by the adversary, hence the name), cannot tell which message the ciphertext is encrypting. The stronger notion, IND-CCA security [47, 43], says that an adversary still cannot distinguish between an encryption of either of the two messages, even if it is allowed to see decryptions of arbitrary ciphertexts (though obviously not the exact challenge ciphertext) both before and after it is given its challenge ciphertext.

2.2 Digital signatures

Digital signatures were introduced at the same time as public-key encryption [81, 80] and can be thought of as the digital analog of a physical signature; that is, they provide evidence that a given message was in fact written by the person who says they wrote it. Like public-key encryption, they have been very well studied since their introduction [49, 76, 85, 18, 34, 57, 77, 3] and have been extended and enhanced in many ways [39, 88, 17, 23, 12, 41, 50, 74], including group signatures themselves.

Formally, there are three algorithms we can consider in the public-key setting: a randomized KeyGen algorithm, a randomized (but deterministic in the case of *unique signatures* [73]) Sign algorithm, and a deterministic Verify algorithm. A user will run the KeyGen algorithm to generate two keys: his public verification key pk and his secret signing key sk; the public key will then be published in some public-key registry. When the user wants to sign a message m, he can compute $\mathsf{Sign}(sk, m)$ to generate a signature σ. A recipient of this message and its signature can look up the public key for the user and compute $\mathsf{Verify}(pk, \sigma, m)$. If this outputs 1 then the recipient can be convinced that the message really did come from the sender; otherwise, if it outputs 0, then the recipient knows that the message was in fact sent by some impostor attempting to forge the sender's signature (or, less cynically, that the signature was just malformed in some way).

In terms of security, definitions for signatures were first given by Goldwasser, Micali, and Rivest [57]. Informally, we would like to say that signatures are *unforgeable*, meaning no one can forge a signature on someone else's behalf. More formally, the strongest notion of security considered by Goldwasser et al. is known as EUF-CMA security, which is short for Existential Unforgeability against Chosen Message Attack; this essentially extends our intuition and says that, even after seeing arbitrarily many signatures on any messages of his choice, an adversary still cannot produce a valid forgery (on any message).

2.3 Zero-knowledge proofs

Zero-knowledge proofs were originally introduced in the 1980s [56, 54] as a way to allow someone to prove that a given statement is true without revealing anything beyond the validity of the statement; even in this early work, such proofs were shown to exist for all languages in NP. Since then, there has been much work done to improve and extend their usefulness [37, 44, 52, 62, 71, 63, 84] and they have been used in a wide variety of applications [60, 72, 31, 10, 40, 32, 9].

A particularly useful kind of zero-knowledge proof is a *non-interactive zero-knowledge proof* (NIZK for short) [16, 51], in which no interaction is required between the prover and the verifier. A NIZK is thus a single message (the proof) send from a prover P to a verifier V, where both parties have access to some common random string R.[2] Before this message can be sent, a Setup algorithm must be run first (either by a trusted third party or in some cases jointly by the prover and the verifier) to obtain the common random string R. We then use the notation $\pi \leftarrow P(R, x, w)$ to mean the proof π computed by the prover for the statement x and using witness w, and $V(R, x, \pi)$ to mean the verification of the proof π for the statement x (and with both parties having access to the random string R).

In terms of security, there are two main properties that we expect from a zero-knowledge proof: soundness and zero knowledge. Informally, the soundness property guarantees that the prover is being honest; that is, that even an all-powerful prover cannot trick the verifier into thinking that a false statement is true. On the other side of things, the zero knowledge property protects the privacy of the prover by guaranteeing that the verifier will not learn anything beyond the validity of the statement (so in particular, will not learn anything about any secret information the prover may have access to).

[2] In many cases a common *reference* string is in fact required, but we stick with the common random string model for simplicity.

3 Definitions and Notation for Group and Ring Signatures

Before we give the formal definitions for group and ring signatures, we can also consider two notions that are used in all of cryptography. The first, a *negligible* function, means a function $\nu(\cdot)$ such that for all $k \in \mathbb{N}$, there exists an integer x_0 such that for all $x > x_0$, $\nu(x) < 1/x^k$; in other words, a function that grows slower than the inverse of any polynomial. We will also consider adversaries that are allowed to make random choices but are constrained to run in time polynomial in the size of their inputs; we refer to such adversaries as *probabilistic polynomial-time* adversaries, or PPT for short.

3.1 Group signatures

Formally, a group signature scheme[3] consists of four algorithms: KeyGen, Sign, Verify, and Trace, where the first two are randomized and the second two are deterministic. The KeyGen algorithm, on input the security parameter 1^k and the number of users 1^n in the group, outputs a group public key pk, a secret key msk intended only for the group master, and a set $\{sk_i\}_{i=1}^n$ of secret keys, where sk_i represents the secret key for user i. The Sign algorithm, on input a secret signing key sk_i and a message m, returns a signature σ on m under sk_i. The Verify algorithm, on input the group public key pk, a signature σ, and a message m, outputs either 0 or 1, depending on whether or not σ really was created by a member of the group. Finally, the Trace algorithm, on input the master secret msk and a signature σ on some message m, outputs either a user identity i or $\perp$ to indicate failure (i.e., that it was unable to determine which user signed the message, or that σ was not a valid signature on m).

In general, there are two main properties we would like from group signature schemes: *anonymity* and *traceability*. Intuitively, anonymity says that a recipient of a group signature should be unable to tell which member of the group formed the signature; formally, we have the following definition:

Definition 3.1. [11] *For a group signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Trace})$*, a given adversary* $\mathcal{A}$ *and a bit* $b \leftarrow \{0,1\}$ *unknown to* $\mathcal{A}$*, define the following game:*

- *Step 1.* $(pk, msk, \{sk_i\}) \leftarrow \mathsf{KeyGen}(1^k, 1^n)$.
- *Step 2.* $\mathcal{A}$ *is now given* pk *and* $\{sk_i\}$ *and may make arbitrarily many requests to a* $\mathsf{Trace}(msk, \cdot, \cdot)$ *oracle to trace any signatures of its choice (i.e., valid signatures on any message* m*, signed using any secret key* sk_i*). It will also keep some state information* s*.*
- *Step 3.* $(m, i_0, i_1) \leftarrow \mathcal{A}(pk, \{sk_i\}, s)$*, where we have* $i_0 \neq i_1$ *and* $1 \leq i_0, i_1 \leq n$*.*
- *Step 4.* $\mathcal{A}$ *is now given* $\sigma \leftarrow \mathsf{Sign}(sk_{i_b}, m)$*. It may again query the* $\mathsf{Trace}(msk, \cdot, \cdot)$ *oracle at will, but this time with the restriction that it cannot query it on* σ*.*
- *Step 5. In the end,* $\mathcal{A}$ *outputs a bit* b'*.*

We say that the group signature is fully anonymous *if for all PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *and a security parameter* k_0 *such that for all* $k > k_0$ *the probability (over the choices of* b *and the randomness used in* KeyGen *and* $\mathcal{A}$*) that* $b' = b$ *is at most* $1/2 + \nu(k)$*.*

As we can see, this definition is quite strong: In the game, the adversary is given access to every single member secret key, and has full access to the tracing oracle both before and after the challenge identities are picked. A slightly weaker setting in which the adversary does not have this oracle access

[3]Here we stick with the most basic definition: a static group with only a group master (i.e., no separate tracer or tracing key). Further variants and their definitions are discussed in Section 4.

turns out to still be meaningful (the difference between the two settings is analogous to the difference between IND-CCA- and IND-CPA-secure encryption; see Section 2.1 for a reminder), and is in fact the setting used (and introduced) by Boneh, Boyen, and Shacham [20], as well as Boyen and Waters [27, 28].

For traceability, we intuitively would like to say that a misbehaving member of the group will be caught, even if members of the group are attempting to alter their own secret keys or even forming collusions to frame other members or otherwise deflect blame. We have the following definition:

Definition 3.2. **[11]** *For a group signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Trace})$ *and a given adversary* $\mathcal{A}$*, define the following game:*

- *Step 1.* $(pk, msk, \{sk_i\}) \leftarrow \mathsf{KeyGen}(1^k, 1^n)$.
- *Step 2.* $\mathcal{A}$ *is now given* pk *and* msk *and is allowed to pick any subset* $\mathcal{C}$ *of users to corrupt. In picking this set* $\mathcal{C}$*,* $\mathcal{A}$ *is given access to a signing oracle, which it can provide with an identity* i *and a message* m *to obtain* $\mathsf{Sign}(sk_i, m)$*; once* $\mathcal{A}$ *has picked a user it is then given access to the secret key for that user and can pick the rest of its users adaptively.*
- *Step 3.* $\mathcal{A}$ *is now allowed to retain access to the signing oracle from Step* 2*, as well as its access to the secret keys of its corrupted users and the master secret key* msk*. At the end of this step,* $\mathcal{A}$ *outputs a pair* (m, σ).

We say that the group signature is fully traceable *if for all such PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *and a security parameter* k_0 *such that for all* $k > k_0$ *the probability (over the randomness used in* KeyGen*,* Sign*, and* $\mathcal{A}$*) that* $\mathsf{Verify}(pk, \sigma, m) = 1$ *and there exists an* i *such that (1)* $\mathsf{Trace}(msk, \sigma) = i$*, (2)* $i \notin \mathcal{C}$*, and (3)* $\mathcal{A}$ *did not query its signing oracle on* (i, m) *is at most* $\nu(k)$.

3.2 Ring signatures

As introduced by Rivest, Shamir, and Tauman [82], ring signatures consist of only two algorithms: Sign and Verify; this encapsulates the intuition that ring signatures are essentially "setup-free" (i.e., don't require the KeyGen algorithm) and unconditionally anonymous (as there is no Trace algorithm). In more recently proposed ring signature schemes, however, a KeyGen algorithm has been added as a way to guarantee that all users have the same kind of keys. Therefore, for the purposes of security definitions we assume that a ring signature scheme consists of three algorithms: KeyGen, Sign, and Verify. Each user will run KeyGen individually; this algorithm, on input the security parameter 1^k, will output a keypair (pk, sk). The Sign algorithm, on input a secret key sk, a ring R (typically just a list of public keys belonging to members of the ring), and a message m, outputs a signature σ on m. Finally, the Verify algorithm, on input the ring R, a signature σ, and a message m, outputs 1 if some member of R created the signature σ on m and 0 otherwise.

Intuitively, we would like ring signatures to be secure in ways similar to group signatures. It turns out we can achieve an anonymity notion very similar to that in Definition 3.1, but without a Trace algorithm we obviously cannot hope to achieve anything that looks like traceability. We would still like to be sure that non-ring members cannot forge signatures, and so we instead consider the slightly weaker property of *unforgeability*. Both these properties were first defined formally by Bender, Katz, and Morselli [15].

As defined by Bender et al., there are three possible levels of anonymity we can achieve: basic anonymity, in which the adversary sees only public keys; anonymity with respect to adversarially-chosen keys, in which the adversary (as the name implies) can pick its own keypairs and thus essentially create its own users; and finally, anonymity with respect to full key exposure, in which the adversary

can continue to pick its own keypairs but also gets to see the secret keys for each user. As this last (and strongest) definition most closely parallels the definition given above for group signatures, we present it here.

Definition 3.3. **[15]** *For a ring signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$*, a given adversary* $\mathcal{A}$ *and a bit* $b \leftarrow \{0,1\}$ *unknown to* $\mathcal{A}$*, define the following game:*

- *Step 1. The* KeyGen *algorithm is run* m *times to obtain a set* $((pk_1, sk_1), \ldots, (pk_m, sk_m))$ *of keypairs.*
- *Step 2.* $\mathcal{A}$ *is now given access to the set of public keys* $S = \{pk_i\}$ *and a signing oracle; i.e., an oracle that, given any index* i*, any ring* R *(so it may be the case that* $R \not\subset S$*), and any message* m*, will output* $\mathsf{Sign}(sk_i, R, m)$*. At the end of this step* $\mathcal{A}$ *will save some state information* s*.*
- *Step 3.* $(i_0, i_1, R, m) \leftarrow \mathcal{A}(s)$*. Note that it again may be the case that* $R \not\subset S$*, but it must be the case that* pk_{i_0} *and* pk_{i_1} *are both in the ring* R*, and that* $i_0 \neq i_1$ *and* $1 \leq i_0, i_1 \leq |R|$*.*
- *Step 4.* $b' \leftarrow \mathcal{A}(\sigma = \mathsf{Sign}(sk_{i_b}, R, m), \{sk_i\})$*.*

We say that the ring signature is anonymous against full key exposure *if for all such PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *and a security parameter* k_0 *such that for all* $k > k_0$ *the probability (over the choices of* b *and the randomness used in* KeyGen*,* Sign*, and* $\mathcal{A}$*) that* $b' = b$ *is at most* $1/2 + \nu(k)$*.*

As with anonymity, we can consider varying degrees of strength in our definitions of unforgeability. The weakest, unforgeability against fixed-ring attacks, considers an adversary who does not get to pick its ring R but is rather handed one at the start. A slightly stronger property, unforgeability against chosen-ring attacks, considers an adversary who *does* pick its own ring. Finally, and strongest, the adversary is allowed to pick any subset of ring members to corrupt, meaning it is given access to their secret keys. Again, this strongest definition most closely parallels the analogous property for group signature (from Definition 3.2) and so we present it here.

Definition 3.4. **[15]** *For a ring signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *and a given adversary* $\mathcal{A}$*, define the following game:*

- *Step 1. The* KeyGen *algorithm is run* m *times to obtain a set* $((pk_1, sk_1), \ldots, (pk_m, sk_m))$ *of keypairs.*
- *Step 2.* $\mathcal{A}$ *is now given* $S = \{pk_i\}$ *and is allowed to pick any subset* $\mathcal{C}$ *of users to corrupt. In picking this set* $\mathcal{C}$*,* $\mathcal{A}$ *is given access to a signing oracle; i.e., an oracle that, given any index* i*, any ring* R *(so it may be the case that* $R \not\subset S$*), and any message* m*, will output* $\mathsf{Sign}(sk_i, R, m)$*. Once* $\mathcal{A}$ *has picked a user it is then given access to the secret key for that user and can pick the rest of its users adaptively. At the end of this step* $\mathcal{A}$ *will save some state information* s*.*
- *Step 3.* $(R^*, m^*, \sigma^*) \leftarrow \mathcal{A}(s)$*.*

We say that the group signature is unforgeable with respect to insider corruption *if for all such PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *and a security parameter* k_0 *such that for all* $k > k_0$ *the probability (over the randomness used in* KeyGen*,* Sign*, and* $\mathcal{A}$*) that* $\mathsf{Verify}(R^*, \sigma^*, m^*) = 1$ *and there exists a* j *such that (1)* $\mathcal{A}$ *did not query its signing oracle on* (j, R^*, m^*) *and (2)* $R^* \subseteq \{pk_i\} \setminus \mathcal{C}$ *is at most* $\nu(k)$*.*

For both the varying degrees of anonymity and unforgeability, Bender et al. prove separation results; that is, that the definitions are distinct and each level up is strictly stronger than the one below it.

4 Background on Group Signatures and Variants

In this section, we consider some of the many possible extensions of the basic four algorithms (KeyGen, Sign, Verify, and Trace) that make up a group signature scheme; we focus in particular on the use of dynamic groups with a group manager and on the notion of *revocation*.

Before we discuss these variants, it is important to first establish some background. As mentioned in Section 1, group signatures were originally introduced by Chaum and van Heyst [42] in 1991, who both proposed them as a useful primitive and provided the first constructions. In 1997, Camenisch and Stadler [36] proposed the first group signature scheme in which the size of both the public key and the signatures did not depend on the size of the group; i.e., their size was constant. Their construction involves a combination of various cryptographic primitives (referred to by Kiayias and Yung as the "single-message and signature-response paradigm" [70]), and in fact this method of construction was followed by a number of schemes for years afterward [5, 70, 6, 7, 20].

In 2003, Bellare, Micciancio, and Warinschi [11] introduced the modern definitions and security properties that we now use for group signatures (and just saw in Section 3.1); they additionally gave a generic construction of a scheme that satisfied these (quite strong) security properties. Their scheme, as we will see in Section 6, combines digital signatures, IND-CCA-secure public-key encryption, and non-interactive zero-knowledge proofs, and works for static groups (i.e., groups in which the set of users is defined at the start) with a group master. Like the Camenisch-Stadler construction, their construction has proved to be quite useful; in fact, the first group signature scheme based on lattices, introduced recently at the end of 2010 by Gordon, Katz, and Vaikuntanathan [59], follows the exact outline of the Bellare et al. construction.

As mentioned above, the security properties defined by Bellare et al. are quite strong, and in fact were not realized by an efficient scheme until a scheme due to Groth in 2006 [61].[4] In 2004, therefore, Boneh, Boyen, and Shacham [20] introduced a slightly weakened definition of security along with a scheme that satisfied this new definition. As mentioned in Section 3, their definition simply removes the access to the Trace oracle from the game in Definition 3.1. Boneh, Boyen, and Shacham argue that, in practice, access to the Trace functionality will be tightly controlled, and so this definition of security is still meaningful.

4.1 Dynamic groups and group managers

The notion of a *group manager*, who helps to issue keys to enrolling members but does not in fact learn these keys, was considered as early as the Camenisch-Stadler scheme [36], and a bit more in depth later by Kiayias and Yung [69] and Kiayias, Tsiounis, and Yung [68]. In 2005, Bellare, Shi, and Zhang [14] gave formal security definitions for groups with a group manager, as well as *dynamic groups*; i.e., groups in which users can be added as time goes by. Recall that in our basic group signature scheme, the KeyGen algorithm was run by the group master at the beginning to both set up the parameters for the group, as well as the secret keys for each of its individual members; each of these secret keys was then presumed to be passed along to the appropriate member along some secure channel. There are two main shortcomings associated with this approach: (1) the group master knows the secret key for each of the members, and (2) the members of the group must be fixed at the start and cannot change over time.

Before we consider the case where both these pitfalls are avoided, it is worth mentioning that these properties do not necessarily have to occur at the same time; that is, we can have a scheme in which users are allowed to enroll at any time, but when they enroll they are still simply handed their

[4] As noted by Groth himself, although the operations in the scheme can be considered efficient and the signatures are of constant size, the constant is far too large for the scheme to be considered remotely practical.

Scheme	Group type/operations	Master/manager?	Assumptions used
CS97 [36]	dynamic join	manager	DLP, strong RSA
BMW03 [11]	static	master	trapdoor permutations
BBS04 [20]	static join but revocation	master	q-SDH, DLIN, random oracle
BW06 [27]	dynamic join and revocation	master	CDH, SGH

Table 1: A comparison of some well-studied group signature schemes. We consider the possible group types (static or dynamic) and the operations supported (addition or revocation), whether the group has a master or a manager, and finally which cryptographic assumptions it relies on for security. As far as assumptions go, we can see that the Camenisch-Stadler scheme relies on the hardness of the Discrete Log Problem and the Strong RSA assumption [8]; the security of the Bellare-Micciancio-Warinschi scheme relies on the existence of trapdoor permutations; the security of the Boneh-Boyen-Shacham scheme relies on the security of the q-SDH (short for Strong Diffie Hellman and introduced by Boneh and Boyen [19]) and DLIN (short for Decision LINear and introduced by the authors) assumptions, as well as the existence of random oracles; and finally the security of the Boyen-Waters scheme relies on the CDH (Computational Diffie Hellman) and SGH (short for SubGroup Hiding and introduced by Boneh, Goh, and Nissim [24]) assumptions.

keys by the group master. In practice, such a scheme can be easily emulated by the standard static construction as follows: During the KeyGen phase, the group master will simply specify the number of group members 1^n as the maximum number of members he would ever expect; he should then have enough keys to continue handing them out throughout the evolution of the group.

While the above scheme seems to solve the problem of dynamic enrollment, the first problem is still left unresolved. To solve both problems at the same time, we need to augment the basic group signature scheme as follows: The KeyGen algorithm is replaced with a Setup algorithm which outputs the group public key pk and (possibly) some common parameters $params$, as well as the group manager secret key msk and a separate tracing key tk. In addition, a new interactive protocol $\mathsf{Join}() \leftrightarrow \mathsf{Enroll}(msk)$ is added that takes place between the enrolling member and the group manager. This protocol is essentially a secure two-party computation, at the end of which user i learns their secret key sk_i but nothing else (so in particular, nothing about the manager's secret key msk), and the group manager learns nothing except that the user is now an enrolled member. Finally, the Trace algorithm is changed to take in the tracing key tk rather than the manager secret key msk, as it is assumed that the tracing authority should be separate from the group manager and thus not be able to enroll users (and vice versa).

As mentioned, this setting was fully formalized by Bellare, Shi, and Zhang [14], who noted that with a weaker issuing authority (i.e., the group manager as opposed to a master), new security notions could be considered.[5] They therefore define the notion of *non-frameability* (also called *strong exculpability* by Boneh, Boyen, and Shacham [20] and expanding on the informal notion of *exculpability* introduced by Ateniese and Tsudik [6]), which says that no coalition of corrupt group members, not even ones that include the group manager, can produce a signature on behalf of another group member.

Finally, we mention that, in addition to considering potentially corrupt group managers, Bellare et al. (and others after them) also consider the notion of potentially corrupt tracing authorities. To this end, many dynamic group signature schemes include another new algorithm in addition to the ones described above; namely, a Judge algorithm. Essentially, the Trace algorithm is now required to output not only the member it believes created the signature, but also a proof that this is so. The Judge algorithm can then decide whether or not this proof is in fact true, so that the tracing authority

[5]Slightly different security definitions were previously given by Kiayias, Tsiounis, and Yung [68] and Kiayias and Yung [69], but we stick here with the formalization due to Bellare et al.

is bound in some sense to be honest.

4.2 Revocation

Another property of a group signature scheme is the opportunity to *revoke* the signing privileges of misbehaving members [7, 25, 45, 6]; that is, perform some operation such that a member who has misbehaved can no longer sign on behalf of the group. As an example of where this might be used, consider that even if a group member publishes their signing key online (or has that information stolen), the group can still continue to function as it should, as the group master can simply revoke the signing privileges of that member. This property is therefore quite desirable for a group signature, as it means that the scheme is able to protect itself and recover from these sorts of failures. One straightforward way to achieve revocation is for the group master (or manager), at the time that a failure of this type occurs, to recreate a public key and a new secret key for each member remaining in the group (or, in the case of the manager, re-run the Join/Enroll algorithm with each member); the members who do not get new keys are thus effectively shut out of the group. In practice, this solution is quite costly, as a message over a secure channel needs to be sent to every group member, and a broadcast message must be sent to all potential verifiers.

A more practical revocation technique, formalized by Boneh and Shacham in 2004 [25], is called *verifier-local revocation*. As the name implies, the work of checking that revoked users cannot create signatures that pass verification is put on the verifier (as opposed to say, the remaining signers). The standard group signature scheme is therefore augmented by giving the Verify algorithm access to not only the public key pk and the signature σ and message m in question, but also to a *revocation list* RL that contains tokens to uniquely identify signers whose member privileges have been revoked. A revoked member therefore will lose any sense of privacy, as their signatures can now be linked by any verifier who simply runs Verify once without the revocation list and once with it; if certain signatures verified in the former case but not in the latter, then the verifier now knows which messages were signed by the revoked member. Because of this, Boneh and Shacham must define a new notion of anonymity, which they call *selfless anonymity*; intuitively, this says that because a user has the ability to "revoke himself," he can in fact check and see (using the above technique) if a particular signature was created using his key. In addition to the scheme introduced by Boneh and Shacham along with the formal definitions, several schemes have been proposed that also satisfy this notion of revocation [78, 4].

In more informal settings, other revocation methods have also been proposed, typically on an ad-hoc basis. Boneh, Boyen, and Shacham [20, Section 6], for example, describe a revocation method for their scheme (which in turn follows a revocation mechanism due to Camenisch and Lysyanskaya [33, 35]) in which a revocation list is again published, but this time is given to remaining signers as well as verifiers. It is then used to update the public key for the group; additionally, remaining signers have the ability to update their secret keys to be consistent with this new public key, while revoked signers do not.

Finally, Boyen and Waters [27, Section 5.4] also describe a way in which their scheme can be extended to support revocation. In their setting, both the Sign and Verify algorithms must be augmented to support a proof, created by the signer, that they are not in fact a revoked member (so there must also be a public list of revoked members); the verifier will then check this proof before proceeding to standard verification. As the signer must create a proof for each revoked member, Boyen and Waters also mention that once there are enough revoked members for this approach to become impractical, the group master can resort to the naïve approach described at the beginning in which the entire group is essentially re-keyed.

5 Background on Ring Signatures

In this section, we discuss previous work in ring signatures. Although the extensions of ring signatures are not as widely varied as those for group signatures, the problem of constructing a secure ring signature has still been approached from a number of different angles, and so we highlight these different approaches here.

5.1 Generic vs. non-generic constructions

In the original scheme of Rivest, Shamir, and Tauman [82], the setup assumptions were quite minimal: Users were assumed only to have generated signing keypairs for any signature scheme whose security relied on the existence of trapdoor permutations. This meant that their construction was fundamentally generic, as it could not rely on any particular properties or forms of the keys. Another generic construction, due to Bender, Katz, and Morselli [15], was proposed in 2006 that satisfied the much stronger definitions of security they defined (the ones we saw in Section 3.2). Their construction, similar to the generic group signature construction due to Bellare, Micciancio, and Warinschi [11], combines public-key encryption (although they require only IND-CPA security, as opposed to IND-CCA), signatures, and a primitive similar to zero-knowledge proofs called ZAPs [48].[6] Finally, we mention that other generic ring signature schemes have been proposed based on a variety of assumptions, including the discrete log assumption [2, 64], the RSA assumption [46], or a mixture of the two [2].

Somewhat surprisingly, the literature for ring signatures with efficient protocols is much less extensive than what we see for generic constructions (note that these are distinct notions, since generic constructions can essentially never be truly efficient). In 2003, Boneh, Gentry, Lynn, and Shacham [23] introduced an efficient ring signature scheme, secure only in the random oracle model. In 2007, Shacham and Waters [87] introduced the first efficient ring signature scheme that was secure without random oracles; we will see an outline of this scheme in Section 7. Also in 2007, Boyen introduced the concept of *mesh signatures* [26], which are a generalization of ring signatures. In the language of Boyen, ring signatures can be viewed as a disjunction of signatures, in which the statement being shown is that at least one member of a ring signed a particular message (so either Member A or Member B or ...). In mesh signatures, more complex structures can be used, and in fact any monotone access structure is supported (e.g., conjunction). As a special case of mesh signatures, Boyen demonstrates an efficient ring signature, using less attractive assumptions than Shacham and Waters but still secure without random oracles.

5.2 Ring signature size

In almost all ring signature constructions, the size of the ring is implicitly assumed to be linear in the number of the members, as the natural way to describe a ring is with a list of the public keys of its members. In a result due to Dodis, Kiayias, Nicolosi, and Shoup [46], however, the authors manage to avoid this linear dependence by arguing that some rings can have short descriptions; e.g., "members of the White House staff." Furthermore, they argue that in practice, certain rings may end up being re-used quite often; if these rings are not being created fresh for every single signature, then they can be assigned some sort of unique description or identifier. Using this intuition, Dodis et al. describe a ring signature scheme with constant-size signatures, as opposed to the linear-size signatures used in

[6] A ZAP is in fact weaker than a zero-knowledge proof, as it achieves a related property called *witness indistinguishability*, which says that the verifier doesn't learn which witness was used by the prover, but not that the verifier learns *nothing* about the witness. The scheme they outline that uses only these three primitives, however, does not actually achieve their strongest notions of security, as it is not anonymous against full key exposure. To achieve this, they require the use of an *oblivious key generator*, which we will not discuss here.

all previous schemes. There is a drawback in the scheme, however, as its security fundamentally relies on the use of random oracles [13]. To address this, Chandran, Groth, and Sahai [38] came up with a scheme that achieves sub-linear size without random oracles; while this is certainly an improvement over linear-size signatures, their signatures are still of size $O(\sqrt{N})$ (where N is the number of members in the ring) as opposed to the constant size achieved by Dodis et al.

6 A Generic Group Signature Construction

In this section, we outline a generic group signature construction, due to Bellare, Micciancio, and Warinschi [11], that satisfies the security requirements described in Definition 3.1. The construction combines digital signatures [58], IND-CCA-secure public-key encryption, and simulation-sound non-interactive zero-knowledge proofs (NIZKs) [84]; for summaries of these three primitives, see Section 2.[7]

Intuitively, the construction works as follows: The KeyGen algorithm will first create keys for the encryption scheme and the signature scheme, as well as the common random string for the NIZK. The group master, for each user, will create a signing keypair and then a signature (under the group master's secret signing key) on the user's identity and public key; this signature will essentially act as a certificate that guarantees the group master really did assign the user a keypair. With the secret key from this keypair, the user can then sign a message; he cannot simply reveal this signature, however, as it completely exposes his identity. He therefore encrypts this signature under the group's encryption scheme, along with the certificate from the group master, and finally forms a NIZK that the ciphertext really does contain this valid certificate and the signature. The final group signature will then consist of this NIZK and the ciphertext; a recipient of this signature can then check that the NIZK is correct to be sure that the message was in fact signed by a member of the group. More formally, the scheme works as follows:

- $\mathsf{KeyGen}(1^k, 1^n)$: Compute keypairs for the encryption and signing schemes as $(pk_e, sk_e) \leftarrow \mathsf{KeyGen}_e(1^k)$ and $(pk_s, sk_s) \leftarrow \mathsf{KeyGen}_s(1^k)$, in addition to a random value $R \leftarrow \{0,1\}^{p(k)}$ that will act as the common random string (CRS) for the NIZK scheme. Set $pk = (R, pk_e, pk_s)$ and $msk = (sk_e, sk_s)$. Now, for each individual secret key, compute the keypair $(pk_i, sk_i) \leftarrow \mathsf{KeyGen}_s(1^k)$ and a certificate $cert_i \leftarrow \mathsf{Sign}(sk_s, (i, pk_i))$, and set user i's secret key to be $gsk_i = (i, sk_i, cert_i)$.

- $\mathsf{Sign}(gsk_i, m)$: First compute a signature $s \leftarrow \mathsf{Sign}(sk_i, m)$. Next, compute some randomness $r \leftarrow \{0,1\}^k$ and use it to compute the ciphertext $c \leftarrow \mathsf{Enc}(pk_e, (i, pk_i, cert_i, s); r)$. Next, compute a proof $\pi \leftarrow P(R, (pk_e, pk_s, m, c), (i, pk_i, cert_i, s, r))$; i.e., a proof that the certificate contained in the ciphertext is in fact a signature on the value (i, pk_i). Finally, form the group signature as $\sigma = (c, \pi)$ and return this value σ.

- $\mathsf{Verify}(pk, \sigma, m)$: Return $V(R, (pk_e, pk_s, m, c), \pi)$.

- $\mathsf{Trace}(msk, \sigma)$: If $V(R, (pk_e, pk_s, m, c), \pi) = 0$ then return $\perp$. Otherwise, compute $m = \mathsf{Dec}(sk_e, c)$, parse it as $m = (i, pk_i, cert_i, s)$, and return i.

The security of the group signature scheme fundamentally relies on the security of the underlying primitives (i.e., the signature scheme, encryption scheme, and simulation-sound NIZKs). Because all these primitives can be generically constructed from trapdoor permutations [47, 83, 84, 51], Bellare, Micciancio, and Warinschi are able to prove the following result:

[7] Recall that in Section 2, we in fact describe standard NIZKs but not simulation soundness. As this is a fairly technical definition we will not present it here, but we refer interested readers to the original paper by Sahai [84].

Theorem 6.1. **[11]** *If there exists a family of trapdoor permutations, then the group signature scheme outlined above is fully anonymous and fully traceable (as outlined in Definitions 3.1 and 3.2).*

Informally, we can describe how the security of the scheme relies on the security of its underlying primitives. First, we consider anonymity, which relies on the IND-CCA security of the encryption scheme and the zero knowledge property of the NIZK. Although the NIZK π is given out directly, the zero knowledge property tells us that a recipient will still not be able to learn anything about the witness used to compute the proof (which completely reveals the user's identity). Similarly, although the encryption of the certificate c is also given out directly, the IND-CCA security of the encryption scheme guarantees that a recipient will not be able to learn the contents of this ciphertext, and thus not learn the certificate or other contents (which, as they are identical to the witness for the NIZK, would again completely reveal the user's identity).

For traceability, we now rely on the soundness of the NIZK and the unforgeability of the signature scheme. By the soundness property of the NIZK, we know that π will not pass verification unless the certificate $cert_i$ and signature s were in fact valid (i.e., $cert_i$ really was signed by the group master and s really was signed by a group member). But, the unforgeability of the signature scheme guarantees that the only way for these two values to be considered valid is if they were in fact signed by the right people (the group master for the former and the possessor of sk_i for the latter), and so we can be assured that any valid group signature will in fact trace back to the appropriate group member.

7 A Ring Signature Construction

In this section, we will see an outline of a ring signature construction due to Shacham and Waters [87]. Unlike the group signature construction we just saw, this construction is not generic, meaning it uses properties of its keys and signatures, which requires them to have specific forms. Additionally, the scheme uses *pairings*, which requires us to introduce some more notation. Given a group G of some order N, we say that G is a *bilinear group* if it is cyclic (meaning every element in G can be written as g^a for some generator g and exponent $a \in \mathbb{Z}/N\mathbb{Z}$) and if there exists some map $e : G \times G \to G_T$ such that e is *nondegenerate*, meaning if $e(x, y) = 1$ for all y then it must be the case that $x = 1$, and *bilinear*, meaning if g is the generator for G then we have $e(g^a, g^b) = e(g, g)^{ab}$ for all $a, b \in \mathbb{Z}/N\mathbb{Z}$.

For the Shacham-Waters construction, we will particularly be interested in bilinear groups with order $N = pq$, where p and q are primes. By the structure theorem for finite abelian groups, we know that if $|G| = N$ then G will have a decomposition of its own into $G = G_p \times G_q$, where G_p is the subgroup of order p and G_q is the subgroup of order q. The *Subgroup Hiding* assumption (SGH for short, and introduced by Boneh, Goh, and Nissim in 2005 [24]) says that a random element h of the subgroup G_q will be indistinguishable from a random element of the full group G. We will also need the *Computational Diffie Hellman* assumption (CDH for short), which says that given $g^a, g^b \in G$ for random $a, b \leftarrow \mathbb{Z}/N\mathbb{Z}$, it is hard to compute g^{ab}.

Finally, the construction relies on the existence of *collision-resistant hash functions*, or CRHFs for short. A CRHF is a hash function $H : \{0,1\}^* \to \{0,1\}^n$ for some fixed n such that it is infeasible to find two messages m_1 and m_2 such that $H(m_1) = H(m_2)$ but $m_1 \neq m_2$. Collision-resistant hash functions, and hash functions in general, have been very well studied in cryptography; we refer the reader to any introductory text in cryptography (e.g., Goldreich [53] or Katz and Lindell [67]) for more information.

Intuitively, the construction works as follows: the bilinear group and related values will be constructed by some trusted third party using a $\mathsf{Setup}(1^k)$ algorithm which outputs a *common reference string* σ_{crs}.[8] Users can then generate their own signing keypairs. When a user wants to sign a message

[8]Using a common reference string is quite common with pairing-based schemes, and in fact many of them are secure

m for some ring R, he first encrypts his public verification key (using the BGN cryptosystem [24], which is IND-CPA secure assuming the SGH assumption) to get a ciphertext C. Next, he creates a zero-knowledge proof π that the value in that ciphertext is in fact exactly one of the public keys contained in R. This can be done by first creating "dummy" ciphertexts for each of the public keys in R that does not belong to him; these dummy ciphertexts C_i will just be encryptions of the identity element for G. For every C_i, he can then provide a zero-knowledge proof π_i (using NIZKs specifically for pairings adapted from those of Groth, Ostrovsky, and Sahai [62], which are secure again assuming SGH) that the value in C_i is either the identity or his public key. Because the BGN encryption scheme is *multiplicatively homomorphic*, meaning if C_1 is an encryption of m_1 and C_2 is an encryption of m_2 then $C_1 \cdot C_2$ is an encryption of $m_1 \cdot m_2$, the ciphertext $C = \prod C_i$ will then be a proper encryption of the user's public key if he behaved honestly (meaning he encrypted his own public key for the appropriate i and the identity for all other i), and an encryption of random garbage otherwise. Finally, the user will sign the message m using his secret signing key; here we will use the pairing-based Waters signature [90], which is secure assuming the CDH assumption. The ring signature will then consist of this Waters signature (S_1, S_2), as well as all the ciphertexts C_i and the NIZKs π_i. For readers interested in the details of the scheme, we have the following outline:

- $\mathsf{Setup}(1^k)$: First, generate a bilinear group G of order $N = pq$ and its generator g, along with the bilinear map $e : G \times G \to G_T$ and an element h such that h generates G_q. Next, pick random exponents $a, b_0 \leftarrow \mathbb{Z}/N\mathbb{Z}$ and compute $A = g^a$, $B_0 = g^{b_0}$, and $\hat{A} = h^a$. Next, pick random generators $u', u_1, u_2, \ldots, u_k$ of G, and define some collision-resistant hash function $H : \{0,1\}^* \to \{0,1\}^k$. The final CRS will then be $\sigma_{crs} = (\mathbb{Z}/N\mathbb{Z}, G, g, e, h, A, B_0, \hat{A}, u', u_1, \ldots, u_k, H)$.
- $\mathsf{KeyGen}(\sigma_{crs})$: Choose a random exponent $b \leftarrow \mathbb{Z}/N\mathbb{Z}$, and set $sk = b$ and $pk = A^b$.
- $\mathsf{Sign}(\sigma_{crs}, sk, R, m)$: Compute $(m_1, \ldots, m_k) = H(M, R)$. Let $n = |R|$, and denote the elements of R as $v_i \in G$ for $1 \leq i \leq n$; furthermore, let i^* be the index such that $v_{i^*} = pk$, where pk is the public key corresponding to the signing key sk being used. Compute an n-tuple of bits $(\beta_1, \ldots, \beta_n)$, where $\beta_i = 1$ if $i = i^*$, and $\beta_i = 0$ otherwise. Next, for each i, pick a random exponent $t_i \leftarrow \mathbb{Z}/N\mathbb{Z}$ and compute
$$C_i = (v_i/B_0)^{\beta_i} h^{t_i} \quad \text{and} \quad \pi_i = ((v_i/B_0)^{2\beta_i - 1} h^{t_i}).$$
Let $C = \prod_i C_i$ and $t = \sum_i t_i$. Finally, choose $r \leftarrow \mathbb{Z}/N\mathbb{Z}$ and compute
$$S_1 = sk \cdot (u' \prod_j u_j^{m_j})^r \cdot \hat{A}^t \quad \text{and} \quad S_2 = g^r.$$
Output the signature $\sigma = ((S_1, S_2), \{(C_i, \pi_i)\}_i)$.
- $\mathsf{Verify}(\sigma_{crs}, R, \sigma, m)$: Compute $(m_1, \ldots, m_k) = H(M, R)$, and again let $n = |R|$ and parse the elements of R as $v_i \in G$ for $1 \leq i \leq n$; in addition, parse the signature $\sigma = ((S_1, S_2), \{C_i, \pi_i)\}_i)$. First, check that the proofs are valid by checking that the equation
$$e(C_i, C_i/(v_i/B_0)) = e(h, \pi_i)$$
holds for all i. If any of the proofs is invalid, reject (i.e., output 0). Otherwise, set $C = \prod_i C_i$ and check if the following equation is satisfied:
$$e(A, B_0 C) = e(S_1, g) \cdot e(S_2^{-1}, u' \prod_j u_j^{m_j}).$$

only in what is called, appropriately enough, the *common reference string model*. We furthermore assume Setup is run by a trusted party, as it is essential for security that no one know the factorization of the group order $N = pq$.

If it is, accept. Otherwise, reject.

As with the generic group signature construction, the security of the ring signature scheme relies fundamentally on the security of its underlying primitives. As mentioned above, both the BGN encryption scheme and the GOS NIZKs are secure using SGH, and the Waters signature scheme is secure using CDH. Shacham and Waters can therefore prove the following theorem:

Theorem 7.1. **[87]** *If SGH is hard in the group G, CDH is hard in G_p, and H is collision resistant, then the above ring signature scheme is anonymous against full key exposure and unforgeable with respect to insider corruption (as outlined in Definitions 3.3 and 3.4).*

Again, we would like to intuitively argue why the ring signature is secure if its component primitives are. For anonymity, we can see that the user's public signing key might be leaked in either the ciphertext C_i containing it, or the NIZK π_i proving that it is the value in C_i; on the other hand, the user's secret signing key might be leaked by the signature (S_1, S_2). Here, we can use SGH as follows: in all of these values, change h from being a generator of G_q to a random element of the whole group G; by the assumption, we know that these changes will go undetected. Now, each of the values is simply a random element of G, meaning the relevant information (i.e., the public or secret signing key) is completely masked by h and no information whatsoever can be recovered about it (cryptographically, we can say that the ciphertext/NIZK/signature values will be information-theoretically independent from the key). As this holds regardless of whether or not someone is in possession of all the secret signing keys, we get anonymity against full key exposure.

To argue unforgeability, we now rely on the collision resistance of the hash function and the unforgeability of the Waters signature, as well as the soundness of the NIZK. First, we note that for any potential forgery (M^*, R^*), it cannot be the case that $H(M^*, R^*) = H(M, R)$ but $(M^*, R^*) \neq (M, R)$ (where (M, R) is some query the adversary made to its signing oracle), as this would violate the collision resistance of H. We can now consider two additional types of forgeries: forgeries such that the adversary encrypted either zero or more than one (i.e., not exactly one) of the public keys in R^*, or forgeries where it did in fact encrypt exactly one public key. The second type of forgery can easily be discounted, as it would imply a forgery for the Waters signature as well, which we assume to be unforgeable. For the first type of forgery, we can either argue that it breaks the soundness property of the NIZK, or "embed" in the CRS a CDH challenge such that this first type of forgery will in fact produce a solution for this challenge, thus breaking the assumption that CDH is hard.

8 Conclusions and Open Problems

In this paper, we have seen a broad overview of group and ring signature schemes, as well as some of their potential applications. Looking at the previous work on group signatures, some open problems are immediately raised. Although the work on group signatures is extensive and varied, there does not yet seem to be any clear winner or scheme that far outstrips the rest. The "holy grail" of sorts would be to combine the best qualities in each scheme and end up with an efficient scheme that supports dynamic groups with a group manager and revocation, secure under mild assumptions and without random oracles, and with short group signatures. Even less ambitious would be an efficient scheme that supports some subset of these properties (e.g., a fully dynamic scheme with a group manager) but meets the strongest definitions of security, as the only schemes we have seen that come close to achieving this level of flexibility provide only CPA-style anonymity.

In terms of the different variations we saw, techniques for revocation seem to be lagging far behind those for dynamic addition or for using group managers. This seems somewhat surprising, as the

problem of getting rid of cheating members in an efficient manner seems to be just as important as, if not more important than, adding users dynamically (and, as mentioned in Section 4, dynamic addition can essentially be "faked" by simply creating too many keys at the start). An in-depth look at revocation, perhaps even with some formal definitions that bind together the various approaches, seems to be an important first step; it would additionally be interesting to see if there were a generic construction (in the style of the group manager/dynamic addition generic construction of Bellare, Shi, and Zhang [14]) that achieved some notion of revocation beyond the naïve one of re-keying the whole group.

For ring signatures, we again see that there is no clear "best" scheme out there. The most obvious goal would therefore be to similarly try to create a "Franken-scheme" that again combines all the best qualities of existing schemes and gives us an efficient scheme with minimal setup assumptions (where ring members have published signing keys, but these keys do not even need to be for the same signature scheme), secure using the strongest definitions under mild assumptions and without random oracles, and with short signatures Unlike the ambitious analog for group signatures, such a scheme actually seems highly unlikely, as any non-generic (and thus efficient) scheme would be hard-pressed to deal with keys that have no specific form; still, even a scheme that required users to all use the same signature scheme but had all the same properties would be a major breakthrough. Perhaps even more important than finding the perfect scheme, finding a true real-world application for ring signatures, analogous to the DAA and VSC applications discussed for group signatures in Section 1, would be extremely valuable and would further motivate research in the field.

Acknowledgements

I gratefully acknowledge Hovav Shacham, my advisor, for suggesting an initial set of papers to look at and for indulging me in many future discussions. I also thank my committee members, Alex Snoeren and Stefan Savage, for taking the time to read this document and providing me with helpful feedback.

References

[1] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. *Journal of Cryptology*, 21(3), July 2008.

[2] M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n signatures from a variety of keys. In Y. Zheng, editor, *Proceedings of Asiacrypt 2002*, volume 2501 of *LNCS*, pages 415–32. Springer-Verlag, Dec. 2002.

[3] ANSI X9.62 and FIPS 186-2. Elliptic curve digital signature algorithm, 1998.

[4] G. Ateniese, J. Camenisch, S. Hohenberger, and B. de Medeiros. Practical group signatures without random oracles, 2005. `http://eprint.iacr.org/2005/385.pdf`.

[5] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In M. Bellare, editor, *Proceedings of Crypto 2000*, volume 1880 of *LNCS*, pages 255–70. Springer-Verlag, Aug. 2000.

[6] G. Ateniese and G. Tsudik. Some open issues and directions in group signatures. In M. Franklin, editor, *Proceedings of Financial Cryptography 1999*, volume 1648 of *LNCS*, pages 196–211. Springer-Verlag, Feb. 1999.

[7] G. Ateniese, G. Tsudik, and D. Song. Quasi-efficient revocation of group signatures. In M. Blaze, editor, *Proceedings of Financial Cryptography 2002*, volume 2357 of *LNCS*, pages 183–97. Springer-Verlag, 2003.

[8] N. Baric and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In W. Fumy, editor, *Proceedings of Eurocrypt 1997*, volume 1233 of *LNCS*, pages 480–494. Springer-Verlag, May 1997.

[9] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham. Delegatable anonymous credentials. In *Proceedings of Crypto 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 108–125. Springer-Verlag, 2009.

[10] M. Belenkiy, M. Chase, C. Erway, J. Jannotti, A. Küpçü, and A. Lysyanskaya. Incentivizing outsourced computation. In *Proceedings of NetEcon 2008*, pages 85–90, 2008.

[11] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In E. Biham, editor, *Proceedings of Eurocrypt 2003*, volume 2656 of *LNCS*, pages 614–29. Springer-Verlag, May 2003.

[12] M. Bellare and S. Miner. A forward-secure digital signature scheme. In *Proceedings of Crypto 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–448. Springer-Verlag, 1999.

[13] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security (CCS) 1993*, pages 62–73, 1993.

[14] M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In A. J. Menezes, editor, *Proceedings of CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–53. Springer-Verlag, Feb. 2005.

[15] A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In S. Halevi and T. Rabin, editors, *Proceedings of TCC 2006*, volume 3876 of *LNCS*, pages 60–79. Springer-Verlag, Mar. 2006.

[16] M. Blum, A. de Santis, S. Micali, and G. Persiano. Non-interactive zero-knowledge. *SIAM Journal of Computing*, 20(6):1084–1118, 1991.

[17] A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Y. Desmedt, editor, *Proceedings of PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46. Springer-Verlag, Jan. 2003.

[18] D. Boneh and X. Boyen. Short signatures without random oracles. In *Proceedings of Eurocrypt 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 54–73. Springer-Verlag, 2004.

[19] D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 56–73. Springer-Verlag, May 2004.

[20] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *Proceedings of Crypto 2004*, volume 3152 of *LNCS*, pages 41–55. Springer-Verlag, Aug. 2004.

[21] D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Computing*, 36(5):1301–28, Dec. 2006.

[22] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Computing*, 32(3):586–615, 2003. Extended abstract in *Proceedings of Crypto 2001*.

[23] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In E. Biham, editor, *Proceedings of Eurocrypt 2003*, volume 2656 of *LNCS*, pages 416–32. Springer-Verlag, May 2003.

[24] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In J. Kilian, editor, *Proceedings of TCC 2005*, number 3378 in LNCS, pages 325–41. Springer-Verlag, Feb. 2005.

[25] D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In B. Pfitzmann and P. Liu, editors, *Proceedings of CCS 2004*, pages 168–77. ACM Press, Oct. 2004.

[26] X. Boyen. Mesh signatures: how to leak a secret with unwitting and unwilling participants. In *Proceedings of Eurocrypt 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 210–227. Springer-Verlag, 2007.

[27] X. Boyen and B. Waters. Compact group signatures without random oracles. In S. Vaudenay, editor, *Proceedings of Eurocrypt 2006*, volume 4004 of *LNCS*, pages 427–44. Springer-Verlag, May 2006.

[28] X. Boyen and B. Waters. Full-domain subgroup hiding and constant-size group signatures. In *Proceedings of PKC 2007*, volume 4450 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 2007.

[29] E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation, Oct. 2004.

[30] J. Camenisch and I. Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In *Proceedings of Asiacrypt 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 331–345. Springer-Verlag, 2000.

[31] J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, and M. Meyerovich. How to win the clonewars: efficient periodic n-times anonymous authentication. In *ACM Conference on Computer and Communications Security (CCS) 2006*, pages 201–210, 2006.

[32] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Proceedings of Eurocrypt 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer-Verlag, 2001.

[33] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In M. Yung, editor, *Proceedings of Crypto 2002*, volume 2442 of *LNCS*, pages 61–76. Springer-Verlag, Aug. 2002.

[34] J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *Proceedings of SCN 2002*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer-Verlag, 2002.

[35] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In M. Franklin, editor, *Proceedings of Crypto 2004*, volume 3152 of *LNCS*, pages 56–72. Springer-Verlag, Aug. 2004.

[36] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). In B. Kaliski, Jr., editor, *Proceedings of Crypto 1997*, volume 1294 of *LNCS*, pages 410–24. Springer-Verlag, Aug. 1997.

[37] R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali. Resettable zero-knowledge. In *Proceedings of the 32st Symposium on the Theory of Computing (STOC)*, pages 235–244, 2001.

[38] N. Chandran, J. Groth, and A. Sahai. Ring signatures of sub-linear size without random oracles. In *Proceedings of Automata, Languages, and Programming 2007*, volume 4596 of *LNCS*, pages 423–434. Springer-Verlag, 2007.

[39] D. Chaum. Blind signatures for untraceable payments. In *Proceedings of Crypto 1982*, Lecture Notes in Computer Science, pages 199–203. Springer-Verlag, 1982.

[40] D. Chaum. Security without identification: transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.

[41] D. Chaum and H. van Antwerpen. Undeniable signatures. In *Proceedings of Crypto 1989*, volume 435 of *Lecture Notes in Computer Science*, pages 212–216. Springer-Verlag, 1989.

[42] D. Chaum and E. van Heyst. Group signatures. In D. W. Davies, editor, *Proceedings of Eurocrypt 1991*, volume 547 of *LNCS*, pages 257–65. Springer-Verlag, Apr. 1991.

[43] R. Cramer and V. Shoup. A practical public key encryption system provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *Proceedings of Crypto 1998*, volume 1642 of *LNCS*, pages 13–25. Springer-Verlag, Aug. 1998.

[44] I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *Proceedings of Eurocrypt 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 418–430, 2000.

[45] X. Ding, G. Tsudik, and S. Xu. Leak-free group signatures with immediate revocation. In T. Lai and K. Okada, editors, *Proceedings of ICDCS 2004*, Mar. 2004.

[46] Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous identification in ad hoc groups. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 609–26. Springer-Verlag, May 2004.

[47] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. *SIAM Journal of Computing*, 30(2):391–437, 200.

[48] C. Dwork and M. Naor. ZAPs and their applications. In *Proceedings of the 41st Symposium on Foundations of Computer Science (FOCS)*, pages 283–293, 2000.

[49] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of Crypto 1984*, pages 10–18, 1984.

[50] S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. *Journal of Cryptology*, 9(1):35–67, 1996.

[51] U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero-knowledge proofs based on a single random string. In *Proceedings of the 31st Symposium on Theory of Computing (STOC)*, pages 308–317, 1990.

[52] E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Proceedings of Crypto 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer-Verlag, 1997.

[53] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, Cambridge, 2001.

[54] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991.

[55] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[56] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. In *Proceedings of 17th Symposium on the Theory of Computing (STOC)*, pages 186–208, 1985.

[57] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, 17(2):281–308, 1988.

[58] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17(2):281–308, 1988.

[59] D. Gordon, J. Katz, and V. Vaikuntanathan. A group signature scheme from lattice assumptions. In *Proceedings of Asiacrypt 2010*, volume 6477 of *LNCS*, pages 395–412. Springer-Verlag, 2010.

[60] J. Groth. Non-interactive zero-knowledge arguments for voting. In *ACNS*, volume 3531 of *Lecture Notes in Computer Science*, pages 467–482. Springer-Verlag, 2005.

[61] J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *Proceedings of Asiacrypt 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 444–459. Springer-Verlag, 2006.

[62] J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero-knowledge for NP. In *Proceedings of Eurocrypt 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358. Springer-Verlag, 2006.

[63] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Proceedings of Eurocrypt 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer-Verlag, 2008.

[64] J. Herranz and G. Sáez. Forking lemmas for ring signature schemes. In T. Johansson and S. Maitra, editors, *Proceedings of Indocrypt 2003*, volume 2904 of *LNCS*, pages 266–79. Springer-Verlag, Dec. 2003.

[65] D. Hofheinz and E. Kiltz. Secure hybrid encryption from weakened key encapsulation. In A. Menezes, editor, *Proceedings of Crypto 2007*, volume 4622 of *LNCS*, pages 553–71. Springer-Verlag, Aug. 2007.

[66] IEEE P1556 Working Group, VSC Project. Dedicated short range communications (DSRC), 2003.

[67] J. Katz and Y. Lindell. *An Introduction to Modern Cryptography*. Chapman and Hall CRC, 2007.

[68] A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 571–89. Springer-Verlag, May 2004.

[69] A. Kiayias and M. Yung. Efficient secure group signatures with dynamic joins and keeping anonymity against group managers. In E. Dawson and S. Vaudenay, editors, *Proceedings of Mycrypt 2005*, volume 3715 of *LNCS*, pages 151–70. Springer-Verlag, Sept. 2005.

[70] A. Kiayias and M. Yung. Group signatures with efficient concurrent join. In R. Cramer, editor, *Proceedings of Eurocrypt 2005*, volume 3494 of *LNCS*, pages 198–214. Springer-Verlag, May 2005.

[71] M. Lepinski, S. Micali, and abhi shelat. Fair zero knowledge. In *Proceedings of 2nd Theory of Cryptography Conference (TCC)*, volume 3378 of *Lecture Notes in Computer Science*, pages 245–263. Springer-Verlag, 2005.

[72] H. Lipmaa, N. Asokan, and V. Niemi. Secure vickrey auctions without threshold trust. In *Proceedings of Financial Cryptography 2002*, volume 2357 of *Lecture Notes in Computer Science*, pages 87–101. Springer-Verlag, 2002.

[73] A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In M. Yung, editor, *Proceedings of Crypto 2002*, volume 2442 of *LNCS*, pages 597–612. Springer-Verlag, Aug. 2002.

[74] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 74–90. Springer-Verlag, May 2004.

[75] R. Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21(4):294–299, 1978.

[76] R. Merkle. A digital signature based on a conventional encryption function. In *Proceedings of Crypto 1988*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378. Springer-Verlag, 1987.

[77] I. Mironov. A short signature as secure as DSA. Unpublished manuscript, 2001.

[78] T. Nakanishi and N. Funabiki. Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps. In B. Roy, editor, *Proceedings of Asiacrypt 2005*, volume 3788 of *LNCS*, pages 533–48. Springer-Verlag, Dec. 2005.

[79] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of Eurocrypt 1999*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer-Verlag, 1999.

[80] M. Rabin. Digitalized signatures and public key functions as intractable as factorization. MIT Technical Report MIT-LCS-TR-212, 1979.

[81] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Commun. ACM*, 21(2):120–6, Feb. 1978.

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

[83] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Symposium on Theory of Computing (STOC)*, pages 387–394. ACM Press, 1990.

[84] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *Proceedings of the 40th Symposium on the Foundations of Computer Science (FOCS)*, pages 543–553, 1999.

[85] C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[86] E. Schwartz, D. Brumley, and J. McCune. Contractual anonymity. In *Proceedings of NDSS 2010*, 2010.

[87] H. Shacham and B. Waters. Efficient ring signatures without random oracles. In *Proceedings of PKC 2007*, volume 4450 of *LNCS*, pages 166–180. Springer-Verlag, 2007.

[88] A. Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of Crypto 1984*, volume 7 of *Lecture Notes in Computer Science*, pages 47–53. Springer-Verlag, 1984.

[89] Trusted Computing Group. Trusted Computing Platform Alliance (TCPA) Main Specification, 2003. Online: `www.trustedcomputinggroup.org`.

[90] B. Waters. Efficient identity-based encryption without random oracles. In *Proceedings of Eurocrypt 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer-Verlag, 2005.