Fully Anonymous Group Signatures without Random Oracles

Jens Groth*

University College London, United Kingdom
jgroth@adastral.ucl.ac.uk

Abstract. We construct a new group signature scheme using bilinear groups. The group signature scheme is practical, both keys and group signatures consist of a constant number of group elements, and the scheme permits dynamic enrollment of new members. The scheme satisfies strong security requirements, in particular providing protection against key exposures and not relying on random oracles in the security proof.

Keywords: Group signatures, certified signatures, bilinear groups.

1 Introduction

Group signatures make it possible for a member of a group to sign messages anonymously so that outsiders and other group members cannot see which member signed the message. The group is controlled by a group manager that handles enrollment of members and also has the ability to identify the signer of a message. Group signatures are useful in contexts where it is desirable to preserve the signer's privacy, yet in case of abuse we want some authorities to have the means of identifying her.

Group signatures were introduced by Chaum and van Heyst [CvH91] and have been the subject of much research. Most of the proposed group signatures have been proven secure in the random oracle model [BR93] and now quite efficient schemes exist in the random oracle model [ACJT00,BBS04,CL04,CG04,FI05,KY05]. The random oracle model has been the subject of criticism though. Canetti, Goldreich and Halevi [CGH98] demonstrated the existence of an insecure signature scheme that has a security proof in the random oracle model. Other works showing weaknesses of the random oracle model are [Nie02,GK03,BBP04,CGH04].

There are a few group signature schemes that avoid the random oracle model. Bellare, Micciancio and Warinschi [BMW03] suggested security definitions for group signatures and offered a construction based on trapdoor permutations. Their security model assumed the group was static and all members were given their honestly generated keys right away. Bellare, Shi and Zhang [BSZ05] strengthened the security model to include dynamic enrollment of members. This security model also separated the group manager's role into two parts: issuer and opener. The issuer is responsible for enrolling members, but cannot trace who has signed a group signature. The opener on the other

* Work done while at UCLA supported by NSF ITR/Cybertrust grant No. 0456717.

hand cannot enroll members, but can open a group signature to see who signed it. Moreover, it was required that this opener should be able to prove that said member made the group signature to avoid false accusations of members. [BSZ05] demonstrated that trapdoor permutations suffice also for constructing group signatures in this model. Both of these schemes use general and complicated primitives and are very inefficient. Groth [Gro06] used bilinear groups to construct a group signature scheme in the BSZ-model, with nice asymptotic performance, where each group signature consists of a constant number of group elements. Still the constant is enormous and a group signature consists of thousands or perhaps even millions of group elements.

There are also a few practical group signature schemes with security proofs in the standard model. Ateniese, Camenisch, Hohenberger and de Medeiros [ACHdM05] give a highly efficient group signature scheme, where each group signature consists of 8 group elements in prime order bilinear groups. This scheme is secure against a non-adaptive adversary that never gets to see private keys of honest members. If a member's key is exposed, however, it is easy to identify all group signatures she has made, so their scheme is not secure in the BMW/BSZ-models.

Boyen and Waters [BW06,BW07] suggest group signatures that are secure against key exposure attacks. Their constructions are secure in a restricted version of the BMW-model where the anonymity of the members relies on the adversary not being able to see any openings of group signatures. In the latter scheme [BW07], the group signatures consist of 6 group elements in a composite order bilinear group. The public key in [BW07] grows logarithmically in the size of the message space though and will for practical purposes typically contain a couple of hundred group elements.

OUR CONTRIBUTION. We propose a new group signature scheme based on prime order bilinear groups. All parts of the group signature scheme, including the group public key and the group signatures, consist of a constant number of group elements. The constants are reasonable for practical purposes; for instance using 256-bit prime order bilinear groups, a group public key would be less than 1kB and a group signature less than 2kB.

We prove under some well-known assumptions, the strong Diffie-Hellman assumption [BB04] and the decisional linear assumption [BBS04], as well as a new assumption that the scheme is secure in the BSZ-model. This means the scheme permits dynamic enrollment of members, preserves anonymity of a group signature even if the adversary can see arbitrary key exposures or arbitrary openings of other group signatures, and separates the role of the issuer and opener such that they can operate independently.

TECHNIQUE. We use in our group signature scheme a certified signature scheme. Certified signatures, the notion stemming from Boldyreva, Fischlin, Palacio and Warinschi, allow a user to pick keys for a signature scheme and use them to sign messages. The user can ask a certification authority to certify her public verification key for the signature scheme. The verification algorithm checks both the certificate and the signature and accepts if both of them are acceptable. A trivial way to build a certified signature schemes is just to let the certification authority output a standard signature on the user's public verification key. Non-trivial solutions such as for instance using an aggregate signature scheme [BGLS03] also exist. Certified signature schemes may be more efficient though since the certificate does not have to be unforgeable. In a certified signature scheme, the requirement is just that it is infeasible to forge a certificate together with a valid signature. We refer to Section 3 for a formal definition.

In our group signature scheme, enrolling members will create a key for a signature scheme and ask the issuer to issue a certificate on their verification key. To make a group signature, the member will make a certified signature. To be anonymous she will encrypt the certified signature and use non-interactive witness-indistinguishable and non-interactive zero-knowledge proofs to demonstrate that the ciphertext contains a valid certified signature.

In order to have efficient non-interactive proofs, it is essential to preserve as much of the bilinear group structure of the encrypted certified signature as possible. In particular, using cryptographic hash-functions or using group elements from one part of the certified signature as exponents in other parts of the certified signature does not work. We will combine the signature scheme of Boneh and Boyen [BB04] with the signature scheme of Zhou and Lin [ZL06] to get a certified signature scheme that is both efficient and relies only on generic group operations.

2 Setup

Let $\mathcal{G}$ be a probabilistic polynomial time algorithm that generates $(p, G, G_T, e, g) \leftarrow \mathcal{G}(1^k)$ such that:

- p is a k-bit prime.
- G, G_T are groups of order p.
- g is a randomly chosen generator of G.
- e is a non-degenerate bilinear map, i.e., $e(g, g)$ is a generator of G_T and for all $a, b \in \mathbb{Z}_p$ we have $e(g^a, g^b) = e(g, g)^{ab}$.
- Group operations, evaluation of the bilinear map, and membership of G, G_T are all efficiently computable.

We will now present some of the security assumptions that will be used in the paper.

DLIN ASSUMPTION. The decisional linear assumption was introduced by Boneh, Boyen and Shacham [BBS04]. The DLIN assumption holds for $\mathcal{G}$, when it is hard to distinguish for randomly chosen group elements and exponents (f, g, h, f^r, g^s, h^t) whether $t = r + s$ or t is random.

q-SDH ASSUMPTION. The strong Diffie-Hellman assumption was introduced by Boneh and Boyen [BB04]. The q-SDH assumption holds for $\mathcal{G}$, when it is hard to find a pair $(m, g^{\frac{1}{1+x}}) \in \mathbb{Z}_p \times G$ when given $g, g^x, g^{x^2}, \ldots, g^{x^{q(k)}}$ as input. In the paper, it suffices to have q being a polynomial.

q-U ASSUMPTION. We will now define the unfakeability assumption. The q-U assumption holds for $\mathcal{G}$ if for any non-uniform polynomial time adversary $\mathcal{A}$ we have:

$$\begin{aligned}
\Pr\Big[&(p, G, G_T, e, g) \leftarrow \mathcal{G}(1^k)\ ;\ x_1, r_1, \ldots, x_{q(k)}, r_{q(k)} \leftarrow \mathbb{Z}_p\ ;\\
&f, h, z \leftarrow G\ ;\ T := e(f, z)\ ;\ a_i := f^{r_i}\ ;\ b_i := h^{r_i} g^{x_i r_i} z\ ;\\
&(V, A, B, m, S) \leftarrow \mathcal{A}(p, G, G_T, e, g, f, h, T, x_1, a_1, b_1, \ldots, x_{q(k)}, a_{q(k)}, b_{q(k)}) :\\
&V \notin \{g^{x_1}, \ldots, g^{x_{q(k)}}\} \ \wedge\ e(A, hV)e(f, B) = T \ \wedge\ e(S, V g^m) = e(g, g)\Big] \approx 0.
\end{aligned}$$

The q-U assumption is implied by a stronger assumption from Zhou and Lin [ZL06] that is similar in nature. A heuristic argument for the assumption is that it holds in the generic group model; see the full paper for a proof.

3 Certified Signatures

Typically, using a signature in a public key infrastructure works like this: A user that wants to set up a signature scheme, generates a public verification key vk and a secret signing key sk. She takes the public key to a certification authority that signs vk and possibly some auxiliary information such as name, e-mail address, etc. We call this the certificate. Whenever the user wants to sign a message, she sends both the certificate and the signature to the verifier. The verifier checks that the certification authority has certified that the user has the public key vk and also checks the user's signature on the message.

In the standard way of certifying verification keys described above, the process of issuing certificates and verifying certificates is separate from the process of signing messages and verifying signatures. Boldyreva, Fischlin, Palacio and Warinschi [BFPW07] show that combining the two processes into one can improve efficiency. As they observe, we do not need to worry about forgeries of the certificate itself, we only need to prevent the *joint* forgery of both the certificate and the signature.

A certified signature scheme [BFPW07], is a combined scheme for signing messages and producing certificates for the verification keys. We will give a formal definition that is tailored to our purposes and slightly simpler than the more general definition given by Boldyreva, Fischlin, Palacio and Warinschi. Formally, a certified signature scheme consists of the following probabilistic polynomial time algorithms.

Setup: $\mathcal{G}$ takes a security parameter as input and outputs a description gk of our setup.

Certification key: $\mathrm{CertKey}$ on input gk outputs a pair (ak, ck), respectively a public authority key and a secret certification key.

Key registration: This is an interactive protocol $\langle \mathrm{User}, \mathrm{Issuer} \rangle$ that generates keys for the user together with a certificate. User takes gk, ak as input, whereas Issuer takes gk, ck as input. If successful User outputs a triple $(vk, sk, cert)$, whereas Issuer outputs $(vk, cert)$. We write $((vk, sk, cert), (vk, cert)) \leftarrow \langle \mathrm{User}(gk, ak), \mathrm{Issuer}(gk, ck) \rangle$ for this process. We call vk the verification key, sk the signing key and $cert$ the certificate. Either party outputs $\perp$ if the other party deviates from the key registration protocol.

Signature: Sign gets a signing key and a message m as input. It outputs a signature σ.

Verification: Ver takes as input $gk, ak, vk, cert, m, \sigma$ and outputs 1 if accepting the certificate and the signature on m. Otherwise it outputs 0.

The certified signature scheme must be correct, unfakeable and unforgeable as defined below.

Perfect correctness: For all messages m we have

$$\Pr \Big[gk \leftarrow \mathcal{G}(1^k) \; ; \; (ak, ck) \leftarrow \mathrm{CertKey}(gk) \; ;$$

$$((vk, sk, cert), (vk, cert)) \leftarrow \langle \mathrm{User}(gk, ak), \mathrm{Issuer}(gk, ck) \rangle \; ;$$
$$\sigma \leftarrow \mathrm{Sign}_{sk}(m) : \mathrm{Ver}(gk, ak, vk, cert, m, \sigma) = 1 \Big] = 1.$$

Unfakeability: We want it to be hard to create a signature with a faked certificate. Only if the verification key has been generated correctly and been certified by the certification authority should it be possible to make a certified signature on a message. For all non-uniform polynomial time adversaries $\mathcal{A}$ we require:

$$\Pr\Big[gk \leftarrow \mathcal{G}(1^k); (ak, ck) \leftarrow \mathrm{CertKey}(gk); (vk, cert, m, \sigma) \leftarrow \mathcal{A}^{\mathrm{KeyReg}}(gk, ak) :$$
$$vk \notin Q \text{ and } \mathrm{Ver}(gk, ak, vk, cert, m, \sigma) = 1\Big] \approx 0,$$

where KeyReg is an oracle that allows $\mathcal{A}$ to sequentially start up new key registration sessions and lets $\mathcal{A}$ act as the user. That is in session i we run $(*, (vk_i, cert_i)) \leftarrow \langle \mathcal{A}, \mathrm{Issuer}(gk, ck) \rangle \; ; \; Q := Q \cup \{vk_i\}$ forwarding all messages to and from $\mathcal{A}$ through the oracle.

Existential M-unforgeability: Let M be a stateful non-uniform polynomial time algorithm. We say the certified signature scheme is existentially M-unforgeable if for all non-uniform polynomial time adversaries $\mathcal{A}$ we have:

$$\Pr\Big[gk \leftarrow \mathcal{G}(1^k) \; ; \; (\mathrm{St}_1, ak) \leftarrow \mathcal{A}(gk) \; ;$$
$$((vk, sk, cert), \mathrm{St}_2) \leftarrow \langle \mathrm{User}(gk, ak), \mathcal{A}(\mathrm{St}_1) \rangle \; ;$$
$$(cert', m, \sigma) \leftarrow \mathcal{A}^{\mathrm{MessageSign}(\cdot)}(\mathrm{St}_2) :$$
$$m \notin Q \text{ and } \mathrm{Ver}(gk, ak, vk, cert', m, \sigma) = 1\Big] \approx 0,$$

where $\mathrm{MessageSign}(\cdot)$ is an oracle that on input a_i runs $(m_i, h_i) \leftarrow M(gk, a_i) \; ; \; \sigma_i \leftarrow \mathrm{Sign}_{sk}(m_i) \; ; \; Q := Q \cup \{m_i\}$ and returns (m_i, h_i, σ_i).
Adaptive chosen message attack corresponds to letting M be an algorithm that on input m_i outputs (m_i, ε). On the other hand, letting M be an algorithm that ignores $\mathcal{A}$'s inputs corresponds to a weak chosen message attack, where messages to be signed by the oracle are chosen without knowledge of vk. In a weak chosen message attack, the h_i's may contain a history of how the messages were selected. In this paper, we only need security against weak chosen message attack.

4 A Certified Signature Scheme

We will construct a certified signature scheme from bilinear groups that is existentially unforgeable under weak chosen message attack. There are two parts of the scheme: certification and signing. For signing, we will use the Boneh-Boyen signature scheme that is secure under weak chosen message attack. In their scheme the public key is $v := g^x$ and the secret signing key is x. A signature on message $m \in \mathbb{Z}_p \setminus \{x\}$ is $\sigma = g^{\frac{1}{x+m}}$. It can be verified by checking $e(\sigma, vg^m) = e(g, g)$. Boneh and Boyen [BB04] proved that this signature scheme is secure against weak chosen message attack under the q-SDH assumption. The existential unforgeability of our certified signature scheme under weak chosen message attack will follow directly from the security of the Boneh-Boyen signature scheme under weak chosen message attack.

What remains is to specify how to generate the verification key v and how to certify it. This is a 2-step process, where we first generate a random $v = g^x$ such that the issuer learns v but only the user learns x. In Section 4.1 we describe in detail the properties we need this key generation protocol to have. In the second step, we use a variation of the signature scheme of Zhou and Lin [ZL06] to certify v.[1]

To set up the certified signature scheme, the certification authority picks random group elements $f, h, z \in G$. The authority key is (f, h, T) and the secret certification key is z so $T = e(g, z)$. To certify a Boneh-Boyen key v the authority picks $r \leftarrow \mathbb{Z}_p$ and sets $(a, b) := (f^{-r}, (hv)^r z)$. The certificate is verified by checking $e(a, hv)e(f, b) = T$. We remark that this is not a good signature scheme, since given v, a, b it is easy to create a certificate for $v' := v^2 h$ as $(a', b') := (a^{\frac{1}{2}}, b)$. For certified signatures it works fine though since we cannot use the faked verification keys to actually sign any messages. The nice part about the certified signature scheme we have suggested here is that a certificate consists of only two group elements and is created through the use of generic group operations. These two properties of the certified signature scheme are what enable us to construct a practical group signature scheme on top of it.

$\mathbf{Setup}(1^k)$
Return $gk := (p, G, G_T, e, g) \leftarrow \mathcal{G}(1^k)$

$\mathbf{CertKey}(gk)$
$f, h, z \leftarrow G$
$T := e(f, z)$
Return $(ak, ck) := ((gk, f, h, T), (ak, z))$

$\mathbf{Sign}_{sk}(m)$
If $x = -m$ return $\perp$
Else return $\sigma := g^{\frac{1}{x+m}}$

$\langle \mathbf{User}(gk, ak), \mathbf{Issuer}(gk, ck) \rangle$
$(x, v) \leftarrow \langle \mathrm{User}(gk), \mathrm{Issuer}(gk) \rangle$
$r \leftarrow \mathbb{Z}_p$
$a := f^{-r}$
$b := (vh)^r z$
$vk := v$; $sk := x$; $cert := (a, b)$
User output: $(vk, sk, cert)$
Issuer output: $(vk, cert)$

$\mathbf{Ver}(gk, ak, vk, cert, m, \sigma)$
Return 1 if
$e(a, vh)e(f, b) = T$
$e(\sigma, vg^m) = e(g, g)$
Else return 0

Fig. 1. The certified signature scheme.

Theorem 1. *The scheme in Figure 1 is a certified signature scheme with perfect correctness for messages in $\mathbb{Z}_p \setminus \{x\}$. It is unfakeable under the q-U assumption and is*

[1] The signature scheme of Zhou and Lin [ZL06] can be used to sign exponents. As they observe, however, it is sufficient to know $v = g^x$ to sign x. In our notation, their scheme computes a signature on x by setting $v = g^x$ and computing the signature (a, b) as $a := f^r, b := (hv)^r z$, where $z = h^{\log_f g}$ so $T = e(g, h)$.

existentially unforgeable under weak chosen message attack under the q-SDH assumption.

Sketch of proof. Perfect correctness follows from the perfect correctness of the key generation protocol.

We will now argue that the certified signature scheme is unfakeable. Part of the key registration protocol is the interactive key generation protocol. We can black-box simulate the view of the adversarial user in each of these key generation protocols. We can therefore pick $x_1, \ldots, x_{q(k)}$ in advance and simulate the key generation such that the adversarial user i get the signing key x_i (or gets no key at all in case it deviates from the protocol). With this modified key registration, $\mathcal{A}$ only sees certificates on $v_1 := g^{x_1}, \ldots, v_{q(k)} := g^{x_{q(k)}}$. These certificates are of the form $a_i := f^{-r_i}$ and $b_i := h^{r_i} g^{x_i r_i} z$. It therefore follows directly from the q-U assumption that it is hard to come up with a certified signature using a new public verification key.

We will now ague that the certified signature scheme is existentially unforgeable under weak chosen message attack. By definition th key generation protocol has the property that it is possible to choose $v := g^x$ in advance and black-box simulate the malicious issuer's view in a protocol that gives it v as output. Now we are in a situation, where v is an honestly chosen Boneh-Boyen verification key and $\mathcal{A}$ only has access to a weak chosen message attack. Existential unforgeability of the certified signature scheme therefore follows from the existential unforgeability of Boneh-Boyen signatures under weak chosen message attack. □

4.1 Key Generation

In the certified signature scheme, we require that the user generates her signing key honestly. We will use an interactive protocol between the user and the issuer that gives the user a uniformly random secret key $x \in \mathbb{Z}_p$, while the issuer learns $v := g^x$. In case either party does not follow the protocol or halts prematurely, the other party will output $\perp$. We will now give a more precise definition of the properties the protocol should have. For notational convenience, define $g^{\perp} = \perp$.

Write $(x, v) \leftarrow \langle \mathrm{User}(gk), \mathrm{Issuer}(gk) \rangle$ for running the key generation protocol between two probabilistic polynomial time interactive Turing machines $\mathrm{User}, \mathrm{Issuer}$ on common input gk giving User output x and Issuer output v. We require that the protocol is correct in the following sense:

$$\Pr\left[gk \leftarrow \mathcal{G}(1^k)\ ;\ (x, v) \leftarrow \langle \mathrm{User}(gk), \mathrm{Issuer}(gk) \rangle : v = g^x\right] = 1.$$

We require that the view of the issuer, even if malicious, can be simulated. More precisely, for any $\delta > 0$ and polynomial time Issuer^* there exists a polynomial time (in k and the size of the input to Issuer^*) black-box simulator S_I, such that for all non-uniform polynomial time adversaries $\mathcal{A}$ we have:

$$\Pr\Big[gk \leftarrow \mathcal{G}(1^k)\ ;\ y \leftarrow \mathcal{A}(gk)\ ;\ x \leftarrow \mathbb{Z}_p\ ;\ v := g^x\ ;\ (g^u, i) \leftarrow S_I^{\mathrm{Issuer}^*(y)}(gk, v) : \mathcal{A}(u, i) = 1\Big]$$

$$- \Pr\Big[gk \leftarrow \mathcal{G}(1^k)\ ;\ y \leftarrow \mathcal{A}(gk)\ ;\ (x,i) \leftarrow \langle \mathrm{User}(gk), \mathrm{Issuer}^*(y)\rangle :$$
$$\mathcal{A}(u,i) = 1\Big] < k^{-\delta},$$

where S_I outputs g^u so $u \in \{\perp, x\}$.

We also require that the view of the user, even if malicious, can be simulated. For any $\delta > 0$ and any polynomial time User^* there exists a polynomial time (in k and the size of the input to User^*) black-box simulator S_U, such that for all non-uniform polynomial time adversaries $\mathcal{A}$ we have:

$$\Pr\Big[gk \leftarrow \mathcal{G}(1^k)\ ;\ y \leftarrow \mathcal{A}(gk)\ ;\ x \leftarrow \mathbb{Z}_p\ ;\ v := g^x\ ;\ (u,i) \leftarrow S_U^{\mathrm{User}^*(y)}(gk, x) :$$
$$\mathcal{A}(u,i) = 1\Big]$$
$$- \Pr\Big[gk \leftarrow \mathcal{G}(1^k)\ ;\ y \leftarrow \mathcal{A}(gk)\ ;\ (u,i) \leftarrow \langle \mathrm{User}^*(y), \mathrm{Issuer}(gk)\rangle :$$
$$\mathcal{A}(u,i) = 1\Big] < k^{-\delta},$$

where S_U outputs $i \in \{\perp, v\}$.

There are many ways in which one can construct a key generation protocol with these properties. One example of a simple 5-move key generation protocol is given in the full paper.

5 Defining Group Signatures

In a group signature scheme there is a group manager that decides who can join the group. Once in the group, members can sign messages on behalf of the group. Members' signatures are anonymous, except to the group manager who can open a signature and see who signed the message. In some scenarios it is of interest to separate the group manager into two entities, an issuer who enrolls members and an opener who traces signers.

We imagine that enrolled member's when joining have some identifying information added to a registry reg. This registry may or may not be publicly accessible. The specifics of how the registry works are not important, we just require that $reg[i]$ only contains content both the issuer and user i agrees on. One option could be that the issuer maintains the registry, but the user has to sign the content of $reg[i]$ for it to be considered a valid entry. User i stores her corresponding secret key in $gsk[i]$. The number i we associate with the user is simply a way to distinguish the users. Without loss of generality, we will assume users are numbered $1, \ldots, n$ according to the time they joined or attempted to join.

Key generation: GKg generates (gpk, ik, ok). Here gpk is a group public key, while ik and ok are respectively the issuer's and the opener's secret key.

Join/Issue: This is an interactive protocol between a user and the issuer. If successful, the user and issuer register a public key vk_i in $reg[i]$ and the user stores some corresponding secret signing key information in $gsk[i]$.

[BSZ05] specify that communication between the user and the issuer in this protocol should be secret. The Join/Issue protocol in our scheme works when all messages are sent in clear though. In our scheme, we will assume the issuer joins users in a sequential manner, but depending on the setup assumptions one is willing to make, it is easy to substitute the $\mathrm{Join}/\mathrm{Issue}$ protocol for a concurrent protocol.

Sign: Group member i can sign a message m as $\Sigma \leftarrow \mathrm{Gsig}(gpk, gsk[i], m)$.

Verify: To verify a signature Σ on message m we run $\mathrm{GVf}(gpk, m, \Sigma)$. The signature is valid if and only if the verification algorithm outputs 1.

Open: The opener has read-access to the registration table reg. We have $(i, \tau) \leftarrow \mathrm{Open}(gpk, ok, reg, m, \Sigma)$ gives an opening of a valid signature Σ on message m pointing to user i. In case the signature points to no member, the opener will assume the issuer forged the signature and set $i := 0$. The role of τ is to accompany $i \neq 0$ with a proof that user i did indeed sign the message.

Judge: This algorithm is used to verify that openings are correct. We say the opening is correct if $\mathrm{Judge}(gpk, i, reg[i], m, \Sigma, \tau) = 1$.

[BSZ05] define four properties that the group signature must satisfy: correctness, anonymity, traceability and non-frameability. We will here give a quick informal description of the properties. We refer to [BSZ05] for details and a discussion of how these security definitions cover and strengthen other security definitions that have appeared in the literature.

Non-frameability: Non-frameability protects the user against being falsely accused of making a group signature, even if both the issuer and the opener are corrupt.

Traceability: When the issuer is honest and the opening algorithm is applied correctly, albeit the opener's key may be exposed, traceability guarantees that a group signature always can be traced back to a member who made it.

Anonymity: An opener knows who made a particular group signature, but provided the opener is honest and the opener's key is kept secret, nobody else should be able to identify the member. Anonymity gives this guarantee even in an environment where all users' keys are exposed and the issuer is corrupt. In the definition, the adversary is also permitted to ask the opener to open group signatures, except the group signature where it is trying to guess who signed it.

A weaker variant of anonymity called CPA-anonymity does not permit the adversary to see openings of other group signatures. The difference between full anonymity and CPA-anonymity is analogous to the difference between security under chosen ciphertext attack and chosen plaintext attack for public-key encryption.

6 Tools

To construct our group signature scheme, we will use the certified signature scheme from Section 4. We will also use several other tools in our construction, namely collision-free hash functions, non-interactive proofs for bilinear groups, strong one-time signatures secure against weak chosen message attack and selective-tag weak CCA-secure cryptosystems.

6.1 Collision-Free Hash-Functions

$\mathcal{H}$ is a generator of collision free hash-functions $\mathrm{Hash} : \{0,1\}^* \to \{0,1\}^{\ell(k)}$ if for all non-uniform polynomial time adversaries $\mathcal{A}$ we have:

$$\Pr\Big[\mathrm{Hash} \leftarrow \mathcal{H}(1^k)\ ;\ x, y \leftarrow \mathcal{A}(\mathrm{Hash}) : \mathrm{Hash}(x) = \mathrm{Hash}(y)\Big] \approx 0.$$

We will use a collision-free hash-function to compress messages before signing them. For this purpose we will require that we can hash down to $\mathbb{Z}_p$, so we want to have $2^{\ell(k)} < p$. We remark that collision-free hash-functions can be constructed assuming the discrete logarithm problem is hard, so the existence of collision-free hash-functions follows from our assumptions on the bilinear group.

6.2 Strong One-Time Signatures

We will use a one-time signature scheme that is secure against an adversary that has access to a single weak chosen message attack. We say the one-time signature scheme is strong, if the adversary can neither forge a signature on a different message nor create a different signature on the chosen message she already got signed. An obvious candidate for such a scheme is the Boneh-Boyen signature scheme [BB04], since this signature scheme is deterministic and hence automatically has the strongness property.

6.3 Non-interactive Proofs for Bilinear Groups

Groth and Sahai [GS07] suggest non-interactive proofs that capture relations for bilinear groups. They look at sets of equations in our bilinear group (p, G, G_T, e, g) over variables in G and $\mathbb{Z}_p$ such as pairing product equations, e.g. $e(x_1, x_2)e(x_3, x_4) = 1$, or multi-exponentiation equations, e.g. $x_1^{\delta_1} x_2^{\delta_2} = 1$. They suggest non-interactive proofs for demonstrating that a set of equations of the form described above has a solution $x_1, \ldots, x_I \in G, \delta_1, \ldots, \delta_J \in \mathbb{Z}_p$ so all equations are simultaneously satisfied. Their proofs are in the common reference string model. There are two types of common reference strings that yield respectively perfect soundness and perfect witness indistinguishability/perfect zero-knowledge. The two types of common reference strings are computationally indistinguishable and they both give perfect completeness. We now give some further details.

[GS07] show that there exists four probabilistic polynomial time algorithms (K, P, V, X), which we call respectively the key generator, the prover, the verifier and the extractor. The key generator takes (p, G, G_T, e, g) as input and outputs a common reference string $crs = (F, H, U, V, W, U', V', W') \in G^8$ as well as an extraction key xk. Given a set of equations, the prover takes crs and a witness $x_1, \ldots, x_I, \delta_1, \ldots, \delta_J$ as input and outputs a proof π. The verifier given crs, a set of equations and π outputs 1 if the proof is valid and else it outputs 0. Finally, the extractor on a valid proof π will extract $x_1, \ldots, x_I \in G$, in other words it will extract part of the witness.

The proofs of [GS07] have perfect completeness: on a correctly generated CRS and a correct witness, the prover always outputs a valid proof. They have perfect soundness: on a correctly generated CRS it is impossible to create a valid proof unless the equations are simultaneously satisfiable. Further, they have perfect partial knowledge: given xk the algorithm X can extract $x_1, \ldots, x_I$ from the proof, such that there exists a solution for the equations that use these $x_1, \ldots, x_I$.

There exists a simulator S_1 that outputs a simulated common reference string crs and a simulation trapdoor key tk. These simulated common reference strings are computationally indistinguishable from the common reference strings produced by K assuming the DLIN problem is hard. On a simulated common reference string, the proofs created by the prover are perfectly witness-indistinguishable: if there are many possible witnesses for the equations being satisfiable, the proof π does not reveal anything about which witness was used by the prover when creating the proof. Further, let us call a set of equations tractable, if it is possible to find a solution, where $x_1, \ldots, x_I$ are the same in all equations, but $\delta_1, \ldots, \delta_J$ are allowed to vary from equation to equation. Tractable equations have perfect zero-knowledge proofs on simulated reference strings: there exists a simulator S_2 that on a simulated reference string crs and a simulation trapdoor key tk produces a simulated proof π for the tractable equations being satisfiable. If the equations are satisfiable, then simulated proofs are perfectly indistinguishable from the proofs a real prover with a witness would form on a simulated reference string.

It will be useful later in the paper to know some technical details of the construction. The values F, H, U, V, W will be used to commit to the variables x as $(c_1, c_2, c_3) := (F^r U^t, H^s V^t, g^{r+s} W^t x)$ for randomly chosen $r, s, t \in \mathbb{Z}_p$. On a real common reference string, they are set up so $U = F^R, V = H^S, W = g^{R+S}$ so the commitment can be rewritten as $(F^{r+Rt}, H^{s+St}, g^{r+s+(R+S)t} x)$. The extraction key is $xk := (\phi, \eta)$ so $F = g^\phi, H = g^\eta$. This permits decryption of the commitment as $x = c_3 c_1^{-\phi} c_2^{-\eta}$. On the other hand, on a simulation reference string, we use $U = F^R, V = H^S, W = g^T$ with $T \neq R + S$, which makes the commitment perfectly hiding.

To commit to a variable $\delta \in \mathbb{Z}_p$ using randomness r, s we use the commitment $(d_1, d_2, d_3) := (F^r (U')^\delta, H^s (V')^\delta, g^{r+s} (W')^\delta)$. On a normal common reference string, we pick $U' = F^R, V' = H^S, W' = g^T$ for $T \neq R + S$. This makes the commitment perfectly binding. On a simulated common reference string, on the other hand, we pick $U' = F^R, V' = H^S, W' = g^{R+S}$. The simulation trapdoor key is $tk := (R, S)$, which permits us to trapdoor open a commitment to 0 to any value δ since $(F^r, H^s, g^{r+s}) = (F^{r-R\delta} (U')^\delta, H^{s-S\delta} (V')^\delta, g^{r+s-(R+S)\delta} (W')^\delta)$.

6.4 Selective-tag Weakly CCA-secure Encryption

We will use a tag-based cryptosystem [MRY04] due to Kiltz [Kil06]. The public key consists of random non-trivial elements $pk = (F, H, K, L) \in G^4$ and the secret key is $sk = (\phi, \eta)$ so $F = g^\phi, H = g^\eta$. We encrypt $m \in \mathbb{G}$ using tag $t \in \mathbb{Z}_p$ and randomness $r, s \in \mathbb{Z}_p$ as $(y_1, \ldots, y_5) := (F^r, H^s, g^{r+s} m, (g^t K)^r, (g^t L)^s)$. The validity of the ciphertext is publicly verifiable, since valid ciphertexts have $e(F, y_4) = e(y_1, g^t K)$ and $e(H, y_5) = e(y_2, g^t L)$. Decryption can be done by computing $m = y_3 y_1^{-\phi} y_2^{-\eta}$. In the group signature scheme, we will set up the cryptosystem with the same F, H as in the common reference string of the non-interactive proofs.

[Kil06] shows that under the DLIN assumption this cryptosystem is selective-tag weakly CCA-secure. By this we mean that it is indistinguishable which message we encrypted under a tag t, even when we have access to a decryption oracle that decrypts ciphertexts under any other tag. Formally, for all non-uniform polynomial time adversaries $\mathcal{A}$ we have:

$$\Pr\Big[gk \leftarrow \mathcal{G}(1^k)\ ;\ t \leftarrow \mathcal{A}(gk)\ ;\ (pk, sk) \leftarrow K(gk)\ ;\ (m_0, m_1) \leftarrow \mathcal{A}^{D_{sk}(\cdot,\cdot)}(pk)\ ;$$
$$y \leftarrow E_{pk}(t, m_0)\ :\ \mathcal{A}^{D_{sk}(\cdot,\cdot)}(y) = 1\Big]$$
$$\approx \Pr\Big[gk \leftarrow \mathcal{G}(1^k)\ ;\ t \leftarrow \mathcal{A}(gk)\ ;\ (pk, sk) \leftarrow K(gk)\ ;\ (m_0, m_1) \leftarrow \mathcal{A}^{D_{sk}(\cdot,\cdot)}(pk)\ ;$$
$$y \leftarrow E_{pk}(t, m_1)\ :\ \mathcal{A}^{D_{sk}(\cdot,\cdot)}(y) = 1\Big],$$

where the oracle returns $D_{sk}(t_i, y_i)$ if $t_i \neq t$.

7 The Group Signature Scheme

The core of our group signature scheme is the certified signature scheme from Section 4. The issuer acts as a certification authority and whenever a new member i wants to enroll, she needs to create a verification key v_i for the Boneh-Boyen signature scheme and get a certificate from the issuer. In the group signature scheme, the verification key and the corresponding secret key is generated with an interactive key generation protocol as defined in Section 4.1. This way both user and issuer know that v_i is selected with the correct distribution and that the user holds the corresponding secret key x_i.

When making a group signature, the member will generate a key pair $(vk_{\text{sots}}, sk_{\text{sots}})$ for a strong one-time signature that is secure under weak chosen message attack. She will sign the message using sk_{sots} and use x_i to sign vk_{sots}. The combination of certified signatures and strong one-time signatures is what makes it hard to forge group signatures.

Group signatures have to be anonymous and therefore we cannot reveal the certified signature. Instead, a group signature will include a non-interactive witness-indistinguishable (NIWI) proof of knowledge of a certified signature on vk_{sots}. Witness-indistinguishability implies that a group signature does not reveal which group member has signed the message. The opener will hold the extraction key for the NIWI proof of knowledge and will be able to extract the certified signature. Whenever an opening is called for, she extracts the signature on vk_{sots}, which points to the member who signed the message. In case no member has certified signed vk_{sots}, the opener points to the issuer since the certified signature has a valid certificate.

The ideas above suffice to construct a CPA-anonymous group signature scheme. To get anonymity even when the adversary has access to the Open oracle, we will encrypt the signature on vk_{sots} with Kiltz' cryptosystem using vk_{sots} as a tag. We will also give an NIZK proof that the encrypted signature is the same as the one used in the NIWI proof of knowledge.

We present the full group signature scheme in Figure 2. Let us explain the non-interactive proofs further. The NIWI proof of knowledge, will demonstrate that there exists a certified signature (a, b, v, σ) on vk_{sots} so

$$e(a, hv)e(f, b) = T\ \wedge\ e(\sigma, vg^{\mathrm{Hash}(vk_{\text{sots}})}) = e(g, g).$$

In the terminology of [GS07], these are two pairing product equations over three variables b, v, σ. The last element a will be public, since we can rerandomize the certificate such that a does not identify the member. [GS07] gives us an NIWI proof of knowledge for these two equations being simultaneously satisfiable that consists of 27 group elements. This proof consists of three commitments to respectively b, v, σ, which consist of 3 group elements each, and two proofs for the committed values satisfying the two equations consisting of 9 group elements each.

In the NIZK proof we have a ciphertext y under tag $\mathrm{Hash}(vk_{\mathrm{sots}})$ and a commitment c to σ from the NIWI proof of knowledge. We wish to prove that the plaintext of y and the committed value in c are the same. The ciphertext is of the form $(y_1, \ldots, y_5) = (F^{r_y}, H^{s_y}, g^{r_y+s_y}\sigma, (g^{\mathrm{Hash}(vk_{\mathrm{sots}})}K)^{r_y}, (g^{\mathrm{Hash}(vk_{\mathrm{sots}})}L)^{s_y})$ and the commitment is of the form $(c_1, c_2, c_3) = (F^{r_c}U^t, H^{s_c}V^t, g^{r_c+s_c}W^t\sigma)$. Setting $r := r_c - r_y, s := s_c - s_y$ we have $(c_1 y_1^{-1}, c_2 y_2^{-1}, c_3 y_3^{-1}) = (F^r U^t, H^s V^t, g^{r+s}W^t)$. On the other hand, if the plaintext and the committed value are different, then no such r, s, t exist. Proving that the plaintext and the committed value are the same, therefore corresponds to proving the simultaneous satisfiability of the following equations over $\phi, r, s, t \in \mathbb{Z}_p$:

$$\phi = 1 \ \wedge \ (c_1^{-1}y_1)^{\phi}F^r U^t = 1 \ \wedge \ (c_2^{-1}y_2)^{\phi}H^s V^t = 1 \ \wedge \ (c_3^{-1}y_3)^{\phi}g^{r+s}W^t.$$

This set is tractable, i.e., if we allow ϕ to take different values in the equations, then there is a trivial solution $\phi = 1$ in the first equation and $\phi = r = s = t = 0$ in the other three equations. Since the set of equations is tractable, there is an NIZK proof for the 4 equations being simultaneously satisfiable. The proof consists of commitments to ϕ, r, s, t, but since the first equation is straightforward we can simply use (U', V', W') as the commitment to ϕ, which makes it easy to verify that the first equation holds. The three commitments to r, s, t each consist of 3 group elements. The three last equations are multi-exponentiations of constants and using the proof of [GS07] each equation costs 2 group elements to prove. The NIZK proof therefore costs a total of 15 group elements.

Theorem 2. *The scheme in Figure 2 is a group signature scheme with perfect correctness. Under the DLIN, q-SDH and q-U assumption and assuming the strong one-time signature scheme is secure against weak chosen message attack and the hash-function is collision resistant, the group signature has anonymity, traceability and non-frameability.*

Sketch of proof. Perfect correctness follows by inspection and the fact that the constituent protocols have perfect correctness and perfect completeness. We will sketch a proof that the group signature is secure, we refer to the full paper for more details.

To argue anonymity we consider a situation where the issuer may be corrupt and the members' keys are exposed. Since the adversary controls the issuer, she can let both corrupt users and honest users join the group. She can also ask the opener to open arbitrary valid group signatures. At some point she will choose two honest members and a message and get a group signature from one of the members. We want to show that she cannot tell which of the honest members made the group signature, as long as she does not ask the opener to open the challenge group signature.

$\mathbf{GKg}(1^k)$
$gk \leftarrow \mathcal{G}(1^k)$; $\mathrm{Hash} \leftarrow \mathcal{H}(1^k)$
$((f,h,T),z) \leftarrow \mathrm{CertKey}(gk)$
$(crs, xk) \leftarrow K_{\mathrm{NI}}(gk)$; $K, L \leftarrow G$
$(F, H, \text{the rest}) \leftarrow \mathrm{Parse}(crs)$; $pk := (F,H,K,L)$
$(gpk, ik, ok) := ((gk, \mathrm{Hash}, f, h, T, crs, pk), z, xk)$

$\mathbf{Join/Issue}(\text{User } i : gpk\ ,\ \text{Issuer} : gpk, ik)$
$((v_i, x_i, a_i, b_i), (v_i, a_i, b_i)) \leftarrow \langle \text{User}, \text{Issuer} \rangle$
User: If $e(a_i, hv_i)e(f, b_i) = T$ set
$reg[i] := v_i$; $gsk[i] := (x_i, a_i, b_i)$

$\mathbf{GSig}(gpk, gsk[i], m)$
$(vk_{\mathrm{sots}}, sk_{\mathrm{sots}}) \leftarrow \mathrm{KeyGen}_{\mathrm{sots}}(1^k)$
(Repeat until $\mathrm{Hash}(vk_{\mathrm{sots}}) \neq -x_i$)
$\rho \leftarrow \mathbb{Z}_n$; $a := a_i f^{-\rho}$; $b := b_i(hv_i)^\rho$
$\sigma := g^{\frac{1}{x_i + \mathrm{Hash}(vk_{\mathrm{sots}})}}$
$\pi \leftarrow P_{\mathrm{NIWI}}(crs, (gpk, a, \mathrm{Hash}(vk_{\mathrm{sots}})), (b, v_i, \sigma))$
$y \leftarrow E_{pk}(\mathrm{Hash}(vk_{\mathrm{sots}}), v_i)$
$\psi \leftarrow P_{\mathrm{NIZK}}(crs, (gpk, y, \pi), (r, s, t))$
$\sigma_{\mathrm{sots}} \leftarrow \mathrm{Sign}_{sk_{\mathrm{sots}}}(vk_{\mathrm{sots}}, m, a, \pi, y, \psi)$
Return $\Sigma := (vk_{\mathrm{sots}}, a, \pi, y, \psi, \sigma_{\mathrm{sots}})$

$\mathbf{GVf}(gpk, m, \Sigma)$
Return 1 if these verifications pass:
$\mathrm{Ver}_{vk_{\mathrm{sots}}}((vk_{\mathrm{sots}}, m, a, \pi, y, \psi), \sigma_{\mathrm{sots}})$
$V_{\mathrm{NIWI}}(crs, (gpk, a, \mathrm{Hash}(vk_{\mathrm{sots}})), \pi)$
$V_{\mathrm{NIZK}}(crs, (gpk, \pi, y), \psi)$
$\mathrm{ValidCiphertext}(pk, \mathrm{Hash}(vk_{\mathrm{sots}}), y)$
Else return 0

$\mathbf{Open}(gpk, ok, m, \Sigma)$
$(b, v, \sigma) \leftarrow X_{xk}(crs,$
$(gpk, a, \mathrm{Hash}(vk_{\mathrm{sots}})), \pi)$
Return (i, σ) if there is i so $v = v_i$
Else return $(0, \sigma)$

$\mathbf{Judge}(gpk, i, reg[i], m, \Sigma, \sigma)$
Return 1 if
$i \neq 0 \ \wedge\ e(\sigma, v_i g^{\mathrm{Hash}(vk_{\mathrm{sots}})}) = e(g,g)$
Else return 0

Fig. 2. The group signature scheme.

The NIZK proof implies that the ciphertext y contains the same Boneh-Boyen signature σ as the NIWI proof of knowledge. The opener can therefore use the decryption key for the tag-based cryptosystem to track down the user instead of extracting it from the NIWI proof of knowledge. This means we do not need the extraction key for the NIWI proof, so we can switch to using a common reference string that gives perfect witness-indistinguishability. The only information about the member now resides in the ciphertext. The existential unforgeability of the one-time signature under weak chosen message attack and the collision-freeness of the hash-function make it infeasible for the adversary to query the opener with a valid group signature that recycles vk_{sots} from the challenge or that collides with $\mathrm{Hash}(vk_{\mathrm{sots}})$. Since $\mathrm{Hash}(vk_{\mathrm{sots}})$ is the tag for the cryptosystem and is never recycled in a query to the opener, the ciphertext does not reveal which member made the group signature.

We have to argue that a user cannot be framed. We consider an unfriendly environment where both the issuer and the opener are corrupt. They are trying to come up with a proof that the user signed a message, a proof that consists of a Boneh-Boyen signature. When joining the group, the user and the issuer engage in a key registration protocol. This protocol gives the user a uniformly random x and a Boneh-Boyen verification key $v = g^x$, without the issuer learning x. Even if the user makes group signatures on arbitrary messages, this just corresponds to signing randomly chosen verification keys for the strong one-time signature scheme. The weak chosen message attack security of the

Boneh-Boyen signature scheme is therefore sufficient to guarantee that the adversary cannot falsely accuse the user of having signed a message that she did not sign.

Finally, we consider an honest issuer that keeps her issuer key secret and an honest opener with an exposed opener key. We have to argue that a valid group signature can always be traced back to a member of the group. By the perfect extractability of the NIWI proof of knowledge, we can extract a valid certified signature on $\mathrm{Hash}(vk_{\mathrm{sots}})$ from the NIWI proof π. The key registration protocol guarantees that all members have correctly generated signing keys. The unfakeability of the certified signature scheme therefore implies that a member has made the group signature. The Boneh-Boyen signature σ is sufficient to trace this member, since it matches a unique verification key v_i. $\square$

EFFICIENCY. If we instantiate the strong one-time signature with the Boneh-Boyen signature scheme a verification key is one group element and a one-time signature is also one group element. We make the element a public. The NIWI proof of knowledge consists of 27 group elements. The ciphertext consists of 5 group elements. The NIZK proof consists of 15 group elements. The total size of a group signature is therefore 50 group elements in G. This is of course much better than the many thousand elements required for a group signature in [Gro06].

In case CPA-anonymity is sufficient, we can consider a lighter version of our group signature, where we omit the ciphertext y and the NIZK proof ψ. This CPA-anonymous group signature scheme would consist of 30 group elements. We observe that regular anonymity implies that the group signature is strong, i.e., even when seeing a message m and a group signature Σ on it, it is not possible to create a different group signature Σ' on m such that it still points to the same member. In CPA-anonymity, however, we do not give the adversary access to an opening oracle and thus mauling signatures is no longer a problem. If we do not care about the group signature being strong, we do not need the strong one-time signature key and we can simply sign $\mathrm{Hash}(m)$ instead of $\mathrm{Hash}(vk_{\mathrm{sots}})$. This reduces the size of the group signatures further to 28 group elements. In comparison, the CPA-anonymous group signature scheme of [BW07] consists of 6 group elements in a composite order group. Since composite order groups rely on the hardness of factoring, these groups are very large and our CPA-anonymous group signatures are therefore comparable in size for practical parameters, perhaps even a bit smaller. However, our CPA-anonymous group signature scheme still supports dynamic enrollment of members and has a group public key gpk consisting of a constant number of group elements.

KEY GENERATION. Since the [BSZ05]-model assumes a trusted key generator it is worth considering how the key generation should be carried out in practice. The trust in our scheme relies on the bilinear group (p, G, G_T, e, g) being generated so the cryptographic assumptions hold and it relies on the hash-function being collision-free. We remark that an advantage of our scheme is that we work over prime order bilinear groups, so it may be possible to use a uniform random string to set up (p, G, G_T, e, g). Also, since the trust is based on a very elementary setup, a bilinear group and a hash-function, it is possible that suitable public standards can be found. One could for instance use SHA-256 as the hash-function.

The non-frameability of the user relies only on the collision-freeness of the hash-function and the cryptographic assumptions in (p, G, G_T, e, g). The rest of the group public key gpk can be generated jointly by the issuer and the opener. The issuer generates the authority key for the certified signature scheme. The opener generates crs and pk, anonymity follows from the opener generating these keys correctly. Since the opener can break anonymity anyway, it is quite reasonable to trust the opener with protecting anonymity. The opener will have to make a zero-knowledge proof of knowledge of the corresponding extraction key to the issuer, since the security proof for traceability relies on the opener being able to actually extract a signature from the NIWI proof of knowledge.

References

[ACHdM05] Giuseppe Ateniese, Jan Camenisch, Susan Hohenberger, and Breno de Medeiros. Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385, 2005. `http://eprint.iacr.org/2005/385`.

[ACJT00] Guiseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure group signature scheme. In *proceedings of CRYPTO '00, LNCS series, volume 1880*, pages 255–270, 2000.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *proceedings of EUROCRYPT '04, LNCS series, volume 3027*, pages 56–73, 2004.

[BBP04] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid encryption problem. In *proceedings of EUROCRYPT '04, LNCS series, volume 3027*, pages 171–188, 2004. Full paper available at `http://eprint.iacr.org/2003/077`.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *proceedings of CRYPTO '04, LNCS series, volume 3152*, pages 41–55, 2004.

[BFPW07] Alexandra Boldyreva, Marc Fischlin, Adriana Palacio, and Bogdan Warinschi. A closer look at pki: Security and efficiency. In *proceedings of PKC '07, LNCS series, volume 4450*, pages 458–475, 2007.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *proceedings of EUROCRYPT '03, LNCS series, volume 2656*, pages 416–432, 2003.

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *proceedings of EUROCRYPT '03, LNCS series, volume 2656*, pages 614–629, 2003.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS '93*, pages 62–73, 1993.

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In *proceedings of CT-RSA '05, LNCS series, volume 3376*, pages 136–153, 2005.

[BW06] Xavier Boyen and Brent Waters. Compact group signatures without random oracles. In *proceedings of EUROCRYPT '06, LNCS series, volume 4004*, pages 427–444, 2006.

[BW07] Xavier Boyen and Brent 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, 2007. Available at `http://www.cs.stanford.edu/~xb/pkc07/`.

[CG04] Jan Camenisch and Jens Groth. Group signatures: Better efficiency and new theoretical aspects. In *proceedings of SCN '04, LNCS series, volume 3352*, pages 120–133, 2004. Full paper available at `http://www.brics.dk/~jg/GroupSignFull.pdf`.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. In *proceedings of STOC '98*, pages 209–218, 1998.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. On the random-oracle methodology as applied to length-restricted signature schemes. In *proceedings of TCC '04, LNCS series, volume 2951*, pages 40–57, 2004.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *proceedings of CRYPTO '04, LNCS series, volume 3152*, pages 56–72, 2004.

[CvH91] David Chaum and Eugène van Heyst. Group signatures. In *proceedings of EUROCRYPT '91, LNCS series, volume 547*, pages 257–265, 1991.

[FI05] Jun Furukawa and Hideki Imai. An efficient group signature scheme from bilinear maps. In *proceedings of ACISP '05, LNCS series, volume 3574*, pages 455–467, 2005.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *proceedings of FOCS '03*, pages 102–113, 2003. Full paper available at `http://eprint.iacr.org/2003/034`.

[Gro06] Jens Groth. Simulation-sound nizk proofs for a practical language and constant size group signatures. In *proceedings of ASIACRYPT '06, LNCS series*, 2006. Full paper available at `http://www.brics.dk/~jg/NIZKGroupSignFull.pdf`.

[GS07] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. Cryptology ePrint Archive, Report 2007/155, 2007. Available at `http://eprint.iacr.org/2007/155`.

[Kil06] Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In *proceedings of TCC '06, LNCS series, volume 3876*, pages 581–600, 2006.

[KY05] Aggelos Kiayias and Moti Yung. Group signatures with efficient concurrent join. In *proceedings of EUROCRYPT '05, LNCS series, volume 3494*, pages 198–214, 2005. Full paper available at `http://eprint.iacr.org/345`.

[MRY04] Philip D. MacKenzie, Michael K. Reiter, and Ke Yang. Alternatives to non-malleability: Definitions, constructions, and applications. In *proceedings of TCC '04, LNCS series, volume 2951*, pages 171–190, 2004.

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *proceedings of CRYPTO '02, LNCS series, volume 2442*, pages 111–126, 2002.

[ZL06] Sujing Zhou and Dongdai Lin. Shorter verifier-local revocation group signatures from bilinear maps. In *proceedings of CANS '06, LNCS series, volume 4301*, pages 126–143, 2006.