Compact Group Signatures Without Random Oracles

Xavier Boyen [*] Brent Waters [†]

October 21, 2005

Abstract

We present the first efficient group signature scheme that is provably secure without random oracles. We achieve this result by combining provably secure hierarchical signatures in bilinear groups with a novel adaptation of the recent Non-Interactive Zero Knowledge proofs of Groth, Ostrovsky, and Sahai. Our scheme has logarithmic signature size in the number of signers; we prove it secure under the Computational Diffie-Hellman and the Subgroup Decision assumptions.

1 Introduction

Group signatures allow any member of a group to sign an arbitrary number of messages on behalf of the group, moreover the identity of the signer will be hidden from all members of the system. Preserving the anonymity of the signer can be important in many applications where the signer does not want to be directly identified with the message that he signed. However, there exist situations where it can be deemed desirable to revoke a signer's anonymity. For example, if a signature certified a malicious program, one would want to identify the party that made the malicious statement. Therefore, in group signatures there exists a special party known as the group manager which has the ability to trace the signer of any given signature.

Almost all group signatures scheme are only provably secure in the random oracle model, where we can only make a heuristic argument about security. Additionally, efficient constructions are based on strong assumptions ranging from the Strong Diffie-Hellman [BBS04, BS04] and Strong RSA [ACJT00, ATS02, CL02] assumptions to the LRSW [CL04, LRSW99] assumption which itself has the challenger act as an oracle. The first construction proved secure in the standard model is due to Bellare et. al. [BMW03]. They give a method of constructing group signatures from any signature scheme by using Non-Interactive Zero Knowledge (NIZK) techniques. However, since they use generic NIZK techniques their scheme is too inefficient to be useful in practice.

We approach the problem of group signatures with the goal of creating an efficient group signature scheme that is provably secure without random oracles under reasonable assumptions. In particular we at least wish to avoid "oracle-like" assumptions since the value of removing random oracles from the proofs of security while using these types of assumptions is dubious.

In order to solve this problem we combine two recent ideas from pairing-based cryptography. First, we derive our underlying signature scheme from the Waters [Wat05] signature scheme that was proven secure under the computational Diffie-Hellman assumption in bilinear groups. We create

[*]Voltage Inc., Palo Alto — `xb@boyen.org`
[†]SRI, Palo Alto — `bwaters@csl.sri.com`

a two-level signature scheme where the first level is the signer identity and the second level is the message to be signed. For example, user ID is given a signature on the first level message "ID" as his private key. Group member ID can sign message M by creating the two-level hierarchical signature on "$\mathsf{ID}.M$". Clearly, the signature σ on "$\mathsf{ID}.M$" from the Waters signature scheme will give away the identity of the signer. To protect his anonymity, a signer, in our scheme, will encrypt the signature components of σ using the Boneh-Goh-Nissim [BGN05] encryption system. Additionally, the signer will attach a NIZK proof that the encrypted signature is a signature on "$X.M$" for $1 \leq X \leq 2^k$, where 2^k is the number of signers in the system. Adapting the recent techniques of Groth, Ostrovsky, and Sahai we are able to get efficient NIZKs for our scheme scheme with $O(k)$ complexity in the signature size, signing time, and verification time, i.e., logarithmic in the number of users. We achieve this efficiency by taking advantage of special properties of the NIZK scheme of Groth, Ostrovsky, and Sahai and avoid the general method of circuit construction.

1.1 Related Work

Group signatures were first introduced by Chaum and Van Heyst [CvH91] as a way to provide anonymity for signers within a group. The anonymity, however, could be revoked by a special third party if necessary.

Until recently, the most efficient group signature constructions [ACJT00, ATS02, CL02] were proved secure under the Strong-RSA assumption introduced by Baric and Pfitzman [BP97]. Boneh, Boyen, and Shacham [BBS04] showed how to construct "short" group signatures using bilinear maps under an assumption they introduced called the Strong Diffie-Hellman assumption. Concurrently, Camenish and Lysyanskaya [CL04] gave another group signature scheme that used bilinear maps. Their scheme was proven secure under the interactive LRSW [LRSW99] assumption. All of the above schemes, however, were only proved secure in the random oracle model.

Bellare, Micciancio, and Warinschi [BMW03] gave the first construction that was provably secure in the standard model. Additionally, they provided formal definitions of the security properties of group signatures, which to that point were only informally understood. Since their methods use general NIZK proof techniques, the resulting schemes are inherently too inefficient to be used in practice.

1.2 Organization

In Section 2 we give background information pertinent to our constructions. In Section 3 we present our group signature scheme that has cost logrithmic in the number of signers. Additionally, we provide a proof of security in Section 4. Next, we describe some extensions to our signature scheme in Section 5. Finally, we conclude in Section 6.

2 Background

We review a number of useful notions from the recent literature on pairing-based cryptography, which we shall need in later sections. First, we briefly review the properties that constitute a group signature scheme and define its security.

We take this opportunity to clarify once and for all that, in this paper, the word "group" by default assumes its algebraic meaning, except in contexts such as "group signature" and "group manager" where it designates a collection of users. There should be no ambiguity from context.

2.1 Group Signatures

A group signature scheme consists of a pentuple of PPT algorithms:

- A group setup algorithm, *Setup*, that takes as input a security parameter 1^λ (in unary) and the number of signers in the group, for simplicity taken as a power of two, 2^k, and outputs a public key PK for verifying signatures, a master key MK for enrolling group members, and a tracing key TK for identifying signers.

- An enrollment algorithm, *Enroll*, that takes the master key MK and an identity ID, and outputs a unique identifier s_{ID} and a private signing key K_{ID} which is to be given to the user.

- A signing algorithm, *Sign*, that takes a group member's private signing key K_{ID} and a message M, and outputs a signature σ.

- A (usually deterministic) verification algorithm, *Verify*, that takes a message M, a signature σ, and a group verification key PK, and outputs either `valid` or `invalid`.

- A (usually deterministic) tracing algorithm, *Trace*, that takes a valid signature σ and a tracing key TK, and outputs an identifier s_{ID} or the failure symbol $\perp$.

There are four types of entities one must consider:

- The group master, which sets up the group and issues private keys to the users. Often, the group master is an ephemeral entity, and the master key MK is destroyed once the group is set up. Alternatively, techniques from distributed cryptography can be used to realize the group master functionality without any real party becoming in possession of the master key.

- The group manager, which is given the ability to identify signers using the tracing key TK, but not to enroll users or create new signing keys for existing users.

- Regular member users, or signers, which are each given a distinct private signing key K_{ID}.

- Outsiders, or verifiers, who can only verify signatures using the public key PK.

We require the following correctness and security properties.

Consistency. The consistency requirements are such that, whenever, (for a group of 2^k users),

$$(\mathsf{PK},\mathsf{MK},\mathsf{TK}) \leftarrow \mathit{Setup}(1^\lambda, 2^k), \qquad (s_{\mathsf{ID}}, K_{\mathsf{ID}}) \leftarrow \mathit{Enroll}(\mathsf{MK},\mathsf{ID}), \qquad \sigma \leftarrow \mathit{Sign}(K_{\mathsf{ID}}, M),$$

we have, (except with negligible probability over the random bits used in *Verify* and *Trace*),

$$\mathit{Verify}(M,\sigma,\mathsf{PK}) = \mathtt{valid}, \qquad \text{and} \qquad \mathit{Trace}(\sigma,\mathsf{TK}) = s_{\mathsf{ID}}.$$

The unique identifier s_{ID} can be used to assist in determining the user ID from the transcript of the *Enroll* algorithm; s_{ID} may but need not be disclosed to the user; it may be the same as ID.

Security. Bellare, Micciancio, and Warinschi [BMW03] characterize the fundamental properties of group signatures in terms of two crucial security properties from which a number of other properties follow. The two important properties are:

Full Anonymity which requires that no PPT adversary be able to decide (with non-negligible probability in excess of one half) whether a challenge signature σ on a message M emanates from user ID_1 or ID_2, where ID_1, ID_2, and M are chosen by the adversary. In the original definition of [BMW03], the adversary is given access to a tracing oracle, which it may query before and after being given the challenge σ, much in the fashion of IND-CCA2 security for encryption.

Boneh, Boyen, and Shacham [BBS04] relax this definition by withholding access to the tracing oracle, thus mirroring the notion of IND-CPA security for encryption. We follow [BBS04] and speak of *CCA2-full anonymity* and *CPA-full anonymity* respectively.

Full Traceability which requires that no coalition of users be able to generate, in polynomial time, a signature that passes the *Verify* algorithm but fails to trace to a member of the coalition under the *Trace* algorithm. According to this notion, the adversary is allowed to ask for the private keys of any user of its choice, adaptively, and is also given the secret key TK meant for tracing—but of course not the enrollment master key MK.

It is noted in [BMW03] that this property implies that of *exculpability* [AT99], which is the requirement that no party, not even the group manager, should be able to frame a honest group member as the signer of a signature he did not make. However, the model of [BMW03] does not consider the possibility of a (long-lived) group master, which could act as a potential framer. To address this problem and achieve the notion of *strong exculpability*, introduced in [ACJT00] and formalized in [KY04, BSZ04], one would need an interactive enrollment protocol, call *Join*, at the end of which only the user himself knows his full private key. We do not further consider exculpability issues in this paper.

We refer the reader mainly to [BMW03] for more precise definitions of these and related notions.

2.2 Bilinear Groups of Composite Order

We review some general notions about bilinear maps and groups, with an emphasis on groups of *composite order* which will be used in most of our constructions. We follow [BGN05] in which composite order bilinear groups were first introduced in cryptography.

Consider two finite cyclic groups G and G_T of same order n, in which the respective group operation is efficiently computable and denoted multiplicatively. Assume the existence of an efficiently computable function $e : G \times G \to G_T$, with the following properties:

- (Bilinearity) $\forall u, v \in G$, $\forall a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$, where the product in the exponent is defined modulo n;
- (Non-degeneracy) $\exists g \in G$ such that $e(g, g)$ has order n in G_T. In other words, $e(g, g)$ is a generator of G_T, whereas g generates G.

If such a function can be computed efficiently, it is called a (symmetric) bilinear map or pairing, and the group G is called a bilinear group. We remark that the vast majority of cryptosystems based on pairings assume for simplicity that bilinear groups have prime order. In our case, it is important that the pairing be defined over a group G containing $|G| = n$ elements, where $n = pq$ has a (hidden) factorization in two large primes, $p \neq q$.

Complexity Assumptions. We shall make use of two complexity assumptions: the first, computational in the prime order subgroup G_p, the second, decisional in the full group G.

The first one is the familiar Computational Diffie-Hellman assumption in bilinear groups, which states that there is no probabilistic polynomial time (PPT) adversary that, given a triple $(g, g^a, g^b) \in G_p^3$ for random exponents $a, b \in \mathbb{Z}_p$, computes $g^{ab} \in G_p$ with non-negligible probability (i.e., with polynomial probability in the bit-size of the algorithm's input). Notice that the CDH assumption in bilinear groups is weaker than—i.e., it is implied by—the computational Bilinear Diffie-Hellman assumption [BF03].

The second assumption we need is the subgroup decision assumption, introduced in [BGN05]; it is based on the hardness of factoring, and is recalled next.

2.3 Subgroup Decision Assumption

Informally, the subgroup decision assumption posits that for a bilinear group G of composite order $n = pq$, the uniform distribution on G is computationally indistinguishable from the uniform distribution on a subgroup of G (say, G_q, the subgroup of order q). The formal definition is based on the subgroup decision problem, which is as follows [BGN05].

The Subgroup Decision Problem. Consider an "instance generator" algorithm $\mathcal{GG}$ that, on input a security parameter 1^λ, outputs a tuple (p, q, G, G_T, e), in which p and q are independent uniform random λ-bit primes, G and G_T are cyclic groups of order $n = pq$ with efficiently computable group operations (over their respective elements, which must have a polynomial size representation in λ), and $e : G \times G \to G_T$ is a bilinear map. Let $G_q \subset G$ denote the subgroup of G of order q. The subgroup decision problem is as follows:

> On input a tuple $(n = pq, G, G_T, e)$ derived from a random execution of $\mathcal{GG}(1^\lambda)$, and an element w selected at random either from G or from G_q, decide whether $w \in G_q$.

The advantage of an algorithm $\mathcal{A}$ solving the subgroup decision problem is defined as $\mathcal{A}$'s excess probability, beyond $\frac{1}{2}$, of outputting the correct solution. The probability is defined over the random choice of instance and the random bits used by $\mathcal{A}$.

2.4 The BGN Homomorphic Cryptosystem

Boneh, Goh, and Nissim [BGN05] recently proposed an efficient encryption system that is "slightly" doubly homomorphic, in that it allows chains of computations on ciphertexts that involve arbitrary additions and at most one multiplication along each path. The system uses bilinear maps and is based on the subgroup decision assumption. The single-bit message cryptosystem is as follows:

KeyGen(1^λ)**:** to generate a public/private key pair given as input a security parameter 1^λ, first run the instance generator to get $(p, q, G, G_T, e) \leftarrow \mathcal{GG}(1^\lambda)$, let $n = pq$, pick two random generators $g, u \in G$, and set $h = u^p$. Notice that g and h are uniform random generators of G and G_q respectively. The public key is $\mathsf{PK} = (n, G, G_T, e, g, h)$. The private key is $\mathsf{SK} = q$.

Encrypt(PK, M)**:** to encrypt a single-bit message $M \in \{0,1\}$, pick a random $r \in \mathbb{Z}_n$ and output the ciphertext $c = g^M h^r$.

Decrypt(SK, c)**:** to decrypt a ciphertext c, calculate c^q. If $c^q = g^0$, output 0; if $c^q = g^q$, output 1; otherwise, output $\perp$. Indeed, notice that $c^q = (g^M h^r)^q = (g^q)^M$ since h has order q and thus vanishes when raised to that power.

Homomorphic Properties. The system is additively homomorphic since multiplication of ciphertexts corresponds to addition of messages. Furthermore, the product of two messages can be obtained *once* by taking the pairing of the two ciphertexts, with suitable randomization, i.e., $c' = e(c_1, c_2) \cdot e(g, h)^t \in G_T$, for some randomly chosen $t \in \mathbb{Z}_p$. Notice that the constant $e(g, h)$ has order q in G_T. Decryption of ciphertexts obtained this way works exactly as above, except that since $(c')^q$ is now in G_T, it is compared to $e(g, g)^0$ and to $e(g, g)^q$ in G_T. Notice that any calculation circuit should be designed to ensure that the final output is in $\{0, 1\}$, which is sufficient for evaluating 2-DNF formulæ in ciphertext space, as the primary application mentioned in [BGN05].

2.5 Non-Interactive Zero-Knowledge Proofs for NP

Groth, Ostrovsky, and Sahai [GOS05] very recently showed how to use the BGN system to construct an efficient statistical zero knowledge proof system (or "argument system", since it only provides computational soundness) for any language in NP. A key component of their result is the observation that the BGN system can be modified to provide perfectly hiding commitment instead of encryption, while ostensibly retaining the homomorphic features that enabled the evaluation of 2-DNF formulæ, without an outsider being able to realize that we made the switch.

Thus, to argue zero knowledge, it suffices to let h be a random generator of G instead of G_q. This renders the "ciphertext" statistically independent of the message it contains, hence realizing perfect (and thus, statistical) hiding. Then, by the subgroup decision assumption, one should be able to restore h to being a generator of G_q without anyone being able to tell (computationally).

Relation to Group Signatures. We shall exploit the simple but powerful idea above to achieve the full-anonymity requirement in our group signatures. This stems from the notion that a group signature can be viewed as a zero-knowledge proof of authority to sign a message on behalf of the group, which is an NP language. We note, however, that a zero-knowledge proof system is not enough in itself, since in group signatures it is necessary to embed authoritative information that will endow the group manager with the ability to trace. To meet both requirements, our strategy makes use of an identity-based signature, or two-level hierarchical signature, in which the user identity is concealed using the NIZK technique. Before delving into all that, however, we review the notion of hierarchical signature.

2.6 Hierarchical Signatures

In an Λ-level hierarchical signature, a message is a tuple of 1 and Λ message components. The crucial property is that a signature on a message, $M_1. \cdots .M_i$, can act as a restricted private key that enables the signing of any extension, $M_1. \cdots .M_i. \cdots .M_j$, of which the original message is a prefix. In a Λ-level signature scheme, the messages must obey the requirement that $1 \leq i \leq j \leq \Lambda$.

We note that this is essentially equivalent to the notion of $(\Lambda-1)$-hierarchical identity-based signature (HIBS), in which the first $\Lambda-1$ levels are viewed as the components of a hierarchical identity, and the last level (which in HIBS parlance is no longer deemed part of the hierarchy) is for the message proper. Our basic group signature uses a two-level hierarchy, though in Section 5 we shall discuss how additional levels can be used to achieve delegation in group signatures.

3 Group Signature Scheme

In this section, we present our group signature scheme, which is based solely on the CDH and the Subgroup Decision assumptions. It is built upon a two-level hierarchical signature scheme, which we describe first.

3.1 Simple Two-Level Hierarchical Signatures

Waters [Wat05] recently offered an efficient identity-based encryption system provably secure under "full" adaptive attacks. The system generalizes easily to a hierarchical IBE of depth $O(\log\lambda)$, and it is a triviality to observe that any Λ-level HIBE scheme also gives an Λ-level hierarchical signature functionality. We describe below the 2-level hierarchical signature scheme (or 1-level IBS) that results from these transformations.

We assume that identities are strings of k bits, and messages strings of m bits. To fix ideas, for group signatures one would have, $k \ll m \approx \lambda$, where λ is the security parameter. The description that follows assumes that g is a generator of G_p, so that all elements in G and G_T are in fact in the respective subgroups of prime order p.

Setup(1^λ)**:** To setup the system, first, a secret $\alpha \in \mathbb{Z}_p$ is chosen at random, from which the value $A = e(g,g)^\alpha$ is calculated. Next, two random integers $y' \in \mathbb{Z}_p$ and $z' \in \mathbb{Z}_p$ and two random vectors $\vec{y} = (y_1,\ldots,y_k) \in \mathbb{Z}_p^k$ and $\vec{z} = (z_1,\ldots,z_m) \in \mathbb{Z}_p^m$ are selected. The public parameters of the system and the master secret key are then given by,

$$\begin{aligned}\mathsf{PP} = \Big(\; & g,\ u' = g^{y'},\ \ u_1 = g^{y_1},\ \ \ldots,\ \ u_k = g^{y_k}, \\ & v' = g^{z'},\ \ v_1 = g^{z_1},\ \ \ldots,\ \ v_m = g^{z_m},\ \ A = e(g,g)^\alpha \;\Big) \in G^{k+m+3} \times G_T, \\ \mathsf{MK} = \ & g^\alpha \ \in G.\end{aligned}$$

The public parameters, PP, also implicitly include k, m, and a description of (p, G, G_T, e).

Extract$(\mathsf{PP},\mathsf{MK},\mathsf{ID})$**:** To create a private key for a user whose binary identity string is $\mathsf{ID} = (\kappa_1 \ldots \kappa_k) \in \{0,1\}^k$, first select a random $r \in \mathbb{Z}_p$, and return,

$$K_{\mathsf{ID}} = \Big(\; g^\alpha \cdot \big(u' \prod_{i=1}^{k} u_i^{\kappa_i}\big)^r,\ \ g^{-r} \;\Big) \in G^2.$$

Sign$(\mathsf{PP}, K_{\mathsf{ID}}, M)$**:** To sign a message represented as a bit string $M = (\mu_1 \ldots \mu_m) \in \{0,1\}^m$, using a private key $K_{\mathsf{ID}} = (K_1, K_2) \in G^2$, select a random $s \in \mathbb{Z}_p$, and output,

$$S = \Big(\ K_1 \cdot \big(v' \prod_{j=1}^{m} v_j^{\mu_j}\big)^s,\ \ K_2,\ \ g^{-s}\ \Big)$$
$$= \Big(\ g^{\alpha} \cdot \big(u' \prod_{i=1}^{k} u_i^{\kappa_i}\big)^r \big(v' \prod_{j=1}^{m} v_j^{\mu_j}\big)^s,\ \ g^{-r},\ \ g^{-s}\ \Big) \in G^3.$$

Verify$(\mathsf{PP}, \mathsf{ID}, M, \sigma)$**:** To verify a signature $S = (S_1, S_2, S_3) \in G^3$ against an identity $\mathsf{ID} = (\kappa_1 \dots \kappa_k) \in \{0,1\}^k$ and a message $M = (\mu_1 \dots \mu_m) \in \{0,1\}^m$, verify that,

$$e(S_1, g) \cdot e(S_2, u' \prod_{i=1}^{k} u_i^{\kappa_i}) \cdot e(S_3, v' \prod_{j=1}^{m} v_j^{\mu_j}) \stackrel{?}{=} A.$$

If the equality holds, output `valid`; otherwise, output `invalid`.

Security from CDH. The scheme's existential unforgeability against adaptive chosen message attacks follows more or less directly from the adaptive-ID security of the Waters IBE, although it relies on a weaker assumption. For completeness, we give a reduction from the CDH assumption without random oracles in Appendix A. There is an unusual aspect to our proof of security, however.

Paradoxically, we can achieve tighter overall security bounds by proving the signature in a hybrid "selective-identity, adaptive-message" model, which combines aspects of the Boneh-Boyen [BB04] IBE proof of security for the first level with the Waters [Wat05] proof for the second level, instead of using the latter at both levels. The first-level proof is followed by an implicit application of the (lossy) selective-ID to adaptive-ID transformation given in [BB04]. We benefit from this because the Boneh-Boyen proof is tight in the selective-ID model, and the subsequent adaptive-ID lifting degrades linearly with the number of possible identities, 2^k, which for a group signature scheme is much smaller and easier to control than the number of adversarial queries. We need the full Waters proof at the second level since the number of possible messages, 2^m, must be large enough to accommodate the output of a hash function. (For very small messages, such as serial numbers, one could use the Boneh-Boyen proof at both levels to achieve even tighter security.)

3.2 Logarithmic-Size Group Signature Scheme

We are now in a position to describe our actual group signature scheme. It is composed of the following algorithms.

Setup(1^λ)**:** The input is a security parameter in unary, 1^λ. Suppose we wish to support up to 2^k signers in the group, and sign messages in $\{0,1\}^m$, where k and m are some functions of λ.

The setup algorithm first chooses $n = pq$ where p and q are random primes of bit size $\Theta(\lambda)$. Let G be a bilinear group of order n and denote by G_p and G_q its subgroups of respective order p and q. Next, the algorithm chooses generators $g \in G$, and $h \in G_q$. It chooses a random exponent $\alpha \in \mathbb{Z}_n$. Finally, it chooses generators $u', u_1, \dots, u_k \in G$ and $v', v_1, \dots, v_m \in G$.

The bilinear group, (n, G, G_T, e), is published together with the public parameters,

$$\mathsf{PP} = \Big(\ g, \ h, \ u', \ u_1, \ \ldots, \ u_k, \ v', \ v_1, \ \ldots, \ v_m, \ A = e(g,g)^{\alpha} \ \Big) \ \in G \times G_q \times G^{k+m+2} \times G_T.$$

The master key for user enrollment, MK, and the group manager's tracing key, TK, are,

$$\mathsf{MK} = g^{\alpha} \ \in G, \qquad\qquad \mathsf{TK} = q \ \in \mathbb{Z}.$$

Enroll$(\mathsf{PP}, \mathsf{MK}, \mathsf{ID})$**:** Suppose we wish to create a group signature key for user ID where $0 \leq \mathsf{ID} < 2^k$. We denote by κ_i the i-th bit of ID. The algorithm chooses a random $s \in \mathbb{Z}_n$ and creates the key for user ID as,

$$K_{\mathsf{ID}} = (K_1, K_2, K_3) = \left(\ g^{\alpha} \cdot \Big(u' \prod_{i=1}^{k} u_i^{\kappa_i}\Big)^{s}, \ \ g^{-s}, \ \ h^{s} \ \right) \ \in G^3.$$

The key for user ID is essentially a private key for identity ID in the Waters IBS scheme, except that we are working in a bilinear group G of composite order, and are adjoining the additional element $h^s \in G_q$.

Sign$(\mathsf{PP}, \mathsf{ID}, K_{\mathsf{ID}}, M)$**:** To sign a message $M = (\mu_1 \ldots \mu_m) \in \{0,1\}^m$, user ID first chooses random exponents $t_1, \ldots, t_k \in \mathbb{Z}_n$, and, for all $i = 1, \ldots, k$, it creates,

$$c_i = u_i^{\kappa_i} \cdot h^{t_i}, \qquad\qquad \pi_i = (u_i^{2\kappa_i - 1} \cdot h^{t_i})^{t_i}.$$

The signer also defines $t = \sum_{i=1}^{k} t_i$ and $c = u' \prod_{i=1}^{k} c_i = (u' \prod_{i=1}^{k} u_i^{\kappa_i}) \cdot h^t$. The set of values, c_i and π_i, are proof that c is well formed. It also lets $V = v' \prod_{i=1}^{m} v_i^{\mu_i}$. Then, it picks two random exponents $\tilde{s}_1, s_2 \in \mathbb{Z}_n$, and creates,

$$\sigma_1 = K_1 \cdot K_3^t \cdot c^{\tilde{s}_1} \cdot V^{s_2}, \qquad\qquad \sigma_2 = K_2 \cdot g^{-\tilde{s}_1}, \qquad\qquad \sigma_3 = g^{-s_2}.$$

If we let $s_1 = \tilde{s}_1 + s$, with s as in the *Enroll* procedure, then we have,

$$\sigma_1 = g^{\alpha} \cdot \Big(u' \prod_{i=1}^{k} u_i^{\kappa_i}\Big)^{s_1} \cdot \Big(v' \prod_{i=1}^{m} v_i^{\mu_i}\Big)^{s_2} \cdot h^{s_1 t} = g^{\alpha} \cdot c^{s_1} \cdot V^{s_2}, \qquad \sigma_2 = g^{-s_1}, \qquad \sigma_3 = g^{-s_2}.$$

The final signature is output as:

$$\sigma = \big(\ \sigma_1, \sigma_2, \sigma_3, \ c_1, \ldots, c_k, \ \pi_1, \ldots, \pi_k \ \big) \ \in G^{2k+3}.$$

Verify$(\mathsf{PP}, \mathsf{ID}, M, \sigma)$**:** The verification proceeds in two phases. In the first phase the verifier will reconstruct c and check to make sure that it is well formed. To do this, it computes,

$$c = u' \prod_{i=1}^{k} c_i, \qquad \text{and checks that,} \quad \forall i = 1, \ldots, k \ : \ e(c_i, u_i^{-1} c_i) \stackrel{?}{=} e(h, \pi_i).$$

This proof shows that all $c_i = u_i^{\kappa_i} h^{t_i}$ for $\kappa_i \in \{0,1\}$, and thus that c is well formed. Next, the verifier focuses on the actual signature. To do so, it derives $V = v' \prod_{i=1}^{m} v_i^{\mu_i}$ from the message, and checks that,

$$e(\sigma_1, g) \cdot e(\sigma_2, c) \cdot e(\sigma_3, V) \stackrel{?}{=} A.$$

This proof shows that $(\sigma_1, \sigma_2, \sigma_3)$ is a valid two-level hierarchical signature, after the blinding factors $h^{s_1 t}$ and h^t cancel each other out in the product after they are respectively paired with g and g^{-s_1}.

If all tests are successful, the verifier outputs `valid`; otherwise, it outputs `invalid`.

Trace$(\mathsf{PP}, \mathsf{TK}, \sigma)$**:** Suppose the tracing algorithm wishes to trace a signature σ, assumed to pass the verification test for some message M that is not needed here. Let κ_i denote the i-th bit of the signer's identity ID that is to be determined. To recover the bits of ID, for each $i = 1, \ldots, k$, the tracer sets,

$$\kappa_i = \begin{cases} 0 & \text{if } (c_i)^q = g^0, \\ 1 & \text{otherwise.} \end{cases}$$

The reconstituted signer identity is output as $\mathsf{ID} = (\kappa_1 \ldots \kappa_k) \in \{0,1\}^k$.

4 Proofs of Security

We now prove the main security properties of our group signature scheme.

4.1 Full Anonymity (under CPA attack)

We prove the security of our scheme in the anonymity game against chosen plaintext attacks. We refer to [BMW03] for the game description, which should also be clear from the proof.

Intuitively, our proof follows from two simple arguments. First, we show that an adversary cannot tell whether h is a random generator of G_q or G by reduction from the subgroup decision problem. Next, we show that if h is chosen from G then the identity of a signer is perfectly hidden.

Theorem 4.1. *Suppose no t-time adversary can solve the subgroup decison problem with advantage at least ϵ_{sd}. Then for every t'-time adversary $\mathcal{A}$ where $t' \approx t$ we have that* $\mathrm{Adv}_A < 2\epsilon_{\mathrm{sd}}$.

We first introduce a hybrid game H_1 in which the public parameters are the same as in the original game except that h is chosen randomly from G instead of G_q. We denote the adversary's advantage in this game as Adv_{A,H_1}.

Lemma 4.2. *For all t'-time adversaries as above,* $\mathrm{Adv}_A - \mathrm{Adv}_{A,H_1} < 2\,\epsilon_{\mathrm{sd}}$.

Proof. Consider an algorithm $\mathcal{B}$ that plays the subgroup decision problem. Upon receiving a subgroup decision challenge (n, G, G_T, e, w) the algorithm $\mathcal{B}$ first creates public parameters for our scheme by setting $h = w$ and choosing all other parameters as the scheme does. It then sends the parameters to $\mathcal{A}$ and plays the anonymity game with it. If w is randomly chosen from G_q then the adversary is playing the normal anonymity game, otherwise, is w is chosen randomly from G then it plays the hybrid game H_1. The algorithm $\mathcal{B}$ will be able to answer all chosen plaintext queries—namely, issue private signing keys for, and sign any message by, any user—, since it knows the master key.

At some point the adversary will choose a message M and two identities ID_1 and ID_2 it wishes to be challenged on (under the usual constraints that it had not previously made a signing key query on ID_x or a signature query on $\mathsf{ID}_x.M$). The simulator $\mathcal{B}$ will create a challenge signature on M, and $\mathcal{A}$ will guess the identity of the signer. If $\mathcal{A}$ answers correctly, then $\mathcal{B}$ outputs $b = 1$, guessing $w \in G_q$; otherwise it outputs $b = 0$, guessing $w \in G$.

Denote by Adv_B the advantage of the simulator $\mathcal{B}$ in the subgroup decision game. As we know that $\Pr[w \in G] = \Pr[w \in G_q] = \frac{1}{2}$, we deduce that,

$$\begin{aligned}\mathrm{Adv}_A - \mathrm{Adv}_{A,H_1} &= \Pr[b = 1 | w \in G_q] - \Pr[b = 1 | w \in G] \\ &= 2 \Pr[b = 1, w \in G_q] - 2 \Pr[b = 1, w \in G] = 2 \,\mathrm{Adv}_B < 2\, \epsilon_{\mathrm{sd}},\end{aligned}$$

since by our hardness assumption Adv_B must be lesser than ϵ_{sd}, given that $\mathcal{B}$ runs in time $t \approx t'$. □

Lemma 4.3. *For any algorithm $\mathcal{A}$, we have that* $\mathrm{Adv}_{A,H_1} = 0$.

Proof. We must argue that when h is chosen uniformly at random from G, instead of G_q in the real scheme, then the challenge signature is statistically independent of the signer identity, ID, in the adversary's view (which might comprise answers to earlier signature queries on ID). Consider the challenge signature,

$$\sigma = \big(\ \sigma_1, \sigma_2, \sigma_3,\ c_1, \ldots, c_k,\ \pi_1, \ldots, \pi_k\ \big),$$

and let us determine what such an adversary might deduce from σ.

First, observe that σ_2 and σ_3 by themselves do not depend on (any of the bits κ_i comprising) the signer identity ID. However, since the adversary is computationally unbounded, we must assume that they reveal s_1 and s_2.

Next, consider $c_i = u_i^{\kappa_i} h^{t_i}$ and the corresponding $\pi_i = (u_i^{2\kappa_i - 1} h^{t_i})^{t_i} = (u_i^{\kappa_i} u_i^{\kappa_i - 1} h^{t_i})^{t_i}$ for each i. There are two competing hypotheses that may be formulated by the adversary: $\kappa_i = 0 \ \vee\ \kappa_i = 1$. For either hypothesis, there is a solution for the ephemeral exponent t_i that explains the observed value of c_i. In other words, in the adversary's view,

$$\forall i \in \{1, \ldots, k\}\ :\ \exists \tau_0, \tau_1 \in \mathbb{Z}_n \quad \text{s.t.} \quad (\kappa_i, t_i) = (0, \tau_0)\ \vee\ (\kappa_i, t_i) = (1, \tau_1) \quad \text{and} \quad c_i = h^{\tau_0} = u_i h^{\tau_1}.$$

Using the last equality we find that the observed value of π_i is compatible with both hypotheses:

$$\big(\pi_i\big|_{\kappa_i=0}\big) = (u_i^{-1} h^{\tau_0})^{\tau_0} = (u_i^{-1} u_i h^{\tau_1})^{\tau_0} = h^{\tau_0 \tau_1} = (h^{\tau_0})^{\tau_1} = (u_i h^{\tau_1})^{\tau_1} = \big(\pi_i\big|_{\kappa_i=1}\big).$$

This all means that, the knowledge of c_i and π_i does not disambiguate the relevant bit $\kappa_i \in \{0, 1\}$. Taken together, all the c_i and π_i do not reveal anything about ID.

Last, we consider $\sigma_1 = g^{\alpha} \cdot c^{s_1} \cdot V^{s_2}$. But this value is just redundant in the eyes of the adversary, since he already knows all the values that determine it, including $\alpha = \log_{e(g,g)} A$.

Therefore, ID is statistically independent of the entire signature σ, which proves the lemma. □

4.2 Full Traceability

We show how to reduce the full traceability of our scheme to the two-level signature scheme described in Section 3.1. We create a simulator that will interact with a challenger for the security game of the Waters signature scheme. If the adversary asks for the secret key of user ID, the simulator will simply ask for a first-level signature on ID and give this to the adversary. If the adversary asks for a signature of message M by user ID the simulator will ask the challenger for a second-level signature on $\mathsf{ID}.M$ and then blind the signature itself.

The adversary will finally output a signature σ^* on some message M^*. In order for the adversary to be successful the signature will need to verify. By the perfect binding properties of the underlying NIZK techniques, a signature can be traced to some user ID^* and we can recover from it a Waters two-level signature on $\mathsf{ID}^*.M^*$; the simulator will submit this as its forgery in its own attack against the underlying signature scheme. The adversary will only be considered successful if he had not asked for the private key of user ID^* and had not queried for a signature on M^* from user ID^*. However, these are precisely the conditions that the simulator needs to abide by to be sucessful in its own game.

One tricky point in our reduction is that the simulator will play the signature game in the subgroup G_p, however the parameters for the group signature scheme are to be given in the group G, and so will be the forgery produced by the adversary. In addition, we note that the adversary is effectively given the factorization $n = pq$, as required by the full anonymity security definition which demands that the tracing key $\mathsf{TK} = q$ be disclosed for this attack. Our formal reduction follows.

Theorem 4.4. *If there exists a (t, ϵ) adversary for the full tracing game then there exists a $(\tilde{t}, \epsilon)$ UF-CMA adversary against the two-level signature scheme, where $t \approx \tilde{t}$.*

Proof. Suppose there exists an algorithm $\mathcal{A}$ that is succesful in the tracing game of our group signature scheme with advantage ϵ. Then we can create a simulator $\mathcal{B}$ that existentially forges signatures in an adaptive chosen message attack against the two-level signature scheme, with advantage ϵ.

The simulator will be given the factorization $n = pq$ of the group order $|G| = n$. As usual, denote by G_p and G_q the subgroups of G of respective order p and q, and by analogy let G_{Tp} and G_{Tq} be the subgroups of G_T of order p and q. The simulator begins by receiving from its challenger the public parameters of the signature game, all in subgroups of order p,

$$\tilde{\mathsf{PP}} = \Big(\ \tilde{g} \ , \tilde{u}' = \tilde{g}^{y'}, \ \tilde{u}_1 = \tilde{g}^{y_1}, \ldots, \ \tilde{u}_k = \tilde{g}^{y_k}, \\ \tilde{v}' = \tilde{g}^{z'}, \ \tilde{v}_1 = \tilde{g}^{z_1}, \ldots, \ \tilde{v}_m = \tilde{g}^{z_m}, \ \tilde{A} = e(\tilde{g}, \tilde{g})^{\alpha} \ \Big) \in G_p^{k+m+3} \times G_{Tp}.$$

The simulator then picks random generators $(f, h, \gamma', \gamma_1, \ldots, \gamma_k, \nu', \nu_1, \ldots, \nu_m) \in G_q^{k+m+4}$ and a random exponent $\beta \in \mathbb{Z}_q$. The simulator publishes the group signature public parameters as,

$$\mathsf{PP} = \Big(\ g = \tilde{g}\, f \ , \ h \ , \ u' = \tilde{u}'\, \gamma' \ , \ u_1 = \tilde{u}_1\, \gamma_1 \ , \ldots, \ u_k = \tilde{u}_k\, \gamma_k \ , \\ v' = \tilde{v}'\, \nu' \ , \ v_1 = \tilde{v}_1\, \nu_1 \ , \ldots, \ v_m = \tilde{v}_k\, \nu_m \ , \ A = \tilde{A} \cdot e(f, f)^{\beta} \ \Big).$$

The distribution of the public key is the same is in the real scheme. The simulator also gives the tracing key $\mathsf{TK} = q$ to the adversary.

Suppose the adversary asks for the private key of user ID. To answer the query, the simulator first asks the challenger for a first-level signature on message ID, and receives back $\tilde{K}_{\mathsf{ID}} = (\tilde{K}_1, \tilde{K}_2) \in G_p^2$. As before, κ_i denotes the i-th bit of ID. The simulator then chooses a random $r \in \mathbb{Z}_q$ and creates the requested key as,

$$K_{\mathsf{ID}} = \Big(\ K_1 = \tilde{K}_1 \cdot f^{\beta} \cdot (\gamma' \prod_{i=1}^{k} \gamma_i^{\kappa_i})^r \ , \ \ K_2 = \tilde{K}_2 \cdot f^{-r} \ , \ \ K_3 = h^{-r} \ \Big).$$

This is a well formed private key in our scheme.

Suppose the simulator is asked for a signature on message $M = (\mu_1 \ldots \mu_m) \in \{0,1\}^m$ by user $\mathsf{ID} = (\kappa_1 \ldots \kappa_k) \in \{0,1\}^k$. The simulator starts as in the real scheme, by choosing random

$t_1, \ldots, t_k \in \mathbb{Z}_n$, defining $t = \sum_{i=1}^{k} t_i$, and creating the values $c_i = u_i^{\kappa_i} \cdot h^{t_i}$ and $\pi_i = (u_i^{2\kappa_i - 1} \cdot h^{t_i})^{t_i}$ for all $i = 1, \ldots, k$. Next, the simulator requests a two-level signature on $\mathsf{ID}.M$ and receives in return $S = (S_1, S_2, S_3) \in G_p^3$. It then chooses random $r_1, r_2 \in \mathbb{Z}_q$ and creates the remaining components,

$$\sigma_1 = S_1 \cdot f^{\beta} \cdot (\gamma' \prod_{i=1}^{k} \gamma_i^{\kappa_i})^{r_1} \cdot (\nu' \prod_{i=1}^{m} \nu_i^{\mu_i})^{r_2} \cdot h^{r_1 t}, \qquad \sigma_2 = S_2 \cdot f^{-r_1}, \qquad \sigma_3 = S_3 \cdot f^{-r_2}.$$

The simulator gives the full signature $\sigma = (\sigma_1, \sigma_2, \sigma_3, c_1, \ldots, c_k, \pi_1, \ldots, \pi_k)$ to the adversary. Again, this is a well-formed signature in our scheme.

Finally, the adversary gives the simulator a forgery $\sigma^* = (\sigma_1, \sigma_2, \sigma_3, c_1, \ldots, c_k, \pi_1, \ldots, \pi_k)$ on message $M^* = (\mu_1 \ldots \mu_m)$. The simulator first checks that the signature verifies, otherwise the adversary is not successful and the simulator can abort. Next, it sets out to trace the identity, ID^*, of the forgery. Let κ_i denote the i-th bit of the string ID^* that is to be determined. For each $i = 1, \ldots, k$, the tracer sets $\kappa_i = 0$ if $(c_i)^q = g^0$, and $\kappa_i = 1$ otherwise. It then reconstitutes $\mathsf{ID}^* = (\kappa_1 \ldots \kappa_k)$. If either the key for ID^* or a signature on M^* by ID^* was previously requested by the adversary, the simulator can safely abort since the adversary was not successful. Otherwise the adversary was successful and the simulator must produce its own forgery.

To see how, recall that for all i we have that $e(c_i, u_i^{-1} c_i) = e(h, \pi_i)$, which has order q in G_T. Therefore, either $c_i \in G_q$ or $c_i\, u_i^{-1} \in G_q$. It follows that $c_i = u_i^{\kappa_i} f^{r'_i}$ for the previously determined $\kappa_i \in \{0, 1\}$ for some unknown r'_i, and therefore, that $c = u' \prod_{i=1}^{k} c_i = (\tilde{u}' \prod_{i=1}^{k} \tilde{u}_i^{\kappa_i}) f^{r'}$ for some r'. Let then $\delta \in \mathbb{Z}_n$ be an integer which is 0 $\pmod q$ and 1 $\pmod p$. The verification equation entails,

$$e(\sigma_1^{\delta}, \tilde{g}) \cdot e(\sigma_2^{\delta}, \tilde{u}' \prod_{i=1}^{k} \tilde{u}_i^{\kappa_i}) \cdot e(\sigma_3^{\delta}, \tilde{v}' \prod_{j=1}^{m} \tilde{v}_j^{\mu_j}) = A^{\delta} = e(\tilde{g}, \tilde{g})^{\alpha\delta}\, e(f, f)^{\beta\delta} = e(\tilde{g}, \tilde{g})^{\alpha} = \tilde{A}.$$

This, however, leaves $S^* = (\sigma_1^{\delta}, \sigma_2^{\delta}, \sigma_3^{\delta}) \in G_p^3$ as the sought forgery on $\mathsf{ID}^*.M^*$ in the underlying hierarchical signature scheme, which the simulator gives to the challenger. Therefore, our simulator will be successful whenever the adversary is. □

5 Extensions

Our framework of creating group signature schemes from hierarchical signature schemes allows us to extend our basic scheme in some interesting ways. We outline a few of these applications in the present section.

5.1 Fast Verification

Perhaps the main drawback of our scheme in terms of practicality, is that, taken at face value, signature verification requires $2k + 3$ pairing computations. However, in all known realizations of the pairing, it turns out that computing a product of pairings is almost as fast as computing a single pairing, thanks to a device akin to multi-exponentiation. Thus, the three pairings in the product in the last verification equation count as one. Furthermore, it is possible to batch the k remaining equations into a single "multi-pairing", using randomization, at the cost of k exponentiations in G: to check that $\forall i = 1, \ldots, k : e(c_i, u_i^{-1} c_i) = e(h, \pi_i)$, the verifier would pick $r_1, \ldots, r_k \in \mathbb{Z}_n$, and test,

$$\prod_{i=1}^{k} \Big(e(c_i^{r_i}, u_i^{-1} c_i) \cdot e(h^{-r_i}, \pi_i) \Big) \stackrel{?}{=} 1.$$

Probabilistic signature verification can thus be performed with a total of 2 multi-pairings and k exponentiations for the c^{r_i}. Notice that since h is constant across all signers for the life of the system, the h^{-r_i} can be computed comparatively very quickly using a few amortized pre-computations.

5.2 Long Messages

Once we have a signature scheme that can sign messages in $\{0,1\}^m$ for large enough $m = \Theta(\lambda)$, it is easy to sign arbitrary messages with the help of a Universal One-Way Hash Function (UOWHF) family $\mathcal{H}$, a description of which is added to the public key. To sign $M \in \{0,1\}^*$, first pick a random index h into the family, which determines a function $H_h \in \mathcal{H}$. Next, let $M' = h \| H_h(M)$, and compute $\sigma' = \mathit{Sign}(\mathsf{PP}, \mathsf{ID}, K_{\mathsf{ID}}, M')$, a signature on M' in the initial scheme. The signature on M is then given by $\sigma = (h, \sigma')$.

Since $|h|$ and $|H_h(M)|$ both grow linearly in the security parameter, it suffices to let $m = \Theta(\lambda)$ with a constant factor large enough to accommodate the two. A standard argument shows that the new scheme is existentially unforgeable under adaptive chosen message attacks whenever the old one was. Also, it is easy to see that this transformation does not affect anonymity or tracing, since it operates only on the message, and does so in a "public" way.

5.3 Delegation

Using a hierarchical signature scheme we can allow for a group signature scheme where a signer can delegate its authority down in a hierarchical manner. Suppose we have an $(\Lambda+1)$-level hierarchical signature scheme where at each level identities can be at most d bits long (except the last level, which must support messages of sufficient size m, as discussed above). Then we can extend the techniques from our basic scheme to create a new group signature scheme that allows for hierarchical identities of up to Λ levels, where someone with an identity at level l can delegate down to a new user at level $l+1$.

To do this we simply extend our scheme to hide identities at all levels. However, this will come with an $O(\Lambda\, d + m)$ cost in signing time, verification time, and signature size.

5.4 Revocation

Keys can be revoked in any group signature scheme in a very generic manner, in which the group master sends a revocation message linear in the number of remaining signers. Upon enrollment, each user is assigned an additional, unique, long-lived decryption key. Then, to revoke a user, the group master would re-key the group signature sub-system, and form a public revocation message that contains the new public key as well as the signing key of each remaining user encrypted under that user's long-term key. (Alternatively, the group master could broadcast a constant size revocation message containing only the new PK, and privately communicate a new key to each signer.)

Using an extension of our methods we can have a constant size revocation message along with an $O(r)$ overhead for our group signature scheme, where r is the number of revoked users. Essentially, the idea is for the signers to attach an additional proof for each revoked user that they are not that user.

These two techniques can be used in conjunction. Most revocation messages can be kept short using the second technique. However, when the number of revoked users becomes too large, the group master issues out a long revocation message to re-key the system.

6 Conclusion

In this paper we presented the first efficient group signature scheme that is provably secure without random oracles, based on bilinear maps. We built our group signature scheme from the Waters two-level hierarchical signatures scheme, where the first level is the identity of the signer and the second level is the signed message. Additionally, we applied the recent NIZK proof techniques of Groth, Ostrovsky, and Sahai in a novel manner to hide the identity of the signer.

We proved the security of our scheme using the subgroup decision and the computational Diffie-Hellman assumptions. Its signing time, verification time, and signature size are all logarithmic in the number of signers.

Our method of using a hierarchical signature scheme allowed us to create clean, modular proofs of security. Additionally, it had the added benefit of allowing for a hierarchical identity structure. We expect our new framework of creating group signatures to enable many other extensions in the future.

Acknowledgements

We thank Dawn Song for suggesting the concept of signature delegation and Dan Boneh for useful comments and suggestions.

References

[ACJT00] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Proceedings of Crypto 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–70. Springer-Verlag, 2000.

[AT99] G. Ateniese and G. Tsudik. Some open issues and directions in group signatures. In *Proceedings of Financial Cryptography 1999*, volume 1648 of *Lecture Notes in Computer Science*, pages 196–211. Springer-Verlag, 1999.

[ATS02] Giuseppe Ateniese, Gene Tsudik, and Dawn Song. Quasi-efficient revocation of group signatures. In *Proceedings of Financial Cryptography 2002*, 2002.

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–38. Springer-Verlag, 2004.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology—CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer-Verlag, 2004.

[BF03] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003. Extended abstract in *Advances in Cryptology—CRYPTO 2001.*

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Proceedings of TCC 2005*, Lecture Notes in Computer Science. Springer-Verlag, 2005.

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Advances in Cryptology—EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–29. Springer-Verlag, 2003.

[BP97] Niko Baric and Birgit Pfitzman. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology—EUROCRYPT 1997*, Lecture Notes in Computer Science, pages 480–94. Springer-Verlag, 1997.

[BS04] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In *Proceedings of ACM CCS 2004*, pages 168–77. ACM Press, 2004.

[BSZ04] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. Cryptology ePrint Archive, Report 2004/077, 2004. `http://eprint.iacr.org/`.

[CL02] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology—CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 61–76. Springer-Verlag, 2002.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology—CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.

[CvH91] David Chaum and Eugène van Heyst. Group signatures. In *Advances in Cryptology—EUROCRYPT 1991*, volume 547 of *Lecture Notes in Computer Science*, pages 257–65. Springer-Verlag, 1991.

[GOS05] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for np. Cryptology ePrint Archive, Report 2005/290, 2005. `http://eprint.iacr.org/`.

[KY04] Aggelos Kiayias and Moti Yung. Group signatures: Provable security, efficient constructions and anonymity from trapdoor-holders. Cryptology ePrint Archive, Report 2004/076, 2004. `http://eprint.iacr.org/`.

[LRSW99] Anna Lysyanskaya, Ron Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In *Proceedings of SAC 1999*, volume 1758 of *Lecture Notes in Computer Science*, pages 184–99. Springer-Verlag, 1999.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology—EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*. Springer-Verlag, 2005.

A Simple Hierarchical Signatures: Reduction from CDH

We reduce the CDH problem to the existential forgery of the signature scheme of Section 3.1.

Theorem A.1. *Consider an adversary $\mathcal{A}$ that existentially forges the simple two-level signature scheme in an adaptive chosen message attack. Assume that $\mathcal{A}$ makes no more that $\ell \ll p$ signature queries and produces a successful forgery with probability ϵ in time t. Then there exists an algorithm $\mathcal{B}$ that solves the CDH problem in G_p with probability $\tilde{\epsilon} \geq \epsilon/(2^{k+2}m\ell)$ in time $\tilde{t} \approx t$.*

Proof. The simulator $\mathcal{B}$ is given an instance $(g, g^a, g^b) \in G_p^3$ of the CDH problem, and must produce g^{ab}. To prepare the simulation, $\mathcal{B}$ first selects a random identity $\mathsf{ID}^* = (\kappa_1^* \ldots \kappa_k^*) \in \{0,1\}^k$, and picks random exponents $y', y_1, \ldots, y_k \in \mathbb{Z}_p$ under the constraint that $y' + \sum_{i=1}^k \kappa_i^* y_i \equiv 0 \pmod{p}$. Next, $\mathcal{B}$ chooses a random $k \in \{0, \ldots, m\}$, and random numbers $x', x_1, \ldots, x_m$ in the interval $\{0, \ldots, 2\ell - 1\}$. It also chooses additional random exponents $z', z_1, \ldots, z_m \in \mathbb{Z}_p$. Last, it picks $t \in \mathbb{Z}_p$ and lets $g_1 = g^a$ and $g_2 = g^b$. The simulator gives to $\mathcal{A}$ the public key,

$$\mathsf{PP} = \Big(g,\ u' = g_2^{y'} g^t,\ \big(u_i = g_2^{y_i}\big)_{i=1,\ldots,k},\ v' = g_2^{x'-2k\ell} g^{z'},\ \big(v_j = g_2^{x_j} g^{z_j}\big)_{j=1,\ldots,m},\ A = e(g^a, g^b) \Big).$$

The corresponding master key, $\mathsf{MK} = g^{ab}$, is unknown to $\mathcal{B}$.

To answer a signature query on ID or $\mathsf{ID}.M$, the simulator $\mathcal{B}$ proceeds as follows. Let $\mathsf{ID} = (\kappa_1 \ldots \kappa_k)$ and $H = y' + \sum_{i=1}^k \kappa_i y_i$. We have that $H \equiv 0 \pmod{p}$ iff $\mathsf{ID} = \mathsf{ID}^*$, except with negligible probability over the choice of y' and y_i.

If $H \not\equiv 0 \pmod{p}$, and thus, $\mathsf{ID} \neq \mathsf{ID}^*$, the simulator picks a random $r \in \mathbb{Z}_p$ and computes the user's private key as,

$$K_{\mathsf{ID}} = (K_1, K_2) = \Big(\ g_1^{-t/H}(u' \prod_{i=1}^k u_i^{\kappa_i})^r,\ \ g_1^{1/H} g^{-r}\ \Big).$$

For $\tilde{r} = r - a/H$, we have that $K_1 = g_1^{-t/H}(g^t g_2^H)^r = g_1^{-t/H} g^{at/H} g_2^a (g^t g_2^H)^{\tilde{r}} = g^{ab}(u' \prod_{i=1}^k u_i^{\kappa_i})^{\tilde{r}}$, and also $K_2 = g_1^{1/H} g^{a/H} g^{-\tilde{r}} = g^{-\tilde{r}}$, which shows that K_{ID} has the correct distribution. To respond to the query, if the signature request was on ID, the simulator simply outputs K_{ID}; otherwise, $\mathcal{B}$ uses K_{ID} to create a signature on M exactly as in the actual scheme, and outputs the result.

If $H \equiv 0 \pmod{p}$, and the request was on ID^* only, $\mathcal{B}$ is unable to compute K_{ID} and must abort the simulation. If instead the request was on $\mathsf{ID}^*.M$ for some $M = (\mu_1 \ldots \mu_m)$, we define $F = -2k\ell + x' + \sum_{j=1}^m x_j \mu_j$ and $J = z' + \sum_{j=1}^m z_j \mu_j$. If $F \equiv 0 \pmod{p}$, then $\mathcal{B}$ cannot produce the requested signature and must also abort in this case. Otherwise, $\mathcal{B}$ picks random $r, s \in \mathbb{Z}_p$, and returns the signature,

$$S = (S_1, S_2, S_3) = \Big(\ g_1^{-J/F} g^{tr} (v' \prod_{j=1}^m u_j^{\mu_j})^s,\ \ g^{-r},\ \ g_1^{1/F} g^{-s}\ \Big).$$

For $\tilde{s} = s - a/F$, we have that $S_1 = g_1^{-J/F} g^{tr} (g^J g_2^F)^s = g_1^{-J/F} g^{tr} g^{aJ/F} g_2^a (g^J g_2^F)^{\tilde{s}} = g^{ab} g^{tr} (g^J g_2^F)^{\tilde{s}} = g^{ab}(u' \prod_{i=1}^k u_i^{\kappa_i})^r (v' \prod_{j=1}^m u_j^{\mu_j})^{\tilde{s}}$, and similarly, $S_3 = g_1^{1/F} g^{a/F} g^{-\tilde{s}} = g^{-\tilde{s}}$, which shows that S has the correct distribution.

Eventually, the adversary outputs a valid forgery $S^* = (S_1^*, S_2^*, S_3^*)$ on $\mathsf{ID}.M^*$ where $M^* = (\mu_1^* \ldots \mu_m^*) \in \{0,1\}^m$. Let $F^* = -2k\ell + x' + \sum_{j=1}^m x_j \mu_j^*$ and $J^* = z' + \sum_{j=1}^m z_j \mu_j^*$. If $\mathsf{ID} \neq \mathsf{ID}^*$ or if $F^* \not\equiv 0 \pmod{p}$, the simulator aborts. Otherwise, the forgery must be of the form, for some $r^*, s^* \in \mathbb{Z}_p$,

$$S^* = \Big(\ g^{ab}(u' \prod_{i=1}^k u_i^{\kappa_i^*})^{r^*} (v' \prod_{j=1}^m u_j^{\mu_j^*})^{s^*},\ \ g^{-r^*},\ \ g^{-s^*}\ \Big) = \big(g^{ab+tr^*+J^*s^*}, g^{-r^*}, g^{-s^*}\big) = (S_1^*, S_2^*, S_3^*).$$

To solve the CDH instance, algorithm $\mathcal{B}$ outputs $(S_1^*) \cdot (S_2^*)^t \cdot (S_3^*)^{J^*} = g^{ab}$.

To conclude, we bound the probability that $\mathcal{B}$ completes the simulation without aborting. First, the guess on the target identity ID^* must be correct, which has probability 2^{-k}. Notice that if $\mathcal{B}$ guessed ID^* correctly, it will never have to abort on a first-level signature query on ID^*. Second, for each second-level query where $\mathsf{ID} = \mathsf{ID}^*$, the conditional probability that $F \not\equiv 0 \pmod p$ given that we have made it so far is at least $1 - 1/(2\ell)$; and since there are at most ℓ such queries, the total probability of not aborting through all of them is greater than $1/2$. Last, the conditional probability that $F^* \equiv 0 \pmod p$ in the challenge stage is at least $1/(2m\ell)$. Since $\mathcal{A}$ succeeds with probability ϵ when the simulation is allowed to finish, and since this probability is independent of the random choices made by $\mathcal{B}$, we conclude that $\mathcal{B}$ succeeds with probability $\tilde{\epsilon} \geq \epsilon/(2^{k+2}m\ell)$. $\square$