Identity-Based Aggregate Signatures

Craig Gentry[1,⋆] and Zulfikar Ramzan[2]

[1] Stanford University
cgentry@cs.stanford.edu
[2] DoCoMo Communications Laboratories USA, Inc.
ramzan@docomolabs-usa.com

Abstract. An *aggregate signature* is a single short string that convinces any verifier that, for all $1 \leq i \leq n$, signer S_i signed message M_i, where the n signers and n messages may all be distinct. The main motivation of aggregate signatures is compactness. However, while the aggregate signature itself may be compact, aggregate signature verification might require potentially lengthy additional information – namely, the (at most) n distinct signer public keys and the (at most) n distinct messages being signed. If the verifier must obtain and/or store this additional information, the primary benefit of aggregate signatures is largely negated.

This paper initiates a line of research whose ultimate objective is to find a signature scheme in which the *total information needed to verify* is minimized. In particular, the verification information should preferably be as close as possible to the theoretical minimum: the complexity of describing which signer(s) signed what message(s). We move toward this objective by developing *identity-based* aggregate signature schemes. In our schemes, the verifier does not need to obtain and/or store various signer public keys to verify; instead, the verifier only needs a description of who signed what, along with two constant-length "tags": the short aggregate signature and the single public key of a Private Key Generator. Our scheme is secure in the random oracle model under the computational Diffie-Hellman assumption over pairing-friendly groups against an adversary that chooses its messages and its target identities adaptively.

1 Introduction

Authentication is crucial for many cryptographic applications. Improving the performance of building blocks, like digital signatures, that provide a means for authentication is therefore an essential goal. While time complexity is a well-known traditional measure for evaluating performance, communication complexity is becoming increasingly important for two reasons. First, consider wireless devices (e.g., PDAs, cell phones, RFID chips, and sensors). Here battery life is often more of a limiting bottleneck than processor speed. Communicating a single bit of data consumes several orders of magnitude more power than executing a basic 32-bit arithmetic instruction [BA05]. Second, consider wireless network

[⋆] This research was conducted while the author was at DoCoMo Labs, USA scenarios (e.g., MANETS, cellular networks, tactical networks, and sensor nets). Here reliable bandwidth may be more of a limiting factor than computation. In these cases it would be preferable to limit the communication requirements (i.e., the size) of a digital signature. An *aggregate signature* is one technique towards achieving this aim.

Aggregate Signatures. In an aggregate signature scheme [BGLS03], multiple signatures can be aggregated into a compact "aggregate signature," even if these signatures are on (many) different documents and were produced by (many) different signers. This is useful in many real-world applications. For example, certificate chains in a hierarchical PKI of depth n consist of n signatures by n different CAs on n different public keys; by using an aggregate signature scheme, this chain can be compressed down to a single aggregate certificate. Another application is secure routing. In Secure BGP [KLS00], each router successively signs its segment of a path in the network, and forwards the collection of signatures associated with the path to the next router; forwarding these signatures entails a high transmission overhead that could be reduced by using aggregate signatures. Aside from compactness, aggregate signatures have other advantages. For example, in scenarios such as database outsourcing [MNT04] and dynamic content distribution [SRF$^+$04] one may want to prevent a malicious party from removing a signature from a collection of signatures without being detected. An aggregate signature scheme makes this possible, since a signature that has been aggregated cannot (under certain conditions) be separated.

Currently, two aggregate signature schemes exist. The first [BGLS03] uses bilinear maps and supports flexible aggregation – i.e., given n individual signatures $\sigma_1, \ldots, \sigma_n$, *anyone* can aggregate them in *any order* into an aggregate signature σ. The second [LMRS04] uses a weaker assumption – namely, certified trapdoor permutations – but it permits only *sequential* aggregation – i.e., the n-th signer must aggregate its own signature into the aggregate signature formed by the first $n-1$ signers.

For both schemes above, the aggregate signature is compact (i.e., its size is independent of n). However, the total information $\mathcal{V}$ needed to verify the aggregate signature – namely, the aggregate signature itself, the public keys of the individual signers, and a description of the respective messages that they signed – is not necessarily compact at all. Of course, $\mathcal{V}$ must (information-theoretically) contain a description $\mathcal{D}$ of what signer signed what message, since the verification information must *convince* the verifier that certain signers signed certain messages. But $|\mathcal{D}|$ can grow slowly with the number of individual signatures n; e.g., in a routing application, one can use IP addresses as identities, and we can reduce communication further since the higher-order bits of the IP addresses of consecutive routers may be identical, so only need to be transmitted once.

Beyond this information-theoretic minimum, however, $\mathcal{V}$ in current aggregate signature schemes must also contain individual signer public keys, whose length is dictated by the security parameter of the signature scheme (not by basic information-theoretic considerations). Theoretically, this means that $|\mathcal{V}| - |\mathcal{D}|$ grows linearly with n. Practically speaking, this means that current aggregate signature schemes may not perform significantly better than traditional signature schemes in situations where verifiers cannot be expected to already have the signers' public keys – e.g., in a dynamic multi-hop network in which a node is unlikely to have a prior relationship with a neighboring node. Clearly, it would be preferable if $\mathcal{V}$ could specify the signers by their identities rather than by their individual public keys.

Identity-Based Signatures. In identity-based cryptography (IBC) [Sha84], the central idea is to simplify public-key and certificate management by using a user's "identity" (e.g., its email address) as its public key. For this to be possible, the IBC system requires a trusted third party, typically called a "Private Key Generator" (PKG), to generate user private keys from its "master secret" and the user's identity. Only the PKG has a traditional "random-looking" public key. In an identity-based encryption (IBE) scheme, the sender encrypts a message using the recipient's identity and the PKG's public key; it need not obtain the recipient's public key and certificate before encrypting, since the recipient has no traditional public key and since the sender knows that the recipient (or an attacker) will not be able to decrypt unless it has received an identity-based private key from the PKG (in effect, an implicit certificate). In an identity-based signature (IBS) scheme, the verifier verifies a signature by using the signer's identity and PKG's public key; the verification information does not include any certificate or any individual public key for the signer.

Research on IBS has experienced a revival in the wake of the discovery – independently by Boneh and Franklin [BF03] and by Cocks [Coc01] – of practical IBE schemes. (Early schemes include [Sha84,FS86,GQ88]; recent schemes and analyses include [CC03,Boy03,LQ04,BNN04].) Unfortunately, IBS does not have the significant infrastructural advantages over traditional public-key signing that IBE has over traditional public-key encryption. In IBE, the fact that the sender does not need to obtain the recipient's public key and certificate before encrypting means that no infrastructure (i.e., public-key infrastructure (PKI)) needs to be deployed to distribute such information to third parties (including non-clients); rather, the authority (the PKG) only needs infrastructure to distribute private keys directly to its clients. On the other hand, IBS and public-key signing (PKS) are analogous infrastructurally: in IBS (resp. PKS), the PKG (resp. CA) sends a private key (resp. certificate) to each client. Thus, the main advantage of IBS over PKS, at least abstractly, turns out to be communication-efficiency, since (unlike PKS) the signer does not need to send an individual public key and certificate with its signature.

This advantage of IBS becomes more compelling when we consider multiple signers, all of which are clients of the same PKG. In this setting, the verifier needs only one traditional public key (the PKG's) to verify multiple identity-based signatures on multiple documents. Unfortunately, current identity-based signatures are not aggregable. Interestingly, multiple-signer IBS therefore has precisely the opposite problem of aggregate signing: for IBS, the public key is (in some sense) aggregable, while the individual signatures are not.

Goals and Challenges. Our goal is simple: a signature scheme (allowing distinct signers to sign distinct documents) in which the total verification information is minimized. We cannot do better than the information-theoretic lower bound of describing who signed what, but we would like to get as close to this lower bound as possible.

Based on the above discussion, one natural approach is to construct an "identity-based aggregate signature" (IBAS) scheme – i.e., a scheme in which the verification information (apart from the required description of who signed what) consists only of a single aggregate signature and a single public key (of the PKG). In a sense, identity-based aggregate signatures would really address the motivating applications considered first in the context of regular (non ID-based) aggregate signatures.

However, there certainly does not appear to be any generic way of combining an aggregate signature scheme with an IBS scheme to achieve this desideratum. To see the difficulty, note that each of the current aggregate signature schemes are deterministic, and with good reason; if each successive signer contributed randomness to the aggregate signature in a trivial way, this randomness would cause the size of the signature to grow linearly with n – hence the signature would not be compact. On the other hand, identity-based signature schemes tend to be randomized; typically, the signer uses the Fiat-Shamir heuristic (which involves choosing a random commitment and treating the output of a hash function as the challenge to which the signer responds) to prove knowledge of the authority's signature on its identity. In short, current approaches for constructing aggregate signatures appear to be fundamentally at odds with current approaches for constructing identity-based signatures. To construct an IBAS scheme, it seems we must somehow find a way to "aggregate the randomness" provided by the multiple signers.

Our Results. Our first contribution is a formal definition of identity-based aggregate signatures and a corresponding formal security model. Second, we describe, as a stepping stone, an identity-based multi-signature scheme (which may be of independent interest). Third, we present a concrete IBAS scheme that meets our definition. As desired, our scheme allows multiple signers to sign multiple documents in such a way that the total verification information, apart from a description of who signed what, consists only of a short aggregate signature (which consists of only 2 group elements and a short (e.g., 160-bit) string) and the PKG's public key (which is also short – about the same size as the PKG's public key in Boneh-Franklin). Our scheme is also very efficient computationally. In fact, it allows more efficient verification than the aggregate signature scheme of [BGLS03], since verification requires only three pairing computations, regardless of the value of n, while [BGLS03] uses $\mathcal{O}(n)$ pairing computations. (Note: verification in our scheme uses $\mathcal{O}(n)$ elliptic curve scalar multiplications, but these can be computed quite quickly.) Later we describe certain extensions and additional benefits of our scheme.

Our scheme is provably secure in the random oracle model, assuming the hardness of computational Diffie-Hellman over groups with bilinear maps. In our security model, the adversary can make q_E adaptive key extraction queries (wherein he receives the signing key corresponding to any ID of his choice), q_S adaptive signature queries (wherein he receives the signature on any message of his choice), and q_H hash queries (wherein he receives the output of a hash function, modeled as a random oracle, on inputs of his choice). The adversary succeeds if he constructs a single non-trivial forgery. The concrete security loss in our scheme is roughly $q_E \cdot q_H \cdot q_S$. While one would prefer a smaller loss, it is worth noting that typical ID-based signature schemes usually suffer from a concrete security loss of roughly $q_E \cdot q_S$ because the simulator usually has to guess the ID and message that will be used in the forgery. We further note that such a quadratic loss is also inherent in schemes where security is proved using the forking lemma [PS96],[PS00]).

We remark that in our scheme all signers must use the same (unique) random string w when signing – this step seems necessary to enable signature aggregation. Choosing such a w may be straightforward in certain settings. For example, if the signers have access to loosely synchronized clocks, then w could be chosen based on the current time. Further, if w is sufficiently long (i.e., accounting for birthday bounds), then it will be statistically unique. In order to alleviate any cost incurred in choosing w, we describe a simple extension of our scheme that allows a signer to securely re-use the same w a constant number of times.

Aside from requiring a common value of w, aggregation in our scheme is very flexible. Anybody can aggregate individual identity-based signatures into an identity-based aggregate signature, and aggregate smaller aggregates into larger aggregates. Moreover, our scheme permits aggregation across multiple trusted authorities; i.e., signers under different PKGs can aggregate their signatures. As a stepping stone to IBAS, we also describe an identity-based multisignature (in which all signers sign the same message) that may be of independent interest.

Other Related Work. Aggregate signatures are related to, but more flexible than, multisignatures [Oka98,OO99,MOR01,Bol03]. Although the term "multisignature" has been used in the literature to denote a variety of different types of schemes, we will use the term to denote an aggregate signature in which all users sign the same message. Aggregate signatures are also tenuously related to threshold signatures [Sho00]. Recall that, in a threshold signature scheme, t signature components from any t signers can be combined into a single signature, for some threshold $t \leq n$. The signers must undergo a large setup cost, they all sign the same message, and the verifier cannot tell which signers contributed components to a complete threshold signature. Secure identity-based threshold signature schemes are known [BZ04].

Subsequent to our work, a recent paper claimed an ID-based aggregate signature construction [CLW05]. However, "identity-based aggregate signatures" may not be the best term to describe this result since each signer S_i that participates in the creation of a signature must first generate a random scalar r_i and *broadcast* r_iP (for a certain elliptic curve point P) to all of the other signers so that they can each compute $(\sum r_i)P$. Signer S_i then inputs $(\sum r_i)P$ and its message m_i into a hash function to obtain a signature scheme via the Fiat-Shamir heuristic.

Later, individual signatures can be aggregated. However, because of the large setup cost (in which the users essentially broadcast their key shares) and the fact that the signature cannot be verified until all of the signers contribute, this scheme actually bears some resemblance to an identity-based *threshold* signature scheme. Also subsequent to our work, Herranz [Her05] describes a Schnorr-based IBAS scheme that permits "partial" aggregation – that is, signatures can only be aggregated if they all come from the same signer.

ORGANIZATION OF THE PAPER. After providing some preliminaries in Section 2, we propose a definition of identity-based aggregate signatures in Section 3, together with the security model. Next, as a stepping stone to our IBAS construction, we give a simple identity-based multisignature scheme in Section 4. We provide our IBAS construction in Section 5 and describe the security proof in Section 6. Finally, we conclude and mention open problem in Section 7.

2 Preliminaries

Let λ denote the security parameter, which will be an an implicit input to the algorithms in our scheme. For a set S, we let $|S|$ denote the number of elements in S, and $x \overset{D}{\leftarrow} S$ denote the experiment of choosing $x \in S$ according to probability distribution D.

2.1 Bilinear Maps

Our IBAS scheme uses a bilinear map, which is often called a "pairing." Typically, the pairing used is a modified Weil or Tate pairing on a supersingular elliptic curve or abelian variety. However, we describe bilinear maps and the related mathematics in a more general format here.

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of some large prime order q. We write $\mathbb{G}_1$ additively and $\mathbb{G}_2$ multiplicatively.

Admissible pairings: We will call $\hat{e}$ an *admissible pairing* if $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is a map with the following properties:

1. Bilinear: $\hat{e}(aQ, bR) = \hat{e}(Q, R)^{ab}$ for all $Q, R \in \mathbb{G}_1$ and all $a, b \in \mathbb{Z}$.
2. Non-degenerate: $\hat{e}(Q, R) \neq 1$ for some $Q, R \in \mathbb{G}_1$.
3. Computable: There is an efficient algorithm to compute $\hat{e}(Q, R)$ for any $Q, R \in \mathbb{G}_1$.

Notice that $\hat{e}$ is also symmetric – i.e., $\hat{e}(Q, R) = \hat{e}(R, Q)$ for all $Q, R \in \mathbb{G}_1$ – since $\hat{e}$ is bilinear and $\mathbb{G}_1$ is a cyclic group.

2.2 Computational Assumptions

The security of our schemes is based on the assumed hardness of the computational Diffie-Hellman (CDH) problem in $\mathbb{G}_1$.

Definition 1 (Computational Diffie-Hellman Problem in $\mathbb{G}_1$ (CDH$_{\mathbb{G}_1}$ Problem)). *Given $P, aP, bP \in \mathbb{G}_1$, as well as an admissible pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, compute abP (for unknown randomly chosen $a, b \in \mathbb{Z}/q\mathbb{Z}$).*

An algorithm $\mathcal{A}$ has an advantage ϵ in solving CDH$_{\mathbb{G}_1}$ if $\Pr[\mathcal{A}(P, aP, bP) = abP] \geq \epsilon$, where the probability is over the choice of P in $\mathbb{G}_1$, the random scalars a and b in $\mathbb{Z}_q$, and the random bits used by $\mathcal{A}$. Our computational assumption is now formally defined as follows.

Definition 2 (Computational Diffie-Hellman Assumption in $\mathbb{G}_1$ (CDH$_{\mathbb{G}_1}$ Assumption)). *We say that the (t, ϵ)-CDH$_{\mathbb{G}_1}$ Assumption holds if no t-time algorithm $\mathcal{A}$ has advantage ϵ in solving the CDH$_{\mathbb{G}_1}$Problem.*

We may occasionally refer to the CDH$_{\mathbb{G}_1}$ Assumption without specifying t or ϵ. The CDH$_{\mathbb{G}_1}$ Assumption underlies the security of numerous cryptosystems (e.g., [BLS01,BGLS03,CC03]), and is weaker than other commonly-used assumptions relating to bilinear maps, such as the "Bilinear Diffie-Hellman" assumption used in Boneh-Franklin (given $P, aP, bP, cP \in \mathbb{G}_1$ and the bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, it is hard to compute $\hat{e}(P, P)^{abc} \in \mathbb{G}_2$).

3 Identity-Based Aggregate Signatures

We now define the procedures involved in an IBAS scheme, and thereafter specify what it means for IBAS scheme to be secure.

3.1 Components of an Identity-Based Aggregate Signature

An IBAS scheme is composed of five algorithms: key generation by the PKG, private key extraction by the PKG for individual users, signing by an individual user, aggregation of multiple individual signatures or aggregates of signatures, and verification of an identity-based aggregate signature:

- KeyGen takes 1^λ as input and outputs a suitable key pair (Pk,Sk).
- KeyExt takes Sk and a user identity ID_i as input and outputs a user private key USk_i.
- Sign takes USk_i, message m_i and possibly some state information w as input and outputs an individual identity-based signature σ_i.
- Agg takes as input Pk, w, two sets of identity-message pairs S_1 and S_2, and two identity-based (aggregate) signatures σ_{S_1} and σ_{S_2} on the identity-message pairs contained in sets S_1 and S_2 respectively; if $\mathsf{Verify}(\mathsf{Pk}, w, S_1, \sigma_{S_1}) = \mathsf{Accept}$ and $\mathsf{Verify}(\mathsf{Pk}, w, S_2, \sigma_{S_2}) = \mathsf{Accept}$, it outputs the identity-based aggregate signature $\sigma_{S_1 \cup S_2}$ on the identity-message pairs in $S_1 \cup S_2$ (where identity-message pairs may be repeated).
- Verify takes as input Pk, w, an identity-based aggregate signature σ_S, and a description of the identity-message pairs in set S, and outputs Accept if and only if σ_S could be a valid output of Sign or Agg for Pk, w and S.

Remark 1. Depending on the instantiation, the state information w may be empty. Also, it is possible that Sign and Agg may be inseparably combined into a single step in certain instantiations – e.g., if the IBAS scheme permits only *sequential* aggregation.

3.2 Security Model

An IBAS scheme should be secure against existential forgery under an adaptive-chosen-message and an adaptive-chosen-identity attack. Informally, existential forgery here means that the adversary attempts to forge an identity-based aggregate signature on identities and messages of his choice.

We formalize the identity-based aggregate signature model as follows. The adversary's goal is the existential forgery of an aggregate signature. We give the adversary the power to choose the identities on which it wishes to forge a signature, the power to request the identity-based private key on all but one of these identities, and the power to choose the state w used in its forgery. The adversary is also given access to a signing oracle on any desired identity. The adversary's advantage $\mathsf{AdvIBAS}_{\mathcal{A}}$ is defined as its probability of success (taken over the coin tosses of the key-generation algorithm and of $\mathcal{A}$) in the following game.

Setup: The adversary $\mathcal{A}$ is given the public key Pk of the PKG, an integer n, and any other needed parameters.

Queries: Proceeding adaptively, $\mathcal{A}$ may choose identities ID_i and request the private key USk_i. Also, $\mathcal{A}$ may request an identity-based aggregate signature σ_S on $(\mathsf{Pk}, w, S, \{m_i\}_{i=1}^{k-1})$ where $S = \{\mathsf{ID}_i\}_{i=1}^{k-1}$. We require that $\mathcal{A}$ has not made a query $(\mathsf{Pk}, w, S', \{m_i'\}_{i=1}^{k-1})$ where $ID_i \in S \cap S'$ and $m_i' \neq m_i$.

Response: For some $(\mathsf{Pk}, \{\mathsf{ID}_i\}_{i=1}^{l}, \{m_i\}_{i=1}^{l})$ for $l \leq n$, $\mathcal{A}$ outputs an identity-based aggregate signature σ_l.

$\mathcal{A}$ wins if σ_l is a valid signature on $(\mathsf{Pk}, \{\mathsf{ID}_i\}_{i=1}^{l}, \{m_i\}_{i=1}^{l})$, and the signature is nontrivial – i.e., for some i, $1 \leq i \leq l$, $\mathcal{A}$ did not request the private key for ID_i and did not request a signature including the pair (ID_i, m_i).

Definition 3. *An IBAS adversary $\mathcal{A}$ $(t, \epsilon, n, q_H, q_E, q_S)$-breaks an IBAS scheme in the above model if: for integer n as above, $\mathcal{A}$ runs in time at most t; $\mathcal{A}$ makes at most q_H hash function queries, at most q_E private key extraction queries and at most q_S signing oracle queries; and $\mathsf{AdvIBAS}_{\mathcal{A}}$ is at least ϵ.*

Definition 4. *An IBAS scheme is $(t, \epsilon, n, q_H, q_E, q_S)$-secure against existential forgery if no adversary $(t, \epsilon, n, q_H, q_E, q_S)$-breaks it.*

4 An Identity-Based Multisignature Scheme

Before presenting our construction of an IBAS scheme, we address, as a stepping stone, the simpler problem of constructing an identity-based ad-hoc multisignature scheme. In this scheme, all signers sign the same message, possibly in a completely decentralized fashion. Thereafter, *any subset* of the individual identity-based signatures on the message can be aggregated by anyone in any order. We use the term "ad hoc" to stress this flexibility.

Interestingly, the individual signatures in this identity-based multisignature scheme are very similar to one-level hierarchical identity-based signatures as presented by Gentry and Silverberg [GS02]. We modify their scheme slightly by hashing the message by itself, rather than together with the signer's identity, to enable aggregation; this makes our security reduction slightly looser. This construction will be instructive as to how one can "aggregate the randomness" provided by multiple signers. The scheme is as follows.

Setup: The Private Key Generator (PKG) generates parameters and keys essentially as in [GS02]. Specifically, it:

1. generates groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q with admissible pairing $\hat{e}\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$;
2. chooses an arbitrary generator $P \in \mathbb{G}_1$;
3. picks a random $s \in \mathbb{Z}/q\mathbb{Z}$ and sets $Q = sP$;
4. chooses cryptographic hash functions $H_1, H_2 : \{0,1\}^* \to \mathbb{G}_1$.

The PKG's public key is $(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, Q, H_1, H_2)$; its secret is $s \in \mathbb{Z}/q\mathbb{Z}$.

Private key extraction: The client with identity ID_i receives the value sP_i from the PKG as its private key, where $P_i = H_1(\mathsf{ID}_i) \in \mathbb{G}_1$.

Individual Signing: To sign m, the signer with identity ID_i:

1. computes $P_m = H_2(m) \in \mathbb{G}_1$;
2. generates random $r_i \in \mathbb{Z}/q\mathbb{Z}$;
3. computes its signature (S_i', T_i'), where $S_i' = r_i P_m + sP_i$ and $T_i' = r_i P$.

Aggregation: Anyone can aggregate a collection of individual signatures (on the same m) into a multisignature. In particular, individual signatures (S_i', T_i') for $1 \le i \le n$ can be aggregated into (S_n, T_n), where $S_n = \sum_{i=1}^n S_i'$ and $T_n = \sum_{i=1}^n T_i'$.

Verification: Let (S_n, T_n) be the multisignature (where n is the number of signers). The verifier checks that:

$$\hat{e}(S_n, P) = \hat{e}(T_n, P_m)\hat{e}(Q, \sum_{i=1}^{n} P_i) \ ,$$

where $P_i = H_1(\mathsf{ID}_i)$ and $P_m = H_2(m)$.

Notice how, although each of the individual identity-based signatures is randomized, the randomness is "aggregated" into the scalar coefficient of P_m, the element of $\mathbb{G}_1$ corresponding to the common message being signed. Notice also that aggregation is perfectly flexible. Users generate their signatures in a decentralized fashion; later, anyone can aggregate them. The users do not need to maintain any state. Verification requires only three pairing computations (and n point *additions*).

In the full version [GR06], we give a proof of the following theorem.

Theorem 1. *Let $\mathcal{A}$ be an adversary that $(t, \epsilon, n, q_E, q_S)$-breaks the identity-based multisignature scheme. Then, there exists an algorithm $\mathcal{B}$ that solves $CDH_{\mathbb{G}_1}$ in time $O(t) + O(\log^3 q)$ with probability at least $\epsilon(1 - 1/q)/64(q_E + q_S)^2$.*

5 Construction of an Identity-Based Aggregate Signature Scheme

In our identity-based multisignature scheme, we were able to aggregate the randomness contributed by the individual signers into the scalar coefficient of the common message point P_m. However, for IBAS, signers may sign distinct messages, and aggregating the signers' randomness seems difficult. Our solution to this problem, at a high level, is simply to create a "dummy message" w that is mapped to an element P_w of $\mathbb{G}_1$ whose scalar coefficient provides a place where individual signers can aggregate their randomness, and to embed messages into individual signatures using a different mechanism. The details follow.

Setup: The Private Key Generator (PKG) generates parameters and keys essentially as above. Specifically, it:

1. generates groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q and an admissible pairing $\hat{e}$: $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$;
2. chooses an arbitrary generator $P \in \mathbb{G}_1$;
3. picks a random $s \in \mathbb{Z}/q\mathbb{Z}$ and sets $Q = sP$;
4. chooses a cryptographic hash functions $H_1, H_2 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_3 : \{0,1\}^* \rightarrow \mathbb{Z}/q\mathbb{Z}$.

The system parameters are $params = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, Q, H_1, H_2, H_3)$. The root PKG's secret is $s \in \mathbb{Z}/q\mathbb{Z}$.

Private key generation: The client with identity ID_i receives from the PKG the values of $sP_{i,j}$ for $j \in \{0,1\}$, where $P_{i,j} = H_1(\mathrm{ID}_i, j) \in \mathbb{G}_1$.

Individual Signing: The first signer chooses a string w that it has never used before. Each subsequent signer checks that it has not used the string w chosen by the first signer. (Alternatively, different signers may arrive at the same w independently – e.g., if they issue signatures according to a pre-established schedule.) To sign m_i, the signer with identity ID_i:

1. computes $P_w = H_2(w) \in \mathbb{G}_1$;
2. computes $c_i = H_3(m_i, \mathrm{ID}_i, w) \in \mathbb{Z}/q\mathbb{Z}$;
3. generates random $r_i \in \mathbb{Z}/q\mathbb{Z}$;
4. computes its signature (w, S'_i, T'_i), where $S'_i = r_i P_w + sP_{i,0} + c_i sP_{i,1}$ and $T'_i = r_i P$.

Aggregation: Anyone can aggregate a collection of individual signatures that use the same string w. For example, individual signatures (w, S'_i, T'_i) for $1 \leq i \leq n$ can be aggregated into (w, S_n, T_n), where $S_n = \sum_{i=1}^n S'_i$ and $T_n = \sum_{i=1}^n T'_i$. Our security proof does not permit the aggregation of individual (or aggregate) signatures that use different w's.

Verification: Let (w, S_n, T_n) be the identity-based aggregate signature (where n is the number of signers). The verifier checks that:

$$\hat{e}(S_n, P) = \hat{e}(T_n, P_w)\hat{e}(Q, \sum_{i=1}^{n} P_{i,0} + \sum_{i=1}^{n} c_i P_{i,1}) \ ,$$

where $P_{i,j} = H_1(\mathrm{ID}_i, j)$, $P_w = H_2(w)$ and $c_i = H_3(m_i, \mathrm{ID}_i, w)$, as above.

Remark 2. This scheme is reasonably efficient. Unlike the BGLS [BGLS03] aggregate signature, this scheme requires a constant number of pairing computations for verification (though the total work is still linear in the number of signers).

Remark 3. If we were to just set the signature to be $sP_{i,0} + c_1 sP_{i,1}$, then after two signatures an adversary will likely be able to recover the values of $sP_{i,0}$ and $sP_{i,1}$ using linear algebra. The purpose of the one-time-use P_w is to disturb this linearity, while providing a place where all the signers can "aggregate their randomness."

Remark 4. To allow each signer to produce k individual identity-based signatures with a single value of w, we can change private key generation so that the client with identity ID_i receives from the PKG the values of $sP_{i,j}$ for $j \in [0, k]$, where $P_{i,j} = H_1(\mathrm{ID}_i, j) \in \mathbb{G}_1$. To sign, the signer computes $c_{ij} = H_3(m_i, \mathrm{ID}_i, w, j)$ for $1 \leq i \leq k$, and sets $S'_i = r_i P_w + sP_{i,0} + \sum_{j=1}^{k} c_{ij} sP_{i,j}$. The result of the signing procedure is the same, and verification is modified in the obvious fashion.

Remark 5. It is possible to aggregate individual identity-based signatures even if the signers have different PKGs, and the security proof goes through. However, to verify such a multiple-PKG identity-based aggregate signature, the verifier needs the public key of every PKG. Thus, from a bandwidth perspective, the single-PKG case is optimal.

6 The Security of Our IBAS Construction

We start by providing some intuition for how an algorithm $\mathcal{B}$ can solve a computational Diffie-Hellman problem – i.e., compute sP' from P, sP, and P' – by interacting with an algorithm $\mathcal{A}$ that breaks our IBAS scheme. The security proof for the multisignature scheme in the full version [GR06] provides additional intuition. During the interaction, $\mathcal{B}$ must either respond correctly to $\mathcal{A}$'s queries, or abort. $\mathcal{A}$ can make several types of queries:

1. H_1 **and Extraction Queries:** $\mathcal{A}$ can ask for the identity-based private keys $sP_{i,j}$ for $j \in \{0, 1\}$ that correspond to identity ID_i. $\mathcal{B}$ handles these queries through its control of the H_1 oracle. In particular, it usually generates $P_{i,j}$ in such a way that it knows $b_{i,j} = \log_P P_{i,j}$; then, it can compute $sP_{i,j} = b_{i,j} sP$. However, $\mathcal{B}$ occasionally sets $P_{i,j} = b_{i,j}P + b'_{i,j}P'$. In this case, $\mathcal{B}$

cannot respond to an extraction query on ID_i, but if $\mathcal{A}$ later chooses ID_i as a target identity, $\mathcal{A}$'s forgery may help $\mathcal{B}$ solve its computational Diffie-Hellman problem.

2. H_2 **queries:** $\mathcal{B}$, through its control over the H_2 oracle, will usually generate P_w in such a way that it knows $d_w = \log_{P'} P_w$, but occasionally generates P_w so that it knows $c_w = \log_P P_w$ instead.
3. H_3 **and signature queries:** $\mathcal{B}$'s control over the H_2 and H_3 oracles helps it to respond to signature queries regarding the tuple $(\mathsf{ID}_i, m_j, w_k)$ when it cannot even extract the private key corresponding to ID_i. How can $\mathcal{B}$ generate valid and consistent values of $P_{i,0}$, $P_{i,1}$, P_{w_k}, $d_{i,j,k} = H_3(\mathsf{ID}_i, m_j, w_k)$, $S'_i = rP_{w_k} + sP_{i,0} + d_{ijk}sP_{i,1}$ and $T'_i = rP$ in such a situation? In particular, how can it generate S'_i, which seems to require that $\mathcal{B}$ know sP'? If $\mathcal{B}$ knows $\log_{P'} P_w$, it can compute the value of r' such that $r'sP_w$ "cancels out" the multiple of sP' that comes from the final two terms; it then sets T'_i to be $r'sP$. If $\mathcal{B}$ doesn't know $\log_{P'} P_w$, it has one more trick it can use; occasionally, $\mathcal{B}$ sets d_{ijk} to be the unique value in $\mathbb{Z}/q\mathbb{Z}$ that causes the multiples of sP' in the final two terms to cancel. In this case, $\mathcal{B}$ can produce a valid signature. Once this unique value is revealed for a given ID_i, it cannot use this trick again (otherwise, the simulation will not be convincing to $\mathcal{A}$).

If $\mathcal{B}$ is lucky, its simulation does not abort and $\mathcal{A}$ produces a forgery on a tuple $(\mathsf{ID}_i, m_j, w_k)$ for which it does not know $\log_P P_{i,j}$, does know $\log_P P_w$, and where d_{ijk} was not chosen using the trick above. In this case, $\mathcal{A}$'s forgery gives $\mathcal{B}$ the value of sP' with extremely high probability.

The following theorem characterizes the security of our IBAS scheme.

Theorem 2. *Let $\mathcal{A}$ be an adversary that $(t, \epsilon, n, q_{H_3}, q_E, q_S,)$-breaks the IBAS scheme. Then, there exists an algorithm $\mathcal{B}$ that solves $CDH_{\mathbb{G}_1}$ in time $O(t) + O(\log^3 q)$ with success probability at least $\epsilon/1024 q_E q_S (q_{H_3} - q_S)$.*

Proof: Algorithm $\mathcal{B}$ is given an instance $(P, Q, P', \hat{e})$ (for $Q = sP$) of the $CDH_{\mathbb{G}_1}$ problem, and will interact with algorithm $\mathcal{A}$ as follows in an attempt to compute sP'.

Setup: $\mathcal{B}$ sets the public key of the PKG to be $(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, Q, H_1, H_2, H_3)$, and it transmits this key to $\mathcal{A}$. Here the H_i's are random oracles controlled by $\mathcal{B}$.

Hash Queries: $\mathcal{A}$ can make an H_1-query, H_2-query, or H_3-query at any time. $\mathcal{B}$ gives identical responses to identical queries, maintaining lists relating to its previous hash query responses for consistency. $\mathcal{B}$ also maintains H_3-list2, which addresses certain special cases of the H_3 simulation. $\mathcal{B}$ responds to $\mathcal{A}$'s H_1-query on (ID_i, j) as follows:

For $\mathcal{A}$'s H_1-query on (ID_i, j) for $j \in \{0, 1\}$:

1. If ID_i was in a previous H_1-query, $\mathcal{B}$ recovers $(b_{i0}, b'_{i0}, b_{i1}, b'_{i1})$ from its H_1-list.
2. Else, $\mathcal{B}$ generates a random $H_1\text{-}coin_i \in \{0, 1\}$ so that $\Pr[H_1\text{-}coin_i = 0] = \delta_1$ for δ_1 to be determined later. If $H_1\text{-}coin_i = 0$, $\mathcal{B}$ generates random $b_{i0}, b_{i1} \in \mathbb{Z}/q\mathbb{Z}$ and sets $b'_{i0} = b'_{i1} = 0$; else, it generates random $b_{i0}, b'_{i0}, b_{i1}, b'_{i1} \in \mathbb{Z}/q\mathbb{Z}$. $\mathcal{B}$ logs $(\mathsf{ID}_i, H_1\text{-}coin_i, b_{i0}, b'_{i0}, b_{i1}, b'_{i1})$ in its H_1-list.

3. $\mathcal{B}$ responds with $H_1(\mathsf{ID}_i, j) = P_{ij} = b_{ij}P + b'_{ij}P'$.

For $\mathcal{A}$'s H_2-query on w_k:

1. If w_k was in a previous H_2-query, $\mathcal{B}$ recovers c_k from its H_2-list.
2. Else, $\mathcal{B}$ generates a random $H_2\text{-}coin_k \in \{0,1\}$ so that $\Pr[H_1\text{-}coin_i = 0] = \delta_2$ for δ_2 to be determined later. $\mathcal{B}$ generates a random $c_k \in (\mathbb{Z}/q\mathbb{Z})^*$. It logs $(w_k, H_2\text{-}coin_k)$ in its H_2-list.
3. If $H_2\text{-}coin_k = 0$, $\mathcal{B}$ responds with $H_2(w_k) = P_{w_k} = c_kP'$; otherwise, it responds with $H_2(w_k) = P_{w_k} = c_kP$.

For $\mathcal{A}$'s H_3-query on $(\mathsf{ID}_i, m_j, w_k)$:

1. If $(\mathsf{ID}_i, m_j, w_k)$ was in a previous H_3-query, $\mathcal{B}$ recovers d_{ijk} from its H_3-list.
2. Else, $\mathcal{B}$ runs an H_1-query on $(\mathsf{ID}_i, 0)$ to recover b'_{i0} and b'_{i1} from its H_1-list. $\mathcal{B}$ generates a random $H_3\text{-}coin_{ijk} \in \{0,1\}$ so that $\Pr[H_3\text{-}coin_{ijk} = 0] = \delta_3$ for δ_3 to be determined later.
   (a) If $H_1\text{-}coin_i = 1$, $H_2\text{-}coin_k = 1$, and $H_3\text{-}coin_{ijk} = 0$, $\mathcal{B}$ checks whether H_3-list2 contains a tuple $(\mathsf{ID}_{i'}, m_{j'}, w_{k'}) \neq (\mathsf{ID}_{i'}, m_{j'}, w_{k'})$ with $\mathsf{ID}_{i'} = \mathsf{ID}_i$. If so, $\mathcal{B}$ aborts. If not, it puts $(\mathsf{ID}_i, m_j, w_k)$ in H_3-list2 and sets $d_{ijk} = -b'_{i0}/b'_{i1} (\mathrm{mod} q)$.
   (b) If $H_1\text{-}coin_i = 0$, $H_2\text{-}coin_k = 0$, or $H_3\text{-}coin_{ijk} = 1$, $\mathcal{B}$ generates a random $d_{ijk} \in (\mathbb{Z}/q\mathbb{Z})^*$.
   (c) $\mathcal{B}$ logs $(\mathsf{ID}_i, m_j, w_k, H_3\text{-}coin_{ijk}, d_{ijk})$ in its H_3-list.
3. $\mathcal{B}$ responds with $H_3(\mathsf{ID}_i, m_j, w_k) = d_{ijk}$.

Extraction Queries: When $\mathcal{A}$ requests the private key corresponding to ID_i, $\mathcal{B}$ recovers $(H_1\text{-}coin_i, b_{i0}, b'_{i0})$. If $H_1\text{-}coin_i = 0$, $\mathcal{B}$ responds with $(sP_{i,0}, sP_{i,1}) = (b_{i0}Q, b_{i1}Q)$. If $H_1\text{-}coin_i = 1$, $\mathcal{B}$ aborts.

Signature Queries: When $\mathcal{A}$ requests a (new) signature on $(\mathsf{ID}_i, m_j, w_k)$, $\mathcal{B}$ first confirms that $\mathcal{A}$ has not previously requested a signature by ID_i on w_k (otherwise, it is an improper query). Then, $\mathcal{B}$ proceeds as follows:

1. If $H_1\text{-}coin_i = H_2\text{-}coin_k = H_3\text{-}coin_{ijk} = 1$, $\mathcal{B}$ aborts.
2. If $H_1\text{-}coin_i = 0$, $\mathcal{B}$ generates random $r \in \mathbb{Z}/q\mathbb{Z}$ and outputs the signature (w_k, S'_i, T'_i), where $S'_i = sP_{i,0} + d_{ijk}sP_{i,1} + rP_{w_k} = b_{i0}Q + d_{ijk}b_{i1}Q + rP_{w_k}$ and $T'_i = rP$.
3. If $H_1\text{-}coin_i = 1$ and $H_2\text{-}coin_k = 0$, $\mathcal{B}$ generates random $r \in \mathbb{Z}/q\mathbb{Z}$ and outputs the signature (w_k, S'_i, T'_i), where

$$\begin{aligned} S'_i &= sP_{i,0} + d_{ijk}sP_{i,1} + (r - (b'_{i0} + d_{ijk}b'_{i1})sc_k^{-1})P_{w_k} \\ &= b_{i0}Q + b'_{i0}sP' + d_{ijk}b_{i1}Q + d_{ijk}b'_{i1}sP' + rc_kP' - (b'_{i0} + d_{ijk}b'_{i1})sP' \\ &= b_{i0}Q + d_{ijk}b_{i1}Q + rc_kP', \text{ and} \\ T'_i &= (r - (b'_{i0} + d_{ijk}b'_{i1})sc_k^{-1})P = rP - (b'_{i0} + d_{ijk}b'_{i1})c_k^{-1}Q \ . \end{aligned}$$

4. If H_1-*coin*$_i$ = H_2-*coin*$_k$ = 1 and H_3-*coin*$_{ijk}$ = 0, $\mathcal{B}$ generates random $r \in \mathbb{Z}/q\mathbb{Z}$ and outputs the signature (w_k, S'_i, T'_i), where $T'_i = rP$, and

$$\begin{aligned} S'_i &= sP_{i,0} + d_{ijk}sP_{i,1} + rP_{w_k} \\ &= b_{i0}Q + b'_{i0}sP' + d_{ijk}b_{i1}Q - (b'_{i0}/b'_{i1})b'_{i1}sP' + rc_kP \\ &= b_{i0}Q + d_{ijk}b_{i1}Q + rc_kP \ . \end{aligned}$$

$\mathcal{A}$'s Response: Finally, with probability at least ϵ, $\mathcal{A}$ outputs $\{\mathsf{ID}_i\}_{i=1}^{l}$ and $\{m_j\}_{j=1}^{l}$ with $l \leq n$, and string w_K, such that there exists $I, J \in [1, l]$ such that it has not extracted the private key for ID_I or requested a signature for $(\mathsf{ID}_I, m_J, w_K)$. In addition, it also outputs an identity-based aggregate signature (w_K, S_l, T_l) satisfying the equation

$$\hat{e}(S_l, P) = \hat{e}(T_l, P_{w_K})\hat{e}(sP, \sum_{i=1}^{l} P_{i,0} + \sum_{i=1}^{l} c_iP_{i,1}) \ ,$$

where $P_{i,b} = H_1(\mathrm{ID}_i, b)$, $P_{w_K} = H_2(w_K)$ and $c_i = H_3(m_j, \mathrm{ID}_i, w_K)$ as required.

$\mathcal{B}$'s Final Action: If it is not the case that the above (I, J, K) can satisfy H_1-*coin*$_I$ = H_2-*coin*$_J$ = H_3-*coin*$_{IJK}$ = 1, then $\mathcal{B}$ aborts. Otherwise, it can solve its instance of $\mathrm{CDH}_{\mathbb{G}_1}$ with probability $1 - 1/q$ as follows.

$\mathcal{A}$'s forgery has the form (S_l, T_l), where $T_l = rP$ and $S_l = rP_{w_K} + \sum_{i=1}^{l} sP_{i,0} + c_isP_{i,1}$, where we let $c_i = H_3(\mathsf{ID}_i, m_j, w_K)$ be the hash of the tuple "signed" by the entity with identity ID_i. Since H_2-*coin*$_k$ = 1, $\mathcal{B}$ knows the discrete logarithm c_K of P_{w_K} with respect to P. It can therefore compute:

$$\begin{aligned} S_l - c_KT_l = \sum_{i=1}^{l} sP_{i,0} + c_isP_{i,1} &= s\left(\sum_{i=1}^{l} b_{i,0}P + b'_{i,0}P' + c_i(b_{i,1}P + b'_{i,1}P')\right) \\ &= s\left(\sum_{i=1}^{l}(b_{i,0} + c_ib_{i,1})\right)P + s\left(\sum_{i=1}^{l}(b'_{i,0} + c_ib'_{i,1})\right)P' \ . \end{aligned}$$

If H_1-*coin*$_i$ = H_3-*coin*$_{ijk}$ = 1 for at least one of the signed tuples, then the probability that $\sum_{i=1}^{l}(b'_{i,0} + c_ib'_{i,1}) \neq 0$ is $1 - 1/q$; if $\sum_{i=1}^{l}(b'_{i,0} + c_ib'_{i,1}) \neq 0$, $\mathcal{B}$ can easily derive sP' from the expression above.

We now demonstrate that the above simulation is perfect. The analysis assumes that $\mathcal{A}$ makes no redundant queries and that $\mathcal{A}$ must make an H_3 query on a tuple $(\mathsf{ID}_i, m_j, w_k)$ before making a signature query on it. Let $\mathcal{E}$ represent the set of extraction query responses that $\mathcal{B}$ has made up to a specified point in the simulation; similarly, let $\mathcal{S}$ be the set of signature query responses, and $\mathcal{H}_i$ be the set of H_i query responses for $i \in \{1, 2, 3\}$. Let $E_{1,*,*}$ be the event that H_1-*coin*$_i$ = 1; here, "*" means that H_2-*coin*$_k$ and H_3-*coin*$_{ijk}$ may each be 0 or 1. Let $E_{1,1,*}$, $E_{1,1,1}$ and $E_{1,1,0}$ denote the corresponding events in the obvious way.

Perfect Simulation: We claim that, if $\mathcal{B}$ does not abort, $\mathcal{A}$'s view is the same as in the "real" attack. In the "real" attack, each of the hash functions H_i

behave like random functions. Then, given the values of $P_{i,j} = H_1(\mathsf{ID}_i, j)$, $P_{w_k} = H_2(w_k)$, and $d_{ijk} = H_3(\mathsf{ID}_i, m_j, w_k)$, we choose a signature uniformly from:

$$\{(w_k, S'_i, T'_i) : S'_i = sP_{i,0} + d_{ijk}sP_{i,1} + rP_w, T'_i = rP, r \in \mathbb{Z}/q\mathbb{Z}\}.$$

Similarly, in the simulation, we choose a signature uniformly from $\{(w_k, S'_i, T'_i) : S'_i = sP_{i,0} + d_{ijk}sP_{i,1} + rP_w, T'_i = rP, r \in \mathbb{Z}/q\mathbb{Z}\}$ given values of $P_{i,j} = H_1(\mathsf{ID}_i, j)$, $P_{w_k} = H_2(w_k)$, and $d_{ijk} = H_3(\mathsf{ID}_i, m_j, w_k)$. Also, the H_i behave like random functions – i.e., they are one-to-one and the outputs are chosen with uniform distribution. The only case in which this may not be obvious is when $H_1\text{-}coin_i = H_2\text{-}coin_k = 1$ and $H_3\text{-}coin_{ijk} = 0$. In this case, unless $\mathcal{A}$ has made a previous H_3 query on $(\mathsf{ID}_i, m_{j'}, w_{k'}) \neq (\mathsf{ID}_i, m_j, w_k)$ for which $H_1\text{-}coin_i = H_2\text{-}coin_{k'} = 1$ and $H_3\text{-}coin_{ij'k'} = 0$ (in which case $\mathcal{B}$ aborts), $\mathcal{B}$ sets $H_3(\mathsf{ID}_i, m_j, w_k)$ to be $-b'_{i0}/b'_{i1}(\text{mod} q)$ (rather than choosing the H_3 output uniformly randomly).

However, the value of $-b'_{i0}/b'_{i1}(\bmod q)$ is itself uniformly random. More specifically, given $\mathcal{A}$'s view up until the H_3 query on $(\mathsf{ID}_i, m_j, w_k)$ – namely, the sets $\mathcal{E}$, $\mathcal{S}$, and $\{H_i\}$ – we have that

$$\Pr[H_3(\mathsf{ID}_i, m_j, w_k) = c \mid \mathcal{E}, \mathcal{S}, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, E_{1,1,0}] = 1/q$$

for every $c \in \mathbb{Z}/q\mathbb{Z}$, as long as $\mathcal{B}$ does not abort. Most surprisingly, the value $H_3(\mathsf{ID}_i, m_j, w_k) = -b'_{i0}/b'_{i1}(\text{mod} q)$ is independent of an H_1 query response on ID_i even though $H_1(\mathsf{ID}_i, j) = b_{ij}P + b'_{ij}P'$, since, given $H_1(\mathsf{ID}_i, 0) = b_{i0}P + b'_{i0}P'$, the pairs (b_{i0}, b'_{i0}) with $b'_{i0} = \log_{P'}(H_1(\mathsf{ID}_i, 0)) - b_{i0}\log_{P'}(P)$ are equally likely. It should be clear that the value of $H_3(\mathsf{ID}_i, m_j, w_k)$ is also independent of H_1 queries on identities other than ID_i, all extraction query responses (since they are completely dependent on H_1 query responses), all H_2 queries, all H_3 queries on tuples other than $(\mathsf{ID}_i, m_j, w_k)$ (again, assuming $\mathcal{B}$ does not abort), and all signature queries on tuples other than $(\mathsf{ID}_i, m_j, w_k)$.

To complete the proof, we need to bound from below the probability that $\mathcal{B}$ aborts. The details are provided in the full version [GR06].

7 Summary and Open Problems

We presented an IBAS scheme which allows distinct signers to sign distinct documents in such a way that the total information needed to verify the signatures is about as close as possible to the information-theoretic minimum. The aggregate signature can be generated in a completely decentralized fashion, without requiring a complicated setup procedure. Our scheme was quite efficient - requiring only 4 elliptic curve scalar multiplications and 2 point additions for signature generation; 2 extra point additions for aggregation; and 3 pairing computations *(independent of the number of signers)*, 1 point multiplication, $2n-1$ point additions, and n scalar multiplication (where n is the number of signatures that are aggregated) for verification. Verification in our scheme is much faster than the BGLS aggregate signature scheme [BGLS03] which requires $O(n)$ pairing computations. Further, our scheme allows aggregation even if the signers have different

PKGs. Finally, our scheme is provably secure in the random oracle model under Computational Diffie-Hellman against an adversary who could choose both its target identities and messages adaptively.

It may be possible to construct practical IBAS schemes using different approaches and assumptions – e.g., based on strong RSA – but, again, aggregating individual signer randomness is a problem. With strong RSA, one might even consider a deterministic scheme, roughly as follows. The PKG publishes a modulus N, a base $a \in \mathbb{Z}_N^*$, and hash functions $H_1 : \{0,1\}^* \rightarrow \{0,1\}^d$ (e.g., $d = 160$) and $H_2 : \{0,1\}^* \rightarrow \mathcal{P}$ (where $\mathcal{P}$ is a suitable set of prime numbers). To a user with identity ID_i who wants to generate up to t signatures, the PKG gives the value $a^{1/P_i}(\mathrm{mod} N)$, where $P_i = \prod_{j\in[1,t]}^{k\in[1,d]} H_2(\mathsf{ID}_i, j, k)$. To sign m for its j-th signature, the user computes $a^{1/P_{i,j,m}}(\mathrm{mod} N)$ for

$$P_{i,j,m} = \prod^{k\in[1,d]} H_2(\mathsf{ID}_i, j, k)^{H_1(\mathsf{ID}_i,j,m)_k},$$

where $H_1(\mathsf{ID}_i, j, m)_k$ is the k-th bit of $H_1(\mathsf{ID}_i, j, m)$. With this approach the "de-accumulation" that a user performs is computationally-intensive if t is reasonably large. One could amortize the expense of de-accumulation by using tree-traversal (pebbling-type) techniques – e.g., as described in [Szy04] – but this restricts the users to using the j-values in order, which makes it less likely that distinct users will use the same j, which increases the amount of verification information.

References

[BA05] K.C. Barr and K. Asanovic. Energy aware lossless data compression. In *Proc. of Mobisys 2005*, 2005.

[BF03] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. of Computing*, 32(3):586–615, 2003.

[BGLS03] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proc. of Eurocrypt 2003*, volume 2656 of *LNCS*, pages 416–432. Springer-Verlag, 2003.

[BLS01] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *Proc. of Asiacrypt 2001*, volume 2248 of *LNCS*, pages 514–532. Springer-Verlag.

[BNN04] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security proofs for identity-based identification and signature schemes. In *Proc. of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 268–286. Springer-Verlag, 2004.

[Bol03] A. Boldyreva. Efficient threshold signature, multisignature and blind signature schemes based on the gap-Diffie-Hellman-group signature scheme. In *Proc. of PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer-Verlag, 2003.

[Boy03] X. Boyen. Multipurpose identity-based signcryption (a swiss army knife for identity-based cryptography). In *Proc. of Crypto 2003*, volume 2729 of *LNCS*, pages 383–399. Springer-Verlag, 2003.

[BZ04] J. Baek and Y. Zheng. Identity-based threshold signature scheme from the bilinear pairings. In *Proc. of ITCC (1)*, pages 124–128, 2004.

[CC03] J.C. Cha and J.H. Cheon. An identity-based signature from gap diffie-hellman groups. In *Proc. of PKC 2003*, volume 2567 of *LNCS*, pages 18–30.

[CLW05] X. Cheng, J. Liu, and X. Wang. Identity-based aggregate and verifiably encrypted signatures from bilinear pairing. In *Proc. of ICCSA 2005*, pages 1046–1054, 2005.

[Coc01] C. Cocks. An identity based encryption scheme based on quadratic residues. In *Proc. of IMA Int. Conf. 2001*, volume 2260 of *LNCS*, pages 360–363.

[FS86] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proc. of Crypto 1986*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.

[GR06] C. Gentry and Z. Ramzan. Identity-Based Aggregate Signatures. Full Version. *Cryptology E-print Archive, 2006.*

[GQ88] L.C. Guillou and J.-J. Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In *Proc. of Crypto 1988*, volume 403 of *LNCS*, pages 216–231. Springer-Verlag, 1988.

[GS02] C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In *Proc. of Asiacrypt 2002*, volume 2501 of *LNCS*, pages 548–566. Springer-Verlag, 2002.

[Her05] J. Herranz. Deterministic identity-based signatures for partial aggregation. Cryptology ePrint Archive, Report 2005/313, 2005. http://eprint.iacr.org/.

[KLS00] S. Kent, C. Lynn, and K. Seo. Secure border gateway protocol (secure-bgp). *IEEE J. Selected Areas in Comm.*, 19(4):582–592, 2000.

[LMRS04] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In *Proc. of Eurocrypt 2004*, volume 9999 of *LNCS*, pages 74–90. Springer-Verlag, 2004.

[LQ04] B. Libert and J.-J. Quisquater. Identity based undeniable signatures. In *Proc. of CT-RSA 2004*, pages 112–125, 2004.

[MNT04] E. Mykletun, M. Narasimha, and G. Tsudik. Signature bouquets: Immutability for aggregated/condensed signatures. In *Proc. of ESORICS 2004*, pages 160–176, 2004.

[MOR01] S. Micali, K. Ohta, and L. Reyzin. Accountable subgroup multisignatures (extended abstract). In *Proc. of CCS 2001*, pages 245–54. ACM Press, 2001.

[Oka98] T. Okamoto. A digital multisignature scheme using bijective public-key cryptosystems. *ACM Trans. Computer Systems*, 6(4):432–441, 1998.

[OO99] K. Ohta and T. Okamoto. Multisignature schemes secure against active insider attacks. *IEICE Trans. Fundamentals*, E82-A(1):21–31, 1999.

[PS96] D. Pointcheval and J. Stern. Security proofs for signature schemes. In *Proc. of Eurocrypt*. Springer-Verlag, 1996.

[PS00] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[Sha84] A. Shamir. Identity-based cryptosystems and signature schemes. In *Proc. of Crypto 1984*, volume 196 of *LNCS*, pages 47–53. Springer-Verlag, 1984.

[Sho00] V. Shoup. Practical threshold signatures. In *Proc. of Eurocrypt 2000*, volume 1807 of *LNCS*, pages 207–220. Springer-Verlag, 2000.

[SRF$^+$04] T. Suzuki, Z. Ramzan, H. Fujimoto, C. Gentry, T. Nakayama, and R. Jain. A system for end-to-end authentication of adaptive multimedia content. In *Proc. of Conference on Communications and Multimedia Security*, 2004.

[Szy04] M. Szydlo. Merkle tree traversal in log space and time. In *Proc. of Eurocrypt*, volume 3027 of *LNCS*, pages 541–554. Springer-Verlag, 2004.