Forward-Secure Multi-Signatures

Manu Drijvers Gregory Neven

DFINITY

Abstract

Multi-signatures allow a group of signers to jointly sign a message in a compact and efficiently verifiable signature, ideally independent of the number of signers in the group. We present the first provably secure forward-secure multi-signature scheme by deriving a forward-secure signature scheme from the hierarchical identity-based encryption of Boneh, Boyen, and Goh (Eurocrypt 2005) and showing how the signatures in that scheme can be securely composed. Multi-signatures in our scheme contain just two group elements (one from each of the base groups) and require one exponentation and three pairing computations to verify.

1 Introduction

There is a long line of work in cryptography on techniques to compress many signatures by different signers while maintaining verifiability under the signers' public keys. Multi-signatures [IN83] compress signatures by n different signers on the same message M into a single compact signature, preferably with size independent of n, while aggregate signatures [BGLS03] can compress signatures on different messages $M_1, \ldots, M_n$.

While somewhat more restricted in terms of functionality, many multi-signature schemes [MOR01, Bol03, MPSW18, DEF+18, BDN18a] offer the additional advantage of having verification time (practically) independent of n. Multi-signatures have recently gained popularity for their use in cryptocurrencies [MPSW18, Poe19] to save precious block space for multi-input transactions, or as an additional layer of security to protect user wallets.

Another long line of work tries to mitigate the damage done to a cryptosystem when the adversary gets hold of the long-term keys. The concept of *forward security* [And00] requires that honest users periodically update their secret keys, so that when their secret key is compromised, the transactions in previous time periods remain secure. For signatures, this means that an adversary who gains access to the signing key in time period $\bar{t}$, is still unable to forge signatures for time periods $t < \bar{t}$.

Given the widespread use of multi-signatures in cryptocurrencies and the considerable financial consequences of key exposure, it makes sense to add forward security to multi-signature schemes. To the best of our knowledge, only a single scheme combining these features has appeared in the literature [SA09] based on ElGamal signatures [ElG85], but without a proof of security.

1.1 Our Contributions

We propose the first provably secure forward-secure multi-signature scheme. Our scheme is based on pairing-friendly elliptic curves and is best understood as being derived from the key structure

of the Boneh-Boyen-Goh (BBG) hierarchical identity-based encryption (HIBE) scheme [BBG05a]. Our signing algorithm is non-interactive, meaning that individual signers can independently generate their contribution to the multi-signature, and any third party can combine their individual contributions into a multi-signature.

Cannetti, Halevi, and Katz [CHK07] described how to derive a forward-secure encryption scheme from a HIBE scheme by embedding the time periods in the identity tree of the HIBE scheme, and by letting the secret key at period t consist of the decryption keys from which the current and all future keys can be derived. It is well known that the key structure of an identity-based encryption scheme naturally yields a signature scheme [BF01, CFH$^+$07] and that a two-level HIBE scheme yields an identity-based signature scheme [GS02]. Analogously, it is not hard to see that the key structure of a HIBE-derived forward-secure encryption scheme also yields a forward-secure signature scheme [BSSW06].

A particular advantage of the BBG HIBE scheme for use as a signature scheme, as opposed to other HIBE schemes [GS02, BB04, Wat05], is that keys in the BBG HIBE become smaller in size as one descends the hierarchy. Keys at the leaves therefore become attractive to use as signatures because they are quite compact, namely one element of $\mathbb{G}_1$ and one element of $\mathbb{G}_2$ when instantiated over a curve with an admissible pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$. Moreover, we show that, similar to other pairing-based multi-signatures [Bol03, LOS$^+$06], signatures on the same message M by multiple signers are easily aggregated through component-wise multiplication into a compact and efficiently verifiable multi-signature.

The straightforward scheme, however, is vulnerable to the same rogue-key attacks that have plagued other multi-signature schemes in the past [MOR01, Bol03, BN06, RY07, MPSW18], meaning that an adversary who chooses his signing key as a function of other signers' keys can forge multi-signatures. We show that the Ristenpart-Yilek technique [RY07] of adding a proof of possession to the public key restores security. We prove our scheme secure in the random-oracle model [BR93] under a variant of the bilinear Diffie-Hellman inversion (BDHI) assumption.

1.2 Related Work

Multi-signature schemes have been proposed based on different assumptions, including RSA [IN83, OO93, BN07], discrete logarithms [HZ93, LHL95, OO99, MOR01, BN06, BCJ08, MWLD10, MPSW18, DEF$^+$18], pairings [Bol03, LOS$^+$06, RY07, BGOY07, LBG09, BDN18a], and lattices [BS16]. The schemes also differ in the way signatures are being generated. All known discrete-logarithm based schemes require at least two rounds of interaction between the signers. Schemes based on RSA tend to require a sequential interaction among signers [IN83] or be fully interactive [OO93, BN07]. The only known non-interactive schemes, i.e., where each signer locally computes his contribution to the multi-signature and anyone can combine these contributions into a multi-signature, are based on pairings [Bol03, LOS$^+$06, RY07, BGOY07, LBG09, BDN18a].

There also exist signature schemes that can compress signatures on different messages. Aggregate signatures [BGLS03] can do so in a non-interactive way, while sequential aggregate signatures [LMRS04, LOS$^+$06, Nev08, BGR12, GOR18] require signers to take turns in adding their signatures to the aggregate. None of these schemes have efficient verification, though, in the sense of being independent of the number of aggregated signatures.

Forward security was originally proposed for key exchange protocols [And00], but was later extended and formalized to signature schemes [BM99]. Generic constructions of forward-secure signature schemes [BM99, MMM02, Kra00] usually organize public keys of any standard signature

scheme in a tree structure for certification and apply clever techniques to make signing, verification, and key update more efficient. Direct constructions have also been proposed based on factoring [AR00], RSA [IR01, KR03].

Finally, Boyen et al. [BSSW06] proposed a forward-secure signature scheme with untrusted key updates, meaning, where key updates can be performed on encrypted versions of the key. They also base their construction on a forward-secure signature scheme derived from the BBG HIBE, just like we do. We make the underlying forward-secure signature scheme explicit here, re-prove it in the asymmetric pairing setting, and show how it can be used as a multi-signature.

2 Preliminaries

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$ be multiplicative groups of prime order q with an admissible pairing function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$. Let g_1 and g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.

In analogy with the weak bilinear Diffie-Hellman inversion problem ℓ-wBDHI* [BBG05b], which was originally defined for Type-1 pairings (i.e., symmetric pairings $e : \mathcal{G} \times \mathcal{G} \to \mathbb{G}_t$), we define the following variant for Type-3 pairings denoted ℓ-wBDHI$^*_3$.

$$\text{Input: } A_1 = g_1^{\alpha}\ ,\ A_2 = g_1^{(\alpha^2)}\ ,\ \ldots\ ,\ A_\ell = g_1^{(\alpha^\ell)}\ ,\ A'_1 = g_2^{\alpha}\ ,\ B \quad \text{for } \alpha \xleftarrow{\$} \mathbb{Z}_q\ ,\ B \xleftarrow{\$} \mathbb{G}_2$$
$$\text{Compute: } e(g_1, B)^{(\alpha^{\ell+1})}$$

The advantage $\mathbf{Adv}^{\ell\text{-wBDHI}^*_3}_{\mathbb{G}_1 \times \mathbb{G}_2}(\mathcal{A})$ of an adversary $\mathcal{A}$ is defined as its probability in solving this problem.

3 Forward-Secure Signatures

3.1 Definition

We use the Bellare-Miner model [BM99] to define syntax and security of a forward-secure signature scheme. A forward-secure signature scheme $\mathcal{FS}$ for a message space $\mathcal{M}$ consists of the following algorithms:

Key generation: $(pk, sk_0) \xleftarrow{\$} \mathsf{Kg}$. The signer runs the key generation algorithm on input the maximum number of time periods T to generate a public verification key pk and an initial secret signing key sk_0 for time period $t = 0$.

Key update: $sk_{t+1} \xleftarrow{\$} \mathsf{Upd}(sk_t)$. The signer updates its secret key sk_t for time period t to sk_{t+1} for the next period using the key update algorithm.

Signing: $\sigma \xleftarrow{\$} \mathsf{Sign}(sk_t, M)$. On input the current signing key sk_t and message $M \in \mathcal{M}$, the signer uses this algorithm to compute a signature σ.

Verification. $b \leftarrow \mathsf{Vf}(pk, t, M, \sigma)$. Anyone can verify a signature σ for on message M for time period t under public key pk by running the verification algorithm, which returns 1 to indicate that the signature is valid and 0 otherwise.

Correctness requires that for all messages $M \in \mathcal{M}$ and for all time periods $t \in \{0, \ldots, T-1]$ it holds that $\mathsf{Vf}(pk, t, M, \mathsf{Sign}(sk_t, M)) = 1$ with probability one if $(pk, sk_0) \xleftarrow{\$} \mathsf{Kg}$ and $sk_{i+1} \leftarrow \mathsf{Upd}(sk_i)$ for $i = 0, \ldots, t-1$.

Unforgeability under chosen-message attack for forward-secure signatures is defined through the following game. The experiment generates a fresh key pair (pk, sk_0) and hands the public key pk to the adversary $\mathcal{A}$. The adversary is given access to the following oracles:

Key update. If the current time period t (initially set to $t = 0$) is less than $T-1$, then this oracle updates the key sk_t to sk_{t+1} and increases t.

Signing. On input a message M, this oracle runs the signing oracle with the current secret key sk_t and message M, and returns the resulting signature σ.

Break in. The experiment records the break-in time $\bar{t} \leftarrow t$ and hands the current signing key $sk_{\bar{t}}$ to the adversary. This oracle can only be queried once, and after it has been queried, the adversary can make no further queries to the key update or signing oracles.

At the end of the game, the adversary outputs its forgery $(t^*, M^*, \sigma^*$. It wins the game if σ^* verifies correctly under pk for time period t^* and message M^*, if it never queried the signing oracle on M^* during time period t^*, and if it queried the corrupt oracle, then it did so in a time period $\bar{t} > t^*$. We define $\mathcal{A}$'s advantage $\mathbf{Adv}^{\text{fu-cma}}_{\mathcal{FS}}(\mathcal{A})$ as its probability in winning the above game.

We also define a selective variant of the above notion, referred to as sfu-cma, where the adversary first has to commit to $\bar{t}$, t^*, and M^*. More specifically, $\mathcal{A}$ first outputs $(\bar{t}, t^*, M^*)$, then receives the public key pk, is allowed to make signature and key update queries until time period $t = \bar{t}$ is reached, at which point it is given $sk_{\bar{t}}$ and outputs its forgery σ^*.

3.2 Construction

The following scheme is essentially the result of applying the Canetti-Halevi-Katz technique to obtain forward security from hierarchical identity-based encryption (HIBE) [CHK07] to the signature scheme determined by the key structure of the Boneh-Boyen-Goh HIBE scheme [BBG05a].

Common parameters. Let $\mathcal{M}$ be the message space of the scheme and let $\mathsf{H}_q : \mathcal{M} \to \{0,1\}^\kappa$ be a hash function that maps messages to bit strings of length κ such that $2^\kappa < q$. Apart from the description of the groups, the common system parameters also contain the maximum number of time slots $T = 2^\ell$ and random group elements $h, h_0, \ldots, h_{\ell+1} \xleftarrow{\$} \mathbb{G}_1$. These parameters could, for example, be generated as the output of a hash function modeled as a random oracle.

Key generation. Each signer chooses $x \xleftarrow{\$} \mathbb{Z}_q$ and computes $y \leftarrow g_2^x$. It sets its public to $pk = y$ and computes its initial secret keys as $sk_0 \leftarrow h^x$.

Key update. Imagine the time periods $0, \ldots, 2^\ell - 1$ as being organized as the leaves of a binary tree of depth ℓ, sorted in increasing order from left to right. Then one can see that the path from the root of the tree to leaf node t is simply the binary representation of $t = t_1 \ldots t_\ell$, where $t_i = 0$ means taking the left branch at level i, and $t_i = 1$ means taking the right branch.

In the same way, each internal node w in the tree can be identified by a bit string $w_1 \ldots w_i$ describing the path from the root, where $i = |w|$ is the level of the node in the tree. Let's

associate to each node w a key of the form

$$(c, d, e_{i+1}, \ldots, e_{\ell+1}) \;=\; \left(g_2^r \;,\; h^x (h_0 \prod_{j=1}^{i} h_j^{w_j})^r \;,\; h_{i+1}^r \;,\; \ldots \;,\; h_{\ell+1}^r \right)$$

for $r \xleftarrow{\$} \mathbb{Z}_q$. Given the key of an internal node $w = w_1 \ldots w_i$, one can derive a key for a descendant node $w' = w_1 \ldots w_k$ for $k > i$ as

$$(c', d', e'_{k+1}, \ldots, e'_{\ell+1}) \;=\; \left(c \cdot g_2^{r'} \;,\; d \cdot \prod_{j=i+1}^{k} e_j^{w_j} \cdot (h_0 \prod_{j=1}^{k} h_j^{w_j})^{r'} \;,\; e_{k+1} \cdot h_{k+1}^{r'} \;,\; \ldots \;,\; e_{\ell+1} \cdot h_{\ell+1}^{r'} \right)$$

for $r' \xleftarrow{\$} \mathbb{Z}_q$.

Let the *minimal cover* of leaves $\{t, \ldots, T-1\}$ be defined as the smallest subset of nodes so that the subset contains an ancestor of all leaves in $\{t, \ldots, T-1\}$, but doesn't contain any ancestor of any leaf in $\{0, \ldots, t-1\}$. The secret key sk_t at time period t then contains keys corresponding to all nodes in the minimal cover C_t of $\{t, \ldots, T-1\}$. To update sk_t to the secret key sk_{t+1} for time period $t+1$, the signer determines the cover C_{t+1} of $\{t+1, \ldots, T-1\}$, derives keys for all nodes in $C_{t+1} \setminus C_t$ using the keys in sk_t, and securely deletes all re-randomization exponents r' as well as all keys in $C_t \setminus C_{t+1}$.

Signing. To generate a signature on message $M \in \{0,1\}^*$ in time period $t \in \mathbb{Z}_T$, the signer derives from the keys in sk_t a key $(c, d, e_{\ell+1})$ for the leaf node $t = t_1 \ldots t_\ell$. It then chooses $r' \xleftarrow{\$} \mathbb{Z}_q$ and outputs

$$(\sigma_1, \sigma_2) \;=\; \left(d \cdot e_{\ell+1}^{\mathsf{H}_q(M)} \cdot \big(h_0 \cdot \prod_{j=1}^{\ell} h_j^{t_j} \cdot h_{\ell+1}^{\mathsf{H}_q(M)}\big)^{r'} \;,\; c \cdot g_2^{r'} \right) .$$

Verification. Anyone can verify a signature $(\sigma_1, \sigma_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ on message M under public key $pk = y$ in time period t by checking whether

$$\mathrm{e}(\sigma_1, g_2) \;=\; \mathrm{e}(h, y) \cdot \mathrm{e}\big(h_0 \cdot \prod_{j=1}^{\ell} h_j^{t_j} \cdot h_{\ell+1}^{\mathsf{H}_q(M)} \;,\; \sigma_2\big) \;.$$

Efficiency. A signature contains one element of $\mathbb{G}_1$ and one element of $\mathbb{G}_2$, or 32+48=80 bytes on a BLS12-381 curve. The secret key contains at most one node key at every level $d = 1, \ldots, \ell$ at any given time, and the key at level d contains one element in $\mathbb{G}_2$ and $\ell - d + 2$ elements in $\mathbb{G}_1$, summing up to at most ℓ elements in $\mathbb{G}_2$ and $\ell^2/2 + 3\ell/2$ elements in $\mathbb{G}_1$ in total. For the BLS12-381 curve and $T = 2^{30}$ time periods, this comes down to at most 16800 bytes of secret key material.

Signing efficiency depends on the structure of the node keys in sk_t, but when the key for the leaf node t is precomputed, it takes one simple exponentiation in $\mathbb{G}_1$, one 2-multi-exponentiation in $\mathbb{G}_1$, and one exponentiation in $\mathbb{G}_2$. By precomputing

$$\begin{aligned} \sigma_{1,1} \;&\leftarrow\; d \cdot \big(h_0 \cdot \prod_{j=1}^{\ell} h_j^{t_j}\big)^{r'} \\ \sigma_{1,2} \;&\leftarrow\; e_{\ell+1} \cdot h_{\ell+1}^{r'} \\ \sigma_2 \;&\leftarrow\; c \cdot g_2^{r'} \;, \end{aligned}$$

the signature can be computed as $\sigma_1 \leftarrow \sigma_{1,1} \cdot \sigma_{1,2}^{\mathsf{H}_q(M)}$ once the message M is known, bringing the online computation down to a single exponentiation. Verification takes one exponentiation in $\mathbb{G}_1$ and three pairings (or one 3-multi-pairing). Key update can take up to ℓ exponentiations in $\mathbb{G}_2$ and $\ell^2/2 + 3\ell/2$ exponentiations in $\mathbb{G}_1$, or 30 exponentiations in $\mathbb{G}_2$ and 495 exponentiations in $\mathbb{G}_1$ for $T = 2^{30}$ time periods. Key updates can of course be entirely precomputed, if necessary.

3.3 Security

Theorem 1. *For any* fu-cma *adversary $\mathcal{A}$ against the above forward-secure signature scheme in the random-oracle model for $T = 2^\ell$ time periods, there exists an adversary $\mathcal{B}$ with essentially the same running time and advantage in solving the $(\ell+1)$-wBDHI$_3^*$ problem*

$$\mathbf{Adv}^{(\ell+1)\text{-wBDHI}_3^*}_{\mathbb{G}_1 \times \mathbb{G}_2}(\mathcal{B}) \;\geq\; \frac{1}{T \cdot q_{\mathrm{H}}} \cdot \mathbf{Adv}^{\text{fu-cma}}_{\mathcal{FS}}(\mathcal{A}) - \frac{q_{\mathrm{H}}^2}{2^\kappa} \;,$$

where q_{H} is the number of random-oracle queries made by $\mathcal{A}$.

Proof. We prove the theorem by first proving that the scheme is selectively secure when the message space $\mathcal{M} = \{0,1\}^\kappa$ and H_q is the identity function, meaning, interpreting a κ-bit string as an integer in $\mathbb{Z}_q$. Full fu-cma security for $\mathcal{M} = \{0,1\}^*$ and with $\mathsf{H}_q : \mathcal{M} \to \{0,1\}^\kappa$ modeled as a random oracle then follows because, given an fu-cma adversary $\mathcal{A}$ in the random-oracle model, one can build a sfu-cma adversary $\mathcal{A}'$ that guesses the time period t^* and the index of $\mathcal{A}$'s random-oracle query for $\mathsf{H}_q(M^*)$, and sets $\bar{t} \leftarrow t^* + 1$. If $\mathcal{A}'$ correctly guesses t^*, then it can use $sk_{\bar{t}}$ to simulate $\mathcal{A}$'s signature, key update, and break-in queries after time $\bar{t}$ until $\mathcal{A}$'s choice of break-in time $\bar{t}'$, at which point it can hand over $sk_{\bar{t}'}$.

If $\mathcal{A}'$ moreover correctly guessed the index of $\mathsf{H}_q(M^*)$, and if $\mathcal{A}$ never made colliding queries $\mathsf{H}_q(M) = \mathsf{H}_q(M')$ for $M \neq M'$, then $\mathcal{A}$'s forgery is also a valid forgery for $\mathcal{A}'$. Note that for $\mathcal{A}$ to be successful, it must hold that $\bar{t}' > t^*$, so it must hold that $\bar{t}' \geq \bar{t}$. The advantage of $\mathcal{A}'$ is given by

$$\mathbf{Adv}^{\text{sfu-cma}}_{\mathcal{FS}}(\mathcal{A}') \;\geq\; \frac{1}{T \cdot q_{\mathrm{H}}} \cdot \mathbf{Adv}^{\text{fu-cma}}_{\mathcal{FS}}(\mathcal{A}) - \frac{q_{\mathrm{H}}^2}{2^\kappa} \;, \tag{1}$$

where q_{H} is an upper bound on $\mathcal{A}$'s number of random-oracle queries.

We show that the scheme with message space $\mathcal{M} = \{0,1\}^\kappa$ and H_q the identity function is sfu-cma-secure under the $(\ell+1)$-wBDHI$_3^*$ assumption by describing an algorithm $\mathcal{B}$ that, given a successful sfu-cma forger $\mathcal{A}'$, solves the $(\ell+1)$-wBDHI$_3^*$ problem. On input $(A_1 = g_1^\alpha, A_2 = g_1^{(\alpha^2)}, \ldots, A_{\ell+1} = g_1^{(\alpha^{\ell+1})}, A_1' = g_2^\alpha, B)$, algorithm $\mathcal{B}$ proceeds as follows.

It first runs $\mathcal{A}$ to obtain $(\bar{t}, t^*, M^*)$. It then sets the public key and public parameters as

$$\begin{aligned}
y &\leftarrow A_1' \\
h &\leftarrow g_1^\gamma \cdot A_{\ell+1} \\
h_0 &\leftarrow g_1^{\gamma_0} \cdot \prod_{i=1}^{\ell} A_{\ell-i+2}^{-t_i^*} \cdot A_1^{-M^*} \\
h_i &\leftarrow g_1^{\gamma_i} \cdot A_{\ell-i+2} \qquad\qquad \text{for } i = 1, \ldots, \ell+1 \;,
\end{aligned}$$

where $\gamma, \gamma_0, \ldots, \gamma_{\ell+1} \xleftarrow{\$} \mathbb{Z}_q$. By setting the parameters as such, $\mathcal{B}$ implicitly sets $x = \alpha$ and $sk_0 = h^x = A_1^\gamma \cdot g_1^{(\alpha^{\ell+2})}$. The goal of the reduction is to extract the value of h^x from $\mathcal{A}'$, allowing $\mathcal{B}$ to easily compute its $(\ell+1)$-wBDHI$_3^*$ solution $\mathrm{e}(g_1, B)^{(\alpha^{\ell+2})}$.

Algorithm $\mathcal{B}$ responds to $\mathcal{A}'$'s oracle queries as follows.

Key update. There is no need for $\mathcal{B}$ to simulate anything beyond keeping track of the current time period t.

Signing. To answer a signing query for message M in time period $t \neq t^*$, $\mathcal{B}$ first internally derives a valid key $(c, d, e_{\ell+1})$ for the leaf corresponding to t, from which it can then compute a signature on M in the standard way. In fact, $\mathcal{B}$ derives a valid key $(c, d, e_{k+1}, \ldots, e_{\ell+1})$ for the k-th level ancestor of the leaf node t, where $k = \min_i(t_i \neq t_i^*)$ is the leftmost bit of t that is different from the same bit in t^*. (Note that k must exist and $1 \leq k \leq \ell$ because $t \neq t^*$.) From this key, $\mathcal{B}$ can further derive a key for the leaf node t and a signature for M in the standard way.

The key for the k-th level ancestor of t must have a structure

$$(c, d, e_{k+1}, \ldots, e_{\ell+1}) \;=\; \left(g_2^r \;,\; h^x \big(h_0 \prod_{i=1}^{k} h_i^{t_i}\big)^r \;,\; h_{k+1}^r \;,\; \ldots \;,\; h_{\ell+1}^r \right)$$

for a uniformly distributed value of r. Focusing on the second component d first, we have that

$$\begin{aligned} d \;&=\; h^x \cdot \left(h_0 \cdot \prod_{i=1}^{k} h_i^{t_i} \right)^r \\ &=\; (g_1^\gamma A_{\ell+1})^\alpha \cdot \left(\left(g_1^{\gamma_0} \cdot \prod_{i=1}^{\ell} A_{\ell-i+2}^{-t_i^*} \cdot A_1^{-M^*} \right) \cdot \prod_{i=1}^{k} \left(g_1^{\gamma_i} \cdot A_{\ell-i+2} \right)^{t_i} \right)^r \\ &=\; A_1^\gamma \cdot g_1^{(\alpha^{\ell+2})} \cdot \left(g_1^{\gamma_0 + \sum_{i=1}^{k} \gamma_i t_i} \cdot A_{\ell-k+2}^{t_k - t_k^*} \cdot \prod_{i=k+1}^{\ell} A_{\ell-i+2}^{-t_i^*} \cdot A_1^{-M^*} \right)^r \;, \end{aligned}$$

where the third equality holds because $t_i = t_i^*$ for $i < k$. (Note that in the product notation $\prod_{i=k+1}^{\ell}$ above, we let the result of the product simply be the unity element if $k \geq \ell$.) Let us denote the four factors between parentheses in the last equation as F_1, F_2, F_3, and F_4, and denote their product as F. If we let $r \leftarrow r' + \frac{\alpha^k}{t_k^* - t_k}$ for a random $r' \xleftarrow{\$} \mathbb{Z}_q$, then we have that

$$d \;=\; A_1^\gamma \cdot g_1^{(\alpha^{\ell+2})} \cdot F^{r'} \cdot F^{\frac{\alpha^k}{t_k^* - t_k}} \;.$$

The first and third factors in this product are easy to compute. The second factor would allow $\mathcal{B}$ to compute the solution its $(\ell+1)$-wBDHI$_3^*$ problem as $\mathrm{e}(g_1^{(\alpha^{\ell+2})}, B)$, so $\mathcal{B}$ cannot

simply compute it. The last factor can be split up as

$$\begin{aligned}
B_1^{\frac{\alpha^k}{t_k^*-t_k}} &= A_k^{\frac{\gamma_0+\sum_{i=1}^{k}\gamma_i t_i}{t_k^*-t_k}} \\
B_2^{\frac{\alpha^k}{t_k^*-t_k}} &= A_{\ell-k+2}^{-\alpha^k} = g_1^{-(\alpha^{\ell+2})} \\
B_3^{\frac{\alpha^k}{t_k^*-t_k}} &= \prod_{i=k+1}^{\ell} A_{\ell+k-i+2}^{\frac{-t_i^*}{t_k^*-t_k}} = \prod_{i=1}^{\ell-k} A_{\ell-i+2}^{\frac{-t_{k+i}^*}{t_k^*-t_k}} \\
B_4^{\frac{\alpha^k}{t_k^*-t_k}} &= A_{k+1}^{\frac{-M^*}{t_k^*-t_k}} \ .
\end{aligned}$$

Because $1 \leq k \leq \ell$, it is clear that all but the second of these can be computed from $\mathcal{B}$'s inputs, and that the second cancels out with the factor $g_1^{(\alpha^{\ell+2})}$ in d, so that it can indeed compute d this way. The other components of the key are also efficiently computable as

$$\begin{aligned}
c &= g_1^{r'} \cdot A_k^{\frac{1}{t_k^*-t_k}} \\
e_i &= h_i^{r'} \cdot A_{\ell+k-i+2} \quad \text{for } i = k+1, \ldots, \ell+1 \\
&= h_{k+i}^{r'} \cdot A_{\ell-i+2} \quad \text{for } i = 1, \ldots, \ell-k+1 \ .
\end{aligned}$$

For a signing query with $t = t^*$ but $M \neq M^*$, $\mathcal{B}$ proceeds in a similar way, but derives the signature (σ_1, σ_2) directly. Algorithm $\mathcal{B}$ can generate a valid signature using a similar approach as we used above to derive a key at level k, because a signature is essentially a key at level $\ell+1$. Namely, $\mathcal{B}$ computes a signature

$$\begin{aligned}
\sigma_1 &= h^x \cdot \left(h_0 \cdot \prod_{i=1}^{\ell} h_i^{t_i^*} \cdot h_{\ell+1}^M \right)^r \\
&= (g_1^\gamma A_{\ell+1})^\alpha \cdot \left(\left(g_1^{\gamma_0} \cdot \prod_{i=1}^{\ell} A_{\ell-i+2}^{-t_i^*} \cdot A_1^{-M^*} \right) \cdot \prod_{i=1}^{\ell} \left(g_1^{\gamma_i} \cdot A_{\ell-i+2} \right)^{t_i^*} \cdot (g_1^{\gamma_{\ell+1}} \cdot A_1)^M \right)^r \\
&= A_1^\gamma \cdot g_1^{(\alpha^{\ell+2})} \cdot \left(g_1^{\gamma_0 + \sum_{i=1}^{\ell} \gamma_i t_i + \gamma_{\ell+1} M} \cdot A_1^{M-M^*} \right)^r \\
\sigma_2 &= g_2^r
\end{aligned}$$

by setting $r \leftarrow r' + \frac{\alpha^{\ell+1}}{M^*-M}$ for $r' \xleftarrow{\$} \mathbb{Z}_q$, so that $\mathcal{B}$ can compute (σ_1, σ_2) from its inputs $A_1, \ldots, A_{\ell+1}$.

Break in. The only nodes in the tree for which $\mathcal{B}$ cannot simulate valid keys using the above approach are nodes that correspond to prefixes of t^*. These nodes are all ancestors of leaf t^*, however, and therefore cannot be present in the minimal cover of leaf $\bar{t} > t^*$. So $\mathcal{B}$ can simulate $sk_{\bar{t}}$ by deriving the needed node keys in the same way as above.

Forgery. When $\mathcal{A}'$ outputs a forgery (σ_1^*, σ_2^*) that satisfies the verification equation

$$\mathrm{e}(\sigma_1^*, g_2) = \mathrm{e}(h, y) \cdot \mathrm{e}\big(h_0 \cdot \prod_{j=1}^{\ell} h_j^{t_j} \cdot h_{\ell+1}^M \ , \ \sigma_2^*\big) \ ,$$

then there exists an $r \in \mathbb{Z}_q$ such that

$$\begin{aligned} \sigma_1^* &= h^\alpha \cdot \big(h_0 \cdot \prod_{i=1}^{\ell} h_i^{t_i^*} \cdot h_{\ell+1}^M\big)^r \\ \sigma_2^* &= g_2^r \ . \end{aligned}$$

By the way that $\mathcal{B}$ chose the parameters $h, h_0, \ldots, h_{\ell+1}$, this means that

$$\sigma_1^* = A_1^\gamma \cdot g_1^{(\alpha^{\ell+2})} \cdot (g_1^r)^{\gamma_0 + \sum_{i=1}^{\ell} \gamma_i t_i^* + \gamma_{\ell+1} M^*} \ ,$$

so that we have that

$$\mathrm{e}(\sigma_1^*, B) = \mathrm{e}(A_1^\gamma, B) \cdot \mathrm{e}(g_1^{(\alpha^{\ell+2})}, B) \cdot \mathrm{e}(g_1, \sigma_2^*)^{\gamma_0 + \sum_{i=1}^{\ell} \gamma_i t_i^* + \gamma_{\ell+1} M^*} \ ,$$

from which $\mathcal{B}$ can easily compute and output its response as $\mathrm{e}(g_1, B)^{(\alpha^{\ell+2})}$. It does so whenever $\mathcal{A}'$ is successful, so that

$$\mathbf{Adv}^{(\ell+1)\text{-wBDHI}^*_3}_{\mathbb{G}_1 \times \mathbb{G}_2}(\mathcal{B}) \geq \mathbf{Adv}^{\text{sfu-cma}}_{\mathcal{FS}}(\mathcal{A}') \ .$$

Together with Equation (1), we obtain the inequality of the theorem statement.

□

4 Forward-Secure Multi-Signatures

The easiest way to turn the forward-secure signature scheme from the previous section into a multi-signature scheme is to observe that the component-wise product $(\Sigma_1, \Sigma_2) = (\prod_{i=1}^n \sigma_{i,1}, \prod_{i=1}^n \sigma_{i,2})$ of a number of signatures $(\sigma_{1,1}, \sigma_{1,2}), \ldots, (\sigma_{n,1}, \sigma_{n,2})$ satisfies the verification equation with respect of the product of public keys $Y = y_1 \cdot \ldots \cdot y_n$. This method of combining signatures is vulnerable to a rogue-key attack, however, where a malicious signer chooses his public key based on that of an honest signer, so that the malicious signer can compute valid signatures for their aggregated public key. The scheme below borrows a technique due to Ristenpart and Yilek [RY07] using proofs of possession to prevent against these types of attack.

4.1 Definitions

In addition to the algorithms of a forward-secure signature scheme in Section 3.1, a forward-secure multi-signature scheme $\mathcal{FMS}$ adds the following algorithms.

Key aggregation: $apk \xleftarrow{\$} \mathsf{KAgg}(pk_1, \ldots, pk_n)$. On input a list of individual public keys $(pk_1, \ldots, pk_n)$, the key aggregation returns an aggregate public key apk, or $\perp$ to indicate that key aggregation failed.

Signature aggregation. $\Sigma \xleftarrow{\$} \mathsf{SAgg}((pk_1, \sigma_1), \ldots, (pk_n, \sigma_n), t, M)$. Anyone can aggregate a given list of individual signatures $(\sigma_1, \ldots, \sigma_n)$ by different signers with public keys $(pk_1, \ldots, pk_n)$ on the same message M and for the same period t into a single multi-signature Σ.

Aggregate verification. $b \leftarrow \mathsf{AVf}(apk, t, M, \Sigma)$. Given an aggregate public key apk, a message M, a time period t, and an aggregate signature Σ, the verification algorithm returns 1 to indicate that all signers in apk signed M in period t, or 0 to indicate that verification failed.

Correctness requires that for all messages $M \in \{0,1\}^*$, for all $n \in \mathbb{Z}$, and for all time periods $t \in \{0, \ldots, T-1\}$, it holds that $\mathsf{AVf}(\mathsf{KAgg}(pk_1, \ldots, pk_n), t, M, \mathsf{SAgg}((pk_1, \mathsf{Sign}(sk_{1,t}, M)), \ldots, (pk_n, \mathsf{Sign}(sk_{n,t})), t, M)) = 1$ with probability one if $(pk_i, sk_{i,0}) \xleftarrow{\$} \mathsf{Kg}$ and $ski, j+1 \xleftarrow{\$} \mathsf{Upd}(sk_{i,j})$ for $i = 1, \ldots, n$ and $j = 0, \ldots, t-1$.

Unforgeability (fu-cma) is defined through a game that is similar to that described in Section 3.1. The adversary is given the public key pk of an honest signer and access to the same key update, signing, and break-in oracles. However, at the end of the game, the adversary's forgery consists of a list of public keys $(pk_1^*, \ldots, pk_n^*)$, a message M^*, a time period t^*, and a multi-signature Σ^*. The forgery is considered valid if

- $pk \in \{pk_1^*, \ldots, pk_n^*\}$,
- Σ^* is valid with respect to the aggregate public key apk^* of $(pk_1^*, \ldots, pk_n^*)$, message M^*, and time period t^*,
- $\bar{t} > t^*$,
- and $\mathcal{A}$ never made a signing query for M^* during time period t^*.

4.2 Construction

Let $\mathsf{H}_{\mathbb{G}_1} : \{0,1\}^* \to \mathbb{G}_1^*$ be a hash function. The multi-signature scheme reuses the key update and signature algorithms from the scheme from Section 3.2, but uses different key generation and verification algorithms, and adds signature and key aggregation.

Key generation. Each signer chooses $x \xleftarrow{\$} \mathbb{Z}_q$ and computes $y \leftarrow g_2^x$ and $y' \leftarrow \mathsf{H}_{\mathbb{G}_1}(\mathtt{PoP}, y)$, where $\mathtt{PoP}$ is a fixed string used as a prefix for domain separation. It sets its public key to $pk = (y, y')$ and computes its initial secret key as $sk_0 \leftarrow h^x$.

Key aggregation. Given public keys $pk_1 = (y_1, y_1'), \ldots, (y_n, y_n')$, the key aggregation algorithm first validates the proofs of possession in the public keys by checking whether

$$\mathrm{e}(y_i', g_2) \;=\; \mathrm{e}(\mathsf{H}_{\mathbb{G}_1}(\mathtt{PoP}, y_i), y_i)$$

for $i = 1, \ldots, n$, or equivalently, whether

$$\mathrm{e}\Big(\prod_{i=1}^{n} y_1', g_2\Big) \;=\; \prod_{i=1}^{n} \mathrm{e}(\mathsf{H}_{\mathbb{G}_1}(\mathtt{PoP}, y_i), y_i)\ .$$

Of course, these verifications only need to be performed once for each key; after that, the key can be marked as valid and used in more key aggregation without verifying again. If these checks pass, then it computes $Y \leftarrow \prod_{i=1}^{n} y_i$ and returns the aggregate public key $apk = y$. If not, it returns $\perp$.

Signature aggregation. Given signatures $(\sigma_{1,1}, \sigma_{1,2}), \ldots, (\sigma_{n,1}, \sigma_{n,2}) \in \mathbb{G}_1 \times \mathbb{G}_2$ on the same message M, the signature aggregation algorithm outputs

$$(\Sigma_1, \Sigma_2) \;=\; \big(\prod_{i=1}^{n} \sigma_{i,1} \;,\; \prod_{i=1}^{n} \sigma_{i,2}\big) \;.$$

Aggregate verification. Given an aggregate signature $(\Sigma_1, \Sigma_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ on message M under aggregate public key $apk = Y$ in time period t, the verifier accepts if and only if $apk \neq \perp$ and

$$\mathrm{e}(\Sigma_1, g_2) \;=\; \mathrm{e}(h, Y) \cdot \mathrm{e}\big(h_0 \cdot \prod_{j=1}^{\ell} h_j^{t_j} \cdot h_{\ell+1}^{\mathsf{H}_q(M)} \;,\; \Sigma_2\big) \;.$$

4.3 Security

Theorem 2. *For any* fu-cma *adversary $\mathcal{A}$ against the above forward-secure multi-signature scheme for $T = 2^\ell$ time periods in the random-oracle model, there exists an adversary $\mathcal{B}$ with essentially the same running time that solves the $(\ell+1)$-wBDHI$^*_3$ problem with advantage*

$$\mathbf{Adv}^{(\ell+1)\text{-wBDHI}^*_3}_{\mathbb{G}_1 \times \mathbb{G}_2}(\mathcal{B}) \;\geq\; \frac{1}{T \cdot q_{\mathrm{H}}} \cdot \mathbf{Adv}^{\text{fu-cma}}_{\mathcal{FMS}}(\mathcal{A}) - \frac{q_{\mathrm{H}}^2}{2^\kappa} \;,$$

where q_{H} is the number of random-oracle queries made by $\mathcal{A}$.

Proof. We prove the theorem by showing that a successful forgery $\mathcal{A}$ against the multi-signature scheme yields a successful forger $\mathcal{A}'$ against the single-signer scheme of Section 3.2 such that

$$\mathbf{Adv}^{\text{fu-cma}}_{\mathcal{FS}}(\mathcal{A}') \;\geq\; \mathbf{Adv}^{\text{fu-cma}}_{\mathcal{FS}}(\mathcal{A}) \;.$$

The theorem then follows from Theorem 1.

On input the parameters $(T, h, h_0, \ldots, h_{\ell+1})$ and a public key y for the single-signer scheme, algorithm $\mathcal{A}'$ chooses $r \xleftarrow{\$} \mathbb{Z}_q^*$ and stores $(y, \perp, g_1^r)$ in a list L. It computes $y' \leftarrow y^r$ and runs $\mathcal{A}$ on the same common parameters and target public key $pk = (y, y')$. Algorithm $\mathcal{B}$ answers all of $\mathcal{A}$'s key update, signing, and break-in oracle queries, as well as random-oracle queries for H_q, by simply relaying queries and responses to and from $\mathcal{A}'$'s own oracles. Queries to the random oracle for $\mathsf{H}_{\mathbb{G}_1}$ are answered as follows.

Random oracle $\mathsf{H}_{\mathbb{G}_1}$. On input z, $\mathcal{A}'$ checks whether there already exists a tuple $(z, \cdot, v) \in L$. If so, it returns v. If not, it chooses $r \xleftarrow{\$} \mathbb{Z}_q^*$, computes $v \leftarrow h^r$, adds a tuple (z, r, v) to L and returns v.

When $\mathcal{A}$ outputs its forgery $(pk_1^*, \ldots, pk_n^*), M^*, t^*, \Sigma^*$, algorithm $\mathcal{A}'$ first computes the aggregate public key apk^* for $(pk_1^*, \ldots, pk_n^*)$, creating additional entries in L if necessary. Let $pk_i^* = (y_i, y_i')$ and $y_i = g_2^{x_i}$. If $apk^* = Y \neq \perp$, then it holds that $y_i' = \mathsf{H}_{\mathbb{G}_1}(\mathsf{PoP}, y_i)^{x_i}$ for all $i = 1, \ldots, n$. From the aggregate verification equation

$$\mathrm{e}(\Sigma_1^*, g_2) \;=\; \mathrm{e}(h, Y) \cdot \mathrm{e}\big(h_0 \cdot \prod_{j=1}^{\ell} h_j^{t_j^*} \cdot h_{\ell+1}^{\mathsf{H}_q(M^*)} \;,\; \Sigma_2^*\big)$$

and the fact that $Y = \prod_{i=1}^{n} y_i = y \cdot g_2^{\sum_{i=1, y_i \neq y}^{n} x_i}$, we have that

$$\mathrm{e}(\Sigma_1^*, g_2) \;=\; \mathrm{e}(h, y) \cdot \mathrm{e}(h, g_2)^{\sum_{i=1, y_i \neq y}^{n} x_i} \cdot \mathrm{e}\big(h_0 \cdot \prod_{j=1}^{\ell} h_j^{t_j^*} \cdot h_{\ell+1}^{\mathsf{H}_q(M^*)} \;,\; \Sigma_2^*\big)$$

$$\Leftrightarrow\; \mathrm{e}(\Sigma_1^* \cdot h^{-\sum_{i=1, y_i \neq y}^{n} x_i}, g_2) \;=\; \mathrm{e}(h, y) \cdot \mathrm{e}\big(h_0 \cdot \prod_{j=1}^{\ell} h_j^{t_j^*} \cdot h_{\ell+1}^{\mathsf{H}_q(M^*)} \;,\; \Sigma_2^*\big) \;.$$

For all $y_i \neq y$, $\mathcal{A}'$ looks up the tuple (y_i, r_i, v_i) in L. We know that $v_i = h^{r_i}$, and hence that $y_i' = h^{r_i x_i}$. By comparing the last equation above to the verification equation of the single-signer scheme, and by observing that $y_i' = h^{r_i x_i}$, we know that the pair

$$\begin{aligned} \sigma_1^* \;&\leftarrow\; \Sigma_1^* \cdot \prod_{i=1, y_i \neq y}^{n} {y_i'}^{-1/r_i} \\ \sigma_2^* \;&\leftarrow\; \Sigma_2^* \end{aligned}$$

is a valid forgery for the single-signer scheme, so $\mathcal{A}'$ outputs (σ_1^*, σ_2^*). □

5 Variants and Extensions

Using internal nodes. For ease of exposition, our scheme only assigns time periods to leaf nodes in the tree. Alternatively, one could follow the approach of [CHK07] to use all nodes in the tree, in a pre-order traversal, as time periods, which will improve the efficiency of the key update algorithm. Time periods are then identified by bit strings of length at most ℓ, rather than exactly ℓ bits, and a signature in time period $t = t_1 \ldots t_d$ is a tuple of the form

$$(\sigma_1, \sigma_2) \;=\; \left(h^x \cdot \big(h_0 \prod_{j=1}^{d} h_j^{t_j} \cdot h_{\ell+1}^{\mathsf{H}_q(M)}\big)^r \;,\; g_2^r\right) .$$

Details are left to the reader.

Non-binary trees. One could try to reduce the key size by using b-ary trees instead of binary trees. A larger value of b reduces the depth of the tree, but increases the amount of key material that must be kept at each level of the tree. To support T time periods, one needs a b-ary tree of depth $\ell = \lceil \log_b T \rceil$. A node key at level d, however, can now take up to $b - 1$ keys of one element in $\mathbb{G}_2$ and $(\ell + d - 2)$ elements of $\mathbb{G}_1$.

The savings effect is quite limited, however, because the disadvantage of needing more keys per level quickly starts dominating the advantage of having less levels. For practical values of T, the maximum size of the secret key will usually be minimal for $b = 3$.

Parallel key timelines. In some applications, a signer may want to maintain several parallel timelines for different usages of a signing key. For example, in a sharded blockchain, the shards may be running in parallel at different speeds, without strict synchronization between the shards. If a time frame of the forward-secure signature scheme corresponds to the block height of a blockchain, for example, then the signer needs to maintain a different key schedule for the different shards.

A trivial approach to "scope" a forward-secure signature scheme is of course to use certificates. More specifically, the signer generates a standard signature key pair (pk, sk) and one forward-secure signature key pair $(pk_{scope}, sk_{scope,0})$ for every scope that he plans to use. The signer creates and publishes certificates for all public keys pk_{scope} using sk and securely deletes sk. (Deleting sk is needed because otherwise an adversary, after breaking in, can re-generate and re-certify a fresh key pair $(pk_{scope}, sk_{scope,0})$ for an existing *scope*.)

A more efficient approach for our particular scheme is to replace the fixed common parameter h with the output of a hash function $\mathsf{H}_{\mathbb{G}_1}(\mathtt{scope}, scope)$. Meaning, during key generation, the signer generates $sk_{scope,0} \leftarrow \{\mathsf{H}_{\mathbb{G}_1}(\mathtt{scope}, scope)^x$ for all relevant scopes *scope* and deletes the master key x. It can then update, sign, and aggregate signatures for each scope separately in the same way as before, but substituting $\mathsf{H}_{\mathbb{G}_1}(\mathtt{scope}, scope)$ for h. Verification of individual signatures and of multi-signatures is also the same as before, substituting $\mathsf{H}_{\mathbb{G}_1}(\mathtt{scope}, scope)$ for h.

Tighter security. The loss in tightness in Equation (1) of $T \cdot q_{\mathrm{H}}$ can be brought down to $T \cdot q_{\mathrm{S}}$ using Coron's technique [Cor00], where q_{S} is the number of signing queries made by the adversary $\mathcal{A}$, by hashing the message into $\mathbb{G}_1$ instead of into $\mathbb{Z}_q$. Namely, a multi-signature would be a tuple $(\Sigma_1, \Sigma_2, \Sigma_3)$ satisfying

$$\mathrm{e}(\Sigma_1, g_2) \;=\; \mathrm{e}(h, Y) \cdot \mathrm{e}\big(h_0 \cdot \prod_{j=1}^{\ell} h_j^{t_j} \;,\; \Sigma_2\big) \cdot \mathrm{e}\big(\mathsf{H}_{\mathbb{G}_1}(\mathtt{msg}, M), \Sigma_3\big) \;.$$

This scheme has the additional advantage of saving up to ℓ elements of $\mathbb{G}_1$ in secret key size, but signatures are one element of $\mathbb{G}_2$ longer than the base scheme. We leave details to the reader.

Alternative key aggregation mechanisms. In situations where public key length is critical, one could alternatively reuse techniques from [MPSW18, BDN18b] where signers' public keys are simply given by $pk_i = y_i = g_2^{x_i}$, but the aggregate public key is computed as $apk \leftarrow \prod_{i=1}^{n} pk_i^{\mathsf{H}_q(\{pk_1,\ldots,pk_n\},pk_i)}$. Individual signatures $(\sigma_{1,1}, \sigma_{1,2}), \ldots, (\sigma_{n,1}, \sigma_{n,2})$ are aggregated as

$$(\Sigma_1, \Sigma_2) \;\leftarrow\; \big(\prod_{i=1}^{n} \sigma_{i,1}^{\mathsf{H}_q(\{pk_1,\ldots,pk_n\},pk_i)} \;,\; \prod_{i=1}^{n} \sigma_{i,2}^{\mathsf{H}_q(\{pk_1,\ldots,pk_n\},pk_i)}\big) \;,$$

so that verification can be performed as usual.

Partial aggregation of multi-signatures. Further savings in terms of signature length can be obtained by partially aggregating multi-signatures. Multi-signatures $(\Sigma_{i,1}, \Sigma_{i,2})$ under aggregate public keys $apk_i = Y_i$ on messages M_i for time periods t_i for $i = 1, \ldots, n$, can be compressed into an aggregate multi-signature $(\Sigma_1, \Sigma_{1,2}, \ldots, \Sigma_{n,2})$ where $\Sigma_1 \leftarrow \prod_{i=1}^{n} \Sigma_{i,1}$, which can be verified by checking that

$$\mathrm{e}(\Sigma_1, g_2) \;=\; \mathrm{e}(h, \prod_{i=1}^{n} Y_i) \cdot \prod_{i=1}^{n} \mathrm{e}\big(h_0 \cdot \prod_{j=1}^{\ell} h_j^{t_{i,j}} \cdot h_{\ell+1}^{\mathsf{H}_q(M_i)} \;,\; \Sigma_{i,2}\big) \;.$$

Care must be taken, however, that either the messages M_i are all different, or that all aggregate public keys $apk_1, \ldots, apk_n$ are "trusted", in the sense that the verifier checks that they are composed of individual public keys with valid proofs of possession. One could enforce the messages M_i to be

all different by including the aggregate public key in the message $M_i = apk_i \| M'_i$, but this has the disadvantage that the aggregate public key (and hence, the set of signers in the aggregate) must be known at the time of signing. Failure to follow these precautions makes the scheme insecure, because for a given aggregate public key apk_1 it is easy to come up with a "rogue" key $apk_2 = g_2^x / apk_1$ that allows an adversary to forge an aggregate signature on any message under apk_1 and apk_2.

Acknowledgements

We would like to thank Jens Groth for his useful feedback.

References

[And00] Ross Anderson. Two remarks on public-key cryptology. Manuscript. Relevant material presented by the author in an invited lecture at the 4th ACM Conference on Computer and Communications Security, CCS 1997, Zurich, Switzerland, April 1–4, 1997, September 2000.

[AR00] Michel Abdalla and Leonid Reyzin. A new forward-secure digital signature scheme. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 116–129, Kyoto, Japan, December 3–7, 2000. Springer, Heidelberg, Germany.

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.

[BBG05a] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany.

[BBG05b] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. Cryptology ePrint Archive, Report 2005/015, 2005. `http://eprint.iacr.org/2005/015`.

[BCJ08] Ali Bagherzandi, Jung Hee Cheon, and Stanislaw Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 08: 15th Conference on Computer and Communications Security*, pages 449–458, Alexandria, Virginia, USA, October 27–31, 2008. ACM Press.

[BDN18a] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 435–464, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany.

[BDN18b] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. Cryptology ePrint Archive, Report 2018/483, 2018. `https://eprint.iacr.org/2018/483`.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany.

[BGOY07] Alexandra Boldyreva, Craig Gentry, Adam O'Neill, and Dae Hyun Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 07: 14th Conference on Computer and Communications Security*, pages 276–285, Alexandria, Virginia, USA, October 28–31, 2007. ACM Press.

[BGR12] Kyle Brogle, Sharon Goldberg, and Leonid Reyzin. Sequential aggregate signatures with lazy verification from trapdoor permutations - (extended abstract). In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 644–662, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany.

[BM99] Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–448, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany.

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06: 13th Conference on Computer and Communications Security*, pages 390–399, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press.

[BN07] Mihir Bellare and Gregory Neven. Identity-based multi-signatures from RSA. In Masayuki Abe, editor, *Topics in Cryptology – CT-RSA 2007*, volume 4377 of *Lecture Notes in Computer Science*, pages 145–162, San Francisco, CA, USA, February 5–9, 2007. Springer, Heidelberg, Germany.

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46, Miami, FL, USA, January 6–8, 2003. Springer, Heidelberg, Germany.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on*

Computer and Communications Security, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[BS16] Rachid El Bansarkhani and Jan Sturm. An efficient lattice-based multisignature scheme with applications to bitcoins. In Sara Foresti and Giuseppe Persiano, editors, *CANS 16: 15th International Conference on Cryptology and Network Security*, volume 10052 of *Lecture Notes in Computer Science*, pages 140–155, Milan, Italy, November 14–16, 2016. Springer, Heidelberg, Germany.

[BSSW06] Xavier Boyen, Hovav Shacham, Emily Shen, and Brent Waters. Forward-secure signatures with untrusted update. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06: 13th Conference on Computer and Communications Security*, pages 191–200, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press.

[CFH+07] Yang Cui, Eiichiro Fujisaki, Goichiro Hanaoka, Hideki Imai, and Rui Zhang. Formal security treatments for signatures from identity-based encryption. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec 2007: 1st International Conference on Provable Security*, volume 4784 of *Lecture Notes in Computer Science*, pages 218–227, Wollongong, Australia, November 1–2, 2007. Springer, Heidelberg, Germany.

[CHK07] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. *Journal of Cryptology*, 20(3):265–294, July 2007.

[Cor00] Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235, Santa Barbara, CA, USA, August 20–24, 2000. Springer, Heidelberg, Germany.

[DEF+18] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. On the security of two-round multi-signatures. Cryptology ePrint Archive, Report 2018/417, 2018. `https://eprint.iacr.org/2018/417`.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[GOR18] Craig Gentry, Adam O'Neill, and Leonid Reyzin. A unified framework for trapdoor-permutation-based sequential aggregate signatures. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 10770 of *Lecture Notes in Computer Science*, pages 34–57, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany.

[GS02] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566, Queenstown, New Zealand, December 1–5, 2002. Springer, Heidelberg, Germany.

[HZ93] Thomas Hardjono and Yuliang Zheng. A practical digital multisignature scheme based on discrete logarithms. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology – AUSCRYPT'92*, volume 718 of *Lecture Notes in Computer Science*,

pages 122–132, Gold Coast, Queensland, Australia, December 13–16, 1993. Springer, Heidelberg, Germany.

[IN83] K. Itakura and K. Nakamura. A public-key cryptosystem suitable for digital multisignatures. Technical report, NEC Research and Development, 1983.

[IR01] Gene Itkis and Leonid Reyzin. Forward-secure signatures with optimal signing and verifying. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 332–354, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

[KR03] Anton Kozlov and Leonid Reyzin. Forward-secure signatures with fast key update. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02: 3rd International Conference on Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 241–256, Amalfi, Italy, September 12–13, 2003. Springer, Heidelberg, Germany.

[Kra00] Hugo Krawczyk. Simple forward-secure signatures from any signature scheme. In S. Jajodia and P. Samarati, editors, *ACM CCS 00: 7th Conference on Computer and Communications Security*, pages 108–115, Athens, Greece, November 1–4, 2000. ACM Press.

[LBG09] Duc-Phong Le, Alexis Bonnecaze, and Alban Gabillon. Multisignatures as secure as the Diffie-Hellman problem in the plain public-key model. In Hovav Shacham and Brent Waters, editors, *PAIRING 2009: 3rd International Conference on Pairing-based Cryptography*, volume 5671 of *Lecture Notes in Computer Science*, pages 35–51, Palo Alto, CA, USA, August 12–14, 2009. Springer, Heidelberg, Germany.

[LHL95] Chuan-Ming Li, Tzonelih Hwang, and Narn-Yih Lee. Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders. In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 194–204, Perugia, Italy, May 9–12, 1995. Springer, Heidelberg, Germany.

[LMRS04] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential aggregate signatures from trapdoor permutations. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 74–90, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.

[LOS+06] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 465–485, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany.

[MMM02] Tal Malkin, Daniele Micciancio, and Sara K. Miner. Efficient generic forward-secure signatures with an unbounded number of time periods. In Lars R. Knudsen, editor,

Advances in Cryptology – EUROCRYPT 2002, volume 2332 of *Lecture Notes in Computer Science*, pages 400–417, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Heidelberg, Germany.

[MOR01] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures: Extended abstract. In *ACM CCS 01: 8th Conference on Computer and Communications Security*, pages 245–254, Philadelphia, PA, USA, November 5–8, 2001. ACM Press.

[MPSW18] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple schnorr multi-signatures with applications to bitcoin. Cryptology ePrint Archive, Report 2018/068, 2018. `https://eprint.iacr.org/2018/068`.

[MWLD10] Changshe Ma, Jian Weng, Yingjiu Li, and Robert H. Deng. Efficient discrete logarithm based multi-signature scheme in the plain public key model. *Des. Codes Cryptography*, 54(2):121–133, 2010.

[Nev08] Gregory Neven. Efficient sequential aggregate signed data. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 52–69, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany.

[OO93] Kazuo Ohta and Tatsuaki Okamoto. A digital multisignature scheme based on the Fiat-Shamir scheme. In Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto, editors, *Advances in Cryptology – ASIACRYPT'91*, volume 739 of *Lecture Notes in Computer Science*, pages 139–148, Fujiyoshida, Japan, November 11–14, 1993. Springer, Heidelberg, Germany.

[OO99] Kazuo Ohta and Tatsuaki Okamoto. Multi-signature schemes secure against active insider attacks. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 82(1):21–31, 1999.

[Poe19] Andrew Poelstra. Musig: A new multisignature standard, 2019. `https://blockstream.com/2019/02/18/musig-a-new-multisignature-standard/`.

[RY07] Thomas Ristenpart and Scott Yilek. The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 228–245, Barcelona, Spain, May 20–24, 2007. Springer, Heidelberg, Germany.

[SA09] N. R. Sunitha and B. B. Amberker. Forward-secure multi-signatures. In Manish Parashar and Sanjeev K. Aggarwal, editors, *Distributed Computing and Internet Technology, 5th International Conference, ICDCIT 2008*, volume 5375 of *Lecture Notes in Computer Science*. Springer, 2009.

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany.