Digital Signatures for Consensus

Sergey Gorbunov* Hoeteck Wee†

Algorand

Abstract

We present a pairing-based signature scheme for use in blockchains that achieves substantial savings in bandwidth and storage requirements while providing strong security guarantees. Our signature scheme supports aggregation on the same message, which allows us to compress multiple signatures on the same block during consensus, and achieves forward security, which prevents adaptive attacks on the blockchain. Our signature scheme can be applied to all blockchains that rely on multi-party consensus protocols to agree on blocks of transactions (such as proof-of-stake or permissioned blockchains).

1 Introduction

Blockchain technologies are quickly gaining popularity for payments, financial applications, and other distributed applications. A blockchain is an append-only public ledger to which anyone can write and read. At the core of the blockchains is a consensus mechanism that allows nodes to agree on changes to the ledger, while ensuring that changes once confirmed cannot be altered; we refer to the latter as the safety requirement. The key question in any blockchain design is: "How to choose and agree on the next block?"

In the first generation of blockchain implementations, such as Bitcoin, Ethereum, Litecoin, the nodes with the largest computational resources choose the next block. These implementations suffer from large computational waste, high transaction costs, low throughput, high latency, and centralization due to the formation of mining pools [25, 7, 13]. To overcome these problems, the current generation of blockchain implementations such as Algorand, Cardano, Ethereum Casper and Dfinity turn to proofs of stake (PoS), where nodes with larger stakes in the system —as measured for instance by the amount of money in their account— are more likely to participate in choosing the next block [24, 16, 11, 20, 14, 9, 18].

At a high level, PoS-based blockchains share the following structure: (a) a committee of selected of users runs a consensus sub-protocol to agree on what block B to be added next, (b) each committee member then signs that block B, and (c) each node then appends a block B to their view of the ledger if it sees sufficiently many committee member signatures on the block B. We refer to this collection of committee signatures on the block B as the *block certificate*. The way in which the committees are selected and the consensus sub-protocol varies quite substantially amongst the various designs.

This work. In this work, we focus on the common cryptographic core of all PoS-based blockchains, namely the signature scheme used by the committee, and how we can simultaneously meet the requirements for efficiency and security.

In terms of efficiency, a major cost of PoS protocols are bandwidth and space needed to propagate committee signatures and to store the block certificate, as well as the computational

*Email: sergey@algorand.com.

†Email: hoeteck@algorand.com.

resources needed for signature verification of these certificates. The former can be mitigated with the use of the BLS signatures [6, 5]; these signatures are aggregatable, namely we can compress N signatures on N possibly distinct messages under N public keys into a single short signature simultaneously validating all N message-key pairs. Unfortunately, BLS signatures do not satisfy the security requirement which we describe next.

In terms of security, we require that the signatures be *forward-secure*. That is, each signature is associated with the current time period in addition to the signed message, and after each time interval, a user's secret key can be updated in such a way that it can only be used to sign messages for future time periods, and but not previous ones. The use of forward-secure signatures prevent *adaptive* attacks on a PoS-based blockchain, where an adversary waits till the agreement on a block B is reached for a round r, then at some time in the future, it corrupts all the committee members that signed the block to obtain their signing keys.[1] Using the keys the adversary can produce a valid certificate for a different block B' for the same round r. Note that this attack goes away if committee members use a forward-secure signature and update their keys as soon as they sign a block B.

1.1 Our Results

We present a pairing-based signature scheme for use in PoS-based blockchains that achieves substantial savings in bandwidth and storage requirements. To support a total of T time periods and a committee of size N, the block certificate comprises just two group elements (in addition to the identities of the committee members), whereas verifying each committee member's signature as well as the block certificate requires only two pairings plus one exponentiations. This essentially matches the efficiency of BLS signatures, cf. Fig 1.1, while also preventing adaptive attacks. In contrast, using existing forward-secure signature yields much larger block certificate of size $O(N)$ to $O(N \log T)$ group elements [2, 21, 19, 23, 8]; this is the case even if we were to instantiate the tree-based constructions with aggregatable BLS signatures.

Our construction combines prior forward-secure signatures based on hierachical identity-based encryption (HIBE) [8, 12, 10, 4] with the simple observation that it suffices to support signature aggregation on the *same* message, since we only need to aggregate and store committee signatures on the same block B. We achieve security under a standard q-type assumption in the random oracle model (or in the generic group model without random oracles).

Overview of our scheme. Starting with a bilinear group (G_1, G_2, G_T) with $e : G_1 \times G_2 \to G_T$ of prime order p and generators g_1, g_2 for G_1, G_2 respectively, a signature on $M \in \mathbb{Z}_p \setminus \{0,1,2\}$ at time t under public key $e(g_1, g_2)^\alpha$ is of the form:

$$\sigma = (g_2^{\alpha + F(t,M)r}, g_2^r) \in G_2^2$$

where the function $F(t, M)$ also depends on some fixed public parameters ($O(\log T)$ group elements) and r is fresh randomness used for signing. Verification relies on the relation:

$$e(g_1, g_2^{\alpha+F(t,M)r}) \cdot e(g_1^{F(t,M)}, g_2^r)^{-1} = e(g_1, g_2)^\alpha$$

where $g_1^{F(t,M)}$ is computed using the public parameters.

Given N signatures $\sigma_1, \ldots, \sigma_N$ on the same message M at time t under N public keys $g_1^{\alpha_1}, \ldots, g_1^{\alpha_N}$, we can produce an aggregate signature σ' on M by computing the coordinate-wise product of $\sigma_1, \ldots, \sigma_N$. Concretely, if $\sigma_i = (g_2^{\alpha_i + F(t,M)r_i}, g_2^{r_i})$, then

$$\sigma' = (g_2^{\alpha_1 + \cdots + \alpha_N + F(t,M)r'}, g_2^{r'})$$

[1] In a typical PoS protocol, a committee is a tiny fraction of the total number of users in the system so that an adaptive adversary can corrupt an entire committee while controlling only a tiny fraction of the total stake. Also, the stakes of the committee members may decrease significantly over time.

scheme	setup + keygen	update	sign	verify	$\|\sigma\|$	$\|\mathsf{sk}\|$
BLS	$O(1)$ exp	–	1 exp	2 pair	1	$O(1)$
our scheme	$O(\log T)$ exp + 1 pair	3 exp	4 exp	2 pair + 1 exp	2	$O((\log T)^2)$

Figure 1: Comparing our scheme with BLS signatures. Here, "exp" and "pair" refer to number of exponentations and pairings respectively. We omit additive overheads of $O(\log T)$ multiplications. The column update refers to amortized update times for t to $t+1$. The columns $|\sigma|$ and $|\mathsf{sk}|$ denote signature and secret key size in terms of group elements.

where $r' = r_1 + \cdots + r_N$.

How to generate and update keys. To complete this overview, we describe a simplified version of the secret keys and update mechanism, where the secret keys are of size $O(T)$ instead of $O((\log T)^2)$. The construction exploits the fact that the function F satisfies

$$F(t, M) = F(t, 0) + M \cdot F'(0, 1)$$

This means that in order to sign messages at time t, it suffices to know

$$\widetilde{\mathsf{sk}}_t = \{g_2^{\alpha + F(t,0)r}, g_2^{F'(0,1)r}, g_2^r\}$$

from which we can compute $g_2^{\alpha + F(t,M)r}, g_2^r$. For security, we will need to randomize r by computing $g_2^{F(t,M)}$ from the public parameters.

The secret key sk_t for time t is given by:

$$\widetilde{\mathsf{sk}}_t, \widetilde{\mathsf{sk}}_{t+1}, \cdots, \widetilde{\mathsf{sk}}_T$$

generated using independent randomness. To update from the key sk_t to sk_{t+1}, we simply erase $\widetilde{\mathsf{sk}}_t$.

To compress the secret keys down to $O((\log T)^2)$ without increasing the signature size, we combine the tree-based approach in [10] with the compact HIBE in [4]. Roughly speaking, each sk_t now contains $\log T$ sub-keys, each of which contains $O(\log T)$ group elements and looks like an "expanded" version of $\widetilde{\mathsf{sk}}_t$.

Generating public parameters. Our signature scheme requires $\log T$ pairs of group elements of the form (g_1^w, g_2^w) for which w is completely hidden from the adversary. We observe that there is an extremely simple and non-interactive MPC protocol for generating these parameters that achieves information-theoretical security as long as there is a single honest party. Each party contributes a random pair $(g_1^{w_i}, g_2^{w_i})$ (and then erases w_i), which anyone can check is well-formed via a pairing $e(g_1^{w_i}, g_2) = e(g_1, g_2^{w_i})$. The output is the coordinate-wise product of these pairs, which implicitly sets $w = \sum w_i$.

1.2 Discussion

Related works. The use of HIBE schemes for forward secrecy originates in the context of encryption [10] and has been used in signatures [8, 12], key exchange [17] and proxy re-encryption [15]. Our signature scheme is quite similar to the forward-secure signatures of Boyen et al. [8] and achieves the same asymptotic complexity; their construction is more complex to order to achieve security against untrusted updates. Our construction can be viewed as an aggregatable forward-secure multi-signature scheme [22].

Non-interactive and incremental aggregation. As mentioned earlier, the signature scheme will be used by commmittee members to sign a block B selected during a consensus sub-protocol. These signatures will then be propagated through the network. Note that our scheme supports non-interactive and incremental aggregation: the former means that signatures can be aggregated by any party after broadcast without communicating with the original signers, and the latter means that we can incorporate a new signature to an aggregate signature to obtain a new aggregate signature. In practice, this means that relay nodes can perform intermediate aggregation on any number of committee signatures and propagate the result, until the block certificate is formed.

Alternative approaches to adaptive security. There are two variants of the adaptive attack: (i) a short-range variant, where an adversary tries to corrupt a committee member prior to completion of the consensus sub-protocol, and (ii) a long-range variant as described earlier. Dfinity, Ouroboros and Caspter cope with the short-range attacks by assuming a delay in attacks that is longer than the running time of the consensus sub-protocol. For long-range attacks, Casper adopts a fork choice rule to never revert a finalized block, and in addition, assumes that clients log on with sufficient regularity to gain a complete update-to-date view of the chain. We note that forward-secure signatures provide a clean solution against both attacks, without the need for fork choice rules or additional assumptions about the adversary and the clients.

Application to permissioned blockchains. Consensus protocols, such as PBFT, are also at the core of many permissioned blockchains (e.g. Hyperledger), where only approved parties may join the network. Our signature scheme can similarly be applied to this setting to achieve forward secrecy, reduce communication bandwidth, and produce compact block certificates.

Security against rogue key attacks. We prove security in a model where the adversary must provide proofs of secret keys [3, 26], that is, knowledge of g_2^{α} corresponding to $e(g_1, g_2)^{\alpha}$; this can be achieved for instance by using a variant of Schnorr protocol combined with the Fiat-Shamir heuristic. In practice, this is a perfectly reasonable assumption since committee members do need to register their signature public keys in advance (for instance, as a transaction in which case the proofs of secret keys are verified by committee members for the block where the transaction appears) before participating in the consensus sub-protocol.

2 Same Message Aggregatable Forward-Secure Signatures

Syntax. A same message aggregatable forward-secure signature scheme consists of a tuple of algorithms $(\mathsf{Setup}, \mathsf{Keygen}, \mathsf{UpdateKey}, \mathsf{Sign}, \mathsf{Aggregate}, \mathsf{Verify})$ satisfying the following requirements:

- $\mathsf{Setup}(1^{\lambda}, T) \rightarrow \mathsf{pp}$. The set-up algorithm takes as input a security parameter λ and an upper bound on the number of time-periods T (encoded as integer). It outputs public parameters pp.
- $\mathsf{Keygen}(\mathsf{pp}) \rightarrow (\mathsf{pk}, \mathsf{sk}_0)$. The key-generation algorithm takes as input the parameter parameters pp. It outputs a public key pk and secret key sk_0 for time 0.
- $\mathsf{UpdateKey}(\mathsf{pp}, \mathsf{sk}_t, t') \rightarrow \mathsf{sk}_{t'}$. The update algorithm takes as input a secret key for a time period $0 \leq t \leq T$ and a new time period $t' > t$ and outputs a secret key $\mathsf{sk}_{t'}$ for the time period t'.
- $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}_t, t, M) \rightarrow \sigma$. The signing algorithm takes as input a secret key for a time period $t \in [T]$ and a message M. It outputs a signature σ.
- $\mathsf{Aggregate}(\sigma_1, \ldots, \sigma_N) \rightarrow \sigma$: The aggregation algorithm takes as input a collection of signatures signing the same message M for the same time period t and outputs an aggregate signature σ.

- $\mathsf{Verify}(\mathsf{pk}_1,\ldots,\mathsf{pk}_N,t,M,\sigma)\rightarrow 0/1$. The verification algorithm takes as input a collection of public keys, a time period t, a message M and a signature σ. It outputs 1 iff σ is a valid signature under the collection of given public keys.

Correctness. For any integer N, time period $t\in[T]$, and all $(\mathsf{pk}_1,\mathsf{sk}_{1,i}),\ldots,(\mathsf{pk}_N,\mathsf{sk}_{N,t})$ that are produced via invocation of algorithms $\mathsf{Keygen},\mathsf{UpdateKey}$, and all messages M, the following condition must hold:

$$\Pr[\mathsf{Verify}(\mathsf{pk}_1,\ldots,\mathsf{pk}_N,i,M,\mathsf{Aggregate}(\sigma_1,\ldots,\sigma_N))=1]>1-\mathrm{negl}(\lambda)$$

where $\mathsf{Sign}(\mathsf{pp},\mathsf{sk}_{j,i},M)\rightarrow\sigma_j$ and the randomness is over the coins of all of the described algorithms.

In addition, we require that for any T, with probability $1-\mathrm{negl}(\lambda)$ over $\mathsf{Setup}(1^\lambda,T)\rightarrow\mathsf{pp},\mathsf{Keygen}(\mathsf{pp})\rightarrow(\mathsf{pk},\mathsf{sk}_0)$, we have that for any $t<t'\leq T$:

$$(\mathsf{sk}_0,\mathsf{UpdateKey}(\mathsf{pp},\mathsf{sk}_0,t'))\approx_s(\mathsf{sk}_0,\mathsf{UpdateKey}(\mathsf{pp},\mathsf{UpdateKey}(\mathsf{pp},\mathsf{sk}_0,t),t'))$$

Security. We describe the security game between an adversary and a challenger in phases.

- **Challenge time-period.** The adversary outputs T along with a challenge time-period $t^*\in[T]$.
- **Key-generation**. The challenger runs $\mathsf{Setup}(1^\lambda,T)\rightarrow\mathsf{pp},\mathsf{Keygen}(\mathsf{pp})\rightarrow(\mathsf{pk},\mathsf{sk}_0),\mathsf{UpdateKey}(\mathsf{sk}_0,t^*+1)\rightarrow\mathsf{sk}_{t^*+1}$ and sends $\mathsf{pp},\mathsf{pk},\mathsf{sk}_{t^*+1}$ to the adversary.[2]
- **Signature queries.** The adversary issues adaptively chosen queries (t,M) for any $t\leq t^*$ and receives signatures $\sigma\leftarrow\mathsf{Sign}(\mathsf{pp},\mathsf{sk}_t,t,M),\mathsf{UpdateKey}(\mathsf{sk}_0,t)\rightarrow\mathsf{sk}_t$ from the challenger.
- **Forgery.** The adversary outputs

$$N,i\in[N],(\mathsf{pk}_j,\mathsf{sk}_{j,0})_{j\neq i},M^*,\sigma^*$$

The adversary wins the game if

- for all signature queries (t,M), we have $t\neq t^*$ or $M^*\neq M$; and
- $\mathsf{Verify}(\mathsf{pk}_1,\ldots,\mathsf{pk}_N,t^*,M^*,\sigma^*)=1$ where $\mathsf{pk}_i:=\mathsf{pk}$.

The scheme is secure if no efficient adversary can win this game with non-negligible probability.

Remark 2.1 (Accounting for proofs of knowledge)**.** *We can modify the security definition to account for proofs of knowledge as follows: (i) the adversary receives a proof of knowledge of* sk_0 *in the key-generation phase, and (ii) the adversary submits proofs of knowledge of* $\mathsf{sk}_{j,0}, j\neq i$ *instead of* $\mathsf{sk}_{j,0}$ *in the forgery phase. We note that the security definition in [26] does not account for (i).*

3 Our Construction

We present our pairing-based same-message aggregatable forward-secure signature scheme in this section.

[2] This allows the adversary to compute any sk_t and signatures on (t,M) for any $t>t^*$.

3.1 Preliminaries

Following [10], we associate time periods with all nodes of the tree according to a pre-order traversal. Prior tree-based forward-secure signatures associate time periods with the only leaf nodes. The CHK approach allows us to reduce the amortized complexity of key updates from t to $t+1$ from $O(\log T)$ exponentiations to $O(1)$ exponentiations.

Bijection. We describe a bijection between $\{1,2\}^{\leq D-1}$ and $[2^D-1]$ for any integer D given by

$$\mathbf{t} = (t_1, t_2, \ldots) \in \{1,2\}^{\leq D-1} \mapsto 1 + \sum_{i=1}^{|\mathbf{t}|} (1 + 2^{D-i}(t_i - 1))$$

For instance, for $D = 3$, this maps $\varepsilon, 1, 11, 12, 2, 21, 22$ to $1, 2, 3, 4, 5, 6, 7$. This induces a natural precedence relation over $\{1,2\}^{\leq D-1}$ where $\mathbf{t} \preceq \mathbf{t}'$ iff either $\mathbf{t}$ is a prefix of $\mathbf{t}'$ or exists $\bar{\mathbf{t}}$ s.t. $\bar{\mathbf{t}}\|1$ is a prefix of $\mathbf{t}$ and $\bar{\mathbf{t}}\|2$ is a prefix of $\mathbf{t}'$. We also write $\mathbf{t}, \mathbf{t}+1$ corresponding to $t, t+1$.

Encoding intervals. Next, we associate any $\mathbf{t} \in \{1,2\}^{\leq D-1}$ with a set $\Gamma_{\mathbf{t}} \subset \{1,2\}^{\leq D-1}$ given by

$$\Gamma_{\mathbf{t}} := \{\mathbf{t}\} \cup \{\bar{\mathbf{t}}\|2 : \bar{\mathbf{t}}\|1 \text{ prefix of } \mathbf{t}\}$$

The sets $\Gamma_{\mathbf{t}}$ satisfy the following properties:

- $\mathbf{t}' \succeq \mathbf{t}$ iff there exists $\mathbf{u} \in \Gamma_{\mathbf{t}}$ s.t. $\mathbf{u}$ is a prefix of $\mathbf{t}'$;
- For all $\mathbf{t}' \succ \mathbf{t}$, we have that for all $\mathbf{u}' \in \Gamma_{\mathbf{t}'}$, there exists $\mathbf{u} \in \Gamma_{\mathbf{t}}$ such that $\mathbf{u}$ is a prefix of $\mathbf{u}'$;
- For all $\mathbf{t}$, we have either $\Gamma_{\mathbf{t}+1} = \Gamma_{\mathbf{t}} \setminus \{\mathbf{t}\}$ or $\Gamma_{\mathbf{t}+1} = (\Gamma_{\mathbf{t}} \setminus \{\mathbf{t}\}) \cup \{\mathbf{t}\|1, \mathbf{t}\|2\}$

The first property is used for verification and for reasoning about security; the second and third properties are used for key updates.

Bilinear groups. A generator $\mathcal{G}$ takes as input a security parameter λ and outputs a description $\mathbb{G} := (p, G_1, G_2, G_T, e)$, where p is a prime of $\Theta(\lambda)$ bits, G_1, G_2 and G_T are cyclic groups of order p, and $e : G_1 \times G_2 \to G_T$ is a non-degenerate bilinear map. We require that the group operations in G_1, G_2 and G_T as well the bilinear map e are computable in deterministic polynomial time with respect to λ. Let $g_1 \in G_1$, $g_2 \in G_2$ be the respective generators. Our scheme relies on the q-SDH assumption, which says that it is hard to compute $g_2^{\alpha^{q+1}}$ given $g_1, g_1^{\alpha}, g_1^{\alpha^2}, \ldots, g_1^{\alpha^q}, g_2, g_2^{\alpha}, g_2^{\alpha^2}, \ldots, g_2^{\alpha^q}$ for a random $\alpha \leftarrow \mathbb{Z}_p$.

3.2 Our Signature Scheme

We assume the bound T is of the form $2^D - 1$. We use the above bijection so that the algorithms take input $\mathbf{t} \in \{1,2\}^{\leq D-1}$ instead of $t \in [T]$. Our signature scheme relies on two auxiliary algorithms Subkey and Delegate which we describe below.

- $\mathsf{Setup}(1^\lambda, 2^D - 1)$. Given (G_1, G_2, G_T) and generators g_1, g_2, sample $w_0, w_1, \ldots, w_D \leftarrow \mathbb{Z}_p$ and output

$$\mathsf{pp} := \left(g_1^{w_0}, \ldots, g_1^{w_D}, g_2^{w_0}, \ldots, g_2^{w_D}\right) \in G_1^{D+1} \times G_2^{D+1}$$

- $\mathsf{Keygen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk}_0)$. Sample $\alpha \leftarrow \mathbb{Z}_p$. Output

$$\mathsf{pk} := e(g_1, g_2)^{\alpha} \in G_T,\ \mathsf{sk}_0 := g_2^{\alpha} \in G_2$$

- $\mathsf{Subkey}(\mathsf{pp}, \mathsf{sk}_0 = g_2^\alpha, \mathbf{x} \in (\mathbb{Z}_p^*)^{\le D})$. Sample $r \leftarrow \mathbb{Z}_p$. Output

$$\widetilde{\mathsf{sk}}_{\mathbf{x}} := \left(g_2^r, g_2^{\alpha + f(\mathbf{x})r}, g_2^{w_{|\mathbf{x}|+1} r}, \ldots, g_2^{w_D r}\right) \in G_2^{D+2-|\mathbf{x}|}$$

  where $f(\mathbf{x}) := w_0 + \sum_{i=1}^{|\mathbf{x}|} w_i x_i$.

- $\mathsf{Delegate}(\mathsf{pp}, \widetilde{\mathsf{sk}}_{\mathbf{x}} = (K_0, K_1, K_{|\mathbf{x}|+1}, \ldots, K_D), \mathbf{x}' \in (\mathbb{Z}_p^*)^{\le D})$. If $\mathbf{x}$ is a prefix of $\mathbf{x}'$, output

$$\widetilde{\mathsf{sk}}_{\mathbf{x}'} := \left(K_0, K_1 \cdot \prod_{i=|\mathbf{x}|+1}^{|\mathbf{x}'|} K_i^{x_i'}, K_{|\mathbf{x}'|+1}, \ldots, K_D\right) \cdot \mathsf{Subkey}(\mathsf{pp}, g_2^0, \mathbf{x}') \in G_2^{D+2-|\mathbf{x}'|}$$

  where we use $\cdot$ to also denote cooordinate-wise multiplication.

- $\mathsf{UpdateKey}(\mathsf{pp}, \mathsf{sk}_{\mathbf{t}} = \left(\widetilde{\mathsf{sk}}_{\mathbf{u}} : \mathbf{u} \in \Gamma_{\mathbf{t}}\right), \mathbf{t}') \to \mathsf{sk}_{\mathbf{t}'}$. For each $\mathbf{u}' \in \Gamma_{\mathbf{t}'}$, compute $\mathbf{u} \in \Gamma_t$ s.t. $\mathbf{u}$ is a prefix of $\mathbf{u}'$ and sample $\widetilde{\mathsf{sk}}_{\mathbf{u}'} \leftarrow \mathsf{Delegate}(\mathsf{pp}, \widetilde{\mathsf{sk}}_{\mathbf{u}}, \mathbf{u}')$. Erase $\mathsf{sk}_{\mathbf{t}}$ and output

$$\mathsf{sk}_{\mathbf{t}'} := \left(\widetilde{\mathsf{sk}}_{\mathbf{u}'} : \mathbf{u}' \in \Gamma_{\mathbf{t}'}\right)$$

- $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}_{\mathbf{t}}, \mathbf{t}, M \in \mathbb{Z}_p^* \setminus \{1,2\})$. Output the first two elements of $\mathsf{Delegate}(\mathsf{pp}, \widetilde{\mathsf{sk}}_{\mathbf{t}}, \mathbf{t} \| M)$. That is,

$$\sigma := \left(g_2^r, g_2^{\alpha + f(\mathbf{t}\|M) r}\right) \in G_2^2$$

- $\mathsf{Aggregate}(\sigma_1, \ldots, \sigma_N)$. Output

$$\sigma := \sigma_1 \cdot \cdots \cdot \sigma_N \in G_2^2$$

- $\mathsf{Verify}(\mathsf{pk}_1, \ldots, \mathsf{pk}_N, \mathbf{t}, M, \sigma = (\sigma', \sigma''))$. Output

$$e(g_1^{f(\mathbf{t}\|M)}, \sigma') \cdot \mathsf{pk}_1 \cdots \mathsf{pk}_N \stackrel{?}{=} e(g_1, \sigma'')$$

Correctness is straight-forward.

3.3 Security

The BBG security proof in [4] shows that under the q-SDH assumption, for any $\mathbf{x}^* \in \{1,2\}^{\le D-1}$, it is hard to compute $\mathsf{Subkey}(\mathsf{sk}_0, \mathbf{x}^*)$ given pp, pk as well as an oracle for $\mathsf{Subkey}(\mathsf{sk}_0, \cdot)$ subject to the constraint that we never query the oracle on a prefix of $\mathbf{x}^*$. In fact, it is hard to even compute the first two elements of $\mathsf{Subkey}(\mathsf{sk}_0, \mathbf{x}^*)$ (i.e., any (σ', σ'') s.t. $\sigma' = \mathsf{sk}_0 \cdot (\sigma'')^{f(\mathbf{x}^*)}$). This implies security of the signature scheme as follows:

- Set $T = 2^D - 1$ and $\mathbf{x}^* = \mathbf{t}^* \| M^*$. (This requires knowing M^* in advance. In the actual scheme, we will hash the message using H, and in the random oracle model, we can first pick a value for $H(M^*)$ and program $H(M^*)$ to this value later. Alternatively, we can rely on adaptive security of the BBG HIBE in the generi group model [1].)

- $\mathsf{pp}, \mathsf{sk}_0$ are the same in both schemes

- To simulate key generation phase, i.e., $\mathsf{sk}_{\mathbf{t}^*+1} = \left(\widetilde{\mathsf{sk}}_{\mathbf{u}'} : \mathbf{u}' \in \Gamma_{\mathbf{t}^*+1}\right)$, we just query $\mathsf{Subkey}(\mathsf{sk}_0, \mathbf{u}') : \mathbf{u}' \in \Gamma_{\mathbf{t}^*+1}$. Here, we use the fact that $\mathbf{u}'$ is not a prefix of $\mathbf{t}^*$ (since $\mathbf{t}^* \not\ge \mathbf{t}^* + 1$) and also not a prefix of $\mathbf{t}^* \| M^*$.

- To simulate signature oracle for $(\mathbf{t}, M)$ where $\mathbf{t} \neq \mathbf{t}^*$ or $M \neq M^*$, we just query $\mathsf{Subkey}(\mathsf{sk}_0, \mathbf{t} \| M)$, since $\mathbf{t} \| M$ is not a prefix of $\mathbf{t}^* \| M^*$. The latter follows from the fact that since $M \neq 1, 2$, we have $\mathbf{t} \| M$ is a prefix of $\mathbf{t}^* \| M^*$ implies $\mathbf{t} \| M = \mathbf{t}^* \| M^*$.

- Given $\sigma = (\sigma', \sigma'')$ that passes the verification check along with $\mathsf{sk}_{j,0}$, $j \neq i$, we must have

$$\sigma' = \mathsf{sk}_0 \cdot (\sigma'')^{f(\mathbf{x}^*)} \cdot \textstyle\prod_{j \neq i} \mathsf{sk}_{j,0}$$

  We can then compute the first two elements of $\mathsf{Subkey}(\mathsf{sk}_0, \mathbf{x}^*)$ using $(\sigma' \cdot \prod_{j \neq i} \mathsf{sk}_{j,0}^{-1}, \sigma'')$.

References

[1] M. Ambrona, G. Barthe, R. Gay, and H. Wee. Attribute-based encryption in the generic group model: Automated proofs and new constructions. In *ACM CCS 17*, pages 647–664. ACM Press, Oct. / Nov. 2017.

[2] M. Bellare and S. K. Miner. A forward-secure digital signature scheme. In *CRYPTO'99, LNCS*, pages 431–448. Aug. 1999.

[3] A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In *PKC 2003*, *LNCS*, pages 31–46. Jan. 2003.

[4] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT 2005*, *LNCS*, pages 440–456. May 2005.

[5] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT 2003*, *LNCS*, pages 416–432. May 2003.

[6] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In *ASIACRYPT 2001*, *LNCS*, pages 514–532. Dec. 2001.

[7] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten. SoK: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy*, pages 104–121. IEEE Computer Society Press, May 2015.

[8] X. Boyen, H. Shacham, E. Shen, and B. Waters. Forward-secure signatures with untrusted update. In *ACM CCS 06*, pages 191–200. ACM Press, Oct. / Nov. 2006.

[9] V. Buterin and V. Griffith. Casper the friendly finality gadget. *CoRR*, abs/1710.09437, 2017.

[10] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT 2003*, *LNCS*, pages 255–271. May 2003.

[11] J. Chen, S. Gorbunov, S. Micali, and G. Vlachos. Algorand agreement: Super fast and partition resilient byzantine agreement. Cryptology ePrint Archive, Report 2018/377, 2018.

[12] S. S. M. Chow, L. C. K. Hui, S.-M. Yiu, and K. P. Chow. Secure hierarchical identity based signature and its application. In *ICICS 04*, *LNCS*, pages 480–494. Oct. 2004.

[13] M. Conti, E. S. Kumar, C. Lal, and S. Ruj. A survey on security and privacy issues of bitcoin. *IEEE Communications Surveys Tutorials*, 20(4):3416–3452, Fourthquarter 2018.

[14] B. David, P. Gazi, A. Kiayias, and A. Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *EUROCRYPT 2018, Part II*, *LNCS*, pages 66–98. Apr. / May 2018.

[15] D. Derler, S. Krenn, T. Lorünser, S. Ramacher, D. Slamanig, and C. Striecks. Revisiting proxy re-encryption: Forward secrecy, improved security, and applications. In *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 219–250. Mar. 2018.

[16] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 51–68, New York, NY, USA, 2017. ACM.

[17] F. Günther, B. Hale, T. Jager, and S. Lauer. 0-RTT key exchange with full forward secrecy. In *EUROCRYPT 2017, Part III*, *LNCS*, pages 519–548. Apr. / May 2017.

[18] T. Hanke, M. Movahedi, and D. Williams. Dfinity technology overview series, consensus system.

[19] G. Itkis and L. Reyzin. Forward-secure signatures with optimal signing and verifying. In *CRYPTO 2001*, *LNCS*, pages 332–354. Aug. 2001.

[20] A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 357–388. Aug. 2017.

[21] H. Krawczyk. Simple forward-secure signatures from any signature scheme. In *ACM CCS 00*, pages 108–115. ACM Press, Nov. 2000.

[22] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters. Sequential aggregate signatures and multisignatures without random oracles. In *EUROCRYPT 2006*, *LNCS*, pages 465–485. May / June 2006.

[23] T. Malkin, D. Micciancio, and S. K. Miner. Efficient generic forward-secure signatures with an unbounded number of time periods. In *EUROCRYPT 2002, LNCS*, pages 400–417. Apr. / May 2002.

[24] S. Micali. ALGORAND: the efficient and democratic ledger. *CoRR*, abs/1607.01341, 2016.

[25] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system,âĂİ http://bitcoin.org/bitcoin.pdf.

[26] T. Ristenpart and S. Yilek. The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In *EUROCRYPT 2007*, *LNCS*, pages 228–245. May 2007.

A Performance Estimates

We present some preliminary performance estimates as a proof of concept; these numbers could be improved with some optimizations and pre-processing.

These estimates are for the Relic implementation of BLS12-381 curve on a MacBook Pro.

scheme	setup	keygen	update	sign	verify	$N = 1000$	$N = 5000$	$\lvert \mathsf{sk}_t \rvert$
$T = 2^{20}$, sig in G_2	28.4	4.58	3.02	4.21	5.01	9.00	25.0	38 kB
$T = 2^{20}$, sig in G_1	28.4	3.98	1.05	1.47	5.72	9.71	25.7	19 kB
$T = 2^{30}$, sig in G_2	42.0	4.58	3.02	4.30	5.02	9.02	25.0	84 kB
$T = 2^{30}$, sig in G_1	42.0	3.98	1.05	1.51	5.76	9.76	25.8	42 kB

Figure 2: Time in ms. The column "update" refers to amortized key updates from t to $t+1$. The columns $N = 1000, N = 5000$ refer to aggregate verification.

B Signature Variants

B.1 Smaller public keys

We could also consider a variant of our signature scheme with smaller public keys, namely $\mathsf{pk} = g_1^{\alpha}$ instead of $e(g_1, g_2)^{\alpha}$. This requires the following additional modifications to the scheme:

- verification requires an additional pairing to compute $e(g_1, g_2)^{\alpha}$; in a proof-of-stake blockchain, we could pre-compute this quantity for the public keys with the highest stakes.
- security of the scheme requires a stronger assumption, namely that it is hard to compute $g_2^{\alpha^{q+1}}$ given $g_1, g_1^{\alpha}, g_1^{\alpha^2}, \ldots, g_1^{\alpha^q}, g_1^{\alpha^{q+1}}, g_2, g_2^{\alpha}, g_2^{\alpha^2}, \ldots, g_2^{\alpha^q}$ for a random $\alpha \leftarrow \mathbb{Z}_p$.

B.2 Proofs of posession

In the variant where we set $\mathsf{pk} = g_1^{\alpha}$, we can replace the proofs of knowledge of secret keys with a BLS signature on the corresponding public key / identity, with the appropriate domain separation in the underlying hash function as in [26] and α as the BLS secret key. The idea is as follows:

- when the adversary registers pk_j corresponding to $(\mathsf{pk}_j, \mathsf{sk}_{j,0}) = (g_1^{\alpha_j}, g_2^{\alpha_j})$, it provides a BLS signature $\sigma_j = H(\mathsf{pk}_j)^{\alpha_j} \in G_2$. The reduction picks $\beta_j \leftarrow \mathbb{Z}_p$ and programs $H(\mathsf{pk}_j) = g_2^{\beta_j}$, upon which it could extract $g_2^{\alpha_j}$ using σ_j^{1/β_j}.
- the security of the scheme also requires a stronger assumption, namely that it is hard to compute $g_2^{\alpha^{q+1}}$ given $g_1, g_1^{\alpha}, g_1^{\alpha^2}, \ldots, g_1^{\alpha^q}, g_1^{\alpha^{q+1}}, g_2, g_2^{\alpha}, g_2^{\alpha^2}, \ldots, g_2^{\alpha^q}.g_2^{\alpha^{q+2}}$ for a random $\alpha \leftarrow \mathbb{Z}_p$.
- in the security reduction in [4], the reduction basically programs $\mathsf{sk}_0 = g_2^{\alpha^{D+1}}$ (with $D = q$). To simulate a BLS signature $\sigma = H(\mathsf{pk})^{\alpha^{D+1}}$, the reduction picks $\beta \leftarrow \mathbb{Z}_p$ and programs $H(\mathsf{pk}) = g_2^{\alpha\beta}$, upon which it can simulate σ given $g_2^{\alpha^{D+2}}$.

B.3 Modified signatures

We could modify the signing algorithm to support $M \in \mathbb{Z}_p^*$ as follows:

- $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}_{\mathbf{t}}, \mathbf{t}, M \in \mathbb{Z}_p^*)$. Output the first two elements of $\mathsf{Delegate}(\mathsf{pp}, \mathsf{sk}_{\mathbf{t}}, \mathbf{t} \| 0^{D-1-|\mathbf{t}|} \| M)$. That is,

$$\sigma := \left(g_2^r, g_2^{\alpha + f(\mathbf{t} \| 0^{D-1-|\mathbf{t}|} \| M) r} \right) \in G_2^2$$

- $\mathsf{Verify}(\mathsf{pk}_1, \ldots, \mathsf{pk}_N, \mathbf{t}, M, \sigma = (\sigma', \sigma''))$. Output

$$e(g_1^{f(\mathbf{t} \| 0^{D-1-|\mathbf{t}|} \| M)}, \sigma') \cdot \mathsf{pk}_1 \cdots \mathsf{pk}_N \stackrel{?}{=} e(g_1, \sigma'')$$