A preliminary version of this paper appeared in the 32nd International Colloquium on Automata, Languages and Programming, ICALP 2005, Lecture Notes in Computer Science Vol. ???, Giuseppe F. Italiano, Catuscia Palamidessi and Moti Yung eds, Springer Verlag, 2005. This is the full version.

Append-Only Signatures

Eike Kiltz* Anton Mityagin† Saurabh Panjwani‡ Barath Raghavan§

May 6, 2005

Abstract

We present a new primitive—Append-only Signatures (AOS)—with the property that any party given an AOS signature $\mathtt{Sig}[M_1]$ on message M_1 can compute $\mathtt{Sig}[M_1 \| M_2]$ for any message M_2, where $M_1 \| M_2$ is the concatenation of M_1 and M_2. We define the security of AOS, present concrete AOS schemes, and prove their security under standard assumptions. In addition, we find that despite its simple definition, AOS is equivalent to Hierarchical Identity-based Signatures (HIBS) through efficient and security-preserving reductions. Finally, we show direct applications of AOS to problems in network security. Our investigations indicate that AOS is both useful in practical applications and worthy of further study as a cryptographic primitive.

Keywords: Algebraic Signatures, Append-only Signatures, Hierarchical Identity-based Signatures

* Department of Computer Science and Engineering, University of California, San Diego, San Diego, USA. Email: `ekiltz@cs.ucsd.edu`. URL: `http://www.kiltz.net/`. Research supported by a DAAD postdoc fellowship.

† Department of Computer Science and Engineering, University of California, San Diego, San Diego, USA. Email: `amityagin@cs.ucsd.edu`. Research supported in part by NSF grants ANR-0129617 and CCR-0208842.

‡ Department of Computer Science and Engineering, University of California, San Diego, San Diego, USA. Email: `panjwani@cs.ucsd.edu`. URL: `http://www.cse.ucsd.edu/users/spanjwan/`. Research supported in part by NSF grant 0313241. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

§ Department of Computer Science and Engineering, University of California, San Diego, San Diego, USA. Email: `barath@cs.ucsd.edu`. Research supported by a NSF Graduate Research Fellowship.

Contents

1 Introduction

In many real-world applications, users and programs alike require notions of delegation to model the flow of information. It is often required that delegation from one party to another enables the delegatee to "append" to the information it received but to do nothing more. For example, in wide-area Internet routing, each network passes a routing path advertisement to its neighboring networks, which then append to it information about themselves and forward the updated advertisement to their neighbors. For security, the route advertisements must be authenticated; intermediate networks must be incapable of modifying routes except according to the protocol (that is, by appending their names to already-received advertisements). Likewise, in the context of secure resource delegation for distributed systems, users need to delegate their share of resources to other users, who may then re-delegate to other users by including their own resources in the pool. In many of these applications, it is desirable that delegation is possible without parties having to share any cryptographic keys and that the authenticity of any information received through a series of delegations is verifiable based only on the identity of the first party in the chain.

To directly address the needs of these applications, we present a new cryptographic primitive called Append-Only Signatures (AOS). Informally, an AOS scheme enables the extension of signed messages and update of the corresponding signatures, without requiring possession of the signing key. That is, any party given an AOS signature $\mathtt{Sig}[M_1]$ on message M_1 can compute $\mathtt{Sig}[M_1 \| M_2]$ for any message M_2, where $M_1 \| M_2$ is the concatenation of M_1 and M_2. The verifier of the final signature needs the initial signer's public key but does not need to know the public keys or any other information from intermediate signers except the message data appended. Clearly, such a scheme cannot be secure according to the standard notion of security for signatures. Instead, we define an AOS scheme to be secure if it is infeasible to forge signatures of messages that are not obtained by extending already-signed messages. We define the security of AOS more formally in Section 2.

We present several provably secure AOS schemes, offering different tradeoffs of flexibility and efficiency. Our first construction shows a generic approach to building AOS schemes from any standard digital signature scheme $\mathcal{SIG}$ using certificate chains. The construction works as follows: The secret and public keys for the AOS scheme are obtained by running the key generator for $\mathcal{SIG}$. For any message $M = M_1 \| M_2 \| \cdots \| M_n$, each M_i being a symbol in some predetermined message space, the AOS signature of M is defined as a sequence of n public keys $pk_1, pk_2, \cdots, pk_n$ (generated using the key generator for $\mathcal{SIG}$) and a sequence of n certificates binding the message symbols to these public keys. The ith certificate in the chain binds the message symbol M_i to the corresponding public key pk_i and is signed using the secret key, sk_{i-1}, corresponding to pk_{i-1}. The secret key, sk_0, of the AOS scheme signs the first certificate in the chain while the secret key sk_n (corresponding to the last public key), is revealed as part of the AOS signature and is used for appending new symbols to M. We observe that if the message space is small enough, we can make use of "weaker", and more efficient, signature schemes without compromising the security of the resulting AOS scheme. Furthermore, using aggregation techniques[BGLS03, LMRS04], the size of the certificate chain can be made a constant, that is, independent of the number of message symbols appended, which leads to shorter AOS signatures than those in the basic scheme. (See Section 3 for more details on this scheme.)

Since signature schemes exist given the existence of one-way functions [Rom90], the above construction implies the existence of AOS under the same assumption. We also present a more efficient construction of AOS for applications in which the message space is constant size and the total number of append operations performed is also constant. This construction is based on a stronger assumption and makes use of pseudorandom generators and collision-resistant hash functions (CRHFs). We remark that both these schemes—one using CRHFs and one based on certificate chains—are in the standard model; neither of them makes use of random oracles.

Relation to Hierarchical Identity-Based Signatures. Identity-Based Signature (IBS) schemes, due to Shamir [Sha85], are signature schemes in which the identity of the signer (for example, her email address) plays the role of his public key. Such schemes assume the existence of a trusted authority that holds a master public-private key pair that is used to assign secret keys to users based on their identities. Anyone can verify signatures on messages signed by a user knowing only the master public key and the identity of that user. Hierarchical IBS (HIBS) schemes, proposed by Gentry and Silverberg [GS02], are a natural generalization of this concept to a setting in which users are arranged in a hierarchy and a user at any level in this hierarchy can delegate secret keys to her descendants based on their identities and her own secret key. To verify the signature created by any user, one needs to know the identity of the user (and her position in the hierarchy) and the public key of the root user.

HIBS can be implemented using certificate chains (as suggested in [GS02]) and the resulting construction bears a strong resemblance to the certificate-based construction of AOS we give in this paper. Upon closer examination, we find that the similarity between the two constructions is not accidental: it is an artifact of the close relationship between the two primitives themselves—AOS and HIBS are, in fact, tightly equivalent. This means that (a) there exist generic transformations from any HIBS scheme into a corresponding AOS scheme and, likewise, from any AOS scheme into a corresponding HIBS scheme; and (b) these transformations are extremely efficient (the derived scheme is as efficient as the scheme being derived from) and highly security-preserving (an adversary attacking the derived scheme can be transformed into an adversary attacking the original one, losing only a constant factor in efficiency and query complexity).

A benefit of this equivalence is that it considerably simplifies the notion of HIBS and makes security analysis for HIBS schemes less onerous: AOS is simpler than HIBS, and, for any HIBS scheme, it is typically easy to find an equivalent AOS scheme whose security properties carry over to the corresponding HIBS scheme. For example, our security proof for certificate-based AOS translates to a security proof for certificate-based HIBS (originally proposed in [GS02]). Although this construction of HIBS was known prior to our work, it was never analyzed in the literature, and, to the best of our knowledge, we give the first proof of security for it. Furthermore, our construction of AOS based on pseudorandom generators and CRHFs yields a novel approach to designing HIBS and can be useful for some restricted scenarios (for example, in a constant-depth hierarchy wherein each user signs messages from a constant-size message space). We remark that both these constructions yield HIBS schemes in the standard model and neither involves the use of computationally intensive bilinear maps (this is in contrast with some recent results on HIBS [CHYC04]). Finally, we believe that AOS is a more intuitive primitive to study because it clearly reflects the requirements of our problem domains (such as secure routing) and is better suited for analyzing the problems that motivate our work.

Related Work. Append-only signatures belong to a general class of primitives called *algebraic signatures*. Informally, an algebraic signature scheme allows the creation of signatures on a new message M using the signatures on some known messages, $M_1, M_2, \ldots, M_n$, and the public key, provided the new message can be obtained from the known messages using some prespecified set of (n-ary) operations, say $\mathcal{O} = \{f_1, f_2, \cdots, f_m\}$. That is, given the signatures, $\texttt{sig}[M_1], \ldots, \texttt{sig}[M_n]$ and the public key, it is easy to compute $\texttt{sig}[f_i(M_1, \ldots, M_n)]$ for any

$f_i \in \mathcal{O}$. In our setting, each f_i has arity 1 and appends some fixed message symbol M_i to an input message M. Security for algebraic signatures is defined in a manner similar to our approach to security of AOS (that is, it should be hard to forge signatures of messages that cannot be obtained by applying the operations in $\mathcal{O}$ to already-signed messages). Examples of algebraic signatures studied in the literature include transitive signatures by Micali and Rivest [MR02], homomorphic signatures by Johnson, Molnar, Song and Wagner [JMSW02], and graph-based algebraic signatures by Hevia and Micciancio [HM02].

Although no obvious relation exists between our primitive and any of the previously studied algebraic signature primitives, we do note that some of the techniques we use in our constructions parallel prior techniques. For example, our construction of AOS schemes using CRHFs can be viewed as a special instance of graph-based algebraic signature schemes studied in [HM02] (although the set of update operations considered there are different from the `append` operation that we consider). Also, [JMSW02] introduces the notion of *redactable signatures*—signatures that allow "deletion" of message symbols without access to the secret key—and one of the constructions for this primitive given in their paper is also an example of graph-based algebraic signatures.

A concept closely related to AOS (and algebraic signatures, in general) is that of *incremental signatures*, proposed by Bellare, Goldreich, and Goldwasser [BGG94, BGG95]. Given an incremental signature on a message M, it is possible to compute the signature on a slightly updated version of M in time proportional to the "amount" of change made to M (rather than on the length of the entire message). The update operation, however, requires access to the initial signer's secret key whereas, in the case of AOS, a message can be updated by any party. Moreover, the update operations considered in [BGG94, BGG95] are `replace`, `insert` and `delete` whereas we are interested in performing `append` operations on messages.

Application to Secure Routing. In our discussion of applications of AOS, which we expand upon in Section 5, our main focus is on the problem of wide-area Internet routing security. We argue that the existence of secure AOS schemes is a sufficient condition for guaranteeing an important security property of routing protocols, namely, authenticity of route announcements. Though routing is an important component of modern communication networks, its security has received little attention from the cryptography community (although a rich literature exists in the computer networks community [KLS00, SRS$^+$04, HPS04, WKvO05]). Most cryptographic protocols assume that the network is an untrusted black box possibly under full control of the adversary; while this is useful when modeling the security of end-to-end protocols, it fails to capture the security of the underlying routing protocols. We are motivated by this apparent gap, as routing is not just an application in which cryptography is required, but a necessary component of the networks used by most cryptographic protocols.

2 Append-only Signatures

Informally, append-only signatures (AOS) are signatures that enable the public extension of existing signatures. That is, any party given an AOS signature `Sig` on a message $(M_1, \ldots, M_n)$ can compute an AOS signature on any message $(M_1, \ldots, M_n, M_{n+1})$. (As in the introduction, one could represent the message $(M_1, \ldots, M_n)$ as the string $M_1 || \ldots || M_n$ which better captures the idea of appending. However, since we want to differentiate between a message of the form "A"$\|$"B" and that of the form "AB" ("A","B" and "AB" being three different message symbols), we prefer to think of messages as n-tuples). Besides the append operation, AOS is the same as ordinary signatures. That is, given only an AOS signature on the message $(M_1, \ldots, M_n)$ it should be infeasible to forge an AOS signature on any message not having $(M_1, \ldots, M_n)$ as a prefix. In terms of conventional signatures, AOS may seem strange, as it allows the "forgery" of signatures on messages not previously obtained. In particular, given a signature on the empty message ε, a signature on any message $(M_1, \ldots, M_n)$ can be computed. In the context of AOS, we view this as a feature, and, as we will show, this is useful in several applications.

We now formally define AOS and the corresponding notion of security. Let $\mathsf{AOS.MSpace}$ be any set of symbols (for example, $\{0,1\}$ or $\{0,1\}^*$). For an integer $n \geq 0$, a message of length n is an n tuple of symbols written as $M[1..n] = (M_1, M_2, \ldots, M_n)$ with $M_i \in \mathsf{AOS.MSpace}$. The special case of $n = 0$ is the empty message, denoted as ε, also written as $M[1..0]$. We use the symbol $\sqsubseteq$ to denote the prefix relation over the messages: for a given message $M[1..n] = (M_1, M_2, \ldots, M_n)$, any message from the set $\{M[1..i], 0 \leq i \leq n\}$ is a prefix. Note that ε is a prefix of any other message.

An append-only signature (AOS) scheme with respect to the message space $\mathsf{AOS.MSpace}$ is a collection of three algorithms: a setup algorithm ($\mathsf{AOS.Setup}$), an append algorithm ($\mathsf{AOS.Append}$), and a verify algorithm ($\mathsf{AOS.Vfy}$), defined as follows:

- $\mathsf{AOS.Setup}$ (the key generation algorithm) takes the security parameter as input and outputs a pair of keys: the public key $\mathsf{AOS.pk}$ and the secret key $\mathsf{Sig}[\varepsilon]$, which is the signature on the empty message ε.

- $\mathsf{AOS.Append}$ (the append algorithm) takes the public key $\mathsf{AOS.pk}$, a signature on a message $M[1..n-1] = (M_1, \ldots M_{n-1})$, of length $n-1$, and a symbol $M_n \in \mathsf{AOS.MSpace}$ and produces a signature on the message $M[1..n] = (M_1, \ldots, M_n)$.

- $\mathsf{AOS.Vfy}$ (the verification algorithm) takes the public key $\mathsf{AOS.pk}$, a message $M[1..n]$, and a signature $\mathtt{sig}$, and returns either $\mathtt{true}$ or $\mathtt{false}$.

All algorithms can be randomized and all of them must be polynomial-time in the security parameter. Additionally, the scheme should have the property that for any pair $(\mathsf{AOS.pk}, \mathsf{Sig}[\varepsilon])$ generated by $\mathsf{AOS.Setup}(1^k)$ and any message $M[1..n] = (M_1, M_2, \ldots, M_n)$ (where n is polynomially bounded in the security parameter), the signature on $M[1..n]$ given by

$$\begin{aligned}\mathtt{sig} = \mathsf{AOS.Append}(\mathsf{AOS.pk}, M[1..n-1], \mathsf{AOS.Append}(\mathsf{AOS.pk}, M[1..n-2], \ldots \\ \ldots, \mathsf{AOS.Append}(\mathsf{AOS.pk}, \mathsf{Sig}[\varepsilon], M_1), \cdots, M_{n-1}), M_n)\end{aligned} \quad (1)$$

should be accepted by a verification algorithm. That is, $\mathsf{AOS.Vfy}(\mathsf{AOS.pk}, M[1..n], \mathtt{sig})$ must return $\mathtt{true}$.

The way an AOS signature is defined in Eq. (1) implies that the only way of appending a sequence of symbols to a given AOS signature is to append the symbols one-by-one. This means that the distribution of an AOS signature created by appending a symbol M_n to a message $(M_1, \cdots, M_{n-1})$ is the same as the distribution of the signature on the message $(M_1, \cdots, M_{n-1}, M_n)$ when generated from scratch (using the secret key $\mathsf{Sig}[\varepsilon]$). This fact ensures history independence of AOS: that is, no party, given an AOS signature, can tell whether the signature was created by the owner of the secret key or whether it passed through multiple parties that appended symbols at every step[1]. History independence is a useful property to have

[1]The above definition precludes trivial schemes of the following form: Let $\mathcal{SGN} = (\mathsf{SGN.G}, \mathsf{SGN.S}, \mathsf{SGN.V})$ be any standard digital signature scheme. Construct an append-only signature scheme using $\mathcal{SGN}$ in which the signature of any message $M[1..n] = (M_1, \cdots, M_n)$ is $(\mathsf{SGN.S}(M[1..n]), n)$ and the program $\mathsf{AOS.Append}$ takes a

in most applications (as already highlighted in previous work on algebraic signatures [JMSW02] and incremental signatures [BGG95]).

Definition 2.1 [AOS-UF-CMA] Let $\mathcal{AOS} = (\mathsf{AOS.Setup}, \mathsf{AOS.Append}, \mathsf{AOS.Vfy})$ be an AOS scheme, let k be the security parameter, and let $\mathcal{A}$ be an adversary. We consider the experiment:

Experiment $\mathbf{Exp}^{\text{aos-uf-cma}}_{\mathcal{AOS},\mathcal{A}}(k)$
  $\mathit{MSGSet} \leftarrow \emptyset$; $(\mathsf{AOS.pk}, \mathsf{Sig}[\varepsilon]) \xleftarrow{\$} \mathsf{AOS.Setup}(1^k)$
  $(M[1..n], \mathtt{sig}) \xleftarrow{\$} \mathcal{A}^{\text{AOSSign}(\cdot)}(\mathsf{AOS.pk})$
  if $\mathsf{AOS.Vfy}(\mathsf{AOS.pk}, M[1..n], \mathtt{sig}) = \mathtt{true}$
    and $\forall J[1..j] \sqsubseteq M[1..n] : J[1..j] \notin \mathit{MSGSet}$
    then return 1 **else return** 0

Oracle $\text{AOSSign}(M[1..n])$
  $\mathit{MSGSet} \leftarrow \mathit{MSGSet} \cup \{M[1..n]\}$
  return $\text{Extract}(M[1..n])$

Oracle $\text{Extract}(M[1..i])$ // defined recursively
  if $i = 0$ **then return** $\mathsf{Sig}[\varepsilon]$
  else if $\mathsf{Sig}[M[1..i]] =$ defined
    then return $\mathsf{Sig}[M[1..i]]$
    else $\mathsf{Sig}[M[1..i]] \xleftarrow{\$} \mathsf{AOS.Append}(\mathsf{AOS.pk}, M[1..i-1], \text{Extract}(M[1..i-1]), M_i)$
      return $\mathsf{Sig}[M[1..i]]$

The aos-uf-cma-advantage of an adversary $\mathcal{A}$ in breaking the security of the scheme $\mathcal{AOS}$ is defined as

$$\mathbf{Adv}^{\text{aos-uf-cma}}_{\mathcal{AOS},\mathcal{A}}(k) = \Pr[\,\mathbf{Exp}^{\text{aos-uf-cma}}_{\mathcal{AOS},\mathcal{A}}(k) = 1\,] ,$$

and $\mathcal{AOS}$ is said to be unforgeable under chosen message attacks (aos-uf-cma-secure) if the above advantage is a negligible function in k for all polynomial-time adversaries $\mathcal{A}$.

Note that in our definition of security, adversary $\mathcal{A}$ is given access to the oracle $\text{AOSSign}(\cdot)$, not to the oracle $\text{Extract}(\cdot)$. (The latter is used internally by $\text{AOSSign}(\cdot)$ to create intermediate signatures.) The history independence property of AOS ensures that the adversary can get no advantage when given the power to decide "how" the signature on any message is to be created by $\text{AOSSign}(\cdot)$ (for example, whether it asks for a message (M_1, M_2) to be signed from scratch or by first signing M_1 and then appending M_2, it would get the same reply in return).

3 Efficient AOS Constructions

3.1 Certificate-Based Append-Only Signatures

We present an efficient construction of a provably-secure AOS scheme based on a public-key signature scheme. Let $\mathcal{SGN} = (\mathsf{SGN.G}, \mathsf{SGN.S}, \mathsf{SGN.V})$ be a signature scheme with a space of public keys $\mathsf{SGN.PKSpace}$ and message space $\mathsf{SGN.MSpace} = \mathsf{AOS.MSpace} \times \mathsf{SGN.PKSpace}$. (A formal definition of a public-key signature scheme including a security definition is given in Appendix A.) That is, messages to be signed by $\mathcal{SGN}$ are tuples of the form (M, pk), where $M \in \mathsf{AOS.MSpace}$ and $pk \in \mathsf{SGN.PKSpace}$. Intuitively, an AOS signature $\mathtt{Sig}$ of a message $M[1..n]$ consists of the following elements:

$$(pk_1, sig_1, \ldots, pk_n, sig_n, sk_n),$$

message $M[1..n]$, its signature (σ, n) and a new symbol $M[n+1]$ and simply outputs (σ, n). Verification of a signature (σ, n) on message $M[1..N]$ $(N \geq n)$ is carried out by testing if σ is the signature, according to $\mathcal{SGN}$, on $M[1..n]$. Although this scheme allows appending to already signed messages in an arbitrary manner, one can easily distinguish between signatures created by such append operations and those created from scratch.

where for $1 \leq i \leq n$, (pk_i, sk_i) are random public/secret key pairs of the public-key signature scheme $\mathcal{SGN}$ and sig_i is a signature on the tuple (M_i, pk_i) under the secret key sk_{i-1}. Note that the secret key sk_i used to sign sig_i is entangled with sig_{i+1} by signing its corresponding public key pk_i, thereby certifying its validity. For this reason, this construction is sometimes referred to as a certificate chain. The signature sig_0 needs to be signed with the secret key sk_0, which we define to be the master secret key.

More formally, we construct the AOS scheme $\mathcal{AOS}1$ with the message space $\mathsf{AOS.MSpace}$ as specified below:

- $\mathsf{AOS.Setup}(1^k)$ (the setup algorithm):
  Run $\mathsf{SGN.G}(1^k)$ to generate a pair (sk_0, pk_0). Set $\mathsf{AOS.pk} \leftarrow pk_0$ and $\mathtt{Sig}[\varepsilon] \leftarrow (sk_0)$. Return $(\mathsf{AOS.pk}, \mathtt{Sig}[\varepsilon])$.

- $\mathsf{AOS.Append}(\mathsf{AOS.pk}, \mathtt{Sig}[M[1..n]], M_{n+1})$ (the append algorithm):
  Parse $\mathtt{Sig}$ as $(pk_1, sig_1, \ldots, pk_n, sig_n, sk_n)$. If $n = 0$, then $\mathtt{Sig}[\varepsilon]$ consists of a single secret key sk_0. Run $\mathsf{SGN.G}(1^k)$ to generate a pair (sk_{n+1}, pk_{n+1}). Compute $sig_{n+1} \leftarrow \mathsf{SGN.S}_{sk_n}(M_{n+1}, pk_{n+1})$. Return $(pk_1, sig_1, \ldots, pk_n, sig_n, pk_{n+1}, sig_{n+1}, sk_{n+1})$.

- $\mathsf{AOS.Vfy}(\mathsf{AOS.pk}, M[1..n], \mathtt{Sig})$ (the verification algorithm):
  Parse $\mathtt{Sig}$ as $(pk_1, sig_1, \ldots, pk_n, sig_n, sk_n)$. If $n = 0$, then $\mathtt{Sig} = (sk_0)$. Set pk_0 to be the master public key $\mathsf{AOS.pk}$. For $i = 1..n-1$ verify that $\mathsf{SGN.V}(pk_{i-1}, sig_i, (M_i, pk_i)) = \mathtt{true}$. If any of the verifications fail, return $\mathtt{false}$. If all the verifications succeed, verify that (sk_n, pk_n) is a valid secret key/public key pair: pick any message $M \in \mathsf{SGN.MSpace}$ and compute $sig \xleftarrow{\$} \mathsf{SGN.S}(sk_n, M)$. Return $\mathtt{true}$ if $\mathsf{SGN.V}(pk_n, sig, M) = \mathtt{true}$ and $\mathtt{false}$ otherwise.

The length of a signature of $\mathcal{AOS}1$ grows linearly with the number of symbols in a message. The efficiency of $\mathcal{AOS}1$ is summarized in Table 1. We prove aos-uf-cma security of $\mathcal{AOS}1$ provided that the original public-key signature scheme $\mathcal{SGN}$ is sig-uf-cma secure (as defined in Appendix A).

Theorem 3.1 The AOS scheme $\mathcal{AOS}1$ is aos-uf-cma secure assuming that the public-key signature scheme $\mathcal{SGN}$ is sig-uf-cma secure.

The full proof of Theorem 3.1 is in Appendix B. Here we sketch the main ideas of why this construction works. Intuitively, in order to break the aos-uf-cma security of $\mathcal{AOS}1$, an adversary has two choices between which we must distinguish. First, she could try to forge a signature on a prefix of a message she already knows the signature of. Since a valid $\mathcal{AOS}1$ signature of this prefix (say, of length n') has to contain the secret key $sk_{n'}$ in cleartext, this would imply a full break of the security of the signature scheme. Second, the adversary could try to forge an AOS signature on a message that is different from all those with known signatures. To do so, the adversary could use existing public/secret key pairs, meaning she has to produce (for some i) a new signature on a tuple (M_i, pk_i) under an unknown secret key and a different message M_i. Otherwise, the adversary breaks the certificate chain. That is, at some position i, the adversary creates a fresh secret-public key pair (sk_i, pk_i) and uses sk_i to create $\mathtt{sig}_i$. However, $\mathtt{sig}_{i-1}$ is a signature on the public key pk_i and the symbol M_{i-1} under the secret key sk_{i-1}. In order to use a new secret key sk_i to create $\mathtt{sig}_i$, the adversary has to forge a signature under the unknown secret key sk_{i-1}. This clearly contradicts the uf-cma security of the signature scheme.

Metric	Certificate-based AOS
Signature length	n $\mathcal{SGN}$ signatures, n $\mathcal{SGN}$ public keys, 1 $\mathcal{SGN}$ secret key
Setup time	$1 \times \mathsf{SGN.G}(\cdot)$
Append time	$1 \times \mathsf{SGN.G}(\cdot)$, $1 \times \mathsf{SGN.S}(\cdot)$
Verify time	$(n+1) \times \mathsf{SGN.V}(\cdot)$, $1 \times \mathsf{SGN.S}(\cdot)$

Table 1: Efficiency of certificate-based AOS. Data is given for messages of length n.

3.2 Shorter Signatures via Aggregation

An aggregate signature scheme, $\mathcal{ASGN} = (\mathsf{ASGN.G}, \mathsf{ASGN.S}, \mathsf{ASGN.AGG}, \mathsf{ASGN.V})$, allows the aggregation of n signatures on n distinct messages from n distinct users into a single signature. Its verification algorithm, $\mathsf{ASGN.V}(n, \cdot)$, takes an aggregated signature, n messages, and n public keys and verifies that the n users signed the n messages. A sequential signature aggregation algorithm assumes to receive the signatures sequentially: given an aggregated signature of $n-1$ messages and a signature on an n^{th} message, it outputs an aggregated signature for all n messages.

When using the certificate-based construction of AOS from Section 3.1, we can use sequential signature aggregation to shrink the size of the signature (without significantly decreasing security or efficiency). To be more precise, the length of an AOS signature of a message of length n can be condensed to one signature of $\mathcal{ASGN}$, n public keys of $\mathcal{ASGN}$, and one secret key of $\mathcal{ASGN}$. We summarize the efficiency of this approach in Table 2. We note that there are two known signature aggregation techniques. The first scheme, given in [BGLS03], is based on bilinear maps. The second scheme (only supporting sequential aggregation) is from [MOR01] and can be based on homomorphic trapdoor permutations (such as RSA). We note that both aggregation schemes are in the random oracle model.

Metric	Certificate-based AOS with aggregation
Signature length	1 $\mathcal{ASGN}$ signature, n $\mathcal{ASGN}$ public keys, 1 $\mathcal{ASGN}$ secret keys
Setup time	$1 \times \mathsf{ASGN.G}(\cdot)$
Append time	$1 \times \mathsf{ASGN.G}(\cdot)$, $1 \times \mathsf{ASGN.S}(\cdot)$, $1 \times \mathsf{ASGN.AGG}(\cdot)$
Verify time	$1 \times \mathsf{ASGN.V}(n, \cdot)$, $1 \times \mathsf{ASGN.V}(1, \cdot)$, $1 \times \mathsf{ASGN.S}(\cdot)$

Table 2: Efficiency of AOS with signature aggregation. Data is given for messages of length n.

3.3 Compact Signatures via the Boneh-Goh-Boyen HIBE

Even if we apply the techniques of signature aggregation to our certificate-based AOS scheme, the signature length remains linear in n. Based on a recent HIBE construction from Boneh, Goh, and Boyen [BBG05] we construct an AOS scheme $\mathcal{AOS}2$ whose signatures are of size square root of the maximum message length. The scheme is in fact a hybrid between the two schemes from [BBG05, BB04] exploiting a common algebraic structure. It is based on bilinear groups. The scheme is described in Appendix D and its efficiency is given in Table 3.

Metric	HIBE-based $\mathcal{AOS}2$
Signature length	$\leq 2\sqrt{d}+1$ elements $\in \mathbb{G}_1$
Setup time	$1 \times e(\cdot,\cdot)$, 1 exp, $2\sqrt{d}+1$ rand number gen $\in \mathbb{G}_1$
Append time	$\leq \sqrt{d}+3$ exp/mult
Verify time	$(\sqrt{d}+1) \times e(\cdot,\cdot)$, $d+\sqrt{d}$ exp/mult

Table 3: Efficiency of $\mathcal{AOS}2$. d represents the maximum message length. $e(\cdot,\cdot)$ is a pairing operation on elements of the group $\mathbb{G}_1$ as used by the HIBE scheme.

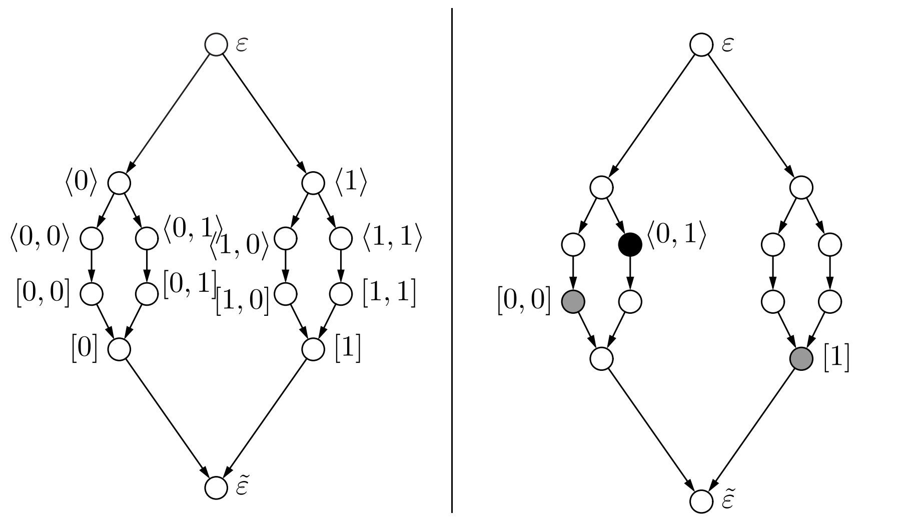

Figure 1: Structure of the hash-tree construction for $d = 2$. The diagram on the left depicts the hash tree. The diagram on the right highlights the node $u = \langle 0, 1\rangle$ (shown in black) and the set of its complements, $\mathrm{Comp}(u)$ (shown in gray).

3.4 AOS via Hash Trees

If the number of symbols in the alphabet $\mathsf{AOS.MSpace}$ is small, AOS can be efficiently implemented using hash trees [Mer88]. This approach suffers from dramatic complexity blowup as the size of the message space increases, but uses only secret-key primitives and provides good security guarantees. We believe that this construction is useful in computationally constrained applications.

Next we construct an AOS scheme $\mathcal{AOS}3$ with fixed message space $\mathsf{AOS.MSpace} = \{0, 1\}$; the messages of $\mathcal{AOS}3$ are limited to length at most d. The construction uses a pseudorandom generator and a collision-resistant hash function (for a formal definition of the two primitives we refer the reader to the textbook of Goldreich [Gol01]). Let $G : \{0, 1\}^k \to \{0, 1\}^{2k}$ be a pseudorandom generator. Denote $G_i : \{0, 1\}^k \to \{0, 1\}^k$ to be the i-th k-bit component of G for $i \in \{0, 1\}$. Let $H : \{0, 1\}^{2k} \to \{0, 1\}^k$ be a collision-resistant hash function.

Consider the left graph T depicted in Figure 1. T consists of the upper tree UT and lower tree LT. The top node ε is called the source and the bottom node $\tilde{\varepsilon}$ is called the destination. Let $\langle v_1, \ldots, v_j\rangle$ denote the node at level j below ε (in the upper tree) such that each $v_i \in \{0, 1\}$ is an index of a node taken at the i-th level on the path from ε to $\langle v_1, \ldots, v_j\rangle$. A mirror image of this node in the lower tree is denoted as $[v_1, \ldots, v_j]$.

Let $u = \langle v_1, \ldots, v_j\rangle$ be any node in the upper tree of the graph. We define the complement of u, denoted $\mathrm{Comp}(u)$, to be the minimal set of nodes in $LT - \{\tilde{\varepsilon}\}$ such that every path from ε to $\tilde{\varepsilon}$ passes through exactly one node from $\{u\} \cup \mathrm{Comp}(u)$. An example of a complement set is given in Figure 1 (the right graph). Let $\neg$ denote the not operator. Then $\mathrm{Comp}(u) = \{[v_1, \ldots, v_{i-1}, \neg v_i] \mid i = 1, \ldots, j\}$.

In $\mathcal{AOS}3$, we associate every node of the graph T with a secret key. Keys are assigned in a top-down manner, starting with a random key for the root ε of UT. Furthermore, for nodes in UT everybody can compute $\mathrm{key}(\langle v_1, \ldots, v_j, 0\rangle)$ and $\mathrm{key}(\langle v_1, \ldots, v_j, 1\rangle)$ from $\mathrm{key}(\langle v_1, \ldots, v_j, 0\rangle)$ (using the pseudo random generators $G_0(\cdot), G_1(\cdot)$) and for nodes in LT everybody can compute $\mathrm{key}([v_1, \ldots, v_j])$ from $\mathrm{key}([v_1, \ldots, v_j, 0])$ and $\mathrm{key}([v_1, \ldots, v_j, 1])$ (using the hash function $H(\cdot, \cdot)$). The nodes between LT and UT are "connected" through the pseudorandom generator $G_0(\cdot)$. The secret key is $\mathrm{key}(\varepsilon)$, the public key is $\mathrm{key}(\tilde{\varepsilon})$. The AOS of a node $\langle v_1, \ldots, v_n\rangle$ (representing the message $M[1..n] = (M_1, \ldots, M_n) \in \{0, 1\}^n$, $n \leq d$) is given by the set of keys $\{\mathrm{key}(x) \mid x \in \{\langle v_1, \ldots, v_n\rangle\} \cup \mathrm{Comp}(\langle v_1, \ldots, v_n\rangle)\}$. Verification of a given AOS is done by computing top-down the corresponding keys in LT and checking if the last key matches the public key $\mathrm{key}(\tilde{\varepsilon})$. The algorithms constituting $\mathcal{AOS}3$ are defined as follows:

- $\mathsf{AOS.Setup}(1^k)$:
  - Pick $\mathrm{key}(\varepsilon) \xleftarrow{\$} \{0, 1\}^k$; $\mathtt{Sig}[\varepsilon] \leftarrow \mathrm{key}(\varepsilon)$.
  - Compute $\mathrm{key}(u) \in \{0, 1\}^k$ for all nodes $u \in T$ as follows:
    - For every node $\langle v_1, \ldots, v_j\rangle \in UT$ recursively compute
      - $\mathrm{key}(\langle v_1, \ldots, v_j\rangle) = G_{v_j}(\mathrm{key}(\langle v_1, \ldots, v_{j-1}\rangle))$.
    - For every node $c = [v_1, \ldots, v_d]$ at the d-th level of LT $\mathrm{key}(c) = G_0(\langle v_1, \ldots, v_j\rangle)$.
    - For the remaining nodes in LT recursively compute
      - $\mathrm{key}([v_1, \ldots, v_j]) = H(\mathrm{key}([v_1, \ldots, v_j, 0]), \mathrm{key}([v_1, \ldots, v_j, 1]))$;
    - The public key is $\mathrm{key}(\tilde{\varepsilon}) = H(\mathrm{key}([0]), \mathrm{key}([1]))$.
  - Return $(\mathrm{key}(\varepsilon), \mathrm{key}(\tilde{\varepsilon}))$.

- $\mathsf{AOS.Append}(\mathtt{Sig}[M[1..n]], M_{n+1})$: //denote $\langle M_1, \ldots, M_n\rangle$ as u
  - Parse $\mathtt{Sig}$ as a set $\{\mathrm{key}(x) \mid x \in \{u\} \cup \mathrm{Comp}(u)\}$.
  - Compute $\mathrm{key}(\langle M_1, \ldots, M_{n+1}\rangle) = G_{M_{n+1}}(\mathrm{key}(\langle M_1, \ldots, M_n\rangle))$.
  - Compute $\mathrm{key}([M_1, \ldots, M_n, \neg M_{n+1}])$ from $\mathrm{key}(\langle M_1, \ldots, M_n\rangle)$ by iterating $G(\cdot)$ and $H(\cdot, \cdot)$
    - // Note that $\mathrm{Comp}(\langle M_1, \ldots, M_{n+1}\rangle) = [M_1, \ldots, M_n, \neg M_{n+1}] \cup \mathrm{Comp}(u)$.
  - Return $\{\mathrm{key}(x) \mid x \in \{\langle M_1, \ldots, M_{n+1}\rangle\} \cup \mathrm{Comp}(\langle M_1, \ldots, M_{n+1}\rangle)\}$.

- $\mathsf{AOS.Vfy}(\mathsf{AOS.pk}, M[1..n], \mathtt{Sig})$:
  - Let $u = \langle M_1, \ldots, M_n\rangle$. Parse $\mathtt{Sig}$ as a set $\{\mathrm{key}(x) \mid x \in \{u\} \cup \mathrm{Comp}(u)\}$.
  - By iterating $G(\cdot)$ and $H(\cdot, \cdot)$ compute $\mathrm{key}(\cdot)$ for all the descendants of $\{u\} \cup \mathrm{Comp}(u)$.
  - If $\mathrm{key}(\tilde{\varepsilon}) = \mathsf{AOS.pk}$ return 1, otherwise return 0.

We give the efficiency of this scheme in Table 4. We prove aos-uf-cma security of the $\mathcal{AOS}3$ scheme assuming the security of the underlying functions $G(\cdot)$ and $H(\cdot, \cdot)$:

Theorem 3.2 If $G(\cdot)$ is a secure pseudorandom generator, $G_0(\cdot)$, $G_1(\cdot)$, are secure one-way functions and $H(\cdot, \cdot)$, $G_0(\cdot)$, and $G_1(\cdot)$ are all collision-resistant hash functions, then $\mathcal{AOS}3$ is aos-uf-cma secure.

The proof can be found in Appendix C.

Metric	Hash-tree based
Signature length	$(n+1)$ k-bit blocks
Setup time	$(2^{d+1}-1)\times G(\cdot)$, $(2^{d}-1)\times H(\cdot,\cdot)$
Append time	$2^{d-n}\times G(\cdot)$, $(2^{d-n-1}-1)\times H(\cdot,\cdot)$
Verify time	$(2^{d-n+1}-1)\times G(\cdot)$, $(2^{d-n}+n-1)\times H(\cdot,\cdot)$

Table 4: Efficiency of the hash-tree based scheme $\mathcal{AOS}3$. d represents the maximum message length, n represents the length of a given message, and k is the security parameter.

3.5 AOS via One-time Signatures

We observe that we can combine the ideas of certificate-based AOS (Section 3.1) and hash-tree AOS (Section 3.4) to gain a more efficient append-only signature scheme when the message space is small. Assume the message space $\mathsf{AOS.MSpace}$ consists of m elements. Then we can use our certificate-based construction $\mathcal{AOS}1$ instantiated with a m-time signature scheme. m-time signatures can be efficiently constructed using hash-trees (see [BM96], [HM02], [Mer88], and [RR02] for the definition and efficient constructions of m-time signatures). In addition, the security proof of $\mathcal{AOS}1$ guarantees unforgeability if $\mathcal{SGN}$ is at least an $|\mathsf{AOS.MSpace}|$-time signature scheme. Note that in contrast to $\mathcal{AOS}3$, the length of the AOS messages in this construction is unbounded[2].

4 Relations between HIBS and AOS

In this section, we show that the concepts of AOS and Hierarchical Identity-based Signatures (HIBS) are in fact equivalent. Before we present the poly-time reductions between the AOS and HIBS, we first review related concepts and relationships.

In Identity-based Signature (IBS) schemes, the identity of a sender (for example, an email address) is used as a public key for verification of the signature. This approach assumes the existence of a trusted party (the certificate authority), which assigns secret keys to all users. The certificate authority has a pair of keys: a master public key and a master secret key; the master secret key is used to delegate keys to the users and the master public key is used for signature verification. Anyone can verify signatures on messages signed by any user, knowing only the master public key and the identity of that user.

Hierarchical Identity-based Signature (HIBS) schemes are a natural generalization of IBS to the setting with a hierarchical organization of users. Assume that the users are structured in a tree with a root being the certificate authority. Descendants of a user are the usernames that contain this user's name as a prefix; the canonical example involves domain names. In HIBS, each individual user can play the role of a certificate authority and can delegate secret keys to his descendants. As in IBS, a secret key allows a user to sign arbitrary messages such that anyone is able to verify the signature knowing only the identity of the sender and the master public key (announced by the certificate authority, who can be viewed as the root user). A formal description of HIBS follows.

Hierarchical Identity-based Encryption (HIBE) assumes the same hierarchy of users as HIBS and provides encryption/decryption mechanisms rather than signatures. In HIBE, anyone can

[2] Similar ideas were used by Abdalla and Reyzin [AR00] who who suggested how to improve the efficiency of binary certification method for constructing forward-secure signatures (see also Bellare and Miner [BM99]).

encrypt data to any user in the hierarchy knowing only the user's identity and the master public key; the ciphertext can only be decrypted using the user's secret key.

As noted by Naor (see Section 6 of [BF03]), any IBE scheme can readily be converted into a public key signature scheme (by interpreting user identities of IBE as messages for a regular signature scheme and defining the signature of a message to be the secret key associated with the corresponding identity). Similarly, any HIBE scheme can be transformed into a HIBS scheme. This was sketched by Gentry and Silverberg [GS02], giving the construction for a HIBS scheme. We note that the converse (transforming HIBS into HIBE) is not known to be possible. Related to these are Forward-secure Signature (FSS) schemes, which modify a secret key over time (while the public key remains the same) such that exposure of the secret key at a certain time period does not allow forgery of signatures from previous time periods. (See [BM99] for an exact definition of FSS.) In [CHK03] it is proved that HIBE implies Forward-secure Encryption (FSE) (which is defined as FSS with signing replaced by encryption). Using the same construction it is easy to show that HIBS implies FSS (as explicitly noted in [CHYC04]). More precisely, a HIBS scheme of depth d can be used to construct a forward-secure scheme providing security for 2^d time steps (using a tree-based construction). The converse (transforming FSS into HIBS) is not known to be possible. Thus, relating HIBE, HIBS, AOS, and FSS, we obtain the following hierarchy:

$$\text{HIBE} \Rightarrow \text{HIBS} \Leftrightarrow \text{AOS} \Rightarrow \text{FSS}$$

In particular, given a secure HIBE scheme, we can construct a secure AOS scheme and given a secure AOS scheme, it is easy to obtain a secure FSS scheme.

4.1 Definition of HIBS

We recall the syntax of Hierarchical Identity-based Signature (HIBS) schemes and the appropriate notions of unforgeability. Let $\mathsf{HIBS.IDSpace}$ be any set of identities (typically $\{0,1\}^*$). For an integer $n \geq 0$, a username at the level n in the tree (called hierarchical identity of depth n) is an (ordered) n-tuple of identities written as $I[1..n] = (I_1, I_2, \ldots, I_n)$ with each $I_i \in \mathsf{HIBS.IDSpace}$. The special case of $n = 0$ is the root identity, denoted as $I[1..0]$ or ε. Further on, we will refer to strings from $\mathsf{HIBS.IDSpace}$ as identities and to n-tuples of them as hierarchical identities. We use the symbol $\sqsubseteq$ to denote the prefix relation over the set of hierarchical identities: for a given hierarchical identity $I[1..n] = (I_1, I_2, \ldots, I_n)$, any hierarchical identity from the set $\{I[1..i], 0 \leq i \leq n\}$ is its prefix. Note that the root identity ε is a prefix of any other hierarchical identity.

A HIBS scheme $\mathcal{HIBS}$ with respect to the message space $\mathsf{HIBS.MSpace}$ and the identity space $\mathsf{HIBS.IDSpace}$ is made up of four algorithms: a setup algorithm $\mathsf{HIBS.Setup}$, a key delegation algorithm $\mathsf{HIBS.KeyDel}$, a signature algorithm $\mathsf{HIBS.Sign}$, and a verification algorithm $\mathsf{HIBS.Vfy}$.

- $\mathsf{HIBS.Setup}$ (the setup algorithm) takes as input a security parameter and generates the master public key $\mathsf{HIBS.pk}$ of the scheme and the secret key of the root identity $\mathsf{HIBS.SK}[\varepsilon]$ (the master secret key).

- $\mathsf{HIBS.KeyDel}$ (the key delegation algorithm) takes as input a hierarchical identity $I[1..n] = (I_1, \ldots, I_n)$, its associated secret key $\mathsf{HIBS.SK}[I[1..n]]$, and an identity $I_{n+1} \in \mathsf{HIBS.IDSpace}$ of its child. It returns a secret key $\mathsf{HIBS.SK}[I[1..n+1]]$ associated with the new hierarchical identity $I[1..n+1] = (I_1, \ldots, I_n, I_{n+1})$.

- $\mathsf{HIBS.Sign}$ (the signing algorithm) takes a hierarchical identity $I[1..n]$, the associated secret key $\mathsf{HIBS.SK}[I[1..n]]$, and a message $M \in \mathsf{HIBS.MSpace}$. It computes a signature on this message M with respect to this identity.

- $\mathsf{HIBS.Vfy}$ (the verification algorithm) takes the master public key $\mathsf{HIBS.pk}$, a hierarchical identity $I[1..n]$, a message M, and a signature $\mathtt{sig}$. It outputs $\mathtt{true}$ or $\mathtt{false}$ depending on whether $\mathtt{sig}$ is a valid signature of M signed by hierarchical identity $I[1..n]$.

All these algorithms can be randomized. All of them must be polynomial-time in the security parameter. Moreover, it is required that for all pairs $(\mathsf{HIBS.pk}, \mathsf{HIBS.SK}[\varepsilon])$ of master public and secret keys output by $\mathsf{HIBS.Setup}$, and for all messages $M \in \mathsf{HIBS.MSpace}$, hierarchical identities $I[1..n]$ and associated secret keys $\mathsf{HIBS.SK}[I[1..n]]$ (recursively generated from the secret key $\mathsf{HIBS.SK}[\varepsilon]$ using the $\mathsf{HIBS.KeyDel}$ algorithm),

$$\mathsf{HIBS.Vfy}(\mathsf{HIBS.pk}, I[1..n], \mathsf{HIBS.Sign}(\mathsf{HIBS.SK}[I[1..n]], I[1..n]), M), M) = \mathtt{true}.$$

Unforgeability of the HIBS scheme $\mathcal{HIBS}$ under chosen-plaintext attacks is formally defined as follows:

Definition 4.1 [HIBS-UF-CMA] Let $\mathcal{HIBS} = (\mathsf{HIBS.Setup}, \mathsf{HIBS.KeyDel}, \mathsf{HIBS.Sign}, \mathsf{HIBS.Vfy})$ be a hierarchical identity-based signature scheme, let k be the security parameter, and let $\mathcal{A}$ be an adversary. We consider the experiment:

Experiment $\mathbf{Exp}^{\text{hibs-uf-cma}}_{\mathcal{HIBS},\mathcal{A}}(k)$
  $\mathit{IDSet} \leftarrow \emptyset$; $(\mathsf{HIBS.pk}, \mathsf{HIBS.SK}[\varepsilon]) \leftarrow \mathsf{HIBS.Setup}(1^k)$
  $(I[1..n], M, \mathtt{sig}) \leftarrow \mathcal{A}^{\textsc{Corrupt}(\cdot),\textsc{Sign}(\cdot,\cdot)}(\mathsf{HIBS.pk})$
  if $\mathsf{HIBS.Vfy}(I[1..n], M, \mathtt{sig}) = \mathtt{true}$
    and $\forall\ J[1..j] \sqsubseteq I[1..n] :\ J[1..j] \notin \mathit{IDSet}$
    and $(I[1..n], M) \notin \mathit{MSGSet}$
  then return 1 **else return** 0

Oracle $\textsc{Corrupt}(I[1..n])$
  $\mathit{IDSet} \leftarrow \mathit{IDSet} \cup \{I[1..n]\}$
  return $\textsc{Extract}(I[1..n])$

Oracle $\textsc{Sign}(I[1..n], M)$
  $\mathit{MSGSet} \leftarrow \mathit{MSGSet} \cup \{(I[1..n], M)\}$
  $\mathsf{HIBS.SK}[I[1..n]] \leftarrow \textsc{Extract}(I[1..n])$
  return $\mathsf{HIBS.Sign}(\mathsf{HIBS.SK}[I[1..n]], I[1..n], M)$

Oracle $\textsc{Extract}(I[1..i])$ // defined recursively
  if $i = 0$ **return** $\mathsf{HIBS.SK}[\varepsilon]$
  else if $\mathsf{HIBS.SK}[I[1..i]] =$ defined
    then return $\mathsf{HIBS.SK}[I[1..i]]$
    else $\mathsf{HIBS.SK}[I[1..i]] \leftarrow \mathsf{HIBS.KeyDel}(\mathsf{HIBS.pk}, I[1..i-1], \textsc{Extract}(I[1..i-1]), I_i)$
      return $\mathsf{HIBS.SK}[I[1..i]]$

The hibs-uf-cma-advantage of an adversary $\mathcal{A}$ in breaking the security of the scheme $\mathcal{HIBS}$ is defined as

$$\mathbf{Adv}^{\text{hibs-uf-cma}}_{\mathcal{HIBS},\mathcal{A}}(k) = \Pr[\,\mathbf{Exp}^{\text{hibs-uf-cma}}_{\mathcal{HIBS},\mathcal{A}}(k) = 1\,]\ ,$$

and $\mathcal{HIBS}$ is said to be existentially unforgeable under chosen message attacks (hibs-uf-cma-secure) if the above advantage is a negligible function in k for all polynomial-time adversaries $\mathcal{A}$. Note that the adversary is given access to the two oracles $\textsc{Corrupt}(\cdot)$ and $\textsc{Sign}(\cdot,\cdot)$, not to the oracle $\textsc{Extract}(\cdot)$. The latter one is only used internally by the experiment.

4.2 Constructing AOS from HIBS

The idea of the reduction is as follows. We set $\mathsf{AOS.MSpace} = \mathsf{HIBS.IDSpace}$ and associate an AOS message $(M_1, \ldots, M_n)$ of length n with the hierarchical identity $I[1..n] = (M_1, \ldots, M_n)$ of depth n. We then define the signature of this message as the secret key $\mathsf{HIBS.SK}[I[1..n]]$ of the hierarchical identity $I[1..n]$.

Given the above analogy between signatures of messages and secret keys of hierarchical identities, we construct an AOS scheme given a HIBS scheme as follows. Appending to a given signature in $\mathcal{AOS}$ is done using key delegation in $\mathcal{HIBS}$. The verification of an AOS signature $\mathsf{HIBS.SK}[I[1..n]]$ is done by signing a random message $M \in \mathsf{HIBS.MSpace}$ under the secret key $\mathsf{HIBS.SK}[I[1..n]]$ and verifying that the resulting signature is valid.

Construction 4.2 Given a HIBS scheme $\mathcal{HIBS} = (\mathsf{HIBS.Setup}, \mathsf{HIBS.KeyDel}, \mathsf{HIBS.Sign}, \mathsf{HIBS.Vfy})$, we construct an AOS scheme $\mathcal{AOS} = (\mathsf{AOS.Setup}, \mathsf{AOS.Append}, \mathsf{AOS.Vfy})$ as follows:

- $\mathsf{AOS.Setup}$: Run the $\mathsf{HIBS.Setup}$ algorithm to generate a pair $(\mathsf{HIBS.pk}, \mathsf{HIBS.SK}[\varepsilon])$ and output $(\mathsf{HIBS.pk}, \mathsf{HIBS.SK}[\varepsilon])$ as the key pair for $\mathcal{AOS}$. $\mathsf{HIBS.SK}[\varepsilon]$ is the signature of an empty message ε.

- $\mathsf{AOS.Append}$: Given the public key $\mathsf{AOS.pk}$, signature $\mathsf{Sig}[M[1..n]]$ of the message $M[1..n]$, and the message M_n to append, the AOS signature on $\mathsf{Sig}[M[1..n+1]]$ is returned as $\mathsf{Sig}[M[1..n+1]] \leftarrow \mathsf{HIBS.KeyDel}(\mathsf{HIBS.pk}, \mathsf{Sig}[M[1..n]], M_{n+1})$.

- $\mathsf{AOS.Vfy}$: Given a public key $\mathsf{AOS.pk}$, a message $M[1..n]$, and a signature $\mathsf{Sig}[I[1..n]]$, the verification algorithm first signs a random message $M \in \mathsf{HIBS.MSpace}$ under hierarchical identity $M[1..n]$ using $\mathsf{Sig}[I[1..n]]$ as a secret key of $M[1..n]$ in $\mathcal{HIBS}$:

$$\mathtt{sig} \leftarrow \mathsf{HIBS.Sign}(I[1..n], \mathsf{Sig}[I[1..n]], M).$$

  Then it outputs the result of the $\mathcal{HIBS}$ verification $\mathsf{HIBS.Vfy}(\mathsf{HIBS.pk}, I[1..n], M, \mathtt{sig})$.

Theorem 4.3 If the HIBS scheme $\mathcal{HIBS} = (\mathsf{HIBS.Setup}, \mathsf{HIBS.KeyDel}, \mathsf{HIBS.Sign}, \mathsf{HIBS.Vfy})$ is hibs-uf-cma secure, then the AOS scheme from Construction 4.2 is aos-uf-cma secure.

We omit the proof of Theorem 4.3.

4.3 Constructing HIBS from AOS

A naive approach to building a HIBS scheme $\mathcal{HIBS}$ from an AOS scheme $\mathcal{AOS}$ is to first append all the identities and then to append a message to be signed. That is, both the identity space $\mathsf{HIBS.IDSpace}$ and the message space $\mathsf{HIBS.MSpace}$ of $\mathcal{HIBS}$ are subsets of the message space $\mathsf{AOS.MSpace}$ of the $\mathcal{AOS}$ scheme. The secret key of a hierarchical identity $I[1..n]$ in this HIBS scheme is exactly the AOS signature of $I[1..n]$ viewed as the AOS message. Delegation in HIBS is equivalent to appending in AOS. Signing a message M with respect to a hierarchical identity $I[1..n]$ is defined by appending M to the AOS signature of $I[1..n]$.

This naive construction is secure only if $\mathsf{HIBS.IDSpace}$ and $\mathsf{HIBS.MSpace}$ are disjoint subsets of $\mathsf{AOS.MSpace}$. If there is some identity J which itself is a valid message, the security of the HIBS scheme can be broken even if its corresponding AOS is secure. An adversary could query the HIBS signing oracle with a message J and some hierarchical identity $I[1..n]$. The resulting signature equals the secret key for the hierarchical identity $(I_1, \dots, I_n, J)$, which violates the security of the HIBS scheme.

Our idea to overcome this problem is to insert a unique identifier between identities and messages. Let $\mathcal{AOS} = (\mathsf{AOS.Setup}, \mathsf{AOS.Append}, \mathsf{AOS.Vfy})$ be a secure AOS scheme with message space $\mathsf{AOS.MSpace}$. Let $\mathsf{HIBS.IDSpace}$ and $\mathsf{HIBS.MSpace}$ be arbitrary subsets of $\mathsf{AOS.MSpace}$ such that there is some symbol Δ from the AOS message space which is not a valid identity for the HIBS scheme (Δ can still be in the HIBS message space). Then we can construct a secure HIBS scheme $\mathcal{HIBS} = (\mathsf{HIBS.Setup}, \mathsf{HIBS.KeyDel}, \mathsf{HIBS.Sign}, \mathsf{HIBS.Vfy})$ with identity space $\mathsf{HIBS.IDSpace}$ and message space $\mathsf{HIBS.MSpace}$ as follows:

Construction 4.4 $\mathcal{HIBS} = (\mathsf{HIBS.Setup}, \mathsf{HIBS.KeyDel}, \mathsf{HIBS.Sign}, \mathsf{HIBS.Vfy})$:

- $\mathsf{HIBS.Setup}(1^k)$: Run the $\mathsf{AOS.Setup}(1^k)$ algorithm to generate a pair $(\mathsf{AOS.pk}, \mathsf{Sig}[\varepsilon])$ and output it as the master public/private key pair for $\mathcal{HIBS}$.

- $\mathsf{HIBS.KeyDel}(\mathsf{HIBS.pk}, \mathsf{HIBS.SK}[I[1..n]], I_{n+1})$: Given the master public key, the secret key of hierarchical identity $I[1..n]$, and a new identity I_{n+1}, the delegation algorithm interprets $\mathsf{HIBS.SK}[I[1..n]]$ as an $\mathcal{AOS}$ signature of $I[1..n]$. It appends to the signature a message I_{n+1} and outputs the resulting signature as the secret key of $I[1..n+1]$:

$$\mathsf{HIBS.SK}[I[1..n+1]] \leftarrow \mathsf{AOS.Append}(\mathsf{HIBS.pk}, \mathsf{HIBS.SK}[I[1..n]], I_n).$$

- $\mathsf{HIBS.Sign}(\mathsf{HIBS.pk}, \mathsf{HIBS.SK}[I_n], M)$: Given a master public key, a secret key of the hierarchical identity $I[1..n]$, and a message M, the sign algorithm for $\mathcal{HIBS}$ interprets $\mathsf{HIBS.SK}[I[1..n]]$ as an $\mathcal{AOS}$ signature of $I[1..n]$. It appends a symbol Δ to $\mathsf{HIBS.SK}[I[1..n]]$ and then appends a message M to the resulting AOS signature to get the final signature $\mathtt{sig}$:

$$\mathtt{sig} \leftarrow \mathsf{AOS.Append}(\mathsf{HIBS.pk}, \mathsf{AOS.Append}(\mathsf{HIBS.pk}, \mathsf{HIBS.SK}[I[1..n]], \Delta), M).$$

- $\mathsf{HIBS.Vfy}(\mathsf{HIBS.pk}, I[1..n], M, \mathtt{sig})$: Given a master public key, a hierarchical identity $I[1..n]$, a signature $\mathtt{sig}$, and a message M, the verification algorithm for $\mathcal{HIBS}$ returns the output of
$$\mathsf{AOS.Vfy}(\mathsf{HIBS.pk}, (I_1, \ldots, I_n, \Delta, M), \mathtt{sig}).$$

Theorem 4.5 If the AOS scheme $\mathcal{AOS} = (\mathsf{AOS.Setup}, \mathsf{AOS.Append}, \mathsf{AOS.Vfy})$ is aos-uf-cma secure, then the HIBS scheme $\mathcal{HIBS}$ from Construction 4.4 is hibs-uf-cma secure.

Proof: Given an adversary $\mathcal{A}$ attacking the hibs-uf-cma security of the $\mathcal{HIBS}$ scheme with success probability $\mathbf{Adv}^{\text{hibs-uf-cma}}_{\mathcal{HIBS},\mathcal{A}}(k)$, we can construct an adversary $\mathcal{B}$ that attacks the aos-uf-cma security of $\mathcal{AOS}$ with success probability

$$\mathbf{Adv}^{\text{aos-uf-cma}}_{\mathcal{AOS},\mathcal{B}}(k) \geq \mathbf{Adv}^{\text{hibs-uf-cma}}_{\mathcal{HIBS},\mathcal{A}}(k).$$

As in Definition 2.1, adversary $\mathcal{B}$ gets input a public key $\mathsf{AOS.pk}$ for the $\mathcal{AOS}$ scheme. $\mathcal{B}$ runs the HIBS-UF-CMA experiment against $\mathcal{HIBS}$ as well as an instance of adversary $\mathcal{A}$. $\mathcal{B}$ gives as input to $\mathcal{A}$ the master public key $\mathsf{HIBS.pk} = \mathsf{AOS.pk}$. Adversary $\mathcal{B}$ answers the oracle queries of adversary $\mathcal{A}$ using the oracle $\text{AOSSign}(\cdot)$ for $\mathcal{AOS}$ as follows:

Corrupt($I[1..n]$):
  $\mathsf{HIBS.SK}[I[1..n]] \xleftarrow{\$} \text{AOSSign}(I[1..n])$
  return $\mathsf{HIBS.SK}[I[1..n]]$ to $\mathcal{A}$.

Sign($I[1..n], M$):
  $\mathtt{sig0} \xleftarrow{\$} \text{AOSSign}(I[1..n])$
  $\mathtt{sig1} \xleftarrow{\$} \mathsf{AOS.Append}(\mathsf{AOS.pk}, \mathtt{sig0}, \Delta)$
  $\mathtt{sig2} \xleftarrow{\$} \mathsf{AOS.Append}(\mathsf{AOS.pk}, \mathtt{sig1}, M)$
  return $\mathtt{sig2}$ to $\mathcal{A}$.

Eventually, adversary $\mathcal{A}$ halts and outputs the triple $(I^*[1..n], M^*, \texttt{sig}^*)$, which consists of a target hierarchical identity $I^*[1..n]$, a message M^*, and a forged signature $\texttt{sig}^*$. Adversary $\mathcal{B}$ then outputs $\texttt{sig}^*$ as the forgery of message $M^*[1..n+2] = (I_1^*, \ldots, I_n^*, \Delta, M^*)$. This completes the description of the simulation.

It is easy to see that $\mathcal{B}$ perfectly simulates the two oracles $\textsc{Corrupt}(\cdot)$ and $\textsc{Sign}(\cdot,\cdot)$; that is, $\mathcal{B}$'s responses on $\mathcal{A}$'s queries are distributed exactly as in the true HIBS-UF-CMA experiment.

Note that if $\texttt{sig}^*$ is a valid signature of M^* with respect to the hierarchical identity $I^*[1..n]$, then $\texttt{sig}^*$ is also a valid AOS signature on the message $M^*[1..n+2]$.

The fact that $\Delta \notin \mathsf{HIBS.IDSpace}$ ensures that the message $M^*[1..n+2]$ was never queried by $\mathcal{B}$ to the oracle $\text{AOS}\textsc{Sign}(\cdot)$ when simulating the oracle $\textsc{Corrupt}(\cdot)$. Furthermore, if $\mathcal{A}$'s forgery is valid, no prefix of the hierarchical identity $I^*[1..n]$ can be queried to the oracle $\textsc{Corrupt}(\cdot)$ by $\mathcal{A}$ and hence no prefix of $M^*[1..n+2]$ was queried to the oracle $\text{AOS}\textsc{Sign}(\cdot)$ by $\mathcal{B}$. Also, $\mathcal{A}$ is not allowed to call oracle $\textsc{Sign}(\cdot)$ for the tuple $(I^*[1..n], M^*)$. This ensures that no prefix of $M^*[1..n+2]$ was ever queried to oracle $\text{AOS}\textsc{Sign}(\cdot)$ when simulating oracle $\textsc{Sign}(\cdot,\cdot)$. Thus, whenever $\mathcal{A}$ outputs a valid forgery, $\mathcal{B}$ wins AOS-UF-CMA game against $\mathscr{AOS}$. ▌

The above reduction uses the existence of a symbol Δ such that $\Delta \in \mathsf{AOS.MSpace}$ but $\Delta \notin \mathsf{HIBS.IDSpace}$. For the case that $\mathsf{HIBS.IDSpace} = \mathsf{AOS.MSpace}$, there is an alternative way of constructing a HIBS from an AOS scheme. We sketch the construction for the interesting binary case—that is, for $\mathsf{HIBS.IDSpace} = \mathsf{AOS.MSpace} = \mathsf{HIBS.MSpace} = \{0, 1\}$.

The HIBS secret key of the hierarchical identity $I[1..n]$ is then defined to be the AOS signature of $(I_1, 0, I_2, 0, \ldots, I_n, 0)$. The HIBS signature of M under the hierarchical identity $I[1..n]$ is defined as the AOS signature of $(I_1, 0, \ldots, I_n, 0, M, 1)$. The security proof of the resulting HIBS scheme is a natural modification of the proof of Theorem 4.5.

4.4 Discussion

Given the equivalence between the AOS and HIBS schemes, one can easily transform all our constructions in Section 3 into provably secure HIBS schemes. Note that since the reductions are tight, an efficient AOS implies an efficient HIBS and vice-versa. The advantages of this indirect approach to designing HIBS are twofold. First, AOS is a much simpler primitive than HIBS; security proofs for AOS schemes are easier to carry out than those for HIBS. Second, some of the tricks used in efficient AOS schemes (for example, the hash-tree based construction) could yield more efficient HIBS constructions.

The certificate-based AOS scheme (Section 3.1) thus naturally transforms a public-key signature scheme into a HIBS scheme. The certificate-based approach to constructing HIBS schemes was mentioned in [GS02] although this fact appears not to be widely known [CHYC04] and we were not able to find any further studies of certificate-based HIBS in the literature. To the best of our knowledge, we are the first to prove security for this scheme. In contrast to identity-based encryption (which is believed to be hard to implement without use of bilinear maps) such HIBS schemes do not utilize pairings, thereby yielding efficient implementations. Moreover, the security proofs are done without using the Random Oracle model.

5 Applications

In this section, we describe two practical scenarios in which append-only signatures are directly applicable.

5.1 Wide-area Routing Protocol Security

An important application of AOS is in the construction of secure routing protocols for the Internet. The Border Gateway Protocol (BGP), which is the primary routing protocol used today in the Internet, has some well-known security weaknesses which require cryptographic solutions. While there have been many proposals for securing BGP in the past [KLS00, HPS04, SRS$^+$04, WKvO05], each must develop its own cryptographic constructions due to the lack of any primitive designed specifically for this application. In the discussion below, we briefly describe Internet routing and explain how our primitive is useful for ensuring one of the most important security requirements in BGP, namely path authenticity. Indeed, providing a sufficient cryptographic primitive for this problem led us to design AOS.

We begin with BGP, the Internet's primary routing protocol, which is tasked with advertising paths from one network to all other networks. Each network, named by an Autonomous System (AS) number, uses BGP to advertise the sets of IP addresses it is responsible for to its neighbor ASes. Each AS, upon receiving such advertisements, appends itself to the list of ASes on the forwarding path and repropagates the advertisement if it adheres to some local policies. When an AS receives two path advertisements for the same IP address space, it must make a decision as to which it wishes to use for its purposes and also to propagate to neighbors. Finally, once the set of routes have converged, these routes are used for packet forwarding; for each packet, the router looks up the destination IP address and forwards it to the neighbor AS as given by the BGP path advertisement.

Unfortunately, this path advertisement process also allows for any intermediate AS to hijack the process by changing advertisements arbitrarily. For example, if an AS truncates the AS path in an advertisement, then its neighbors will receive an advertisement shorter than the true path (typically causing them to prefer it). In the worst case, an AS can use this attack to convince its neighbors to forward all their traffic to it, which it could then modify or drop at will. (There are several other classes of attacks against BGP, but path modification and truncation are the most significant.)

Append-only Signatures can easily be applied to solve this problem as follows. Suppose that an AS R_0 wishes to announce routes for some IP prefix using the above path advertisement process. It first generates an AOS public-private key pair, distributes the public key $\mathsf{AOS.pk}$ throughout the network (this can be done with the help of a trusted authority that certifies public keys of ASes as in [KLS00, HPS04, WKvO05]) and to every neighboring AS R_{i_1}, sends the usual BGP information relating to the single-node path (R_0) along with the AOS signature $\mathsf{AOS.Append}(\mathsf{AOS.pk}, \mathsf{Sig}[\varepsilon], R_{i_1})$. In order to continue the advertisement process, R_{i_1} sends to each of its own neighbors R_{i_2} a BGP announcement containing the path (R_0, R_{i_1}) and the AOS signature $\mathsf{AOS.Append}(\mathsf{AOS.pk}, \mathtt{Sig}[R_{i_1}], R_{i_2})$. In other words, R_0 appends the label of its neighbor R_{i_1} into the AOS signature chain and R_{i_1} further appends the label of R_{i_2} into it[3]. The

[3] This may seem a bit unintuitive because each AS appends the "succeeding" AS's identity, rather than its own, into the AOS signature. However, this is important to ensure security of the protocol; otherwise, a malicious AS can make a path arbitrarily long by appending random ASes into the path before it finally appends itself. The only mischief it can perform in the above protocol is to create an arbitrary loop starting and ending at itself; this can be easily detected by any downstream AS.

advertisement process continues in this manner until all ASes in the network receive information about a path to R_0. Each recipient can verify the validity of the announced path by verifying the corresponding AOS signature using the public key $\mathsf{AOS.pk}$. If the AOS scheme is secure according to our definition (defn. 2.1), then all that a malicious AS can do is append the label of one of its neighbors into the AOS signature chain (since each neighbor R_i can check that the AS it receives an advertisement from was the last to be appended before R_i[4]).

In practice, the number of path advertisements received by any AS to a given source AS R_0 is extremely small: as observed in real routing data [HPS04], the odds that an AS receives more than 15 path advertisements coming from the same source are about 1 in a 1000. This allows the use of efficient m-time signature schemes (as in Section 3.5), with m equal to 15, in order to implement the AOS scheme in the above protocol and obtain a reasonable level of security.

5.2 Secure Delegation of Resources

Fu *et al.* describe the SHARP system for distributed resource management, in which users wish to share their resources with each other in a fully decentralized system [FCC+03]. Central to this system is the notion of a claim—users are issued claims on resources which they present upon resource use. These claims are signed such that they can be verified to be valid by third parties. Furthermore, users can delegate their claims to others, restricting them in the process. In this setting, the resource owner wishes to place some restrictions on how her resources are delegated. Their setting allows for a direct application of AOS. Quite simply, each resource provider, when initially issuing a resource claim, appends to an AOS the amount of resources to be given. Upon delegation, subsequent parties simply append to the signature what fraction of the existing claim is to be delegated. Upon claiming resources, the holder of the sub-delegated claim cannot use more than the fraction of the original resources indicated by the AOS signature.

6 Final Remarks and Open Problems

6.1 Finalization of AOS signature

An interesting feature of append-only signature schemes which might be needed by some applications is the ability to "finalize" the signature, that is, to modify the signature of a message in the way which prohibits any further appending. The general solution to this problem is to use a special symbol Θ (from the message space) to denote the end of the message. When one wants to finalize the signature of some message, he should append Θ to the signature. Messages that contain symbol Θ in the middle of the message (not as the last symbol) are therefore considered to be invalid.

6.2 Restricted AOS

In AOS, anyone can append and verify signatures. In certain scenarios, however, one may want to restrict the ability to append messages to a limited group of users. Still, anyone should be able to verify the signatures. We call this extension of AOS *Restricted Append-Only Signatures* (RAOS).

[4]The security of this solution relies on the assumption that AS-to-AS links are authenticated in some standard manner, for example using Message Authentication Codes (MACs) or existing infrastructure like IPSec, as done in [HPS04, WKvO05]. Also, the AOS-based approach is not resilient to collusions between multiple malicious ASes, as is the case with all proposals for securing BGP that we are aware of.

Let $\mathcal{U}$ be a group of users allowed to perform the append operation. We assume that all members of the group $\mathcal{U}$ are given some key K of an (symmetric) encryption scheme $\mathcal{ENC} = (\mathsf{ENC}, \mathsf{DEC})$.

We modify a given AOS scheme to get a RAOS scheme as follows: Define the RAOS signature of a message $M[1..n] = (M_1, \ldots, M_n)$ as the tuple

$$\mathtt{Sig}' = (\mathsf{ENC}_K(\mathtt{Sig}(M[1..n])), \mathtt{Sig}(M_1, \ldots, M_n, \Theta)),$$

where Θ is the finalization symbol from the last paragraph. In order to append the message M_{n+1} to a given RAOS signature on $M[1..n]$, a member of the group $\mathcal{U}$ decrypts the first part of the RAOS signature with her key K to obtain $\mathtt{Sig}(M[1..n])$. She then appends M_{n+1} using the original $\mathsf{AOS.Append}$ algorithm. Finally, she outputs the new RAOS signature tuple by encrypting $\mathtt{Sig}(M[1..n+1])$ and appending Θ to $\mathtt{Sig}(M[1..n+1])$. Note that without knowledge of the key K, the AOS signature $\mathtt{Sig}(M[1..n])$ remains secret and hence appending cannot be performed. Public verification is done by verifying if the second part of the RAOS signature is a valid AOS signature on the message $(M_1, \ldots, M_n, \Theta)$.

6.3 Shorter AOS signatures

Given that wide-area routing protocols propagate a large number of messages, compact signatures are desirable. Thus we raise an open problem of whether it is possible to build an AOS scheme with constant signature length (in both message length and maximal message length). This problem is equivalent to building a HIBS scheme where secret keys of the users have constant length (in the depth of the given user in the hierarchy and in the maximal depth of the hierarchy).

So far the best we can get is the construction from Section 3.3 which provides an AOS scheme that can sign messages of length up to n symbols with signatures of length $O(\sqrt{n})$. This construction translates to a HIBS scheme with n-level deep hierarchy, where the secret key of each user has length $O(\sqrt{n})$.

Acknowledgments

We would like to thank Mihir Bellare (for suggesting an improvement to the proof of Theorem 3.1) and Daniele Micciancio (for some useful insight about the definition of AOS). Thanks also to the anonymous reviewers for pointing out errors and suggesting improvements.

References

[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-Verlag, Berlin, 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-Verlag, Berlin, Germany.

[BBG05] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, Lecture Notes in Computer Science, page ???? Springer-Verlag, Berlin, Germany, May 22–26, 2005.

[BF03] Dan Boneh and Matthew K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.

[BGG94] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental cryptography - the case of hashing and signing. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, pages 57–66. Springer-Verlag, Berlin, Germany, August 21–25, 1994.

[BGG95] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental cryptography and application to virus protection. In *27th Annual ACM Symposium on Theory of Computing*, pages 57–66, Las Vegas, Nevada, USA, May 29 – June 1, 1995. ACM Press.

[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-Verlag, Berlin, Germany.

[BM96] Daniel Bleichenbacher and Ueli M. Maurer. Optimal tree-based one-time digital signature schemes. In *STACS '96: Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science*, pages 363–374. Springer-Verlag, 1996.

[BM99] Mihir Bellare and Sara 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-Verlag, Berlin, Germany.

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271, Warsaw, Poland, May 4–8, 2003. Springer-Verlag, Berlin, Germany.

[CHYC04] S. S. M. Chow, L. C. K. Hui, S. M. Yiu, and K. P. Chow. Secure hierarchical identity based signature and its application. In *Proceedings of ICICS 2004*, pages 480–494, 2004.

[FCC+03] Yun Fu, Jeffrey Chase, Brent Chun, Stephen Schwab, and Amin Vahdat. SHARP: an architecture for secure resource peering. In *Proceedings of the 19th ACM Symposium on Operating System Principles (SOSP)*, Bolton Landing, NY, October 2003.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.

[GS02] C. Gentry and A. 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-Verlag, Berlin, Germany.

[HM02] A. Hevia and D. Micciancio. The provable security of graph-based one-time signatures and extensions to algebraic signature schemes. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 379 – 396, Queenstown, New Zealand, December 1–5, 2002. Springer-Verlag, Berlin, Germany.

[HPS04] Yih-Chun Hu, Adrian Perrig, and Marvin Sirbu. SPV: secure path vector routing for securing BGP. In *Proceedings of the ACM SIGCOMM Conference*, pages 179–192, 2004.

[JMSW02] Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. Homomorphic signature schemes. In Bart Preneel, editor, *Topics in Cryptology – CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 244–262, San Jose, CA, USA, February 18–22, 2002. Springer-Verlag, Berlin, Germany.

[KLS00] Stephen Kent, Charles Lynn, and Karen Seo. Secure border gateway protocol (S-BGP). *IEEE Journal on Selected Areas in Communications*, 18(4):582–592, 2000.

[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-Verlag, Berlin, Germany.

[Mer88] Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378, Santa Barbara, CA, USA, August 16–20, 1988. Springer-Verlag, Berlin, Germany.

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

[MR02] Silvio Micali and Ronald L. Rivest. Transitive signature schemes. In Bart Preneel, editor, *Topics in Cryptology – CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 236–243, San Jose, CA, USA, February 18–22, 2002. Springer-Verlag, Berlin, Germany.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd Annual ACM Symposium on Theory of Computing*, pages 387–394, Baltimore, Maryland, USA, May 14–16, 1990. ACM Press.

[RR02] Leonid Reyzin and Natan Reyzin. Better than biba: Short one-time signatures with fast signing and verifying. In *Proceedings of 7th Australasian Conference ACSIP*, 2002.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53, Santa Barbara, CA, USA, August 19–23, 1985. Springer-Verlag, Berlin, Germany.

[SRS+04] Lakshminarayanan Subramanian, Volker Roth, Ion Stoica, Scott Shenker, and Randy Katz. Listen and whisper: Security mechanisms for bgp. In *Proceedings of USENIX/ACM NSDI*, 2004.

[WKvO05] Tao Wan, Evangelos Kranakis, and P.C. van Oorschot. Pretty secure BGP (psBGP). In *Proceedings of ISOC NDSS*, 2005.

A Public Key Signature Schemes

A public key signature scheme $\mathcal{SGN} = (\mathsf{SGN.G}, \mathsf{SGN.S}, \mathsf{SGN.V})$ is a collection of three algorithms: a key generation algorithm $\mathsf{SGN.G}$, a signing algorithm $\mathsf{SGN.S}$, and a verification algorithm $\mathsf{SGN.V}$. These algorithms must be polynomial time in the security parameter and should have the following input/output specification:

- $\mathsf{SGN.G}$ takes as input the security parameter 1^k and outputs a secret key/public key pair (sk, pk). The public key also includes some system parameters like the description of the message space $\mathsf{SGN.MSpace}$.
- $\mathsf{SGN.S}$ takes as input a message $M \in \mathsf{SGN.MSpace}$ and produces a string *sig* which is called a signature of a message M.
- $\mathsf{SGN.V}$ takes as input a public key pk, a message M and a signature *sig* and returns either `true` or `false`.

The verification algorithm must accept all signatures produced by the signing algorithm, namely the following should hold for every (sk, pk) produced by $\mathsf{SGN.G}(k)$, every message $M \in \mathsf{SGN.MSpace}$ and every choice of random coins:

$$\mathsf{SGN.V}_{pk}(M, \mathsf{SGN.S}_{sk}(M)) = \mathtt{true}.$$

Next we define unforgeability under chosen message attacks (sig-uf-cma) for a signature scheme.

Definition A.1 [SIG-UF-CMA] Let $\mathcal{SGN}$ be a signature scheme, let k be a security parameter, and let $\mathcal{A}$ be an adversary. We consider the following experiment:

Experiment $\mathbf{Exp}^{\text{sig-uf-cma}}_{\mathcal{SGN},\mathcal{A}}(1^k)$	**Oracle** $\textsc{Sign}(M)$
$\quad MSGSet \leftarrow \emptyset;$	$\quad sig \xleftarrow{\$} \mathsf{SGN.S}_{sk}(M)$
$\quad (sk, pk) \xleftarrow{\$} \mathsf{SGN.G}(1^k)$	$\quad MSGSet \leftarrow MSGSet \cup \{M\}$
$\quad (M, sig) \xleftarrow{\$} \mathcal{A}^{\textsc{Sign}(\cdot)}(pk)$	$\quad$ **return** *sig*
$\quad$ **if** $\mathsf{SGN.V}(M, sig) = \mathtt{true}$	
$\qquad$ **and** M not in $MSGSet$, **return** 1	
$\quad$ **else return** 0	

The sig-uf-cma advantage of an adversary $\mathcal{A}$ in breaking security of the scheme $\mathcal{SGN}$ is defined as

$$\mathbf{Adv}^{\text{sig-uf-cma}}_{\mathcal{SGN},\mathcal{A}}(k) = \Pr[\,\mathbf{Exp}^{\text{sig-uf-cma}}_{\mathcal{SGN},\mathcal{A}}(k) = 1\,].$$

Signature scheme $\mathcal{SGN}$ is said to be *unforgeable under chosen message attacks* (sig-uf-cma) if the above advantage is negligible function in k for all polynomial-time adversaries $\mathcal{A}$.

B Proof of Theorem 3.1

We show that for any adversary $\mathcal{A}$ against $\mathcal{AOS}1$, there exists an adversary $\mathcal{B}$ against $\mathcal{SGN}$ running in about the same time as $\mathcal{A}$ such that

$$\mathbf{Adv}^{\text{aos-uf-cma}}_{\mathcal{AOS}1,\mathcal{A}}(k) \leq s \cdot \mathbf{Adv}^{\text{sig-uf-cma}}_{\mathcal{SGN},\mathcal{B}}(k).$$

The reduction factor s is the upper bound on the number of messages M for which $\texttt{Sig}[M]$ is defined by the $\textsc{Extract}(\cdot)$ oracle in the experiment $\mathbf{Exp}^{\text{forge-cma}}_{\mathcal{AOS}1,\mathcal{A}}(k)$. This bound should be known by $\mathcal{B}$ before she runs the simulation of $\mathcal{A}$ and the bound should hold for any choice of the public key, for any random coins of $\mathcal{A}$ and for any oracle responses. The number s could be also upper bounded by $q_e \cdot d$, where q_e is the maximal number of $\text{AOS}\textsc{Sign}(\cdot)$ queries made by $\mathcal{A}$ and d is the maximal length of the queries.

Now we proceed to the construction of adversary $\mathcal{B}$ who attacks the unforgeability of the public key signature scheme $\mathcal{SGN}$. In the sig-uf-cma experiment the challenger runs a key generation algorithm $\mathsf{SGN.G}(1^k)$ to get a pair of keys (sk, pk) and gives pk as input to $\mathcal{B}$ as well as an access to the $\mathsf{SGN.S}_{sk}(\cdot)$ oracle.

During its execution $\mathcal{B}$ will construct a set T that at each moment of time will contain all the messages M for which $\texttt{Sig}[M]$ is already defined by $\textsc{Extract}(\cdot)$. This set plays the same role as in the original aos-uf-cma experiment. The only difference is that now, for each message $M[1..n] \in T$, we will keep not only a signature $\texttt{Sig}[M[1..n]]$ but also a pair of elements $(sk_{M[1..n]}, pk_{M[1..n]})$. The latter will correspond to the secret key/public key pair for $\mathcal{SGN}$ generated by $\textsc{Extract}(\cdot)$ oracle when computing the signature for $M[1..n]$. The pseudocode for the adversary $\mathcal{B}$ is given below:

Adversary $\mathcal{B}(1^k, pk)$:
  Pick $guess \xleftarrow{\$} [1, \ldots, s]$; $ctr \leftarrow 1$; $T \leftarrow \emptyset$
  if $guess = 1$ **then** $pk_\epsilon \leftarrow pk$; $sk_\epsilon \leftarrow \texttt{false}$
  else $(sk_\epsilon, pk_\epsilon) \xleftarrow{\$} \mathsf{SGN.G}(1^k)$
  Set $\mathsf{AOS.pk} \leftarrow pk_\epsilon$ and define $\texttt{Sig}[\epsilon] \leftarrow \{sk_\epsilon\}$
  Add ϵ to T
  Run $\mathcal{A}(1^k, \mathsf{AOS.pk})$ and answer its $\text{AOS}\textsc{Sign}(\cdot)$ queries as follows:

    $\text{AOS}\textsc{Sign}(M[1..n])$:
      $\texttt{Sig} \xleftarrow{\$} \textsc{Extract}(M[1..n])$; parse $\texttt{Sig}$ as $\{pk_{M[1]}, sig_1, \ldots, pk_{M[1..n]}, sig_n, sk_{M[1..n]}\}$
      if $sk_{M[1..n]} \neq \texttt{false}$ **then return** $\texttt{Sig}$ to $\mathcal{A}$ **else** halt $\mathcal{B}$; **return** $\texttt{Failure}$.

    $\textsc{Extract}(M[1..n])$: // defined recursively
      if $M[1..n] \in T$ **then return** $\texttt{Sig}[M[1..n]]$
      else $\texttt{Sig} \xleftarrow{\$} \textsc{Extract}(M[1..n-1])$; parse $\texttt{Sig}$ as $\{pk_{M[1]}, sig_1, \ldots, pk_{M[n-1]}, sig_{n-1}, sk_{M[n-1]}\}$
        $ctr \leftarrow ctr + 1$
        if $guess = ctr$ **then** $M^+[1..n^+] \leftarrow M[1..n]$; set $(sk_{M^+[1..n^+]}, pk_{M^+[1..n^+]}) \leftarrow (\texttt{false}, pk)$
          else $(sk_{M[1..n]}, pk_{M[1..n]}) \xleftarrow{\$} \mathsf{SGN.G}(1^k)$
        if $sk_{M[1..n-1]} \neq \texttt{false}$ **then** $sig_n \xleftarrow{\$} \mathsf{SGN.S}_{sk_{M[1..n-1]}}(m_n, pk_{M[1..n]})$
          else get sig_n by querying (m_n, pk) to the $\mathsf{SGN.S}_{sk}(\cdot)$ oracle.
        Define $\texttt{Sig}[M[1..n]] = \{pk_{M[1]}, sig_1, \ldots, pk_{M[1..n]}, sig_n, sk_{M[1..n]}\}$
        Add $M[1..n]$ to T

return $\mathtt{Sig}[M[1..n]]$.

Eventually $\mathcal{A}$ halts and outputs $(M^*[1..n], \mathtt{Sig}^*)$, where $\mathtt{Sig}^* = \{pk_1^*, sig_1^*, \ldots, pk_n^*, sig_n^*, sk_n^*\}$.
Find the maximal index $n^* \in [0 \ldots n]$ such that
$M^*[1..n^*] \in T$ and s.t. $pk_i^* = pk_{M^*[1..i]}$ for all $i = 1 \ldots n^*$.
if $M^*[1..n^*] \neq M^+[1..n^+]$ **then return** `Failure`
else if $n^* = n$ **then** pick a pair (m', pk') which was not queried to $\mathsf{SGN.S}_{sk}(\cdot)$;
compute $sig' \xleftarrow{\$} \mathsf{SGN.S}_{sk_n^*}(m', pk')$; output $\{(m', pk'), sig'\}$
if $n^* < n$ **then** output $\{(m_{n^*+1}^*, pk_{n^*+1}^*), sig_{n^*+1}^*\}$.

At a high level, the adversary $\mathcal{B}$ works as follows. Before running $\mathcal{A}$, it randomly selects an integer $guess \xleftarrow{\$} [1..s]$ which corresponds to an index of some message that will be queried to the Extract$(\cdot)$ oracle. Note that all messages are selected by $\mathcal{A}$, so $\mathcal{B}$ does not know in advance the actual message $M^+[1..n^+]$. (It is defined only at the time when $guess$-th new message is queried to Extract$(\cdot)$.) Next, $\mathcal{B}$ runs $\mathcal{A}$ and simulates the AOSSign$(\cdot)$ oracle. $\mathcal{B}$ follows the protocol of the original oracle to compute signatures of all the messages that do not contain the guessed one as a prefix. For the guessed message $M^+[1..n^+]$, the adversary $\mathcal{B}$ sets $pk_{M^+[1..n^+]}$ to be equal to the input public key pk and $sk_{M^+[1..n^+]} \leftarrow \mathtt{false}$. Therefore, if $\mathcal{A}$ queries the guessed message to the AOSSign oracle, $\mathcal{B}$ declares `Failure` since she does not know the secret key sk which corresponds to pk. However, $\mathcal{B}$ still can correctly answer all the AOSSign queries that contain $M^+[1..n^+]$ as a prefix by using the $\mathsf{SGN.S}_{sk}(\cdot)$ oracle that is given by the $\mathbf{Exp}^{\text{sig-uf-cma}}$ experiment. Also, since Extract$(\cdot)$ is recursive, any message is added to T only after all its prefixes are already in T.

Let $\mathcal{A}$ terminate and output a forgery of a message $M^*[1..n]$. $\mathcal{B}$ assumes that the forgery is valid and that her guessed message $M^+[1..n^+]$ is equal to the prefix $M^*[1..n^*]$ of the forged message. $M^*[1..n^*]$ is the longest prefix of $M^*[1..n]$ such that all the public keys $pk_1^*, \ldots pk_{n^+}^*$ from the signature $\mathtt{Sig}^*$ match the stored public keys $pk_{M^*[1]}, \ldots pk_{M^*[1..n^+]}$. In this case, $\mathcal{B}$ can easily make a forgery for $\mathcal{SGN}$: if $n^+ < n^*$ then $sig_{n^++1}^*$ is a forgery for $\mathcal{SGN}$, otherwise (if $n^+ = n^*$) the secret key sk_{n^*} should match the unknown secret key sk.

Next we bound the advantage of $\mathcal{B}$. There are several events in the experiment we must consider.

$$\mathsf{FAIL} : \mathcal{B} \text{ does output } \mathtt{Failure}.$$

Without loss of generality, we can assume that if $\mathcal{B}$ does not fail then $\mathcal{A}$ always outputs a forgery $\mathtt{Sig}^*$ for some message $M^*[1..n]$ and that no prefix of $M^*[1..n]$ was queried to AOSSign$(\cdot)$ (if this does not hold, $\mathcal{A}$ automatically loses). The second event is defined as

$$\mathsf{FORGE} : \text{the forgery of } \mathcal{A} \text{ is valid,}$$

that is, $\mathsf{AOS.Vfy}(\mathsf{AOS.pk}, \mathtt{Sig}^*, M^*[1..n]) = 1$. This event is defined only if $\mathcal{B}$ does not fail. Finally, we consider random variables $M^+[1..n^+]$ and $M^*[1..n^*]$. The former random variable, $M^+[1..n^+]$, corresponds to a message, whose signature was defined at the time when $\mathcal{B}$ sets $guess = ctr$. If no such query was made, we set $M^+[1..n^+] \leftarrow \perp$. The latter random variable, $M^*[1..n^*]$, corresponds to the "good" prefix of the forgery $M^*[1..n]$ returned by $\mathcal{A}$. It is defined only if $\mathcal{B}$ does not fail. We define GUESS to be the event that $\mathcal{B}$ guesses the message $\mathcal{A}$ outputs a forgery on:

$$\mathsf{GUESS} : M^+[1..n^+] = M^*[1..n^*].$$

We observe that if $\mathcal{B}$ does not fail, $\mathcal{A}$ wins and the guess is correct; $\mathcal{B}$ then outputs a valid forgery of the signature scheme $\mathcal{SGN}$. Therefore

$$\mathbf{Adv}^{\text{sig-uf-cma}}_{\mathcal{SGN},\mathcal{B}}(k) \geq \Pr[\neg\mathsf{FAIL} \ \wedge \ \mathsf{FORGE} \ \wedge \ \mathsf{GUESS}]. \tag{2}$$

The analysis is based on the following two claims. The first claim establishes that if $\mathcal{B}$ guessed the right value for $M^+[1..n^+]$ and $\mathcal{B}$ does not output `failure` then the simulation of $\mathcal{A}$ is perfect and we have

Claim B.1 $\mathbf{Adv}^{\text{aos-uf-cma}}_{\mathcal{AOS}1,\mathcal{A}}(k) = \Pr[\mathsf{FORGE} \mid \neg\mathsf{FAIL} \ \wedge \ \mathsf{GUESS}]$.

The second claim shows that the probability of a correct guess is exactly $1/s$:

Claim B.2 $\Pr[\neg\mathsf{FAIL} \ \wedge \ \mathsf{GUESS}] = \frac{1}{s}$.

We will settle the two claims later. Combining Claims B.1 and B.2 with Eqn. (2) we get

$$\begin{aligned}
\mathbf{Adv}^{\text{sig-uf-cma}}_{\mathcal{SGN},\mathcal{B}}(k) &\geq \Pr[\mathsf{FORGE} \ \wedge \ \neg\mathsf{FAIL} \ \wedge \ \mathsf{GUESS}] \\
&= \Pr[\mathsf{FORGE} \mid \neg\mathsf{FAIL} \ \wedge \ \mathsf{GUESS}] \cdot \Pr[\neg\mathsf{FAIL} \ \wedge \ \mathsf{GUESS}] \\
&= \frac{1}{s} \cdot \mathbf{Adv}^{\text{aos-uf-cma}}_{\mathcal{AOS}1,\mathcal{A}}(k),
\end{aligned}$$

which completes the proof of the theorem.

Proof of of Claim B.1: We will show that in an AOSSign$(\cdot)$ query $M[1..n]$ made by $\mathcal{A}$, $\mathcal{B}$ declares `Failure` if $M[1..n] = M^+[1..n^+]$ and otherwise returns a signature which is distributed identically to the original aos-uf-cma experiment against $\mathcal{AOS}1$. This implies

$$\Pr[\mathsf{FORGE} \mid \neg\mathsf{FAIL}] = \mathbf{Adv}^{\text{aos-uf-cma}}_{\mathcal{AOS}1,\mathcal{A}}(k). \tag{3}$$

Since the output of $\mathcal{A}$ is independent from the choice of *guess*, then the above equality holds also assuming $M^+[1..n^+] = M^*[1..n^*]$:

$$\Pr[\mathsf{FORGE} \mid \neg\mathsf{FAIL} \ \wedge \ \mathsf{GUESS}] = \mathbf{Adv}^{\text{aos-uf-cma}}_{\mathcal{AOS}1,\mathcal{A}}(k).$$

It is left to show (3): that if $\mathcal{B}$ does not return `Failure` then the input of $\mathcal{A}$ is identically distributed to the original aos-uf-cma experiment.

First, from the construction of $\mathcal{B}$ we see that the claim is true for messages that do not contain the guessed message $M^+[1..n^+]$ as a prefix. In this case, the Extract oracle uses the $\mathsf{AOS.Append}$ algorithm of $\mathcal{AOS}1$ to append the signatures and therefore all signatures are distributed identically to the ones constructed by the Extract oracle in the original aos-uf-cma experiment.

In the case when $\mathcal{A}$ queries $M[1..n] = M^+[1..n^+]$ to the AOSSign$(\cdot)$ oracle, the AOSSign$(\cdot)$ oracle calls Extract$(M^+[1..n^+])$ to get $\mathtt{Sig}[M^+[1..n^+]]$. The signature $\mathtt{Sig}[M^+[1..n^+]]$ returned by the Extract oracle to has a form $\{pk_1, sig_1, \ldots, pk_{n^+}, sig_{n^+}, sk_{n^+}\}$, where $sk_{n^+} = \mathtt{false}$ and $\mathcal{B}$ declares `Failure` on such a query.

We are left to show what happens when the AOSSign query $M[1..n]$ contains $M^+[1..n^+]$ as a prefix and $n^+ < n$. We know that signatures of all the prefixes of length smaller that n^+ are correctly distributed. The recursive Extract oracle constructs $\mathtt{Sig}[M[1..n^+]]$ and appends it using the $\mathsf{SGN.S}_{sk}(\cdot)$ oracle. The signature $\mathtt{Sig}[M[1..n^+]]$ has the form $\{pk_1, sig_1, \ldots, pk_{n^+}, sig_{n^+}, sk_{n^+}\}$, where $pk_{n^+} = pk$ and $sk_{n^+} = \texttt{false}$. The input public key pk was generated by $\mathsf{SGN.G}(1^k)$ and thus pk_{n^+} is correctly distributed. To append $\texttt{Sig}[M[1..n^+]]$ with M_{n^++1}, $\textsc{Extract}(\cdot)$ computes $(sk_{n^++1}, pk_{n^++1}) \leftarrow \mathsf{SGN.G}(1^k)$, queries $sig_{sk}(M_{n^++1}, pk_{n^++1})$ and sets $\texttt{Sig}[M[1..n^+ + 1]] \leftarrow \{pk_1, sig_1, \ldots, pk_{n^++1}, sig_{n^++1}, sk_{n^++1}\}$. This signature is correctly distributed and all the following signatures are constructed by appending this one using $\mathsf{AOS.Append}$. ▮

Proof of of Claim B.2: Let $\mathcal{A}$ terminate and output a forgery for $M^*[1..n]$. Without loss of generality we can assume that no prefix of $M^*[1..n]$ is contained in $MSGSet$. Thus the target message $M^*[1..n^*]$ must be different from all the AOSSign queries made by $\mathcal{A}$. The previous claim established that $\mathcal{A}$'s AOSSign queries are totally independent from the choice of $M^+[1..n^+]$.

The index of the guessed message was uniformly chosen from $[1..s]$. If there exists an AOSSign query equal to $M^+[1..n^+]$, then $\mathcal{B}$ declares $\texttt{Failure}$; otherwise $\mathcal{A}$ outputs message $M^*[1..n^*]$ which must be different from all the AOSSign queries. Therefore the probability that $\mathcal{B}$ does not fail and that $M^*[1..n^*] = M^+[1..n^+]$ is exactly $1/s$. ▮

C Proof of Theorem 3.2

In this appendix we will establish the following: for any adversary $\mathcal{A}$ against aos-uf-cma security of $\mathcal{AOS}3$, there exists an adversary $\mathcal{B}$ (against collision-resistance of G_0, G_1 and H), $\mathcal{C}$ (against one-wayness of G_0 and G_1) and $\mathcal{D}$ (against pseudo-randomness of G) all running in about the same time as $\mathcal{A}$ such that

$$\mathbf{Adv}^{\text{aos-uf-cma}}_{\mathcal{AOS}3,\mathcal{A}}(k) \;\leq \sum_{S\in\{H,G_0,G_1\}} \mathbf{Adv}^{\text{cr}}_{S,\mathcal{B}}(k) + 3{\cdot}2^d \left(\sum_{S\in\{G_0,G_1\}} \mathbf{Adv}^{\text{1-way}}_{S,\mathcal{C}}(k) \;+\; 2d\cdot \mathbf{Adv}^{\text{prg}}_{G,\mathcal{D}}(k) \right).$$

Here $\mathbf{Adv}^{\text{xxx}}_{S,\mathcal{B}}(k)$ denotes the advantage of a (polynomially bounded) adversary $\mathcal{B}$ attacking the xxx security of the primitive S, where xxx can be cr (collision resistance), 1-way (one-wayness), or prg (pseudorandomness). For a formal definition we refer the reader to the textbooks of Goldreich [Gol01, Gol04]).

We denote the original aos-uf-cma experiment played by $\mathcal{A}$ as Exp. Consider an arbitrary adversary $\mathcal{A}$ that is run in Exp. At the key-generation stage the challenger picks a random secret key $\text{key}(\varepsilon)$, computes the values of $\text{key}(\cdot)$ on all the nodes in the graph T, and gives a public key $\text{key}(\tilde{\varepsilon})$ to $\mathcal{A}$. The adversary outputs a forgery $(M^*[1..t], \texttt{Sig}^*)$ and with probability $\mathbf{Adv}^{\text{aos-uf-cma}}_{\mathcal{AOS}3,\mathcal{A}}(k)$, $\mathcal{A}$'s forgery is valid. Let $u^* = \langle M^*_1, \ldots, M^*_t\rangle$ and parse $\texttt{Sig}^*$ as the set $\{\text{key}^*(x) \mid x \in \{u^*\} \cup \text{Comp}(u^*)\}$. Let $\text{Desc}(\text{u}^*)$ be the set $\{u^*\} \cup \text{Comp}(u^*)$ including all their (lower) descendants in the graph T. Then the signature on u^* defines $\text{key}^*(\cdot)$ on the set $\text{Desc}(\text{u}^*)$. We define the event

$$\mathsf{FORGE} : \text{The forgery of } \mathcal{A} \text{ in experiment } \mathsf{Exp} \text{ is valid},$$

i.e. $\text{key}^*(\tilde{\varepsilon}) = \text{key}(\tilde{\varepsilon})$. We define KEYS as the event that $\text{key}^*(x) = \text{key}(x)$ on all $x \in \text{Desc}(\text{u}^*)$ in experiment Exp. Then we have

$$\begin{aligned} \mathbf{Adv}^{\text{aos-uf-cma}}_{\mathcal{AOS}3,\mathcal{A}}(k) &= \Pr[\,\mathsf{FORGE}\,] \\ &= \underbrace{\Pr[\,\mathsf{FORGE} \wedge \mathsf{KEYS}\,]}_{p_1} + \underbrace{\Pr[\,\mathsf{FORGE} \wedge \neg\mathsf{KEYS}\,]}_{p_2}. \end{aligned}$$

Intuitively, we are going to show that in the first case there is an efficient attack on the pseudorandomness of G or the one-wayness of G_0 or G_1 and in the second case one can break collision-resistance of H, G_0 or G_1.

We will bound p_1 and p_2 in turn. First we bound p_2. Note that $\neg E$ means that there exists some $x \in \mathrm{Desc}(\mathrm{u}^*)$ such that $\mathrm{key}^*(x) \neq \mathrm{key}(x)$. Hence $\mathsf{FORGE} \wedge \neg\mathsf{KEYS}$ means that among the possible descendants $x \in \mathrm{Desc}(\mathrm{u}^*)$ satisfying $\mathrm{key}^*(x) \neq \mathrm{key}(x)$, there must exist some x together with one of its children y, such that $\mathrm{key}^*(x) \neq \mathrm{key}(x)$ but $\mathrm{key}^*(y) = \mathrm{key}(y)$ (since we require $\mathrm{key}^*(\tilde{\varepsilon}) = \mathrm{key}(\tilde{\varepsilon})$).

Consider the following adversary $\mathcal{B}$ (attacking the collision resistance of G_0, G_1 or H) that runs the aos-uf-cma experiment for $\mathcal{A}$. $\mathcal{B}$ picks a random secret key $\mathrm{key}(\varepsilon)$, recursively computes $\mathrm{key}(x)$ for all nodes x in the graph T, and returns the public key $\mathrm{key}(\tilde{\varepsilon})$ to $\mathcal{A}$. The input of $\mathcal{A}$ is distributed identically to the experiment $\mathbf{Exp}^{\text{aos-uf-cma}}_{\mathcal{AOS}3,\mathcal{A}}(k)$. Thus with probability p_2, $\mathcal{A}$ outputs a valid forgery $\{\mathrm{key}^*(x) \mid x \in \{u^*\} \cup \mathrm{Comp}(u^*)\}$ for some node $u^* \in UT$ such that there exists a node $x \in T$ (x is some descendant x of $\{u^*\} \cup \mathrm{Comp}(u^*)$) and its child y such that $\mathrm{key}^*(x) \neq \mathrm{key}(x)$ but $\mathrm{key}^*(y) = \mathrm{key}(y)$. Such an x can be identified by $\mathcal{B}$ in the same time as verifying a signature. If $x \in UT$ then $\{\mathrm{key}^*(x), \mathrm{key}(x)\}$ is a collision either for G_0 or G_1 ($G_i(\mathrm{key}^*(x)) = G_i(\mathrm{key}(x))$ for some $i \in \{0, 1\}$). Otherwise let x' be another parent of y, then $H(\mathrm{key}^*(x), \mathrm{key}^*(x')) = H(\mathrm{key}(x), \mathrm{key}(x'))$ and $\{(\mathrm{key}^*(x), \mathrm{key}^*(x')), (\mathrm{key}(x), \mathrm{key}(x'))\}$ is a collision for $H(\cdot, \cdot)$. Therefore

$$p_2 \leq \mathbf{Adv}^{\mathrm{cr}}_{H,\mathcal{B}}(k) + \mathbf{Adv}^{\mathrm{cr}}_{G_0,\mathcal{B}}(k) + \mathbf{Adv}^{\mathrm{cr}}_{G_1,\mathcal{B}}(k).$$

We now proceed to bound p_1. First consider a single adversary $\mathcal{C}_0$ participating in the oneway experiments for both G_0 and G_1 simultaneously. The adversary $\mathcal{C}_0$ will run $\mathcal{A}$ and in the case $\mathcal{A}$ successfully forges $\mathcal{AOS}3$, she will win in either one of the experiments (that is, finding a preimage for G_0 or G_1 of a given input value C). Second, consider an adversary $\mathcal{C}_1$ participating in the same oneway experiment and that only slightly differs from the description of adversary $\mathcal{C}_0$. The adversary $\mathcal{C}_1$ will be used to bound probability p_1.

Finally, using a hybrid argument, we will connect the two adversaries by showing that the views of adversary $\mathcal{A}$ when run by $\mathcal{C}_0$ and $\mathcal{C}_1$ in the execution of the experiment are computational indistinguishable assuming G is a pseudorandom generator.

We describe the two adversaries $\mathcal{C}_0$ and $\mathcal{C}_1$ as follows:

Adversary $\mathcal{C}_i(1^k, c)$:
  $\mathrm{key}(\varepsilon) \xleftarrow{\$} \{0,1\}^k$ and define $\mathrm{key}(\cdot)$ recursively on all the descendants.
  Pick a random node $u^+ = \langle M_1^+, \ldots, M_n^+ \rangle \xleftarrow{\$} UT \setminus \{\varepsilon\} \cup \{d\text{-th level of } LT\}$.
  if $i = 0$ **then** redefine $\mathrm{key}(u^+) \leftarrow c$ **else** do nothing
  Run $\mathcal{A}(1^k, \mathrm{key}(\tilde{\varepsilon}))$ and answer all the $\textsc{Sign}(M[1..t])$ queries made by $\mathcal{A}$ as follows:
    if $(M[1..t] = M^+[1..t])$ **and** $(t < n)$ **then return** abort.
    else set $u = \langle M_1, \ldots, M_t \rangle$; **return** $\mathtt{Sig} = \{\mathrm{key}(x) \mid x \in \{u\} \cup \mathrm{Comp}(u)\}$.
  Eventually $\mathcal{A}$ terminates and outputs a forgery $\mathtt{Sig}^*$ for some message $M^*[1..t]$.
  Denote $\langle M_1^*, \ldots, M_t^* \rangle$ as u^* ; parse $\mathtt{Sig}^*$ as $\{\mathrm{key}^*(x) \mid x \in u^* \cup \mathrm{Comp}(u^*)\}$.
  if $M^*[1..t] = M^+[1..n-1]$ **then return** $\mathrm{key}^*(u^*)$
  else return abort.

For $i \in \{0, 1\}$ denote by Exp_i the simultaneous oneway experiment when run under adversary $\mathcal{C}_i$. We define the two events

$$\mathsf{ABORT}_{\mathcal{C}_i} : \text{Adversary } \mathcal{C}_i \text{ aborts in experiment } \mathsf{Exp}_i$$
$$\mathsf{KEYS}_{\mathcal{C}_i} : \mathrm{key}^*(x) = \mathrm{key}(x) \text{ for all } x \in \mathrm{Desc}(\mathrm{u}^*) \text{ in experiment } \mathsf{Exp}_i$$

Furthermore, we define the two probabilities

$$q_i = \Pr[\,\mathsf{KEYS}_{\mathcal{C}_i} \wedge \neg\mathsf{ABORT}_{\mathcal{C}_i}\,]$$

in experiment Exp_i. Intuitively, in the remainder of the proof we will show that the value q_0 is related to the onewayness of G_0 and G_1, that q_1 can be lower bounded using p_1, and that $|q_0 - q_1|$ is small.

Consider the experiment Exp_0. Assuming $\mathcal{C}_0$ does not abort, u^+ is a child of u^* and hence we have $G_{M_n^+}(\mathrm{key}^*(u^*)) = \mathrm{key}(u^+) = c$ if the event $\mathsf{KEYS}_{\mathcal{C}_0}$ happens. Hence adversary $\mathcal{C}_0$ outputs $\mathrm{key}^*(u^*)$ which is valid preimage of $G_{M_n^+}(c)$:

$$q_0 \leq \mathbf{Adv}^{\text{1-way}}_{G_0,\mathcal{C}_0}(k) \;+\; \mathbf{Adv}^{\text{1-way}}_{G_1,\mathcal{C}_0}(k)$$

in experiment Exp_0. We will now lower bound probability q_1 in experiment Exp_1.

Claim C.1 $q_1 \geq p_1/(3 \cdot 2^d)$.

We rewrite q_1 as

$$q_1 = \Pr[\,\mathsf{KEYS}_{\mathcal{C}_1} \mid \neg\mathsf{ABORT}_{\mathcal{C}_1}\,] \cdot \Pr[\neg\mathsf{ABORT}_{\mathcal{C}_1}\,].$$

The event $\neg\mathsf{ABORT}_{\mathcal{C}_1}$ in Exp_1 happens if and only if the right value for u^+ was guessed by $\mathcal{C}_1$, i.e. if u^+ is a child of u^*. Hence, assuming $\mathcal{C}_1$ does not abort, the view of $\mathcal{A}$ in experiment Exp_1 is identically distributed to the view of $\mathcal{A}$ the experiment Exp. Hence with probability at least p_1, $\mathcal{A}$ outputs a valid forgery $(u^*, \mathtt{Sig}^*)$ such that $\mathrm{key}^*(x) = \mathrm{key}(x)$ on all descendants $x \in \mathrm{Desc(u^*)}$ in Exp_1:

$$\Pr[\,\mathsf{KEYS}_{\mathcal{C}_1} \mid \neg\mathsf{ABORT}_{\mathcal{C}_1}\,] \geq \Pr[\,\mathsf{FORGE} \wedge \mathsf{KEYS}\,] = \; p_1$$

Since the choice of u^+ in the experiment Exp_1 is totally independent from the inputs of $\mathcal{A}$ (and u^+ is chosen randomly from the set of $3 \cdot 2^d - 2$ nodes), we have

$$\Pr[\neg\mathsf{ABORT}_{\mathcal{C}_1}\,] \;\;=\;\; 1/(3 \cdot 2^d - 2) \geq 1/(3 \cdot 2^d).$$

This completes the proof of the claim.

For $i \in \{0, 1\}$ define View_i to be the view of adversary $\mathcal{A}$ in experiment Exp_i. The following lemma establishes that the views of $\mathcal{A}$ in the experiments Exp_0 and Exp_1 are computationally indistinguishable.

Lemma C.2 $\mathcal{A}$ cannot distinguish between the two distributions View_0 and View_1 better than with probability $2d \cdot \mathbf{Adv}^{\mathrm{prg}}_{G,\mathcal{D}}(k)$ for some adversary $\mathcal{D}$ running in about the same time as $\mathcal{A}$.

We will prove the lemma later. As an implication of Lemma C.2 we get that $|q_0 - q_1| \leq 2d \cdot \mathbf{Adv}^{\mathrm{prg}}_{G,\mathcal{D}}(k)$. Thus

$$\begin{aligned}
p_1 &\leq 3 \cdot 2^d \cdot q_1 \\
&\leq 3 \cdot 2^d (q_0 + 2d \cdot \mathbf{Adv}^{\mathrm{prg}}_{G,\mathcal{D}}(k)) \\
&\leq 3 \cdot 2^d (\mathbf{Adv}^{\text{1-way}}_{G_0,\mathcal{C}}(k) + \mathbf{Adv}^{\text{1-way}}_{G_1,\mathcal{C}}(k)) + 2d \cdot \mathbf{Adv}^{\mathrm{prg}}_{G,\mathcal{D}}(k))
\end{aligned}$$

Combining the bounds on p_1 and p_2 we complete the proof of the theorem:

$$p_1 + p_2 \;\;\leq \sum_{S \in \{H, G_0, G_1\}} \mathbf{Adv}^{\mathrm{cr}}_{S,\mathcal{B}}(k) \;+\; 3 \cdot 2^d \left(\sum_{S \in \{G_0, G_1\}} \mathbf{Adv}^{\text{1-way}}_{S,\mathcal{C}}(k) \;+\; 2d \cdot \mathbf{Adv}^{\mathrm{prg}}_{G,\mathcal{D}}(k) \right).$$

Proof of of Lemma C.2: Each of the experiments Exp_0 and Exp_1 is completely determined by the values of key$(\cdot)$ on the following subset of nodes $R \subset T$:

$$R = \{\langle M_1^+, \ldots, M_j^+\rangle \mid j = 1, \ldots, n\} \cup \{\langle M_1^+, \ldots, M_{j-1}^+, \neg M_j^+\rangle \mid j = 1, \ldots, n\}$$

In Exp_1, key(ε) is chosen at random and the keys on the rest of R is generated by iterating $G(\cdot)$. On the other hand, in Exp_0 the values of key$(\cdot)$ on R are defined in the same way except for key$(\langle M_1^+, \ldots, M_n^+\rangle)$, which is replaced by the challenge string for the onewayness experiment. Our goal is to show that from adversary $\mathcal{A}$'s view, these two experiments are computational indistinguishable.

We define a sequence of n hybrid adversaries $\mathcal{C}_1[1], \ldots, \mathcal{C}_1[n]$ as follows: Let $\mathcal{C}_1[1] = \mathcal{C}_1$ and the i-th adversary $\mathcal{C}_1[i]$ behaves like $\mathcal{C}_1$ except that it assigns random keys key$(\cdot)$ on the nodes $\{\langle M_1^+, \ldots, M_j^+\rangle, \langle M_1^+, \ldots, M_{j-1}^+, \neg M_j^+\rangle \mid j = 1, \ldots, i\}$ and uses $G(\cdot)$ to compute key$(\cdot)$ on the rest of the nodes. The only difference between adversaries $\mathcal{C}_1[i]$ and $\mathcal{C}_1[i+1]$ is that in the first case key$(\langle M_1^+, \ldots, M_i^+\rangle)$, key$(\langle M_1^+, \ldots, \neg M_i^+\rangle)$ are pseudorandom and in the second case they are perfectly random. Let $\mathsf{Exp}_1[i]$ be the experiment when run under adversary $\mathcal{C}_1[i]$ and let $\mathsf{View}_1[i]$ be the random variable describing the view of adversary $\mathcal{A}$ when run in experiment $\mathsf{Exp}_1[i]$. Then, for each $1 \le i \le n-1$, $\mathcal{A}$ cannot distinguish between $\mathsf{View}_1[i]$ and $\mathsf{View}_1[i+1]$ any better than with probability $\mathbf{Adv}^{\mathrm{prg}}_{G,\mathcal{D}}(k)$ for some adversary $\mathcal{D}$ who runs in the time needed to run $\mathcal{A}$.

Next, consider a new adversary $\mathcal{C}_0'$ which is defined as $\mathcal{C}_0$, just key$(\langle M_1^+, \ldots, M_n^+\rangle)$ is replaced by a random string from $\{0,1\}^k$. Analogously define Exp_0' and View_0'. Since $G(\cdot)$ is a pseudorandom generator, $\mathcal{A}$ cannot distinguish between the View_0 and View_0' any better than with probability $\mathbf{Adv}^{\mathrm{prg}}_{G,\mathcal{D}}(k)$.

Analogous to the first sequence of hybrid adversaries, consider a sequence of adversaries $\mathcal{C}_0'[1], \ldots, \mathcal{C}_0'[n]$. Here $\mathcal{C}_0'[1] = \mathcal{C}_0'$ and the i-th adversary $\mathcal{C}_0'[i]$ is identical to $\mathcal{C}_1[i]$, except the value key$(\langle M_1^+, \ldots, M_n^+\rangle)$ is replaced by a truly random string. Again define $\mathsf{Exp}_0'[i]$ and $\mathsf{View}_0'[i]$ as above. A similar argument shows that $\mathcal{A}$ cannot distinguish between $\mathsf{View}_0'[i]$ and $\mathsf{View}_0'[i+1]$ any better than with probability $\mathbf{Adv}^{\mathrm{prg}}_{G,\mathcal{D}}(k)$. Finally, the experiments $\mathsf{Exp}_0'[n]$ and $\mathsf{Exp}_1[n]$ are identical so are $\mathsf{View}_0'[n]$ and $\mathsf{View}_1[n]$. This establishes a sequence of hybrids from View_0 to View_1:

$$\mathsf{View}_0 \approx \mathsf{View}_0' \equiv \mathsf{View}_0'[1] \approx \ldots \approx \mathsf{View}_0'[n] \equiv \mathsf{View}_1[n] \approx \ldots \approx \mathsf{View}_1[1] \equiv \mathsf{View}_1,$$

where $\approx$ means that the distributions of the two random variables are *computational indistinguishable* and $\equiv$ means they are identical. $\mathcal{A}$ cannot distinguish between any consecutive pair of the views any better than with probability $\mathbf{Adv}^{\mathrm{prg}}_{G,\mathcal{D}}(k)$, therefore total probability that $\mathcal{A}$ will distinguish between View_0 and View_1 is no more than

$$2n \cdot \mathbf{Adv}^{\mathrm{prg}}_{G,\mathcal{D}}(k) \le 2d \cdot \mathbf{Adv}^{\mathrm{prg}}_{G,\mathcal{D}}(k).$$

∎

D AOS with short key size

In this section we review the hybrid HIBE scheme from Boneh, Boyen and Goh (see Section 4 of [BBG05]) ($\mathcal{BBG} - \mathcal{HIBE}$) achieving short private keys. By the results from Section 4 we know that a HIBE scheme implies an AOS scheme. Motivated by this reduction, we present a concrete AOS scheme (called $\mathcal{AOS2}$) based on $\mathcal{BBG} - \mathcal{HIBE}$. In presenting the concrete scheme, we are able to make some (straightforward) efficiency improvements over the generic reduction by making the AOS verification algorithm deterministic (instead of probabilistic).

The main intention of this section is to demonstrate that AOS (and hence also HIBS) can be carried out with secret key size of order "square root of the length of the signed message".

We briefly review the necessary facts about bilinear maps and bilinear groups. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be groups with the following properties.

- $\mathbb{G}_1$ is an additive groups of prime order q.
- $\mathbb{G}_1$ has generator P.
- There is a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.

We have stipulated that our groups should have a *bilinear map* $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. This should satisfy the conditions below.

Bilinear: For all $U, V \in \mathbb{G}_1, a, b \in \mathbf{Z}$, $e(aU, bV) = e(U, V)^{ab}$

Non-degenerate: $e(P, P) \neq 1_{\mathbb{G}_2}$

We say that $\mathbb{G}_1$ is a *bilinear group* if there exists a group $\mathbb{G}_2$ with $|\mathbb{G}_2| = |\mathbb{G}_1| = q$ and a bilinear map e satisfying the conditions above; moreover, the group operations in $\mathbb{G}_1$ and $\mathbb{G}_2$ and e must be efficiently computable.

Let $\mathbb{G}_1$ be a bilinear group. We want to build an AOS scheme for messages of length at most d. For simplicity assume that d has an integer root; that is, $l = \sqrt{d}$. Any message $M[1..n] = (M_1, \ldots, M_n)$ of length $n \leq d$ can be represented in matrix form as

$$M[1..n] = \begin{pmatrix} M_1 & M_2 & \ldots & M_l \\ M_{l+1} & M_{l+2} & \ldots & M_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ M_{(l-1)l+1} & M_{(l-1)l+2} & \ldots & M_{l^2} \end{pmatrix} = \begin{pmatrix} M_{1,1} & M_{1,2} & \ldots & M_{1,l} \\ M_{2,1} & M_{2,2} & \ldots & M_{2,l} \\ \vdots & \vdots & \ddots & \vdots \\ M_{l,1} & M_{l,2} & \ldots & M_{l,l} \end{pmatrix} .$$

We denote the induced index mapping as $\pi : \{1, \ldots, d\} \to \{1, \ldots, l\} \times \{1, \ldots, l\}$ with $\pi(n) = (n_1, n_2)$. Here n_1 is called the *row index* and n_2 the *column index*.

We may now describe $\mathcal{AOS2} = (\mathsf{AOS.Setup}, \mathsf{AOS.Append}, \mathsf{AOS.Vfy})$ giving its constituting algorithms.

- $\mathsf{AOS.Setup}(1^k)$: Pick two random generators $P, Q \in \mathbb{G}_1$ and a random element $\alpha \in \mathsf{Z}_q$. Next, pick random elements $G_1, \ldots, G_l$ and $H_1, \ldots, H_l \in \mathbb{G}_1$. The public key consists of

  $$\mathsf{AOS.pk} = (P, G_1, \ldots, G_l, H_1, \ldots, H_l, e(\alpha P, Q)).$$

  The root key $\mathsf{Sig}[\varepsilon]$ is αQ.

- $\mathsf{AOS.Append}$: Let $(n_1, n_2) = \pi(n)$. For a message $M[1..n]$ the signature is defined as

$$\begin{aligned} \mathsf{Sig}[M[1..n]] \quad = \quad & (\alpha Q + \sum_{i=1}^{n_1-1} r_i (G_i + \sum_{j=1}^{l} M_{i,j} H_j) + r'_{n_1}(G_{n_1} + \sum_{j=1}^{n_2} M_{n_1,j} H_j), \\ & r_1 P, \ldots, r_{n_1-1} P, r'_{n_1} P, r'_{n_1} H_{n_2+1}, \ldots, r'_{n_1} H_l) \in \mathbb{G}_1^{l+1+n_1-n_2}, \end{aligned}$$

where $r_1, \ldots, r_{n_1-1}$, as well as r'_{n_1} are random elements from Z_q. We now describe the append operation. Given the public key $\mathsf{AOS.pk}$, a signature $\mathsf{Sig}[M[1..n-1]]$, and a new symbol M_n, we want to output a valid signature of the message $M[1..n]$.

For the append process we have to distinguish between two cases:
Case 1: $\pi(n-1) = (n_1, n_2 - 1)$ (namely n and $n-1$ have the same row index). Then the signature on message $M[1..n-1]$ has the form

$$\Big(\alpha Q + \sum_{i=1}^{n_1-1} r_i(G_i + \sum_{j=1}^{l} M_{i,j}H_j) + r_{n_1}(G_{n_1} + \sum_{j=1}^{n_2-1} M_{n_1,j}H_j),$$
$$r_1P, \ldots, r_{n_1}P, r_{n_1}H_{n_2}, \ldots, r_{n_1}H_l\Big) = (A, A_1, \ldots, A_{n_1}, B_{n_2}, \ldots, B_l).$$

Pick a random $r \in \mathsf{Z}_q$ and output the signature $\mathsf{Sig}[M[1..n]]$ as

$$\Big(A+M_{n_1,n_2}B_{n_2}+r(G_{n_1}+\sum_{j=1}^{n_2} M_{n_1,j}H_j), A_1, \ldots, A_{n_1-1}, A_{n_1}+rP, B_{n_2+1}+rH_{n_2+1}, \ldots, B_l+rH_l\Big).$$

It is easy to verify that the resulting signature is of the right form with $r'_{n_1} = r_{n_1} + r$.

Case 2: If $\pi(n-1) = (n_1-1, l)$ (the row indexes of n and $n-1$ differ). Then the signature on message $M[1..n-1]$ has the form

$$\Big(\alpha Q + \sum_{i=1}^{n_1} r_i(G_i + \sum_{j=1}^{l} M_{i,j}H_j), r_1P, \ldots, r_{n_1-1}P\Big) = (A, A_1, \ldots, A_{n_1-1}).$$

In this case we have $\pi(n) = (n_1, 1)$ and to generate a signature for the message $M[1..n]$ chose a random $r \in \mathsf{Z}_q$ and compute $\mathsf{Sig}[M[1..n]]$ as

$$\Big(A + r(M_{n_1,1}H_1 + G_{n_1}), A_1, \ldots, A_{n_1-1}, rP, rH_2, \ldots, rH_l\Big).$$

- $\mathsf{AOS.Vfy}$. Given a signature $\mathsf{Sig}[M[1..n]]$ and a message $M[1..n]$, verification is carried out as follows: Verify if

$$e(\mathsf{Sig}[M[1..n]], P) = e(\alpha P, Q) \cdot \prod_{i=1}^{n_1-1} (e(r_iP, G_i + \sum_{j=1}^{l} M_{i,j}H_j)) \cdot e(r'_{n_1}P, G_{n_1} + \sum_{j=1}^{n_2} M_{n_1,j}H_j)$$

The length of the signature of $\mathcal{AOS}2$ grows linearly with the square root of the length of the message. Security is proved with respect to the $l+1$-*Bilinear Diffie-Hellman Exponent (BDHE) assumption*. (See [BBG05] for an exact definition.)

Theorem D.1 Suppose the computational $l+1$-BHDE assumptions holds in the group $\mathbb{G}_1$. Then the $\mathcal{AOS}2$ scheme is selective message aos-uf-cma secure.

Here *selective message aos-uf-cma* security is defined similar to Definition 2.1. The difference is that in the security experiment, the adversary has to commit to the message she is going to forge the signature for *before* the public/secret keys are issued and the public key is given to her. In the random oracle model, the scheme can be modified to get a full unforgeable AOS scheme. However, the security reduction is not tight and only allows for signatures on messages of constant length. Again, we refer the reader to [BBG05] for more details.

The proof of the theorem follows that of the $\mathcal{BBG}-\mathcal{HIBE}$ scheme from [BBG05] and is omitted here. We note that since we are dealing with signatures instead of encryption, we can prove security of our scheme with respect to a computational assumption (rather than a decisional assumption as the original $\mathcal{BBG}-\mathcal{HIBE}$ scheme).