A preliminary version of this paper appears in *Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS 2007*, pp. 276–285, S. De Capitani di Vimercati and P. Syverson eds., ACM Press, 2007. This is the full version.

Ordered Multisignatures and Identity-Based Sequential Aggregate Signatures, with Applications to Secure Routing

Alexandra Boldyreva* Craig Gentry† Adam O'Neill‡

Dae Hyun Yum§

Abstract

We construct two new multiparty digital signature schemes that allow multiple signers to sequentially produce a compact, fixed-length signature. First, we introduce a new primitive that we call *ordered multisignatures* (OMS), which allows signers to attest to a common message as well as the order in which they signed. Our OMS construction substantially improves computational efficiency and scalability over any existing scheme with suitable functionality. Second, we design a new identity-based sequential aggregate signature scheme, where signers can attest to different messages and signature verification does not require knowledge of traditional public keys. The latter property permits savings on bandwidth and storage as compared to public-key solutions. In contrast to the only prior scheme to provide this functionality, ours offers improved security that does not rely on synchronized clocks or a trusted first signer. We provide formal security definitions and support the proposed schemes with security proofs under appropriate computational assumptions. We focus on potential applications of our schemes to secure network routing, but we believe they will find many other applications as well.

Keywords: Digital signatures, identity-based signatures, multisignatures, aggregate signatures, pairings, network security.

*School of Computer Science, College of Computing, Georgia Institute of Technology, 266 Ferst Drive, Atlanta, GA 30332, USA. E-mail: `aboldyre@cc.gatech.edu`. URL: `http://www.cc.gatech.edu/∼ aboldyre`. Supported in part by NSF CAREER award 0545659.

†Dept. of Computer Science, Stanford University, USA. E-Mail: `cgentry@cs.stanford.edu`.

‡School of Computer Science, College of Computing, Georgia Institute of Technology, 266 Ferst Drive, Atlanta, GA 30332, USA. E-mail: `amoneill@cc.gatech.edu`. URL: `http://www.cc.gatech.edu/∼ amoneill`. Supported in part by the grant of the first author.

§Pohang University of Science and Technology, Korea. E-Mail: `dhyum@postech.ac.kr`.

Contents

1 Introduction

1.1 Overview

The current Internet design largely lacks the principles of AAA: Authentication, Authorization, and Accountability. It is understood that incorporation of these principles would make tackling security and reliability problems more tractable.

A large body of recent research focuses on identifying weak points in the current design and proposing fixes to the deployed infrastructure. For example, the Secure Border Gateway Protocol (S-BGP) initiative (and its variants) [27, 1, 29, 49, 47, 15, 18, 16], whose primary goal is to patch authenticity of route announcements in BGP, a path-vector protocol used in Internet routing, is currently under consideration for standardization by the IETF. While it is accepted that new security measures are necessary, many remain skeptical about the prospects of widespread adoption and deployment in the near future. The main technical reason is that secure networking adds additional overhead to in-use protocols. We view our role as cryptographers in this regard as designing suitable provably-secure mechanisms to address some of the identified weaknesses, which maintain as best as possible the design goals of the original protocols, especially in terms of router processing time and storage, bandwidth overhead, scalability, etc.

In line with this view, this work introduces two new "multiparty" digital signature schemes for efficiently enhancing authenticity in several network routing applications. Our schemes offer important performance and security improvements as compared to previous candidate solutions. We show that they provably provide security according to the corresponding security definitions (that are of independent interest) and under appropriate computational assumptions.

We clarify that we do *not* attempt to rigorously analyze all possible threats and assumptions about adversarial abilities in the network routing applications we discuss. Indeed, much work remains to be done in this regard, and the specific security requirements in these applications remains an issue of contention [40]. What we suggest is that our schemes appear to be useful towards future resolution of some of the security concerns that have been raised. We next discuss our schemes and their applications in more detail.

1.2 Ordered Multisignatures

Definition and motivation. We introduce a new primitive that we call *ordered multisignatures* (OMS). A multisignature scheme [6, 32] is a public-key primitive that allows multiple signers who want to sign some message to produce a single compact (constant-size) signature convincing a verifier that each signer signed the message. However, some network routing applications that we discuss below require verifying the *path* (i.e. the ordered list of routers) in which a packet travels to reach its destination, where routers have incentive to lie. Although by using multisignatures the routers could each sign (a fixed part of) the packet while keeping total packet overhead due to signatures fixed to a constant, this would be insufficient from a security standpoint because it does not allow to verify the order in which they signed.

Ensuring signing order in multisignatures has been previously addressed, but the constructions all require multiple rounds of interaction among signers (sometimes even in key generation) in order to produce the single constant-size signature, which is not suitable for the routing-based applications we consider. (We discuss these works in more detail later.) We point out that one way to ensure signing order in a (non-interactive) multisignature scheme would be to have each signer use a separate public-private key-pair for signing messages in each position on a path. But the resulting scheme would be impractical due to large combined key size. On the other hand, aggregate signature schemes [10, 34, 32, 3], which allow multiple signers to sign *different* messages while keeping total signature size constant, can of course immediately provide the needed functionality if the signers sign, in addition to the packet, their position on the path, but they are also computationally much less efficient than multisignatures. As an alternative, the OMS primitive we introduce produces a compact (constant-size) multisignature, uses constant-size keys, is "sequential" in that signers sign one after another and no further interaction among the signers is required, and ensures authenticity of both the signing order and that of the message.

Further contributions. After introducing and defining the new primitive, we propose a formal security model for OMS. It adapts the notion of security for multisignatures first presented in [6] to also ensure authenticity of the signing order. Intuitively, a secure OMS scheme, in addition to being secure as a "plain" (un-ordered) multisignature scheme, must enforce an additional unforgeability with respect to the ordering of the signers. We then provide a construction that we prove secure in our model, under a standard computational assumption on the groups equipped with the "bilinear maps" (aka. pairings) we use, in the random oracle (RO) model of [5]. As compared to known aggregate schemes, our construction offers substantial computational savings. Namely, the work per required on both signing and verification is essentially *constant* in the number of signatures currently in the OMS. Section 3.2 gives detailed efficiency comparisons.

Applications. We sketch some potential applications of our OMS scheme in more detail. One problem that appears suitable, raised in [23], is "data plane" security in S-BGP. This means allowing autonomous systems (ASes, i.e. networks under control of a single entity such as Georgia Tech or AT&T) to verify that data packets they send and receive/forward actually travel via previously-authenticated AS paths. (Authentication of AS paths is handled in the "control plane" by route attestation, explained in the following subsection.) To do so, a data packet should be signed, in order, by egress routers of ASes that forward it, allowing ingress routers to accept and forward only packets that followed an authenticated path, and the originating AS to later verify that the packet actually took an authenticated path to reach its destination (an OMS attesting to which could be piggybacked onto traffic on the reverse path).

Another setting where OMS could help arises in the recent in-band network troubleshooting system Orchid [39, 38]. In order to quickly and accurately diagnose faults (e.g. packet drops, re-orderings, duplications) along a flow from a sender to a receiver, Orchid has routers along the flow "mark" a fixed-size header in the data packets being sent. The first packet triggers a probe, which is sent to find out which routers are on the path. Later, a certain pattern of marks in the data packets by a router can implicate packet re-ordering or duplication by the *previous* router on the path, according to the data collected by the probe. When deployed across multiple networks (i.e. ASes), a router may wish to "frame" a router in another network by making it appear that the latter is directly upstream from it. Thus the probe should be signed, in order, by all the routers on the path, before fault data collected by the receiver can be considered authentic.

We suggest that the computational savings our scheme provides over existing solutions is desirable in the above-mentioned applications because it (1) distributes processing time more equitably amongst routers, (2) offers a sizable gain in total processing time, and (3) scales much better in the number of routers or ASes in the network.

Related work. As we mentioned, verifiability of signing order in multisignatures has been considered before, specifically in [21, 22, 13, 37, 46, 14], where they are usually called "structured" or "order-specified." However, this line of work is in the *interactive* setting, meaning the schemes they consider require multiple rounds of communication between co-signers (sometimes even in key generation, requiring a separate interactive key-generation protocol for each subgroup of signers), which is impractical in applications we consider. Other differences between these works and ours include that they mainly treat more complicated structures on the group of signers than just linear ordering, and they do not give concrete applications of their schemes. In the interactive setting, the literature on "plain" multisignatures is extensive; see [6] for a comprehensive account.

One-way signature chains [41] are designed for a different setting than ours, in which a signer attests to which specific signers came before her (cf. Remark 3.4), and moreover their construction does not provide any efficiency gain over existing aggregate signature schemes.

1.3 Identity-based Sequential Aggregate Signatures

Motivation and previous work. It has been pointed out in numerous works and further explored and tested in detail by [49] that aggregate signatures [10, 34, 32, 3], which allow multiple signers to sign different messages while keeping total signature size constant, can be used to address route announcement authenticity in S-BGP while significantly reducing associated bandwidth overhead and memory space for signatures. According to the proposal, each AS forwarding an update message should add its signature on the label of the *next* AS on the advertised route, so that route authenticity can be verified upon receipt of the aggregate. (This is a simplification that omits some details and optimizations such as "signature amortization;" see [49].) This technique is to prevent an unauthorized AS from extending the path and means that that signers need to sign genuinely *different* messages (as opposed to applications of OMS).

However, any public-key-infrastructure-based cryptographic proposal for networking applications requires all parties to know the authentic public keys of all other parties involved. This means that, in routing-based applications, these protocols incur the setup and storage overhead of distributing the public keys and corresponding certificates of all users out-of-band, and participating routers storing them indefinitely. Otherwise, public keys (which cannot be aggregated) and certificates of the signer in each signature would always have to be sent along with the latter for verification, defeating the purpose of using constant-size multiparty signatures to minimize bandwidth in the first place. As noted in previous works [25, 4, 48], identity-based cryptography [12], in which an arbitrary string (e.g. an IP address) acts as a user's public key (the corresponding private key for which can be obtained by authenticating oneself to a trusted private key generator or PKG) and verifying a signature requires knowledge only of a sender's identity in addition to a "master" public key of the PKG, can offer a superior alternative for such applications (subject to various trade-offs). This is because most of the information needed for verifying an aggregate signature is then already contained in the description of "who signed what." It is a compelling setting in which to design and deploy aggregate schemes.

Yet the only (non-interactive) identity-based aggregate signature scheme to date is that of [25], which has the restriction that signers in a given aggregate must agree on a "common nonce" never used by any of them before; indeed, if a signer ever re-uses such a nonce in two different signatures, it then becomes simple to forge a signature by that signer on any message of one's choice. From a functionality perspective, then, in order for the scheme to remain non-interactive, one possibility would be to simply trust the first signer in an aggregate (when signing is done in a "sequential" fashion) to pick a fresh random nonce each time. But there is no reason for this trust. Alternatively, one could rely on synchronized clocks of the signers and instantiate the nonce as a time-stamp; however, an honest computer's perceived clock-time could be altered by a simple virus or after a power failure, which would lead to new potential attacks in practice. Therefore, the above restriction seems rather imprudent from the standpoint of security.

Contributions. After defining the primitive, we design a security model for identity-based sequential aggregate signatures (IBSAS) which adapts the security model of [34] to the identity-based setting. ("Sequential" means that, as for OMS, signatures are aggregated one-by-one as the aggregate-so-far moves along the path, as is natural in the routing-based applications we consider.) Then we provide the first construction of an IBSAS scheme that does *not* place any such "common nonce" restriction on the signers. At a high level, this is achieved by not "aggregating the randomness" produced by the signers on a single group element in an aggregate signature as in previous schemes. (See Section 4.2 for more details.) We prove our construction secure in the RO model under a suitable modification of a computational assumption previously used e.g. in [35, 17, 2]. To help justify the new assumption, we its prove it holds in the generic bilinear group model of [8]. This proof constitutes a "heuristic" security argument showing the assumption holds unless adversarial algorithms exploit specific properties of the underlying algebraic group (i.e. special properties beyond its basic structure), which has become a common way of building confidence in new cryptographic assumptions about the "bilinear" groups we use (see e.g. [8, 9]).

Applications in more detail. As we mentioned, our scheme seems to fit in nicely with route attestation in S-BGP, especially because storage overhead of the protocol has been cited as a major concern [16, 49]. Identities here would roughly consist of an organization name, AS number, and IP address range, which all together are vastly smaller than traditional public keys and certificates. Each identity would be bound to a secret key by a PKG, e.g. ICANN (Internet Corporation for Assigned Names and Numbers). Actually, in practice, the PKG would be in a hierarchy rooted at the latter (cf. [30]), whereby it can delegate generation of user secret keys to descendant PKGs, signatures under which can be verified with the public keys of the PKGs along the path to the root. (We provide an extension of our IBSAS scheme based on [26] that allows this to be done efficiently.) Note that the overhead associated with obtaining and storing these keys – which is equivalent to that of obtaining and storing public keys of a hierarchical certificate authority (CA) – is typically much smaller than that of obtaining traditional public keys and certificates of the signers themselves, which the identity-based setting eliminates.

Further related work. Herranz and Galindo et al. [28, 24] obtain results about identity-based signature schemes permitting aggregation of signatures from the same signer only. Append-only signatures [31] is an interesting public-key primitive suggested for use in S-BGP route attestation, but no construction yielding less than $\omega(\sqrt{n})$-size signatures for n signers is currently known. We clarify that [25] appears to be the only previous (non-interactive) identity-based aggregate signature scheme in the literature; another recent scheme of [19] is interactive. Interactive (i.e. multi-round) identity-based multisignatures are also studied in [4].

1.4 Versions of this Paper and Corrections

This full version of the paper corrects several typos and mistakes from the proceedings version, as well as includes all proofs omitted from the latter. In particular, our security model for IBSAS schemes given in Definition 4.2 has changed. We initially claimed that our IBSAS scheme additionally met an "enhanced" notion of security beyond what is typically required of aggregate signature schemes. (We are unaware of any concrete application of the enhanced definition to secure routing.) We elaborate on this and define such an enhanced security definition for IBSAS in Appendix A for completeness.

2 Preliminaries

Notation and conventions. Let $\mathbb{Z}$ denote the set of integers, $\mathbb{N}$ the positive integers, and $\mathbb{Z}_n$ the integers modulo a number $n \geq 2$. If $\mathbb{G}$ is a prime-order group then $\mathbb{G}^*$ is its set of generators, i.e. $\mathbb{G} = \mathbb{G} \setminus 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ denotes the identity in $\mathbb{G}$. We denote by $\{0,1\}^*$ the set of all (binary) strings of finite length and by ε the empty string. If X is a string then $|X|$ is its length in bits.

If X, Y are strings then $X \| Y$ denotes an encoding from which X and Y are uniquely recoverable. If S is a finite set then $s \stackrel{\$}{\leftarrow} S$ means that s is selected uniformly at random from S. For any $l \in \mathbb{N}$, we often write $s_1, s_2, \ldots, s_l \stackrel{\$}{\leftarrow} S$ as shorthand for $s_1 \stackrel{\$}{\leftarrow} S$; $s_2 \stackrel{\$}{\leftarrow} S$; ... ; $s_l \stackrel{\$}{\leftarrow} S$. The notation $b \stackrel{\delta}{\leftarrow} \{0,1\}$ means that a bit b is assigned value 1 with some probability $0 \leq \delta \leq 1$ and 0 otherwise. If A is a randomized algorithm then $x \stackrel{\$}{\leftarrow} A(y, z, \ldots)$ means that x is assigned the output of running S on inputs $y, z, \ldots$. If A is deterministic then we drop the dollar sign above the arrow. In either case, $A(y, z, \ldots) \Rightarrow x$ denotes that A outputs x after being run on inputs $y, z, \ldots$. All algorithms considered in this paper are efficient and possibly randomized unless indicated otherwise. An adversary is an algorithm with an overlying experiment. By convention, the running-time of an adversary includes that of its overlying experiment.

Bilinear maps. Our schemes use bilinear maps (aka. pairings). Let $\mathbb{G}, \mathbb{G}_T$ be groups of the same prime order p. Following a convention in the cryptographic community, we write both groups multiplicatively. A *pairing* is an efficiently computable map $\mathbf{e}\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that the following two conditions hold:

- Bilinearity: For all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, we have $\mathbf{e}(u^a, v^b) = \mathbf{e}(u, v)^{ab}$.
- Non-degeneracy: For any generator $g \in \mathbb{G}^*$, we have $\mathbf{e}(g, g) \neq 1_{\mathbb{G}_T}$, i.e. $\mathbf{e}(g, g)$ generates $\mathbb{G}_T$.

Observe that $\mathbf{e}(\cdot, \cdot)$ is symmetric since $\mathbf{e}(g^a, g^b) = \mathbf{e}(g, g)^{ab} = \mathbf{e}(g^b, g^a)$.

Definition 2.1 We call an algorithm $\mathcal{G}$ that outputs (descriptions of) $p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}$ as above a *bilinear-group generation algorithm*, and $\mathbb{G}$ a *bilinear group*. ▌

In practice, $\mathbb{G}$ is typically a subgroup of the group of rational points of an elliptic curve over a finite field. Using embedding degree (the degree of certain extension of the ground field) $k = 2$, for standard security levels (meaning discrete log computation in $\mathbb{G}$ is believed to take at least 2^{80} basic operations), elements in $\mathbb{G}$ can be represented using about 512 bits. It is also currently possible to reduce this length to 237 bits for the same security level by choosing $k = 6$, but there are fewer suitable curves known in this case [11]. It is possible that this bit-length will be further reduced in future research. Note that we purposely do not consider the "asymmetric" setting, as in $\mathbf{e}\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ on groups $\mathbb{G}_1 \neq \mathbb{G}_2$, because, although in this case elements in $\mathbb{G}_1$ could be represented using only 160 bits in this case, representation of elements in $\mathbb{G}_2$ would then require at least 1024 bits (due to the "MOV" attack [36]). Since our signatures would contain elements of both, their total length would actually be longer.

Although the bit-length of the representation of elements in $\mathbb{G}$ is 512 bits with embedding degree $k = 2$, for computational efficiency the *order* of $\mathbb{G}$ is usually be chosen to be about 2^{160}. In particular, this means that exponentiations in $\mathbb{G}$ use exponents of only about 160 bits in length. With embedding degree $k = 2$, the cost of computing a pairing is currently about that of two RSA decryptions using CRT preprocessing; with $k = 6$, the cost is about twice as much. See recent benchmarks at [33]. While pairing computation is expensive, on-going algorithmic advances and hardware implementations may bring this cost down.

Below, we formulate some computational problems in such groups that we use for our security proofs. Note that we omit formally defining what it means for these problems to be "hard" and therefore we do not explicitly make any assumptions about them as such, but such definitions and assumptions can be easily derived from our treatment in standard ways.

CDH problem. First we recall the well-known *computational Diffie-Hellman problem* (CDH) in bilinear groups.

Definition 2.2 Fix a bilinear group generator $\mathcal{G}$. We define the *CDH*-advantage of an algorithm A relative to $\mathcal{G}$ as

$$\mathbf{Adv}_{\mathcal{G}}^{\mathrm{CDH}}(A) \stackrel{\mathrm{def}}{=} \Pr\Big[\, C = g^{ab} \;:\; (p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \stackrel{\$}{\leftarrow} \mathcal{G}\;;\; g \stackrel{\$}{\leftarrow} \mathbb{G}^*\;;\; a, b \stackrel{\$}{\leftarrow} \mathbb{Z}_p\;;\; C \stackrel{\$}{\leftarrow} A(p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, g^a, g^b) \,\Big] \;.$$

LRSW PROBLEM. We next recall the *LRSW problem* (LRSW), which was introduced in [35] and has subsequently been used in other works, including [17, 2].

Definition 2.3 Fix a bilinear group generator $\mathcal{G}$. For $(p, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ output by $\mathcal{G}$, we define for all $x, y \in \mathbb{Z}_p$ the associated oracle $\mathcal{O}_{x,y}^{\mathrm{LRSW}}(\cdot)$, which takes input $m \in \mathbb{Z}_p$ and is defined as

Oracle $\mathcal{O}_{x,y}^{\mathrm{LRSW}}(m)$
$u \stackrel{\$}{\leftarrow} \mathbb{G}^*$
Return (u^{x+mxy}, u^y, u)

We then define the *LRSW*-advantage of an algorithm A relative to $\mathcal{G}$ as

$$\mathbf{Adv}_{\mathcal{G}}^{\mathrm{LRSW}}(A) \stackrel{\mathrm{def}}{=} \Pr\Big[\, C = (m', v^{a+m'ab}, v^b, v) \;:\; (p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \stackrel{\$}{\leftarrow} \mathcal{G}\;;\; g \stackrel{\$}{\leftarrow} \mathbb{G}^*\;;\; a, b \stackrel{\$}{\leftarrow} \mathbb{Z}_p \;;\; C \stackrel{\$}{\leftarrow} A^{\mathcal{O}_{a,b}^{\mathrm{LRSW}}(\cdot)}(p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, g^a, g^b) \,\Big] \;,$$

where $m' \in \mathbb{Z}_p$ was not queried by A to its oracle and any $v \neq 1_{\mathbb{G}}$.

M-LRSW PROBLEM. We introduce a related computational problem that we call the *modified-LRSW problem* (M-LRSW), defined in a similar way to the above.

Definition 2.4 Fix a bilinear group generator $\mathcal{G}$. For $(p, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ output by $\mathcal{G}$, we define for all $a, b \in \mathbb{Z}_p$ and $g, u, v \in \mathbb{G}^*$ the associated oracle $\mathcal{O}_{g,u,v,a,b}^{\mathrm{M\text{-}LRSW}}(\cdot)$, which takes input $m \in \mathbb{Z}_p$ and is defined as

Oracle $\mathcal{O}_{g,u,v,a,b}^{\mathrm{M\text{-}LRSW}}(m)$
If $m = 0$ then return $\perp$
$r \stackrel{\$}{\leftarrow} \mathbb{Z}_p$
Return $(u^{mr} g^{ab}, v^r g^{ab}, g^r)$

We then define the *M-LRSW*-advantage of an algorithm A relative to $\mathcal{G}$ as

$$\mathbf{Adv}_{\mathcal{G}}^{\mathrm{M\text{-}LRSW}}(A) \stackrel{\mathrm{def}}{=} \Pr\Big[\, C = (m', u^{m'x} g^{ab}, v^x g^{ab}, g^x) \;:\; (p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \stackrel{\$}{\leftarrow} \mathcal{G}\;;\; g, u, v \stackrel{\$}{\leftarrow} \mathbb{G}^* \;;\; a, b \stackrel{\$}{\leftarrow} \mathbb{Z}_p\;;\; C \stackrel{\$}{\leftarrow} A^{\mathcal{O}_{g,u,v,a,b}^{\mathrm{M\text{-}LRSW}}(\cdot)}(g, u, v, g^a, g^b) \,\Big] \;,$$

where $m' \in \mathbb{Z}_p$ was not queried to the oracle and any $x \in \mathbb{Z}_p$.

Intuitively, the difference between the M-LRSW and LRSW problems is that in the former, the oracle provided to A forms its tuple to return by raising some fixed group elements (meaning these group elements are the same across all invocations of the oracle), namely $u, v, g \in \mathbb{G}$, to polynomials evaluated at a random exponent $r \in \mathbb{Z}_p$, while conversely in the latter a random group element u is chosen by the oracle and is raised to polynomials evaluated at fixed exponents $x, y \in \mathbb{Z}_p$.

We clarify that we call the former the *modified*-LRSW problem because of syntactic similarity only; we do not claim any other relation between them. In Section 5, we show that the M-LRSW is hard in the generic bilinear group model of [8]. This has become a standard way of building confidence in the hardness of computational problems in groups equipped with bilinear maps.

3 Ordered Multisignatures

Ordered multisignatures (OMS) are a natural extension of the notion of multisignatures [6] in which, intuitively, a (constant-size) ordered multisignature on a message attests not only to the fact that some specified group of signers signed it (as in a "plain" multisignature scheme), but also to the order in which they signed. Note that such (non-interactive) schemes are "sequential" by nature. As discussed in the Introduction, potential applications include BGP data-plane security [23] and security in in-band fault localization [39, 38]. One can easily construct an OMS scheme from any aggregate signature scheme; however, the benefit of our OMS construction is that it significantly improves computational efficiency and scalability over existing aggregate signature schemes.

3.1 OMS Schemes and Their Security

Syntax. We formally define the syntax of an OMS scheme.

Definition 3.1 We specify an OMS scheme $\mathsf{OMS} = (\mathsf{OPg}, \mathsf{OKg}, \mathsf{OSign}, \mathsf{OVf})$ by four algorithms:

- A *parameter generation algorithm* OPg that returns some global information I for the scheme. This algorithm can be run by a trusted third-party or standards bodies.
- A *key generation algorithm* OKg run by a user that on the input global information I returns a public-private key-pair (pk, sk).
- A *signing algorithm* OSign run by a user that on inputs its secret key sk, a message $m \in \{0,1\}^*$, an OMS-so-far σ, and a list of $i-1$ public keys $L = (pk_1, \ldots, pk_{i-1})$. It returns a new OMS σ', or $\perp$ if the input is deemed invalid.
- A *verification algorithm* OVf that on inputs a list of public keys $(pk_1, \ldots, pk_n)$, a message m, and an OMS σ returns a bit.

For consistency, we require that the probability that $\mathsf{OVf}(L_n, m, \sigma_n) \Rightarrow 1$ is 1, for all $n \in \mathbb{N}$ and all $m \in \{0,1\}^*$, where the probability is over the experiment

$$
\begin{array}{l}
I \stackrel{\$}{\leftarrow} \mathsf{OPg} \; ; \; (pk_1, sk_1), \ldots, (pk_n, sk_n) \stackrel{\$}{\leftarrow} \mathsf{OKg}(I) \\
\sigma_0, L_0 \leftarrow \varepsilon \\
\text{For } i = 1, \ldots, n \text{ do} \\
\quad \sigma_i \stackrel{\$}{\leftarrow} \mathsf{OSign}(sk_i, m, \sigma_{i-1}, L_{i-1}) \\
\quad L_i \leftarrow (pk_1, \ldots, pk_i).
\end{array}
$$

We also require that $\mathsf{OSign}(sk, m, \sigma, L) \Rightarrow \perp$ if $|L| > 1$ and $\mathsf{OVf}(L, m, \sigma) \Rightarrow 0$ (see Remark 3.3). ▮

Security. We adapt the notion of security for multisignatures given in [6] to our context. Intuitively, a secure OMS scheme, in addition to being secure as a "plain" multisignature scheme, must enforce an additional unforgeability with respect to the ordering of the signers; it should not be possible to re-order the positions of honest signers in an OMS, even if all other signers are malicious. (Note that this also implies that ordered multisignatures cannot be "adversarially combined;" e.g. a forger who sees two ordered multisignatures on a message m by signers (A, B) and (separately) by (C, D) cannot produce a single ordered multisignature on m by signers (A, B, C, D). Security of plain multisignatures does not prevent this.)

Similarly to the model of [6], we require users to prove knowledge of their secret keys during public-key registration with a CA. For simplicity, this is modeled by requiring an adversary to hand over secret keys of malicious signers. This is known as the registered- or certified-key model.

Definition 3.2 Let $\mathsf{OMS} = (\mathsf{OPg}, \mathsf{OKg}, \mathsf{OSign}, \mathsf{OVf})$ be an OMS scheme. We consider the following *UF-OMS experiment* associated to OMS and a forger F with access to an oracle, which runs in three stages.

Setup: The experiment first runs OPg to obtain output I and then generates a challenge key-pair (pk, sk) by running OKg on input I.

Attack: F runs on inputs I, pk. F may query a key registration oracle with a key-pair (pk', sk') and coins c used for key generation, which records pk' as *registered* if $\mathsf{OKg}(I; c) \Rightarrow (pk', sk')$. (This is a simplified model of a possibly more-complex key registration protocol with a CA that involves proofs of knowledge of secret keys.) F also has access to a signing oracle $\mathcal{O}_{\mathsf{OSign}}(sk, \cdot, \cdot, \cdot)$, which on inputs m, σ, L returns $\perp$ if not all public keys in L are registered and $\mathsf{OSign}(sk, m, \sigma, L)$ otherwise.

Forgery: Eventually, F halts with outputs a list of public keys $L^* = (pk_1^*, \ldots, pk_n^*)$, a message m^*, and a purported OMS signature σ^*. This output is considered to be a *forgery* if it holds that (1) $\mathsf{OVf}(L^*, m^*, \sigma^*) = 1$, (2) $pk_{i^*}^* = pk$ for some $i^* \in \{1, \ldots, n\}$, (3) all public keys in L^* except pk are registered, and (4) F did not query m^*, σ', L' to its signing oracle where $|L'| = i^* - 1$ for any $\sigma' \in \{0,1\}^*$.

We define the *UF-OMS*-advantage $\mathbf{Adv}^{\text{UF-OMS}}_{\mathsf{OMS}}(F)$ of F against OMS as the probability that F outputs a forgery in the above experiment, taken over the coin flips of the parameter generation algorithm, the oracles, and any by F itself. We say that F *outputs lists* of length at most $n_{\max}$ if all its lists of public keys used in calls to its signing oracle have length at most $n_{\max} - 1$ and that in its final output (i.e. L^* above) has length at most $n_{\max}$. ▮

Remark 3.3 Our security model does not capture the natural requirement that an honest user should only sign at position i in an OMS if there are really currently $i-1$ signers in it. (As is not the case in secure routing protocols, we do not assume that a signer knows *a priori* its signing position. Instead, she is to obtain this information from the data transmitted by the previous signer.) Otherwise, an adversary that modifies data in transit might simply tell the third signer on the path to sign at the tenth position, and the tenth to sign at the third, for example; the resulting OMS is not required to be invalid. The way we ensure this requirement is instead by the syntactic condition that the signing algorithm in the OMS definition above implicitly must verify validity of the signature-so-far relative to the other data in its input, in order to confirm the signing position.

Remark 3.4 Note that our security model guarantees authenticity of the message signed by an honest signer and her position in an OMS, but not of which specific signers signed before or will sign after her. For example, it would not correspond to a forgery in our model if an OMS σ on a message m valid for public keys (pk_1, pk_2, pk_3) is modified by a malicious signer to some σ' on m valid for (pk_1', pk_2, pk_3'), where pk_1', pk_3' belong to the latter. But this seems to be acceptable in the applications we consider:

- In in-band fault localization [39, 38], reports of packet loss or reordering by a particular router typically indicate a problem upstream, so a main security property we want is that an honest router should not appear to a receiver collecting fault statistics to be further upstream than it actually is — but this does not concern *who* is upstream from the router.
- In S-BGP data plane security [23], since the previously-authenicated AS paths that a data packet may travel are known, if such a packet (having been signed and verified by the previous nodes who have received it to be traveling on an authenticated path) is incorrectly routed to

a malicious node, our security model still ensures the latter cannot modify the packet to then be accepted by an honest node.

However, it is beyond the scope of this paper to rigorously analyze the security requirements needed in these emerging applications (cf. [40]).

3.2 Our OMS Construction and Analysis

THE SCHEME. Our construction extends Boldyreva's multisignature scheme [6] to suitably encode in an OMS the ordering of the signers in addition to the message they signed, by using a technique similar to that of [32]. Our scheme yields a constant-size OMS consisting of 2 group elements (about 1024 or 474 bits depending on implementation details; see Section 2) and is substantially more efficient than all existing aggregate signature alternatives. Unlike these alternatives, it requires essentially *constant work* (in the number of current signers in the OMS) by a user on both signing and verification (see below).

Construction 3.5 Let $\mathcal{G}$ be a bilinear-group generation algorithm. To it we associate the following construction:

Global Parameters: The algorithm first runs $\mathcal{G}$ to obtain output $(p, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ and chooses a random generator $g \in \mathbb{G}^*$ and cryptographic hash function $\mathrm{H}\colon \{0,1\}^* \to \mathbb{G}$. (The analysis will model the latter as a random oracle (RO) [5], adjusting security definitions accordingly.) It returns $(p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, \mathrm{H})$ as the global information I for the scheme.

Key Generation: On input I, the algorithm chooses random $s, t, u \in \mathbb{Z}_p$ and returns $(S = g^s, T = g^t, U = g^u)$ as pk and (s, t, u) as sk.

Signing: On inputs $sk_i, m, \sigma, L = (pk_1, \ldots, pk_{i-1})$, the algorithm first verifies that $\mathsf{OVf}(L, m, \sigma) \Rightarrow 1$ (as defined below) and if not, outputs $\perp$. (This step is skipped for a first signer, i.e. if $i = 1$, for whom σ is defined as $(1_{\mathbb{G}}, 1_{\mathbb{G}})$.) Then it parses σ as (Q, R) and chooses random $r \in \mathbb{Z}_p$. It computes:
1. $R' \leftarrow R \cdot g^r$
2. $X \leftarrow (R')^{t_i + iu_i}$
3. $Y \leftarrow (\prod_{j=1}^{i-1} T_j (U_j)^j)^r$
4. $Q' \leftarrow \mathrm{H}(m)^{s_i} \cdot Q \cdot X \cdot Y$

Finally, it returns (Q', R').

Verification: On inputs $(pk_1, \ldots, pk_n), m, \sigma$, the algorithm first checks that all of $pk_1, \ldots, pk_n$ are distinct and outputs 0 if not.[1] Then it parses σ as (Q, R) and checks if

$$\mathbf{e}(Q, g) \stackrel{?}{=} \mathbf{e}(\mathrm{H}(m), \prod_{i=1}^{n} S_i) \cdot \mathbf{e}(\prod_{i=1}^{n} T_i (U_i)^i, R) \,.$$

If so, it outputs 1. If not, it outputs 0. Consistency follows straightforwardly from the bilinearity condition of a pairing. ▮

Thus, an ordered multisignature in our scheme on a message m by n signers with public keys $pk_1, \ldots, pk_n$, respectively, has the form

$$\left(\mathrm{H}(m)^{\sum_i^n s_i} (g^{\sum_i^n t_i + iu_i})^{\sum_i^n r_i},\ g^{\sum_i^n r_i}\right) \,,$$

where r_i is the randomness chosen by the i-th signer.

[1] This is needed for our security proof, but in all applications we consider repeating signers in the "signature path" is not needed anyway.

SECURITY. Intuitively, the following implies that our OMS scheme is secure (in the RO model) if the CDH is hard relative to its associated bilinear-group generator $\mathcal{G}$.

Theorem 3.6 Let $\mathcal{G}$ be a bilinear-group generation algorithm and OMS be the associated OMS construction given by to Construction 3.5. Suppose there exists a forger F against OMS in the RO model that makes at most at most q_{h} queries to its hash oracle, at most q_{s} queries to its signing oracle, and outputs lists of length at most $n_{\max} \geq 1$. Then there is an algorithm B against the CDH relative to $\mathcal{G}$ such that

$$\mathbf{Adv}^{\text{UF-OMS}}_{\mathsf{OMS}}(F) \;\; \leq \;\; n_{\max} e(q_{\mathrm{s}}+1) \cdot \mathbf{Adv}^{\text{CDH}}_{\mathcal{G}}(B) \; . \tag{1}$$

Furthermore, the running-time of B is at most that of A plus $\tau(\mathcal{G}) \cdot O((q_{\mathrm{h}} + n_{\max}(q_{\mathrm{s}}+1))$, where $\tau(\mathcal{G})$ is the maximum time for an exponentiation in the bilinear groups output by $\mathcal{G}$. ▌

Above and in Theorem 4.5, e denotes the base of the natural logarithm.

Proof: See Appendix B. ▌

Interestingly, our security proof relies on specific properties of Boldyreva's multisignature scheme [6], in the sense that if the recent standard model (random oracle devoid) multisignature scheme of Lu et al. [32] is "substituted" for the former in our OMS construction, our approach to proving security no longer seems to work. Our proof also leverages a technique of [7], originally developed for achieving "selectively-secure" identity-based encryption in the standard model.

RUNNING-TIME ANALYSIS. In our efficiency analysis, we assume that $|\mathbb{G}| = 2^{160}$, i.e. $|p| = 160$; see Section 2. Then, step 1 in the signing algorithm requires one 160-bit exponentiation. (By which we mean that the bit-length of the exponent here is about 160 bits.) In typical applications, steps 2, 3, and 4 can essentially be executed together in the time of one 3-term multi-exponentiation, which is faster than computing 1.5 individual exponentiations. This ignores the cost of computing (we re-name $i-1$ as n here for consistency with the below) $\prod_{j=1}^{n} T_j(U_j)^j$ in step 3, so let us justify this. Computing $\prod_{j=1}^{n}(U_j)^j$ requires n $O(\log n)$-bit exponentiations. So, even if n is a hundred, this is only about the cost of computing three 160-bit exponentiations. (In most applications, n will be much less.) Thus, signing time will remain dominated by the 3 pairing computations in the verification call – which can be reduced to the time of about 2.5 individual pairing computations using a technique of [43]) – and similarly verification requires essentially *constant work* in the number of signers in the OMS.

EFFICIENCY COMPARISON WITH [32]. As noted in the Introduction, one can construct an OMS scheme from any aggregate signature scheme, basically by having the i-th signer add its signature on $m \parallel i$ to the aggregate-so-far, where m is the common message. (We also enforce the requirement in Definition 3.1 that the signing algorithm of the derived OMS scheme verify validity of the signature-so-far in case this is not done by the signing algorithm of the aggregate scheme already, which only affects the comparison with [10] below anyway.) In terms of performance, the best alternative to our OMS scheme seems to be obtained from the "RO version" of the recent aggregate scheme of Lu et al. [32, Section 3.4], which, for basically the same amount of security and signature size,[2] requires an additional n 160-bit exponentiations on both signing and verification (where n is the number of signers currently in the aggregate).

Note that while an n-term multi-exponentiation could be used for these, it would require $(2^n - 2)+2(160-1)$ multiplications (a 160-bit exponentiation by the square-and-multiply method requires

[2]Though [32] claims that if using "asymmetric" pairings, as in $\mathbf{e}\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, their aggregate signatures can have length 320 bits, this appears to be an oversight, since their signatures, like ours, would also contain an element of G_2, whose representation, as we mentioned, would actually require much longer bit-length in this case.

240 multiplications on average) and thus would only provide a speed-up for relatively small values of n. Moreover, it would incur an extra $512\cdot(2^n-1)$ bits of memory usage. We also stress that the bases for these n exponentiations vary from aggregate to aggregate. So, regardless of the computational technique employed, without requiring a prohibitively large amount of memory for pre-computation (and routing platforms are quite memory-constrained in the first place) the cost of computing the n exponentiations will still grow linearly in n. Therefore, the RO version of [32] scales more poorly and provides less equitable distribution of processing time amongst signers (e.g. different ASes) as n grows, giving it more limited applicability in real-world deployment scenarios as compared to our OMS scheme.

Efficiency comparison with [10, 34]. Aggregate signature length is just 160 bits and signing is, strictly speaking, more efficient in the aggregate scheme of [10] than in our OMS scheme, but verification is vastly slower, requiring a linear amount of *pairing computations* in the number of signatures in an aggregate, and verifying the aggregate-so-far is needed anyway upon signing in the derived OMS scheme (cf. Remark 3.3). (We comment that even if it was not, our OMS scheme may still be preferable to [10] due to slow verification time in the latter, which could be of particular concern with denial-of-service attacks on a verifier.) In most routing-based applications, however, a fixed 1024-bit packet size overhead for signatures is still within reason, and, as discussed in Section 2, some implementations may reduce this overhead to less than 500 bits, which is even more manageable.

Finally, we observe that the RSA-based aggregate scheme of [34] either requires proofs of knowledge of RSA keys on key-registration or having the signers' public exponents bigger than their 1024-bit moduli. As for RSA the former are much more expensive than those for discrete log and are not used in practice, this means that their scheme will similarly require a linear number of such costly 1024-bit exponentiations on both signing and verification.

4 Identity-Based Sequential Aggregate Signatures

It has been pointed out in numerous works and tested in [49] that aggregate signatures [10, 34, 32], which allow multiple signers to sign different messages while keeping total signature size constant, can be used to address route announcement authenticity in S-BGP while significantly reducing associated bandwidth overhead. According to the proposal, each AS forwarding an update message should add its signature on the address of the *next* AS on the route (the latter is to prevent an unauthorized AS from picking up the path and makes OMS insufficient here), so that route authenticity can be verified upon receipt of the aggregate.

However, as explained in the Introduction, using public-key schemes that necessitate a public-key infrastructure (PKI) dramatically increases set-up and storage or bandwidth overhead of secure networking protocols. But in the identity-based setting [12], where an arbitrary identifier can be used as a public key, most of the information needed to verify a signature is already included in the transmission anyway. We treat sequential aggregate signatures in this setting. IBSAS schemes appear well-suited for secure routing applications and in particular for route attestation in S-BGP, where storage overhead of the protocol is of particular concern [16]. Our construction improves upon the security of a previous solution in this setting by removing an undue restriction on the signers, making it more useful in practice.

4.1 IBSAS Schemes and Their Security

Syntax. We formally define the syntax of an IBSAS scheme.

Definition 4.1 We specify an *identity-based sequential aggregate signature* (IBSAS) scheme (cf. [25]) $\mathsf{AS} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Sign}, \mathsf{Vf})$ by four algorithms:

- A *setup algorithm* Setup initially run by the trusted private-key generator (PKG) to generate its master public key mpk and corresponding master secret key msk.
- A deterministic *private-key derivation* algorithm KeyDer run by the PKG on inputs msk, ID for any user's identity $ID \in \{0,1\}^*$, to generate the private key sk_{ID} for user ID.
- A *signing algorithm* Sign run by a user ID on inputs its secret key sk_{ID}, a message $m \in \{0,1\}^*$, a list $((ID_1, m_1), \ldots, (ID_{i-1}, m_{i-1}))$ of identity-message pairs, and an aggregate-so-far σ. It returns a new aggregate signature σ', or $\perp$ to indicate that the input was deemed invalid.
- A *verification algorithm* Vf that on inputs the master public key mpk, a list $((ID_1, m_1), \ldots, (ID_n, m_n))$ of identity-message pairs, and an IBSAS σ returns a bit.

For consistency, we require that the probability $\mathsf{Vf}(mpk, L_n, \sigma_n) \Rightarrow 1$ is 1, for all $n \in \mathbb{N}$ and all $\{(ID_i, m_i) \mid 1 \leq i \leq n,\ ID_i \in \{0,1\}^*,\ m_i \in \{0,1\}^*\}$, where the probability is over the experiment

$$
\begin{array}{l}
(mpk, msk) \stackrel{\$}{\leftarrow} \mathsf{Setup} \\
\text{For all } i = 1, \ldots, n \text{ do} \\
\quad sk_{ID_i} \stackrel{\$}{\leftarrow} \mathsf{KeyDer}(msk, ID_i) \\
\sigma_0, L_0 \leftarrow \varepsilon \\
\text{For } i = 1, \ldots, n \text{ do} \\
\quad \sigma_i \stackrel{\$}{\leftarrow} \mathsf{Sign}(sk_{ID_i}, m_i, L_{i-1}, \sigma_{i-1}) \\
\quad L_i \leftarrow ((ID_1, m_1), \ldots, (ID_i, m_i)). \quad \blacksquare
\end{array}
$$

SECURITY. Our notion of security for IBSAS adapts of the notion of security for sequential aggregate signatures presented in [34] to the identity-based setting. It captures the intuition that a forger who can adaptively (1) obtain signatures of users on messages of its choice to be appended to an aggregate-so-far and (2) "corrupt" users by requesting their private keys, should not be able to subsequently "frame" a user as having appended its signature on a message to an aggregate-so-far which it did not. As discussed previously in [4], it is important here that the forger is able to adaptively corrupt users, unlike in the public-key setting, where wlog it receives a public key for just one honest user.

Definition 4.2 Let $\mathsf{AS} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Sign}, \mathsf{Vf})$ be an IBSAS scheme. We consider the following *UF-IBSAS experiment* associated to AS and a forger F with access to two oracles, which runs in three stages.

Setup: The experiment first generates a master key-pair (mpk, msk) by running Setup.

Attack: F runs on input mpk with access to a key-derivation oracle $\mathsf{KeyDer}(msk, \cdot)$ and signing oracle $\mathcal{O}_{\mathsf{Sign}}(\cdot,\cdot,\cdot,\cdot)$. The first operates according to the above definition of the private-key derivation algorithm for IBSAS. The second on inputs an identity ID, a message m, a list of identity-message pairs $L = ((ID_1, m_1), \ldots, (ID_{i-1}, m_{i-1}))$, and an aggregate-so-far σ, sets $sk_{ID} \leftarrow \mathsf{KeyDer}(msk, ID)$ and returns

$$\mathsf{Sign}(sk_{ID}, m, ((ID_1, m_1), \ldots, (ID_{i-1}, m_{i-1})), \sigma)\ .$$

Forgery: Eventually, F halts with outputs a list of identity-message pairs $L^* = ((ID_1^*, m_1^*), \ldots, (ID_n^*, m_n^*))$ and a purported aggregate signature σ^*. This output is considered to be a forgery if (1) $\mathsf{Vf}(mpk, L^*, \sigma^*) \Rightarrow 1$ and (2) there exists some $i^* \in \{1, \ldots, n\}$ such that F did not query $ID_{i^*}^*$ to its key-derivation oracle and did not query $(ID_{i^*}^*, m_{i^*}^*, ((ID_1^*, m_1^*), \ldots, (ID_{(i^*-1)}^*, m_{(i^*-1)}^*)), \sigma)$ to its signing oracle for any $\sigma \in \{0,1\}^*$.

We define the *UF-IBSAS*-advantage $\mathbf{Adv}_{\mathsf{AS}}^{\text{UF-IBSAS}}(F)$ of F against AS as the probability that F outputs a forgery in the above experiment, taken over the coin flips of the setup algorithm, the oracles, and any by F itself. We say that F *outputs lists* of length at most $n_{\max}$ if all its lists of identity-message pairs used in calls to its signing oracle have length at most $n_{\max} - 1$ and that in its final output (i.e. L^* above) has length at most $n_{\max}$. ∎

Comparison to Previous Definitions. Our definition of security for IBSAS, similarly to that for public-key sequential aggregate signatures in [34], makes the requirement that a signature appended to an aggregate cannot be re-used in another aggregate in which the signers and their messages that come before it are different. This requirement is not made in [32], where the signatures in a sequentially-formed aggregate are inherently "unordered." We also note that this requirement is not captured in the security model of [25] in the identity-based setting, which however applies to non-sequential schemes as well.

4.2 Our IBSAS Construction and Analysis

The scheme. We present our IBSAS construction, which is inspired by the recent scheme of [25]. Our scheme yields constant-size aggregate signatures of 3 group elements (about 1536 or 711 bits depending on implementation details; see Section 2) and is reasonably efficient. In particular, verifying an aggregate signature in our scheme requires a small constant (in the number of signatures in an aggregate) amount of pairing computations, though a linear amount of exponentiations. As we explain below, our construction improves functionality/security over the scheme of [25] by lifting a "common nonce" restriction that can lead to some attacks on the scheme in practice.

Construction 4.3 Let $\mathcal{G}$ be a bilinear-group generation algorithm. To it we associate the following construction:

Setup: The algorithm first runs $\mathcal{G}$ to obtain output $(p, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ and chooses a random generators $u, v, g \in \mathbb{G}^*$, a random $\alpha \in \mathbb{Z}_p$, and cryptographic hash functions $\mathrm{H}_1\colon \{0,1\}^* \to \mathbb{G}$ and $\mathrm{H}_2\colon \{0,1\}^* \to \mathbb{Z}_p^*$. (The analysis will model the latter as random oracles (ROs) [5], adjusting security definitions accordingly.) It returns $(p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, u, v, g, g^\alpha, \mathrm{H}_1, \mathrm{H}_2)$ as the mpk and α as the msk.

Key Derivation: On inputs msk and $ID \in \{0,1\}^*$, the algorithm returns $\mathrm{H}_1(ID)^\alpha$ as sk_{ID}.

Signing: On inputs $sk_{ID_i}, m_i, L = ((ID_1, m_1), \ldots (ID_{i-1}, m_{i-1})), \sigma$, the algorithm first parses σ as (X, Y, Z). (This step is skipped for a first signer, i.e. if $i = 1$, for whom σ is defined as $(1_\mathbb{G}, 1_\mathbb{G}, 1_\mathbb{G})$.) It chooses a random $r \in \mathbb{Z}_p$. Below, we let s_j denote $ID_1 \| m_1 \| \ldots \| ID_j \| m_j$ for $j \geq 1$. The algorithm computes:

1. $X' \leftarrow u^{r \prod_{j=1}^{i} \mathrm{H}_2(s_j)} \cdot \mathrm{H}_1(ID)^\alpha$
2. $Y' \leftarrow v^r \cdot \mathrm{H}_1(ID_i)^\alpha$

Finally, it returns

$$(X \cdot X', Y^{1/\mathrm{H}_2(s_i)} \cdot Y', Z^{1/\mathrm{H}_2(s_i)} \cdot g^r) .$$

Above, a term $1/\mathrm{H}_2(s)$ for a string s means $\mathrm{H}_2(s)^{-1} \mod p$.

<u>*Verification*</u>: On inputs $mpk, ((ID_1, m_1), \ldots, (ID_n, m_n)), \sigma$, the algorithm first returns 0 if not all of $ID_1, \ldots, ID_n$ are distinct.[3] Then it parses σ as (X, Y, Z) and verification proceeds in two steps. In the first step, it checks if

$$\mathbf{e}(Y, g) \stackrel{?}{=} \mathbf{e}(v, Z) \cdot \mathbf{e}(\prod_i^n \mathrm{H}_1(ID_i)^{1/(\prod_{j=i+1}^n \mathrm{H}_2(s_j))}, g^\alpha) .$$

If not, the algorithm returns 0. If so, it continues to the next step, where it computes $Z' \leftarrow Z^{\prod_{i=1}^n \mathrm{H}_2(s_i)}$ and then checks if

$$\mathbf{e}(X, g) \stackrel{?}{=} \mathbf{e}(u, Z') \cdot \mathbf{e}(\prod_i^n \mathrm{H}_1(ID_i), g^\alpha) .$$

If not, the algorithm returns 0. If so, the algorithm returns 1. Consistency follows straightforwardly from the bilinearity condition of a pairing. ▮

Thus, an aggregate signature in our scheme on messages $m_1, \ldots, m_n$ by signers $ID_1, \ldots, ID_n$, respectively, has the form

$$\left(\prod_i^n u^{r_i \prod_{j=1}^i \mathrm{H}_2(s_j)} \cdot \mathrm{H}_1(ID_i)^\alpha, \ \prod_i^n (v^{r_i} \cdot \mathrm{H}_1(ID_i)^\alpha)^{1/(\prod_{j=i+1}^n \mathrm{H}_2(s_j))}, \ \prod_i^n g^{r_i/(\prod_{j=i+1}^n \mathrm{H}_2(s_j))} \right) ,$$

where $r_i \in \mathbb{Z}_p$ is the randomness chosen by the i-th signer ID_i.

Remark 4.4 Note that our construction does not verify validity of the aggregate-so-far in its signing algorithm. (For comparison, the public-key aggregate signature schemes of [32, 34] both require such verification on signing for their security proofs, while the scheme of [10], which however requires an amount of pairing computations linear in the number of signatures in an aggregate to verify it, does not.) This turns out to be significant in the context of route attestation in S-BGP, where, for efficiency reasons, it is desirable to perform "lazy" verification of route attestations [30, 49]. This means an incoming attestation is only verified if the route is later chosen as a "best path," which (if it happens at all) is *after* the current AS has already added its signature to the aggregate-so-far and forwarded the result. An aggregate scheme that does not require such verification on signing allows this optimization to be done safely, without losing the security proof for the scheme.

SECURITY. Intuitively, the following establishes that our IBSAS scheme is secure (in the RO model) if the M-LRSW is hard relative to its associated bilinear-group generator $\mathcal{G}$.

Theorem 4.5 Let $\mathcal{G}$ be a bilinear-group generation algorithm and AS be the associated IBSAS scheme given by Construction 4.3. Suppose there exists a forger F against AS in the RO model that make at most $q_{\mathrm{h}_1}, q_{\mathrm{h}_2}$ queries to its $\mathrm{H}_1, \mathrm{H}_2$ hash oracles, at most q_{k} queries to its key-derivation oracle, at most q_{s} queries to its signing oracle, and outputs lists of length at most $n_{\max} \geq 1$. Then there is an algorithm B against the M-LRSW relative to $\mathcal{G}$ such that

$$\mathbf{Adv}_{\mathsf{AS}}^{\text{UF-IBSAS}}(F) \ \leq \ e(n_{\max} + q_{\mathrm{k}}) \cdot \mathbf{Adv}_{\mathcal{G}}^{\text{M-LRSW}}(B) \ + \ \frac{q_{\mathrm{h}_2}(q_{\mathrm{h}_2} - 1)}{2 \cdot 2^{l_{\min}(\mathcal{G})}} ,$$

where $l_{\min}(\mathcal{G})$ is the minimum bit-length of the order p of a bilinear group output by $\mathcal{G}$. Furthermore, B makes at most q_{s} queries to its M-LRSW oracle and its running-time is at most that of A plus

[3]This check is needed for our security proof but does not constitute a significant restriction because, as previously mentioned, in all applications we consider repeating signers in an aggregate is unnecessary. Further, in S-BGP route attestation, repeats in an AS path constitutes a security vulnerability and must not be allowed anyway.

$\tau(\mathcal{G})\cdot O(q_{h_1}+q_k+q_s+n_{max}) + \upsilon(\mathcal{G})\cdot O(n_{max})$, where $\tau(\mathcal{G})$ is the maximum time for an exponentiation in a bilinear group output by $\mathcal{G}$ and $\upsilon(\mathcal{G})$ is the maximum time for a multiplication in $\mathbb{Z}_p$. ▌

Proof: See Appendix C. ▌

Comparison with [25] and design rationale. For comparison, we recall that the recent identity-based aggregate signature scheme of [25] produces an aggregate signature on messages $m_1, \ldots, m_n$ by signers $ID_1, \ldots, ID_n$, respectively, of the form

$$\left(H_3(w)^{\sum_i^n r_i} \cdot \prod_i^n H_1(0\|ID_i)^\alpha \cdot \prod_i^n H_1(1\|ID_i)^{\alpha H_2(w\|ID_i\|m_i)},\ g^{\sum_i^n r_i} \right) ;$$

here the private key of user ID_i consists of the pair $(H_1(0\|ID_i)^\alpha, H_1(1\|ID_i)^\alpha)$, $H_1, H_3 \colon \{0,1\}^* \to \mathbb{G}$ and $H_2 \colon \{0,1\}^* \to \mathbb{Z}_p$ are hash functions, and w is a nonce picked by the first signer. The element $H_3(w)$ provides a "common place" for the signers to "aggregate their randomness;" indeed, [25] remarks that this seems necessary to enable aggregation of individual signatures. However, for security, the string w is then required to be a "common nonce" specific to each aggregate, meaning each of $ID_1, \ldots, ID_n$ above need to be sure that they have not used it in any other aggregate before. Indeed, if any signer repeats the same w in two different signatures, it becomes simple for an adversary to forge a signature by the signer on any message, via simple linear algebraic attacks in the exponent (cf. [25, Remark 4]). Unfortunately, this restriction still leaves their scheme vulnerable in practice in a different way: typical implementations of the scheme in the context of secure routing protocols would likely use a time-stamp as w and require signers to check that an aggregate-so-far has w sufficiently close to the signer's current clock-time, but then the possibility of malicious altering of the latter, say by installing a simple virus that can get no information about the secret key, introduces potential real-world attacks.

Our IBSAS construction, however, shows that such a "common place" on which to aggregate the randomness chosen by the signers is not necessary to enable aggregation and is the first such scheme whose security does not rely on this "common nonce" restriction. To see how this works, first ignore the Y component of an aggregate in our scheme as well as the first step in the verification algorithm. Notice that the randomness r_i chosen by the i-th signer is used as the exponent on $u^{\prod_{j=1}^{i} H_2(s_j)}$, which varies across signers. Moreover, since randomness chosen by a signer changes accordingly across different signatures and is not public, the kind of linear-algebraic attacks mentioned above no longer work. However, this "simplified" version of our scheme is still not secure. For example, consider an individual signature by user ID on a message m in this scheme, which looks like

$$\left(X = u^{rH_2(ID\|m)} \cdot H_1(ID)^\alpha,\ Z = g^r \right) .$$

Given (X, Z), an adversary could easily produce a forgery (X', Z') of a signature by ID on a message m' of its choice (assuming $H_2(ID\|m')$ is not zero) by setting $X' \leftarrow X$ and $Z' \leftarrow Z^{H_2(ID\|m)/H_2(ID\|m')}$. To correct for this, our final scheme adds another component to the signature, namely $Y = v^r \cdot H_1(ID)^\alpha$, which is intended to prevent the above attack by allowing the verifier to detect when the value of Z has been so-tampered with (the first step of the verification algorithm).

It is worth noting that it is not obvious how signatures of the form (X, Y, Z) above can still be combined to form a single compact aggregate signature. Our IBSAS construction demonstrates a way to do this.

Further discussion. We caution that the M-LRSW problem we introduce to prove our IBSAS scheme secure is quite strong and as-yet untested by cryptanalysts. However, in Section 5, we prove its hardness in the generic bilinear group model of [8]. Intuitively, this result means that breaking it in practice is likely to nevertheless require a significant new advance in our current understanding of the appropriate elliptic curve groups in which our scheme can be implemented. We also point out that, given the "common nonce" restriction in the previous scheme of [25] (which, under this restriction, is shown to be secure based on CDH) and the potential attacks on that scheme in practice that can result, it is unclear why one should prefer to use the former, even if one strongly favors schemes that are "proven secure" based on more standard computational problems.

Extension to hierarchical PKGs. We sketch an extension of our IBSAS scheme to the setting where users' private keys are generated via PKGs in a hierarchy. (That is, where there are multiple PKGs situated as nodes in a tree, and any PKG can delegate user secret key generation to descendant PKGs; a user obtains its private key from one of them determined by the particular application.) This is useful for S-BGP (cf. [30]). Although this functionality can be achieved in a generic way by using any public-key signature scheme to create certificates binding PKGs to their public keys appropriately, the advantage of our extension is that it eliminates the need for these certificates as well as the overhead of using an additional scheme.

The extension, based on the scheme of [26], goes as follows. Let P_i denote a PKG at a level i in the hierarchy. In the initial setup, the root PKG P_0 selects $\mathbb{G}, \mathbb{G}_T, \mathbf{e}, u, v, g, \mathrm{H}_1, \mathrm{H}_2$ as in the basic scheme and additionally chooses another cryptographic hash function $\mathrm{H}_3 \colon \{0,1\}^* \to \mathbb{G}$. The global parameters are $\mathbb{G}, \mathbb{G}_T, \mathbf{e}, u, v, g, \mathrm{H}_1, \mathrm{H}_2, \mathrm{H}_3$, and in addition each P_i chooses its own random $\alpha_i \in \mathbb{Z}_p$ as its secret key and publishes g^{α_i} as its public key. For a PKG P_l below the root, P_l's parent provides it with the secret value $S_l = \prod_{j=1}^{l} \mathrm{H}_3(P_1 \| \ldots \| P_l)^{\alpha_{l-1}}$, where $(P_1, \ldots, P_l)$ is the path in the tree from the root PKG P_0 to P_l. As the private key for a user ID, a PKG P_i provides $sk_{ID} = S_i \cdot \mathrm{H}_1(ID)^{\alpha_i}$ (where $S_0 = 1_{\mathbb{G}}$ by convention). The signing and verification algorithms of our scheme can then be extended straightforwardly. Verifying an aggregate containing a signature by a user ID whose PKG is P_i requires the verifier to have obtained the public keys of all the PKGs on the path in the tree from P_0 to P_i. But as long as the hierarchy is not too large, a value $e_i = \mathbf{e}(\mathrm{H}_3(P_1 \| \ldots \| P_i), g^{\alpha_{i-1}})$ can be cached, and thus verification does not take much longer than in the basic scheme. (Recall that if e_i is cached, $\mathbf{e}(\mathrm{H}_3(P_i)^{\beta}, g^{\alpha_{i-1}})$ for $\beta \in \mathbb{Z}_p$ can be computed as e_i^{β}; one can also use "compressed pairings" [44] here.) Adapting Definition 4.2 to the hierarchical setting can be done following [26]; security of the extended scheme follows from [26, Theorem 2] and Theorem 4.5.

5 On the Hardness of M-LRSW

While it would certainly be preferable to prove the security of our IBSAS scheme under the hardness of a more established computational problem like CDH, given the new functionality that the scheme provides, this is not always possible. We aim here to more carefully justify our use of the M-LRSW. The generic group model [45] models an inability of algorithms to use group representation (i.e. special properties of a group beyond the mere fact that it is a group) in solving a computational problem. Below, we establish a strong lower bound on the hardness of the M-LRSW in an extension of the generic group model to the bilinear group setting [8]. This has become a standard way of building confidence in the hardness of new computational problems in bilinear groups (see e.g. [8, 9]).

The generic bilinear group model. We briefly recall the model of [8], making only minor syntactic changes. For simplicity, we fix a bilinear-group generation algorithm $\mathcal{G}$ that always outputs some fixed $(p, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$. Let g, g_T be generators of $\mathbb{G}, \mathbb{G}_T$, respectively. An algorithm A being executed in the model means that it is run by a corresponding *generic bilinear group experiment* that encodes elements of these groups given to A as random strings of length $\lceil \log p \rceil$ via injective maps $\xi, \xi_T \colon \mathbb{Z}_p \to \{0,1\}^{\lceil \log p \rceil}$, where $\xi(a)$ is the encoding of g^a and $\xi_T(a)$ the encoding of $(g_T)^a$ for all $a \in \mathbb{Z}_p$. That is, any group elements in A's input (its input being that in its usual experiment, as well as $1_{\mathbb{G}}$), in A's oracle queries and the responses it gets back, and in A's output are so-encoded. At any time-step, A can in particular query one of two group operation oracles for $\mathbb{G}, \mathbb{G}_T$, respectively, with encodings $\gamma_1, \gamma_2 \in \{0,1\}^{\lceil \log p \rceil}$ and a bit b to get back $\xi(\xi^{-1}(\gamma_1) \cdot (\xi^{-1}(\gamma_2))^{-b})$, where "·" denotes the corresponding group operation. Likewise, it can query a bilinear map oracle with encodings $\gamma_3, \gamma_4 \in \{0,1\}^{\lceil \log p \rceil}$ to get back $\xi_T(\mathbf{e}(\xi^{-1}(\gamma_3), \xi^{-1}(\gamma_4)))$. We use the same notation for the algorithm's advantage when executed in the model as for its advantage in its usual experiment.

Our result. Intuitively, the following establishes that the M-LRSW is (unconditionally) hard in the generic bilinear group model.

Theorem 5.1 Let $\mathcal{G}$ be as above. Suppose there is an algorithm A solving the M-LRSW relative to $\mathcal{G}$ in the generic bilinear group model that runs in time at most t and makes at most q_{s} queries to its oracle. Then we have that

$$\mathbf{Adv}_{\mathcal{G}}^{\text{M-LRSW}}(A) \;\le\; \frac{4\binom{t+3q_{\mathrm{s}}+5}{2}+3}{p} \,. \;\blacksquare$$

Proof: See Appendix D. ▌

As

$$\binom{t+3q_{\mathrm{s}}+5}{2} \;\le\; (t+3q_{\mathrm{s}}+5)^2 \,,$$

the theorem shows that, asymptotically, an algorithm's advantage in solving the M-LRSW in the generic bilinear group model can increase at most quadratically in the work it performs. This is fairly standard, e.g. for computing discrete logs. In practice, q_{s} corresponds roughly to the maximum number of signatures that an adversary may see, so could be set to about, say, 2^{30}.

6 Conclusions and Open Problems

This work presented two new cryptographic schemes for use in securing several network routing applications, which we believe to be more attractive in practice than existing alternatives. To conclude, we point out some interesting open problems in this area. Our results indicate that it would be useful to devise an identity-based OMS scheme that is more efficient than existing identity-based aggregate constructions. It also remains an excellent open problem to devise an identity-based aggregate signature scheme based on a more standard computational problem (e.g. CDH), but without the limitations of previous constructions. Secondarily, it is important to devise such schemes secure in the standard model (without random oracles).

7 Acknowledgments

We thank Nick Feamster, Murtaza Motiwala, Krzysztof Pietrzak, Michel Abdalla, Younho Lee, Brent Waters, Gene Tsudik, Xavier Boyen, Farbod Shokrieh, Yael Tauman Kalai, and the anonymous reviewers of CCS 2007 for helpful comments, suggestions, and references. Alexandra Boldyreva was supported in part by NSF CAREER award 0545659. Craig Gentry was supported by the Herbert Kunzel Stanford Graduate Fellowship. Adam O'Neill was supported in part by the above-mentioned grant of the first author. Dae Hyun Yum was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD), KRF-2006-352-D00177.

References

[1] W. Aiello, J. Ioannidis, and P. McDaniel. Origin authentication in interdomain routing. In *ACM CCS*, 2003. (Cited on page 3.)

[2] M.-H. Au, W. Susilo, and Y. Mu. Practical compact e-cash. Cryptology ePrint Archive, Report 2007/148, 2007. `http://eprint.iacr.org/`. (Cited on page 6, 8.)

[3] M. Bellare and C. Namprempre and G. Neven. Unrestricted Aggregate Signatures. In *ICALP*, 2007. (Cited on page 3, 5.)

[4] M. Bellare and G. Neven. Identity-based multi-signatures from RSA. In *CT-RSA*, 2007. (Cited on page 5, 6, 14.)

[5] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS*, 1993. (Cited on page 4, 11, 15.)

[6] A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the Gap-Diffie-Hellman-Group signature scheme. In *Public Key Cryptography*, 2003. (Cited on page 3, 4, 5, 9, 11, 12.)

[7] D. Boneh and X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *EUROCRYPT*, 2004. (Cited on page 12, 24.)

[8] D. Boneh and X. Boyen. Short signatures without random oracles. In *EUROCRYPT*, 2004. (Cited on page 6, 8, 17, 18.)

[9] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT*, 2005. (Cited on page 6, 18.)

[10] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT*, 2003. (Cited on page 3, 5, 12, 13, 16.)

[11] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. In *Journal of Cryptology*, Volume 17 , Issue 4, 2004. (Cited on page 7.)

[12] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, 1984. (Cited on page 5, 13.)

[13] M. Burmester, Y. Desmedt, H. Doi, M. Mambo, E. Okamoto, M. Tada, and Y. Yoshifuji. A structured ElGamal-type multisignature scheme. In *PKC*, 2000. (Cited on page 4.)

[14] C.-Y. Lin, T.-C. Wu, and F. Zhang. A Structured Multisignature Scheme from the Gap Diffie-Hellman Group. Cryptology ePrint Archive, Report 2003/090, 2003. `http://eprint.iacr.org/`. (Cited on page 4.)

[15] K. Butler and W. Aiello. Optimizing BGP security by exploiting path stability. In *ACM CCS*, 2006. (Cited on page 3.)

[16] K. Butler, F. Farley, P. McDaniel, and J. Rexford. A survey of BGP security. Apr. 2005. `http://www.research.att.com/jrex/`. (Cited on page 3, 6, 13.)

[17] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO*, 2004. (Cited on page 6, 8.)

[18] H. Chan, D. Dash, A. Perrig, and H. Zhang. Modeling adoptability of secure BGP protocols. In *ACM SIGMETRICS*, 2006. (Cited on page 3.)

[19] X. Cheng, J. Liu, and X. Wang Identity-based aggregate and verifiably encrypted signatures from bilinear pairing. In *ICCSA*, 2005. (Cited on page 6.)

[20] J.-S. Coron. On the exact security of full domain hash. In *CRYPTO*, 2000. (Cited on page 23, 27.)

[21] H. Doi, E. Okamoto, and M. Mambo. Multisignature schemes for various group structures. In *Symposium on Cryptography and Information Security*, 1994. (Cited on page 4.)

[22] H. Doi, E. Okamoto, M. Mambo, and T. Uyematsu. Multisignature scheme with specified order. In *Conference on Communication, Control, and Computing*, 1999. (Cited on page 4.)

[23] N. Feamster, H. Balakrishnan, and J. Rexford. Some foundational problems in interdomain routing. In *HotNets*, 2004. (Cited on page 4, 9, 10.)

[24] D. Galindo, J. Herranz, and E. Kiltz. On the generic construction of identity-based signatures with additional properties. In *ASIACRYPT*, 2006. (Cited on page 6.)

[25] C. Gentry and Z. Ramzan. Identity-based aggregate signatures. In *Public Key Cryptography*, 2006. (Cited on page 5, 6, 14, 15, 17, 18.)

[26] C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In *ASIACRYPT*, 2002. (Cited on page 6, 18.)

[27] G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. Mcdanniel, and A. Rubin. Working around BGP: An incremental approach to improving security and accuracy in interdomain routing. In *NDSS*, 2003. (Cited on page 3.)

[28] J. Herranz. Deterministic identity-based signatures for partial aggregation. *J. Comput.*, 49(3), 2006. (Cited on page 6.)

[29] Y.-C. Hu, A. Perrig, and M. Sirbu. SPV: Secure path vector routing for securing BGP. In *ACM SIGCOMM*, 2004. (Cited on page 3.)

[30] S. Kent, C. Lynn, J. Mikkelson, and K. Seo. Secure border gateway protocol (S-BGP) - Real world performance and deployment issues. In *NDSS*, 2000. (Cited on page 6, 16, 18.)

[31] E. Kiltz, A. Mityagin, S. Panjwani, and B. Raghavan. Append-only signatures. In *ICALP*, 2005. (Cited on page 6.)

[32] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters. Sequential aggregate signatures and multisignatures without random oracles. In *EUROCRYPT*, 2006. (Cited on page 3, 5, 11, 12, 13, 15, 16, 23.)

[33] B. Lynn. The pairing-based crypto library. `http://crypto.stanford.edu/pbc`. (Cited on page 7.)

[34] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In *EUROCRYPT*, 2004. (Cited on page 3, 5, 13, 14, 15, 16.)

[35] A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *Selected Areas in Cryptography*, 1999. (Cited on page 6, 8.)

[36] A. Menezes, T. Okamoto, S. A. Vanstone, Reducing elliptic curve logarithms to logarithms in a finite field. In *IEEE Transactions on Information Theory*, Vol. 39, Num. 5, 1993. (Cited on page 7.)

[37] S. Mitomi and A. Miyaji. A multisignature scheme with message flexibility, order flexibility and order verifiability. In *ACISP*, 2000. (Cited on page 4.)

[38] M. Motiwala, A. Bavier, and N. Feamster. In-band network path diagnosis. *Georgia Tech Technical Report GT-CS-07-07.* (Cited on page 4, 9, 10.)

[39] M. Motiwala and N. Feamster. Position paper: Network troubleshooting on data plane coattails. In *WIRED*, 2006. (Cited on page 4, 9, 10.)

[40] S.M. Bellovin. Position paper: Workable Routing Security. In *WIRED*, 2006. (Cited on page 3, 11.)

[41] A. Saxena and B. Soh. One-way signature chaining - a new paradigm for group cryptosystems. Cryptology ePrint Archive, Report 2005/335, 2005. `http://eprint.iacr.org/`. (Cited on page 5.)

[42] J. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. In *Journal of the ACM*, Vol. 27, 1980. (Cited on page 33.)

[43] R. Granger and N.P. Smart On computing products of pairings. Cryptology ePrint Archive, Report 2006/172, 2006. `http://eprint.iacr.org/`. (Cited on page 12.)

[44] M. Scott and P.S.L.M. Barreto. Compressed Pairings. In *CRYPTO*, 2004. (Cited on page 18.)

[45] V. Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, 1997. (Cited on page 18.)

[46] M. Tada. An order-specified multisignature scheme secure against active insider attacks. In *Australian Conference on Information Security and Privacy*, 2002. (Cited on page 4.)

[47] T. Wan, E. Kranakis, and P. van Oorschot. Pretty secure BGP, psBGP. In *NDSS*, 2005. (Cited on page 3.)

[48] S. Xu and Y. and W. Susilo. Online/offline signatures and multisignatures for AODV and DSR routing security. In *Australasian Conference on Information Security and Privacy*, 2006. (Cited on page 5.)

[49] M. Zhao, S. Smith, and D. Nicol. Aggregated path authentication for efficient BGP security. In *ACM CCS*, 2005. (Cited on page 3, 5, 6, 13, 16.)

A An "Enhanced" Security Model for IBSAS

In the proceedings version of this paper, we incorrectly claimed a proof that our IBSAS construction (i.e. Construction 4.3) additionally meets an "enhanced" notion of security for such schemes. This enhanced definition can be formulated using the following experiment.

Definition A.1 Let $\mathsf{AS} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Sign}, \mathsf{Vf})$ be an IBSAS scheme as defined in Section 4. An "enhanced" experiment with a forger F with access to two oracles, is as follows:

Setup: The experiment generates a master key-pair (mpk, msk) by running Setup and gives mpk to F.

Attack: F then runs on input mpk with access to two oracles $\mathsf{KeyDer}(msk, \cdot)$ and $\mathcal{O}_{\mathsf{Sign}}(\cdot, \cdot, \cdot)$, the first of which operates according to the definition of IBSAS. The second on inputs a list of "current" signer-message pairs $((ID_1, m_1), \ldots, (ID_{i-1}, m_{i-1}))$, a list of signer-message pairs "to-add" $((ID_i, m_i), \ldots, (ID_k, m_k))$, and an aggregate-so-far σ executes:
1. $\sigma' \leftarrow \sigma$
2. For $j = i$ to k do: $\sigma' \xleftarrow{\$} \mathsf{Sign}(sk_{ID_j}, m_j, ((ID_1, m_1), \ldots, (ID_{j-1}, m_{j-1}), \sigma')$
3. Return σ'

Forgery: Eventually, F halts with outputs a list of pairs of users and messages $L^* = ((ID_1^*, m_1^*), \ldots, (ID_n^*, m_n^*))$ and a purported aggregate signature σ^*. This output is considered to be a forgery if (1) $\mathsf{Vf}(L^*, \sigma^*) \Rightarrow 1$ and (2) there does not exist an $i^* \in \{1, \ldots, n\}$ such that a valid signature for $((ID_1^*, m_1^*), \ldots, (ID_{i^*}^*, m_i^*))$ was returned to F by its signing oracle and all of $ID_{i^*+1}^*, \ldots, ID_n^*$ were queried by F to its key-derivation oracle. (By convention we assume that a valid signature for the empty list was always returned to F by its signing oracle.)

A particular "attack" that the enhanced definition considers as a forgery is "sub-aggregate extraction:" for example, after being given a valid aggregate signature corresponding to the identity-message list $((ID_1, m_1), (ID_2, m_2))$ with uncorrupted identities ID_1, ID_2, an adversary should not be able to then "extract" an individual signature by ID_1 on m_1 (even if ID_2, but of course not ID_1, is later corrupted). Note that this is not met by a "trivial" aggregate signature construction of concatenating individual signatures. We are not aware of any concrete application of the enhanced definition to secure routing. However, we conjecture our IBSAS construction to meet it, although we are unable to prove it based on the M-LRSW problem we define.

B Proof of Theorem 3.6

We construct a "simulator" B that on inputs $p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, g^a, g^b$, runs F to solve the CDH.

THE SIMULATOR. For simplicity, we assume F always queries messages to its hash oracle prior to using them in its signing queries and its final output, and that F never repeats a hash query. The description of the simulator B for the proof is given in Figure 1. In responding to hash queries by F, using Coron's technique [20] we have B assign query m a bit (aka. δ-value) $\delta[m]$ equal to 1 with probability δ, for some value of $0 \le \delta \le 1$ that we optimize later. Intuitively, B hopes that F never queries a message with δ-value 0 to its signing oracle where the honest signer is at the k^*-th position in the OMS, but that F's forgery contains such a message and that the position of the honest signer in the forgery is k^*.

ANALYSIS. We first need the following claim.

Claim B.1 On executions of B on which it does not abort, F's view (consisting of its input and answers it receives to its oracle queries) in the simulation provided by B comes from an identical distribution to that in its real UF-OMS experiment.

Proof: We first note that, as in the proof of [32, Theorem 3.1], the "signature reconstruction" technique used by B to answer F's signing queries provides responses coming from the same distribution as in the UF-OMS experiment because the OMS-so-far is verified and re-randomized in the signing algorithm, and the distribution of the resulting OMS is independent of the order in which

Simulator $B(p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, g^a, g^b)$
$k^* \xleftarrow{\$} \{1, \ldots, n_{\max}\}$; $t, u \xleftarrow{\$} \mathbb{Z}_p$
Initialize arrays $\mathrm{K}, \delta, \mathrm{H}, \mathrm{E}$ to everywhere undefined
Run F on inputs $(p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, \mathrm{H}), (g^a, (g^a)^{-uk^*} g^t, (g^a)^u)$:
  On key registration query (pk', sk', c):
  If $\mathsf{OKg}(I; c) \Rightarrow (pk', sk')$ then
    $\mathrm{K}[pk'] \leftarrow sk'$; Return 1
  Else return 0
  On hash query m:
  $\mathrm{E}[m] \xleftarrow{\$} \mathbb{Z}_p$; $\delta[m] \xleftarrow{\delta} \{0, 1\}$
  If $\delta[m] = 1$ then $\mathrm{H}[m] \leftarrow g^{\mathrm{E}[m]}$; Else $\mathrm{H}[m] \leftarrow g^b g^{\mathrm{E}[m]}$
  Return $\mathrm{H}[m]$
  On signing query $(m_j, \sigma, L = (pk_1, \ldots, pk_{i-1}))$:
  Parse σ as (Q, R) and set it as $(1_{\mathbb{G}}, 1_{\mathbb{G}})$ if $i = 1$
  If $i > 1$ and $\mathsf{OVf}(m_j, \sigma, L) \Rightarrow 0$ or $\exists z \in \{1, \ldots, i-1\}$ such that $\mathrm{K}[pk_z]$ is undefined
    then return $\perp$
  If $\delta[m_j] = 0$ and $i = k^*$ then abort
  $r \xleftarrow{\$} \mathbb{Z}_p$; Let $\mathrm{K}[pk_z] = (s_z, t_z, u_z)$ for all $z \in \{1, \ldots, i-1\}$
  If $\delta[m_j] = 1$ then {
    $Q' \leftarrow (g^a)^{\mathrm{E}[m_j]}((g^a)^{-uk^*} g^t (g^a)^{iu})^r$; $R' \leftarrow g^r$
    $Q'' \leftarrow Q' \cdot \prod_{k=1}^{j-1} (g^{s_k})^{\mathrm{E}[m_j]} g^{(t_k + k u_k) r}$ }
  Else {
    $Q' \leftarrow (g^a)^{\mathrm{E}[m_j]} (g^b)^{-t/(u(i-k^*))}((g^a)^{-uk^*} g^t (g^a)^{iu})^r$; $R' \leftarrow g^r (g^b)^{1/(u(k^*-i))}$
    $Q'' \leftarrow Q' \cdot \prod_{k=1}^{j-1} (g^b g^{\mathrm{E}[m_j]})^{s_k} (g^r (g^b)^{-1/(u(i-k^*))})^{(t_k + k u_k)}$ }
  Return (Q'', R')
Let $(L^* = (pk_1^*, \ldots, pk_n^*), m^*, \sigma^* = (Q^*, R^*))$ be the output of F
If L^*, σ^* is not a forgery (relative to H-values) then return $\perp$
Let $i^* \in \{1, \ldots, n\}$ satisfy condition (2) of a forgery
If $\delta[m^*] = 1$ or $i^* \neq k^*$ then abort
Let $\mathrm{K}[pk_z^*] = (s_z, t_z, u_z)$ for all $z \in \{1, \ldots, n\}$ such that $z \neq i^*$
$Q \leftarrow Q^* / (\prod_{j \neq i^*} (g^b g^{\mathrm{E}[m^*]})^{s_j} (R^*)^{t_j + j u_j})$; $Z \leftarrow Q / ((R^*)^t (g^a)^{\mathrm{E}[m^*]})$
Return Z

Figure 1: Simulator B for the proof of Theorem 3.6.

---

the individual components of each signers were actually combined (which is why we require verifying the OMS-so-far in the signing algorithm anyway in Definition 3.1). Moreover, in the case that F makes a signing query m such that $\delta[m] = 0$ but the position of the honest signer in the OMS is some $i \neq k^*$, our code for B first uses a trick of Boneh and Boyen [7] to first create the honest signer's individual component in the OMS with the correct distribution from F's perspective. To see this, we can write the honest signer's component (Q', R') in this case as

$$\begin{aligned} Q' &= (g^a)^{\mathrm{E}[m_j]}(g^b)^{-t/(u(i-k^*))}((g^a)^{-uk^*}g^t(g^a)^{iu})^r \\ &= (g^a)^{\mathrm{E}[m_j]}g^{-bt/(u(i-k^*))}(g^{(i-k^*)au}g^t)^r \\ &= (g^a)^{\mathrm{E}[m_j]}g^{ab}(g^{(i-k^*)au}g^t)^{r-b/(u(i-k^*))} \\ &= (g^b g^{\mathrm{E}[m_j]})^a(g^{(i-k^*)au}g^t)^{r-b/(u(i-k^*))}\ ; \\ R' &= g^r(g^b)^{-1/(u(i-k^*))} \\ &= g^{r-b/(u(i-k^*))}\ , \end{aligned}$$

which, given that $r \in \mathbb{Z}_p$ is chosen randomly by B, indeed comes from the same distribution from the perspective of F as the response given to it in its real experiment. Now, it is straightforward to verify that when B does not abort, it provides F with a view coming from an identical distribution to that in its real experiment during the rest of the simulation as well. ▌

Now based on the code for B, conditions (1) and (3) of a forgery in Definition 3.2, and by using the bilinearity and non-degeneracy properties of a pairing, we have that on run of B on which it does not abort and on which F produces a forgery with signature (Q^*, R^*), where $R^* = g^{r^*}$ for some $r^* \in \mathbb{Z}_p$ unknown to B, B's output can be written as

$$\begin{aligned} g^{ab}g^{\mathrm{E}[m_j]a}((g^a)^{-uk^*}g^t(g^a)^{k^*u})^{r^*}/((g^a)^{\mathrm{E}[m_j]}(g^{r^*})^t) &= g^{ab}g^{\mathrm{E}[m_j]a}((g^a)^{k^*-k^*}g^t)^{r^*}/((g^a)^{\mathrm{E}[m_j]}(g^{r^*})^t) \\ &= g^{ab}g^{\mathrm{E}[m_j]a}(g^t)^{r^*}/((g^a)^{\mathrm{E}[m_j]}(g^{r^*})^t) \\ &= g^{ab}\ , \end{aligned}$$

which is a solution to its input CDH problem instance. In light of the above claim, then, we can wlog consider runs of the CDH game played by B and of the UF-OMS experiment with F using randomly-chosen coin sequences drawn from a common finite space of coins, where if F produces a forgery on a run of its experiment using some chosen sequence of coins from this space then, on a run of B using the same coins, the latter provides input and oracle replies to F identical to that its real UF-OMS experiment and hence outputs a solution to its CDH instance if it does not abort. Let forge be the event that F produces a forgery when run in its UF-OMS experiment. Let abort be the probability that B aborts. Then we have the probability $\mathbf{Adv}^{\mathrm{CDH}}_{\mathcal{G}}(B)$ that B succeeds in solving the CDH relative to $\mathcal{G}$ is bounded as follows:

$$\begin{aligned} \mathbf{Adv}^{\mathrm{CDH}}_{\mathcal{G}}(B) &= \Pr\left[\,\mathsf{forge} \wedge \overline{\mathsf{abort}}\,\right] \\ &= \Pr\left[\,\overline{\mathsf{abort}} \mid \mathsf{forge}\,\right] \cdot \Pr\left[\,\mathsf{forge}\,\right] \\ &= \Pr\left[\,\overline{\mathsf{abort}} \mid \mathsf{forge}\,\right] \cdot \mathbf{Adv}^{\mathrm{UF\text{-}OMS}}_{\mathsf{OMS}}(F)\ . \end{aligned}$$

Note that the probabilities above are taken over the choice of the coin sequence drawn from the common finite space, and that the last equality is by definition. To continue the analysis, we make the following claim.

Claim B.2 We claim that

$$\Pr\left[\,\overline{\mathsf{abort}} \mid \mathsf{forge}\,\right] \ \geq\ (1/n_{\max}) \cdot (1-\delta) \cdot \delta^{q_{\mathrm{s}}}\ .$$

Proof: Note that the probability that B aborts seems difficult to analyze directly, because F may repeat messages in its queries to its signing oracle, which have the same δ-values. Instead, we analyze it by looking at a run of B and of the UF-OMS experiment with F using the same coin sequence drawn from the common finite space of coins, on which F outputs a forgery on a message m^* in the latter. Let abort_s be the event that B aborts when responding to a signing query by F and abort_f be the event that B aborts after F outputs a forgery. Let goodf be the event that B sets $\delta[m^*] = 0$ and $k^* = i^*$, where i^* is the position of the honest signer in forgery that F outputs in its experiment when run on the same coin sequence. Then we have

$$\begin{aligned}
\Pr\left[\,\overline{\mathsf{abort}} \mid \mathsf{forge}\,\right] &= \Pr\left[\,\overline{\mathsf{abort}_s} \wedge \overline{\mathsf{abort}_f} \mid \mathsf{forge}\,\right] \\
&= \Pr\left[\,\overline{\mathsf{abort}_s} \wedge \overline{\mathsf{abort}_f} \wedge \mathsf{goodf} \mid \mathsf{forge}\,\right] \\
&= \Pr\left[\,\overline{\mathsf{abort}_f} \mid \overline{\mathsf{abort}_s} \wedge \mathsf{goodf} \wedge \mathsf{forge}\,\right] \cdot \Pr\left[\,\overline{\mathsf{abort}_s} \wedge \mathsf{goodf} \mid \mathsf{forge}\,\right] \\
&= \Pr\left[\,\overline{\mathsf{abort}_s} \wedge \mathsf{goodf} \mid \mathsf{forge}\,\right] \; . \qquad (2)
\end{aligned}$$

For the second equality above, we use the fact that $\overline{\mathsf{abort}_f}$ occurs only if goodf does, by definition of B. For the last, we use that $\Pr\left[\,\overline{\mathsf{abort}_f} \mid \overline{\mathsf{abort}_s} \wedge \mathsf{goodf} \wedge \mathsf{forge}\,\right] = 1$, because abort_f cannot occur if goodf does. Now, consider the set of distinct messages $S = \{m_1, \ldots, m_k\}$ that F queries to its signing oracle when executed in its experiment. We assume wlog that $m^* = m_k$, where m^* is the message on which F forges. Let badq_j be the event that F when executed by B makes a signing query of the form $m_j, \sigma, L = (pk_1, \ldots, pk_{k^*-1})$ for which the reply is not $\perp$. We next claim the following sequence of inequalities:

$$\begin{aligned}
\Pr\left[\,\overline{\mathsf{abort}_s} \wedge \mathsf{goodf} \mid \mathsf{forge}\,\right] &= \Pr\left[\mathsf{goodf} \wedge \bigwedge_{j=1}^{k} \delta[m_j] = 1 \vee (\delta[m_j] = 0 \wedge \overline{\mathsf{badq}_j}) \mid \mathsf{forge}\right] \\
&\geq \Pr\left[\mathsf{goodf} \wedge (\delta[m_k] = 0 \wedge \overline{\mathsf{badq}_k}) \bigwedge_{j=1}^{k-1} \delta[m_j] = 1 \mid \mathsf{forge}\right] \\
&\geq \Pr\left[\mathsf{goodf} \wedge \bigwedge_{j=1}^{k-1} \delta[m_j] = 1 \mid \mathsf{forge}\right] \\
&\geq \frac{1}{n_{\max}} \cdot (1-\delta) \cdot \delta^{k-1} \\
&\geq \frac{1}{n_{\max}} \cdot (1-\delta) \cdot \delta^{q_{\mathrm{s}}} \; .
\end{aligned}$$

Above, the first equation is just by the definition of B. The next follows by dropping events in disjunctions. For the third, we use the fact that if goodf occurs and B sets all δ-values except $\delta[m^*]$ to 1, then bad_k (where $m^* = m_k$) cannot occur. This is because, by condition (4) in the definition of a forgery, F does not make a signing query of the form $m^*, \sigma, L = (pk_1, \ldots, pk_{i^*-1})$ for any $\sigma \in \{0,1\}^*$ on the run of its experiment and hence on the run of B, because the latter provides the same input and oracle replies to F as the former in this case when run on the same coins. The fourth follows from the fact that all B always sets k^* and any δ-values independently of each other and of F. Finally, the last uses $1 \leq k \leq q_{\mathrm{s}}$ and $0 \leq \delta \leq 1$. Substituting the last inequality into equation (2) above proves the claim. ▌

Using the above claim, we now have

$$\mathbf{Adv}_{\mathcal{G}}^{\mathrm{CDH}}(B) \;\geq\; (1/n_{\max}) \cdot (1-\delta) \cdot \delta^{q_{\mathrm{s}}} \cdot \mathbf{Adv}_{\mathsf{OMS}}^{\mathrm{UF\text{-}OMS}}(F) \; . \qquad (3)$$

To finish the analysis, let us define for $0 \leq \delta \leq 1$ the function

$$f(\delta) \;\stackrel{\mathrm{def}}{=}\; \delta^{q_{\mathrm{s}}} \cdot (1-\delta) \; .$$

It is not hard to see that f is maximized at $\delta_{\mathrm{OPT}} = q_{\mathrm{s}}/(q_{\mathrm{s}}+1)$, at which $f(\delta_{\mathrm{OPT}}) \geq 1/(e(q_{\mathrm{s}}+1))$. Setting δ to δ_{OPT} in the description of B and substituting the above for $f(\delta)$ in equation (3), then re-arranging terms, gives equation (1) in Theorem 3.6.

Finally, to justify the running-time analysis of B, we take into account our convention to include in the running-time of F that of its overlying experiment. B's extra work is one additional exponentiation on each hash query F makes, as well as an additional number of exponentiations linear in the number signers in the OMS (of which there are at most $n_{\max}$) on each signing query and on a forgery. This takes time at most $\tau(\mathcal{G}) \cdot O(q_{\mathrm{h}} + n_{\max}(q_{\mathrm{s}}+1))$, as asserted. ▌

C Proof of Theorem 4.5

We construct a "simulator" B that runs F in order to solve the M-LRSW. Recall that B gets input $p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, u, v, g^a, g^b$, as well as access to an associated oracle $\mathcal{O}^{\text{M-LRSW}}_{g,u,v,a,b}(\cdot)$ that on input $k \in \mathbb{Z}_p$ executes

$$\begin{array}{l} \text{If } k = 0 \text{ then return } \perp \\ r \stackrel{\$}{\leftarrow} \mathbb{Z}_p \\ \text{Return } (u^{kr} g^{ab}, v^r g^{ab}, g^r). \end{array}$$

B wants to output $(k', u^{k'x} g^{ab}, v^x g^{ab}, g^x)$ for some $k' \in \mathbb{Z}_p$ it does not query to its oracle and any $x \in \mathbb{Z}_p$ of its choice.

The simulator. We assume wlog that F never repeats a query to its H_1 and H_2 hash oracles. Under some further simplifying assumptions on F given below, the description of B is given in Figure 2. In responding to F's queries to its H_1-oracle, using Coron's technique [20] we have B assign H_1-query ID a bit (aka. δ-value) $\delta[ID]$ equal to 1 with some probability $0 \leq \delta \leq 1$ that we optimize later. Moreover, in B's code, we assume for simplicity that when F makes a signing query $ID_i, m_i, ((ID_1, m_1), \ldots, (ID_{i-1}, m_{i-1})), \sigma$, it has previously queried identity ID_k to its H_1-oracle for all $k \in \{1, \ldots, j\}$; similarly, on F's final output $L^* = ((ID_1^*, m_1^*), \ldots, (ID_n^*, m_n^*)), \sigma^*$, we assume wlog it has queried identity ID_k to its H_1-oracle for all $k \in \{1, \ldots, n\}$ and string s_j for all $j \in \{1, \ldots, n\}$ to its H_2-oracle. Intuitively, B hopes that F does not ask for the secret key of any ID with $\delta[ID] = 0$, but that exactly one such identity occurs in its forgery.

Analysis. For the analysis, let $\mathsf{collide}$ be the event that B outputs (ρ, X, Y, Z) such that it has previously queried ρ to its M-LRSW oracle, let forge be the event that F produces a forgery according to Definition 4.2 when executed by B, and let abort be the event that B aborts. (We emphasize that forge is defined with respect to an execution of the simulator B here, unlike in the proof of Theorem 3.6 in Appendix B.) We claim that the probability $\mathbf{Adv}^{\text{M-LRSW}}_{\mathcal{G}}(B)$ that B succeeds in solving the M-LRSW relative to $\mathcal{G}$ is bounded as follows:

$$\begin{aligned} \mathbf{Adv}^{\text{M-LRSW}}_{\mathcal{G}}(B) &\geq \Pr\left[\,\mathsf{forge} \wedge \overline{\mathsf{collide}} \wedge \overline{\mathsf{abort}}\,\right] \\ &= \Pr\left[\,\mathsf{forge} \wedge \overline{\mathsf{collide}} \mid \overline{\mathsf{abort}}\,\right] \cdot \Pr\left[\,\overline{\mathsf{abort}}\,\right] \\ &= \Pr\left[\,\mathsf{forge} \setminus \mathsf{collide} \mid \overline{\mathsf{abort}}\,\right] \cdot \Pr\left[\,\overline{\mathsf{abort}}\,\right] \\ &= \left(\Pr\left[\,\mathsf{forge} \mid \overline{\mathsf{abort}}\,\right] - \Pr\left[\,\mathsf{collide} \mid \overline{\mathsf{abort}}\,\right]\right) \cdot \Pr\left[\,\overline{\mathsf{abort}}\,\right] . \end{aligned} \tag{4}$$

To see the first inequality above, consider a run of B in which it does not abort, and suppose F's output $L^* = ((ID_1^*, m_1^*), \ldots, (ID_n^*, m_n^*)), \sigma^*$ when executed by B is a forgery. Let $i^* \in \{1, \ldots, n\}$ satisfy condition (2) of a forgery. Based on the description of B, condition (1) of a forgery, and by

Simulator $B^{\mathcal{O}^{\text{M-LRSW}}_{g,u,v,a,b}}(p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, u, v, g^a, g^b)$
Initialize arrays $\mathrm{E}, \delta, \mathrm{H}_1, \mathrm{H}_2$ to everywhere undefined
Run F on input $p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, u, v, g, g^a, \mathrm{H}_1, \mathrm{H}_2$:
  On H_1-query ID:
  $\mathrm{E}[ID] \xleftarrow{\$} \mathbb{Z}_p$; $\delta[ID] \xleftarrow{\delta} \{0,1\}$
  If $\delta[ID] = 1$ then $\mathrm{H}_1[ID] \leftarrow g^{\mathrm{E}[ID]}$; Else $\mathrm{H}_1[ID] \leftarrow g^b g^{\mathrm{E}[ID]}$
  Return $\mathrm{H}_1[ID]$
  On H_2-query x:
  $\mathrm{H}_2[x] \xleftarrow{\$} \mathbb{Z}_p^*$; Return $\mathrm{H}_2[x]$
  On signing query $(ID_i, m_i, ((ID_1, m_1), \ldots, (ID_{i-1}, m_{i-1})), \sigma)$:
  Parse σ as (X, Y, Z) and set it as $(1_{\mathbb{G}}, 1_{\mathbb{G}}, 1_{\mathbb{G}})$ if $i = 1$
  $\pi \leftarrow \prod_{j=1}^{i} \mathrm{H}_2[s_j]$
  If $\delta[ID_i] = 0$ then {
    $(X', Y', Z') \xleftarrow{\$} \mathcal{O}^{\text{M-LRSW}}_{g,u,v,a,b}(\pi)$
    $X' \leftarrow X' \cdot (g^a)^{\mathrm{E}[ID_i]}$; $Y' \leftarrow Y' \cdot (g^a)^{\mathrm{E}[ID_i]}$
    $X \leftarrow X \cdot (X')$; $Y \leftarrow Y^{1/\mathrm{H}_2[s_i]} \cdot Y'$; $Z \leftarrow Z^{1/\mathrm{H}_2[s_i]} \cdot Z'$ }
  Else {
    $r \xleftarrow{\$} \mathbb{Z}_p$; $X \leftarrow X \cdot u^{\pi r} (g^a)^{\mathrm{E}[ID_i]}$
    $Y \leftarrow Y^{1/\mathrm{H}_2[s_i]} \cdot v^r (g^a)^{\mathrm{E}[ID_i]}$
    $Z \leftarrow Z^{1/\mathrm{H}_2[s_i]} \cdot g^r$ }
  Return (X, Y, Z)
  On key-derivation query ID:
  If $\delta[ID] = 0$ then abort ; Else return $(g^a)^{\mathrm{E}[ID]}$
Let $(L^* = ((ID_1^*, m_1^*), \ldots, (ID_n^*, m_n^*)), \sigma^*)$ be the output of F
If L^*, σ^* is not a forgery (relative to $\mathrm{H}_1, \mathrm{H}_2$-values) then return $\perp$
Let $i^* \in \{1, \ldots, n\}$ satisfy condition (2) of a forgery
If $\delta[ID_{i^*}^*] = 1$ or there exists $\delta[ID_{i'}] = 0$ for some $1 \le i' \ne i^* \le n$ then abort
Parse σ^* as (X, Y, Z) and let s_j^* denote $ID_1^* \| m_1^* \| \ldots \| ID_j^* \| m_j^*$ for all $1 \le j \le n$
For all $1 \le j \ne i^* \le n$ do
  $X \leftarrow X/(g^a)^{\mathrm{E}[ID_j]}$; $Y \leftarrow Y/(g^a)^{\mathrm{E}[ID_j]/(\prod_{l=j+1}^{n} \mathrm{H}_2[s_l^*])}$
$X \leftarrow X/(g^a)^{\mathrm{E}[ID_{i^*}]}$; $Y \leftarrow Y^{(\prod_{l=i^*+1}^{n} \mathrm{H}_2[s_l^*])}/(g^a)^{\mathrm{E}[ID_{i^*}]}$; $Z \leftarrow Z^{(\prod_{l=i^*+1}^{n} \mathrm{H}_2[s_l^*])}$
$\rho \leftarrow \prod_{l=1}^{i^*} \mathrm{H}_2[s_l^*]$; Return (ρ, X, Y, Z)

Figure 2: Simulator B for the proof of Theorem 4.5.

---

using the bilinearity and non-degeneracy properties of a pairing, we have that B's output can be written as

$$\left(\rho,\ \prod_{j=1}^{n} u^{z \prod_{l=1}^{j} \mathrm{H}_2[s_l^*]} \cdot g^{ab},\ \prod_{j=1}^{n} v^{z \prod_{l=i^*+1}^{n} \mathrm{H}_2[s_l^*] / \prod_{l=j+1}^{n} \mathrm{H}_2[s_l^*]} \cdot g^{ab},\ \prod_{j=1}^{n} g^{z \prod_{l=i^*+1}^{n} \mathrm{H}_2[s_l^*] / \prod_{l=j+1}^{n} \mathrm{H}_2[s_l^*]} \right) ,$$

for some $z \in \mathbb{Z}_p$ unknown to B, where $\rho = \prod_{j=1}^{i^*} \mathrm{H}_2[s_j^*]$. Now, letting

$$x \;=\; z \sum_{j=1}^{n} \cdot \left(\prod_{l=i^*+1}^{n} \mathrm{H}_2[s_l^*] \;/\; \prod_{l=j+1}^{n} \mathrm{H}_2[s_l^*] \right) \;,$$

we have the equality

$$x\rho \;=\; x \prod_{j=1}^{i^*} \mathrm{H}_2[s_l^*] \;=\; z \sum_{j=1}^{n} \cdot \prod_{l=1}^{j} \mathrm{H}_2[s_l^*] \;.$$

Notice that the last term is equal to the exponent on u in the second component of B's output above. So, the output is a solution to the M-LRSW (meaning causes the M-LRSW game to output 1) if B did not previously query ρ to its M-LRSW oracle, i.e. $\mathsf{collide}$ did not occur. This gives us the first inequality. To lower-bound the last line, we use the following claims.

Claim C.1 We claim that

$$\Pr\left[\,\overline{\mathsf{abort}}\,\right] \;\geq\; \delta^{q_{\mathrm{k}}}(1-\delta)\delta^{n_{\max}-1} \;.$$

Proof: Let abort_k be the event that F makes a key-derivation query ID such that $\delta[ID] = 0$. Let abort_f be the event that B aborts after F outputs a forgery, meaning $((ID_1^*, m_1^*), \ldots, (ID_n^*, m_n^*))$, where $i^* \in \{1, \ldots, n\}$ satisfies the condition (2) of a forgery in Definition 4.2, has either $\delta[ID_{i^*}^*] = 1$ or $\delta[ID_j^*] = 0$ for some $1 \leq j \neq i^* \leq n$. Let $t = \#j \in \{1, \ldots, n\}$ such that F queried ID_j^* to its key-derivation oracle. Then we claim that

$$\begin{aligned}
\Pr\left[\,\overline{\mathsf{abort}}\,\right] &= \Pr\left[\,\overline{\mathsf{abort}_k} \;\wedge\; \overline{\mathsf{abort}_f}\,\right] \\
&= \Pr\left[\,\overline{\mathsf{abort}_k}\,\right] \cdot \Pr\left[\,\overline{\mathsf{abort}_f} \mid \overline{\mathsf{abort}_k}\,\right] \\
&\geq \delta^{q_{\mathrm{k}}} \cdot \Pr\left[\,\overline{\mathsf{abort}_f} \mid \overline{\mathsf{abort}_k}\,\right] \\
&= \delta^{q_{\mathrm{k}}} \cdot (1-\delta) \cdot 1^t \cdot \delta^{n-1-t} \\
&\geq \delta^{q_{\mathrm{k}}} \cdot (1-\delta) \cdot \delta^{n_{\max}-1} \;.
\end{aligned}$$

To see the third line above, note that when F makes a key-derivation query ID it has no information about $\delta[ID]$, which is set to 1 with probability δ (and recall q_{k} is an upper-bound on the number of key-derivation queries F makes). The fourth line similarly follows from the fact that F has no information about $\delta[ID_{i^*}^*]$, because it did not query ID_{i^*} to its key-derivation oracle. Moreover, consider ID_j^* for each $1 \leq j \neq i^* \leq n$. If F did not query ID_j^* to its key-derivation oracle, then it has no information about $\delta[ID_j^*]$, whereas if F did query ID_j^* to its key-derivation oracle then it knows $\delta[ID_j^*] = 1$ (because if $\delta_j = 0$ then B would have aborted already on key-derivation query ID_j^*). The last line uses $0 \leq \delta \leq 1$ and $1 \leq n-1 \leq n_{\max} - 1$. ▌

Claim C.2 We claim that

$$\Pr\left[\,\mathsf{forge} \mid \overline{\mathsf{abort}}\,\right] \;=\; \mathbf{Adv}_{\mathsf{AS}}^{\text{UF-IBSAS}}(F) \;.$$

Proof: To see this, we can consider runs of the M-LRSW game played by B and of the UF-IBSAS experiment with F using randomly-chosen coin sequences drawn from a common finite space of coins. Imagine a new game where we first run B using a randomly-chosen coin sequence from this space and then run the UF-IBSAS experiment with F using the same coins. Note that on executions of B where it does not abort, F's view in the simulation comes from a distribution identical to that in its real experiment. So, the probability of forge given that abort does not occur is the same as the probability that F outputs a forgery when run in the new game given that B did not abort when run in this game, which is clearly just $\mathbf{Adv}_{\mathsf{AS}}^{\text{UF-IBSAS}}(F)$ as desired. ▌

Claim C.3 We claim that

$$\Pr\left[\,\mathsf{collide} \mid \overline{\mathsf{abort}}\,\right] \;\le\; \frac{q_{\mathrm{h}_2}(q_{\mathrm{h}_2}-1)}{2p}\,.$$

Proof: Looking at B's code, note that collide occurs just when B does not abort and F outputs a forgery, but a value $\pi = \prod_{l=1}^{j} \mathrm{H}_2[s_l]$ queried by B to its M-LRSW oracle is such that $\pi = \rho = \prod_{l=1}^{i^*} \mathrm{H}_2[s_l^*]$, for some sequence $(s_1,\ldots,s_j) \neq (s_1^*,\ldots,s_{i^*}^*)$ (i.e. $s_1\|\cdots\|s_j \neq s_1^*\|\cdots\|s_{i^*}^*$) defined relative to one of F's signing queries. (The non-equality here is due to the fact that by condition (2) of a forgery we know that F did not make a query $(ID_i^*, m_i^*, ((ID_1^*, m_1^*),\ldots,(ID_{i^*-1}^*, m_{i^*-1}^*)), \sigma')$ for any $\sigma' \in \{0,1\}^*$ to its signing oracle.)

Call a sequence $(q_1,\ldots,q_k)$ for $k \geq 1$ where each q_i is some query made by F to its H_2-oracle *valid* if for all $1 \leq i \leq k$ there is a string $a_i = b_i\|c_i$ (where b_i is an identity and c_i is a message) such that $q_i = q_{i-1}\|a_i$, where a_0 is the empty string. In particular, $(s_1^*,\ldots,s_{i^*}^*)$ and $(s_1,\ldots,s_j)$ above must be valid. So if we think of $\prod_{l=1}^{k} \mathrm{H}_2[s_l]$ as the hash value of $(s_1,\ldots,s_k)$, the condition that all hash values of valid sequences are different implies collide does not occur. Notice that two valid sequences cannot be "mixed" to create a new valid sequence; namely, a sequence $(s_1,\ldots,s_{i-1},s_i',s_{i+1},\ldots,s_k)$ from two valid sequences $(s_1,s_2,\ldots,s_k)$ and $(s_1',s_2',\ldots,s_k')$ is not valid unless the condition $s_i' = s_i$ is met (which also implies $s_j' = s_j$ for $1 \leq j \leq i$). So each query by F to its H_2-oracle introduces at most one new valid sequence. Thus, there are at most q_{h_2} valid sequences in total and the probability of collide is upper-bounded by the birthday bound:

$$\begin{aligned}
\Pr\left[\,\mathsf{collide} \mid \overline{\mathsf{abort}}\,\right] &\le \Pr[\mathsf{C}_1 \vee \mathsf{C}_2 \vee \cdots \vee \mathsf{C}_{q_{\mathrm{h}_2}}] \\
&\le \Pr[\mathsf{C}_1] + \Pr[\mathsf{C}_2] + \cdots + \Pr[\mathsf{C}_{q_{\mathrm{h}_2}}] \\
&\le \frac{0}{p} + \frac{1}{p} + \cdots + \frac{q_{\mathrm{h}_2}-1}{p} \\
&= \frac{1 + 2 + 3 + \cdots + (q_{\mathrm{h}_2}-1)}{p} \\
&= \frac{q_{\mathrm{h}_2}(q_{\mathrm{h}_2}-1)}{2p}\,,
\end{aligned}$$

where C_i is the event that the hash value of the i-th valid sequence defined during execution of F collides with one of the previous ones. ▌

Plugging the previous claims into equation (4), we now have

$$\mathbf{Adv}_{\mathcal{G}}^{\text{M-LRSW}}(B) \;\ge\; \left(\mathbf{Adv}_{\mathsf{AS}}^{\text{UF-IBSAS}}(F) - \frac{q_{\mathrm{h}_2}(q_{\mathrm{h}_2}-1)}{2p}\right)\cdot \delta^{q_{\mathrm{k}}}(1-\delta)\delta^{n_{\max}-1}\,. \tag{5}$$

To complete the analysis, let us define the function

$$\begin{aligned}
f(\delta) &\stackrel{\text{def}}{=} \delta^{q_{\mathrm{k}}}(1-\delta)\delta^{n_{\max}-1} \\
&= \delta^{n_{\max}+q_{\mathrm{k}}-1}(1-\delta)\,.
\end{aligned}$$

Denoting the exponent $n_{\max} + q_{\rm k} - 1$ by z, it is not hard to see that f is maximized at $\delta_{\rm OPT} = z/(z+1)$, for which we have $f(\delta_{\rm OPT}) \geq 1/(e(z+1)) = 1/(e(n_{\max}+q_{\rm k}))$. Setting δ to $\delta_{\rm OPT}$ in the code for B and substituting the above for $f(\delta)$ in (5) then re-arranging the inequality yields equation (2) in Theorem 4.5.

Finally, to justify our running-time analysis of B, recall our convention to include in the running-time of F that of its overlying experiment. On answering H_1 and key-derivation queries by F, B's overhead is at most one exponentiation in $\mathbb{G}$. On each signing query, B's overhead is at most a constant number of exponentiations in $\mathbb{G}$. After a forgery, B's overhead is at most a linear number of exponentiations in $\mathbb{G}$ and (by re-using computation appropriately) a linear number of multiplications in $\mathbb{Z}_p$ in the number of signers in the forged aggregate signature, of which there are at most $n_{\max}$. Summing, this takes time at most $\tau(\mathcal{G}) \cdot O(q_{\rm h_1} + q_{\rm k} + q_{\rm s} + n_{\max}) + \upsilon(\mathcal{G}) \cdot O(n_{\max})$. Moreover, the number of oracle queries made by B to its M-LRSW oracle is at most $q_{\rm s}$, as asserted. ▌

D Proof of Theorem 5.1

We construct a "simulator" B that on input $(p, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ runs A and tries to simulate for it its corresponding generic bilinear group experiment while behaving a bit differently. But B does not do so in order to solve any computational problem itself; rather, we construct B such that we are able to bound the advantage of A in solving the M-LRSW in its generic bilinear group experiment by bounding the success probability of A in solving the M-LRSW when executed by B.

The simulator. Consider the simulator B given in Figure 3, which runs A. Intuitively, B internally represents (the discrete logs of) group elements in $\mathbb{G}, \mathbb{G}_T$ as multivariate (actually, multilinear) polynomials over $\mathbb{Z}_p$, in indeterminates corresponding to what are randomly-chosen elements of $\mathbb{Z}_p$ in the generic bilinear group experiment with A. Analogously to in the latter, these polynomials are encoded as random strings (stored in arrays ξ, ξ_T) before being given to A. We assume wlog that all such encodings in A's oracle queries and in its final output are "legitimate" encodings of elements in the appropriate group, where by legitimate we mean that they were previously received by A either as part of its input or in response to one of its previous queries. As a consequence, in B's code we write components of F's oracle queries and its output like $\xi[h]$ for some polynomial h, with it being understood that B can find h by scanning the array for $\xi[h]$.

Analysis. We first make the following claim.

Claim D.1 On runs of B on which it does set flag bad, A's view in the simulation provided by B comes a distribution identical to that in its generic bilinear group experiment.

Proof: We show that when B does not set bad, the encodings of group elements given to A in the simulation provided by B come from an identical distribution as in the generic bilinear group experiment with A. Here we re-name the encoding functions in the generic bilinear group experiment with A as ξ', ξ'_T. Consider the map $\phi\colon \mathbb{Z}_p[X, Y, R_u, R_v, R_1, \ldots, R_{q_{\rm s}}] \to \mathbb{Z}_p$ given by $q \to q(x, y, r_u, r_v, r_1, \ldots, r_{q_{\rm s}})$ for all such polynomials q, which in particular maps polynomials encoded by B via its arrays ξ, ξ_T to integers in $\mathbb{Z}_p$.

In the case that B does not set bad, we can view ϕ as mapping the polynomials encoded by B to the discrete logs of group elements encoded in the generic bilinear group experiment with A, in the sense that their images under ϕ take the same distribution as the latter. To see this, first observe that the image of each indeterminate under ϕ takes the same distribution as the corresponding discrete logs in $\mathbb{Z}_p$ randomly chosen by the generic bilinear group experiment with A. So, when B performs addition (on answering A's group operation queries in $\mathbb{G}, \mathbb{G}_T$) and multiplication (on

Simulator $B(\mathbb{G}, \mathbb{G}_T, \mathbf{e}, g)$:
Maintain lists L, L_T of ordered pairs $(h, \xi[h])$,
  where $h \in \mathbb{Z}_p[X, Y, R_u, R_v, R_1, \ldots, R_{q_s}]$ and $\xi[h] \in \{0,1\}^{\lceil \log p \rceil}$
Initialize arrays ξ, ξ_T to everywhere undefined
$ctr \leftarrow 0$; $\xi[1], \xi[R_u], \xi[R_v], \xi[X], \xi[Y] \xleftarrow{\$} \{0,1\}^{\lceil \log p \rceil}$
Initialize L as $((1, \xi[1]), (R_u, \xi[R_u]), (R_v, \xi[R_v]), (X, \xi[X]), (Y, \xi[Y]))$ (and L_T as empty)
Run A on input $p, \xi[1], \xi[R_u], \xi[R_v], \xi[X], \xi[Y]$:
  On $\mathbb{G}$-operation query $(\xi[a_1], \xi[a_2], b)$: /* $a_1, a_2 \in \mathbb{Z}_p[X, Y, R_u, R_v, R_1, \ldots, R_{q_s}]$ */
  $f \leftarrow a_1 + (-1)^b a_2$
  If $(f, \xi[f]) \in L$ for some $\xi[f] \in \{0,1\}^{\lceil \log p \rceil}$ then return $\xi[f]$
  Else $\xi[f] \xleftarrow{\$} \{0,1\}^{\lceil \log p \rceil}$; Add $(f, \xi[f])$ to L ; Return $\xi[f]$
  On $\mathbb{G}_T$-operation query $(\xi_T[a_1], \xi_T[a_2], b)$: /* $a_1, a_2 \in \mathbb{Z}_p[X, Y, R_u, R_v, R_1, \ldots, R_{q_s}]$ */
  $f \leftarrow a_1 + (-1)^b a_2$
  If $(f, \xi_T[f]) \in L_T$ for some $\xi_T[f] \in \{0,1\}^{\lceil \log p \rceil}$ then return $\xi_T[f]$
  Else $\xi_{T,f} \xleftarrow{\$} \{0,1\}^{\lceil \log p \rceil}$; Add $(f, \xi_{T,f})$ to L_T ; Return $\xi_T[f]$
  On pairing query $(\xi[a_1], \xi[a_2])$: /* $a_1, a_2 \in \mathbb{Z}_p[X, Y, R_u, R_v, R_1, \ldots, R_{q_s}]$ */
  $f \leftarrow a_1 \cdot a_2$
  If $(f, \xi_T[f]) \in L_T$ for some $\xi_T[f] \in \{0,1\}^{\lceil \log p \rceil}$ then return $\xi_T[f]$
  Else $\xi_T[f] \xleftarrow{\$} \{0,1\}^{\lceil \log p \rceil}$; Add $(f, \xi_T[f])$ to L_T ; Return $\xi[g]$
  On $\mathcal{O}^{\textbf{M-LRSW}}_{g,u,v,x,y}$ query α: /* $\alpha \in \mathbb{Z}_p$ */
  If $\alpha = 0$ return $\perp$
  $ctr \leftarrow ctr + 1$; $f_1 \leftarrow \alpha R_{ctr} R_u + XY$; $f_2 \leftarrow R_{ctr} R_v + XY$; $f_3 \leftarrow R_{ctr}$
  For $q = 1$ to 3 do
    If $(f_q, \xi[f_q]) \notin L$ for some $\xi[f_q] \in \{0,1\}^{\lceil \log p \rceil}$ then
      $\xi[f_q] \xleftarrow{\$} \{0,1\}^{\lceil \log p \rceil}$; Add $(f_q, \xi[f_q])$ to L
  Return $(\xi[f_1], \xi[f_2], \xi[f_3])$
Let $(\alpha^*, \xi[f_i], \xi[f_j], \xi[f_k])$ be the output of A
$x, y, r_u, r_v, r_1, \ldots, r_{q_s} \xleftarrow{\$} \mathbb{Z}_p$
If there exist $(f, \xi[f]), (f', \xi[f']) \in L$ or
$(f_T, \xi_T[f_T]), (f'_T, \xi_T[f'_T]) \in L_T$ such that
  $f(x, y, r_u, r_v, r_1, \ldots, r_{q_s}) = f'(x, y, r_u, r_v, r_1, \ldots, r_{q_s})$ but $\xi[f] \neq \xi[f']$
  or $f_T(x, y, r_u, r_v, r_1, \ldots, r_{q_s}) = f'_T(x, y, r_u, r_v, r_1, \ldots, r_{q_s})$ but $\xi_T[f_T] \neq \xi_T[f'_T]$ then $\mathsf{bad} \leftarrow \mathsf{true}$
$f'_j \leftarrow R_v f_k + XY$; $f'_i \leftarrow \alpha^* R_u f_k + XY$
If α^* was not queried to $\mathcal{O}^{\text{M-LRSW}}_{g,u,v,x,y}$ but
  $f_i(x, y, r_u, r_v, r_1, \ldots, r_{q_s}) = f'_i(x, y, r_u, r_v, r_1, \ldots, r_{q_s})$ and
  $f_j(x, y, r_u, r_v, r_1, \ldots, r_{q_s}) = f'_j(x, y, r_u, r_v, r_1, \ldots, r_{q_s})$ then return 1 ; Else return 0

Figure 3: Simulator B for the proof of Theorem 5.1.

---

answering A's pairing queries) of two polynomials, the real experiment could equivalently perform addition and multiplication of their images under ϕ. But ϕ is a ring homomorphism. So, when B provides A with an encoding $\xi[f+g]$ or $\xi_T[f \cdot g]$ for polynomials f, g, the real experiment could provide $\xi'(\phi(f) + \phi(g)) = \xi'(\phi(f+g))$, and similarly in the second case. Now if B does not set bad then ϕ is restricted here to polynomials on which it is injective, so in this case their images under

ϕ indeed take an identical distribution as the discrete logs encoded by the real experiment.

The claim now follows by the fact that values of ξ, ξ_T in the simulation and of ξ', ξ'_T in the real experiment both take independent random strings in $\{0,1\}^{\lceil \log p \rceil}$. ▮

Now consider executions of B and the generic bilinear group experiment with A using randomly-chosen coin sequences drawn from a common finite space. Let BAD be the event that B sets flag bad, $\mathsf{success}$ be the event that A when executed in its generic bilinear group experiment outputs a solution to the M-LRSW problem (i.e. causing its experiment to return 1) and, as usual, $B \Rightarrow 1$ be the event that B outputs 1. We claim that the probability $\mathbf{Adv}_{\mathcal{G}}^{\text{M-LRSW}}(A)$ that A solves the M-LRSW problem in its generic bilinear group experiment is bounded as follows:

$$\begin{aligned}
\mathbf{Adv}_{\mathcal{G}}^{\text{M-LRSW}}(A) &\leq \Pr\left[\,\mathsf{success} \wedge \overline{\mathsf{BAD}}\,\right] + \Pr[\,\mathsf{success} \wedge \mathsf{BAD}\,] \\
&\leq \Pr\left[\,\mathsf{success} \mid \overline{\mathsf{BAD}}\,\right] \cdot \Pr\left[\,\overline{\mathsf{BAD}}\,\right] + \Pr[\,\mathsf{BAD}\,] \\
&= \Pr\left[\,B \Rightarrow 1 \mid \overline{\mathsf{BAD}}\,\right] \cdot \Pr\left[\,\overline{\mathsf{BAD}}\,\right] + \Pr[\,\mathsf{BAD}\,] \\
&= \Pr\left[\,B \Rightarrow 1 \wedge \overline{\mathsf{BAD}}\,\right] + \Pr[\,\mathsf{BAD}\,] \\
&\leq \Pr[\,B \Rightarrow 1\,] + \Pr[\,\mathsf{BAD}\,]\,. \qquad (6)
\end{aligned}$$

Note that the probabilities here are taken over the choice of the coin sequence used both for execution of B and the generic bilinear group experiment with A. The third line follows from the above claim and observing that on execution of B for which it does not set bad, B returns 1 just when A would solve the M-LRSW on the corresponding run of its real experiment given by the map ϕ in the above proof. So it remains to bound the probabilities in the last inequality. We do so via the following two claims. But first we need to recall the following fact, which follows from the Schwartz-Zippel Lemma [42].

Fact D.2 Fix a non-zero polynomial $f \in \mathbb{Z}_p[X_1, \ldots, X_k]$ of total degree d. Then the probability that $f(x_1, \ldots x_k) = 0$ when $x_1, \ldots, x_k \in \mathbb{Z}_p$ are chosen independently at random is at most d/p.

Note that a polynomial equality $g_1 = g_2$ can be rewritten as $g_1 - g_2 = 0$. Thus, when evaluating g_1, g_2 at a randomly chosen point, the probability that the former equality holds is the same as the latter, so, assuming $g_1 \neq g_2$, we can apply above fact to bound the probability of the former, with $g_1 - g_2$ playing the role of f. Moreover, in this case the total degree of $g_1 - g_2$ is bounded by the maximum of the total degree of either. We use this observation repeatedly below.

Claim D.3 The probability of BAD is bounded as

$$\Pr[\,\mathsf{BAD}\,] \leq \frac{4\binom{t+3q_{\mathrm{s}}+5}{2}}{p}\,.$$

Proof: Notice that, during execution of B, polynomials initially present in L and those later added by B as a result of queries made by A to its M-LRSW oracle have total degree at most 2. Moreover, A's making a group operation query in $\mathbb{G}$ causes B to add a linear combination of such polynomials to L, so in fact this bound applies to all polynomials in L. On the other hand, if A makes a pairing query then B multiplies two polynomials in L and puts the result, which has total degree at most 4, into L_T. A's making a group operation query in $\mathbb{G}_T$ similarly causes B to add to L_T only a linear combination of polynomials already in L_T, hence all polynomials in L_T have total degree at most 4.

Now, at the end of B's execution there are at most $\gamma + 3q_{\mathrm{s}} + 5$ polynomials in the two lists combined, where γ is the total number of queries A has made to its group operation oracles and its pairing oracles. We can now apply Fact D.2 to the first "If" check in the condition for setting bad, namely $f(x, y, r_u, r_v, r_1, \ldots, r_{q_s}) = f'(x, y, r_u, r_v, r_1, \ldots, r_{q_s})$ where $f \neq f'$, and similarly to the second "If" check. Namely, an instance of the former holds with probability at most $2/p$, and the latter with at most $4/p$. Taking the maximum here and a union bound over all pairs of polynomials in the lists combined yields the claim. ▮

Claim D.4 We claim that

$$\Pr[\, B \Rightarrow 1\,] \;\leq\; \frac{3}{p}\,.$$

Proof: Consider first the equality $f_j(x, y, r_u, r_v, r_1, \ldots, r_{q_s}) = f'_j(x, y, r_u, r_v, r_1, \ldots, r_{q_s})$ in B's code, which must hold in order for B to return 1. Initially, let us suppose that $f_j \neq f'_j$. Then, since $f'_j = R_v f_k + XY$ by construction where f_k is in L, and we have previously seen that all polynomials in L have total degree at most two, f'_j has total degree at most 3. So by Fact D.2 the above equality holds with probability at most $3/p$, and the claim follows.

Thus, for the remainder of the proof, we assume that $f_j = f'_j$. We next want to show that $f'_i \neq f_i$. We first claim that the polynomial f_k can then be written as a sum $R_l + \beta$, for some $l \in \{1, \ldots, q_s\}$ and some $\beta \in \mathbb{Z}_p$ (we assume for simplicity that $q_s \geq 1$). This follows from the fact that the polynomial $f'_j = R_v f_k + XY$ must be in the list L (because f_j is, and we are assuming $f_j = f'_j$), as follows. Notice from the code for B that the set of polynomials containing $1, R_u, R_v, X, Y$ as well as $\alpha_c R_u R_c + XY, R_v R_c + XY, R_c$ for all $c \in \{1, \ldots, q_s\}$, where $\alpha_c \in \mathbb{Z}_p$ is the first component of the c-th query made by A to its M-LRSW oracle, forms a basis for the vector space of polynomials over $\mathbb{Z}_p$ spanned by polynomials in L, and L only contains polynomials in this span. In particular, the only basis polynomials containing R_v are $R_v R_c + XY$ for all $c \in \{1, \ldots, q_s\}$ and R_v itself. One can check from this that having f_k of the above form, so that we may write $f'_j = R_v R_l + \beta R_v + XY$, is the only way that f'_j can be in this span and hence in L.

Now, this means f'_i has the form $\alpha^* R_u(R_l + \beta) + XY = \beta R_u + \alpha^* R_u R_l + XY$ for some $\alpha^* \in \mathbb{Z}_p$ that was not queried by A to its M-LRSW oracle. Considering again the basis given above for the vector space of polynomials spanned by polynomials in L, we see that no such polynomial can be in L, because if it were then $(\beta R_u + \alpha^* R_u R_l + XY) - \beta R_u = \alpha^* R_u R_l + XY$ would be in the span, which it is not because f'_i is not a scalar multiple of $\alpha_l R_u R_l + XY$ (using the fact that $\alpha^* \neq \alpha_l$, since α_l was queried by A to its oracle and α^* was not), and no other basis polynomial contains an $R_u R_l$ term. Therefore $f'_i \neq f_i$ as desired, since f_i is in L. Thus f_i has total degree at most 2, and since $f'_i = \alpha^* R_u f_k + XY$ by construction where f_k is in L, f'_i has total degree at most 3. So $f_i(x, y, r_u, r_v, r_1, \ldots, r_{q_s}) = f'_i(x, y, r_u, r_v, r_1, \ldots, r_{q_s})$, which must hold for B to output 1, holds with probability at most $3/p$ by Fact D.2, and the claim follows. ▮

Finally, plugging the previous two claims into (6) above and then combining terms gives equation (2) in Theorem 5.1 as desired. ▮