Unrestricted Aggregate Signatures

Mihir Bellare[1] Chanathip Namprempre[2] Gregory Neven[3]

June 2006

Abstract

Secure use of the BGLS [7] aggregate signature schemes is restricted to the aggregation of distinct messages (for the basic scheme) or per-signer distinct messages (for the enhanced, prepend-public-key version of the scheme). We argue that these restrictions preclude interesting applications, make usage of the schemes error-prone and are generally undesirable in practice. Via a new analysis and proof, we show how the restrictions can be lifted, yielding the first truly unrestricted aggregate signature scheme. Via another new analysis and proof, we show that the distinct signer restriction on the sequential aggregate signature schemes of [15] can also be dropped, yielding an unrestricted sequential aggregate signature scheme.

[1] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: `mihir@cs.ucsd.edu`. URL: `http://www-cse.ucsd.edu/users/mihir`.

[2] Electrical Engineering Department, Thammasat University, Rangsit Campus, Klong Luang, Patumtani, Thailand 12121. Email: `nchanath@engr.tu.ac.th`. URL: `http://www.engr.tu.ac.th/~nchanath`.

[3] Department of Electrical Engineering, Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, B-3001 Heverlee-Leuven, Belgium and Département d'Informatique, Ecole Normale Supérieure, 45 rue d'Ulm, 75005 Paris, France. Email: `Gregory.Neven@esat.kuleuven.be`. URL: `http://www.neven.org`.

1 Introduction

AGGREGATE SIGNATURES. An aggregate signature (AS) scheme [7] is a digital signature scheme with the additional property that a sequence $\sigma_1, \ldots, \sigma_n$ of individual signatures —here σ_i is the signature, under the underlying base signature scheme, of some message m_i under some public key pk_i— can be condensed into a single, compact aggregate signature σ that simultaneously validates the fact that m_i has been signed under pk_i for all $i = 1, \ldots, n$. There is a corresponding aggregate verification process that takes input $(pk_1, m_1), \ldots, (pk_n, m_n), \sigma$ and accepts or rejects. Aggregation is useful to reduce bandwidth and storage, and is especially attractive for mobile devices like sensors, cell phones, and PDAs where communication is more power-expensive than computation and contributes significantly to reducing battery life.

SCHEMES. Boneh et. al. [7] present an aggregate signature scheme based on the BLS signature scheme [9]. We call it $\mathcal{AS}$-*1* and represent it succinctly in the first row of Table 1. $\mathcal{AS}$-*1*, however, has some limitations. As the table shows, the aggregate verification process, on inputs $(pk_1, m_1), \ldots, (pk_n, m_n), \sigma$, rejects if the messages $m_1, \ldots, m_n$ are not distinct. The restriction is crucial because, without it, as shown in [7], the scheme is subject to a forgery attack. The consequence, however, is to restrict the ability to aggregate to settings where the messages signed by the signers are all different. BGLS recognize this limitation and suggest a workaround. Specifically, they say: "It is easy to ensure the messages are distinct: The signer simply prepends her public key to every message she signs ..." [7, Section 3.2]. They stop short of specifying a scheme in full, but since it is clearly their intention to "reduce to the previous case," our understanding is that they are advocating the modified scheme in which the signature of a message m under pk is the BLS signature of the *enhanced message* $M = pk\|m$ under pk while aggregate verification is done by applying the aggregate verification procedure of $\mathcal{AS}$-*1* to $(pk_1, pk_1\|m_1), \ldots, (pk_n, pk_n\|m_n), \sigma$. However, if so, in this scheme, which we call $\mathcal{AS}$-*2*, the aggregate verification process will reject unless the enhanced messages $pk_1\|m_1, \ldots, pk_n\|m_n$ are distinct. (Why? Because the aggregate verification process of $\mathcal{AS}$-*1* rejects unless the messages are distinct, and the role of the messages is now played by the enhanced messages.) The consequence is that the ability to aggregate is restricted to settings where the enhanced messages signed by the signers are all different. That is, the limitations have been reduced, but not eliminated.

OUR RESULT. We ask whether there exists a truly unrestricted proven-secure aggregate signature scheme. Namely, there should be no distinctness-based restriction of any kind, whether on messages or enhanced messages. We show that the answer is yes. Our result is a new, direct analysis of the security of enhanced-message signature aggregation which shows that the distinctness condition in the aggregate verification process of $\mathcal{AS}$-*2* —namely that this process rejects if any two enhanced messages are the same— can be dropped without compromising security. In other words, an unrestricted scheme can be obtained by the natural adaptation of $\mathcal{AS}$-*2* in which the distinctness condition in the verification is simply removed and all else is the same. This scheme, which we denote $\mathcal{AS}$-*3*, is summarized in the last row of Table 1. The fact that $\mathcal{AS}$-*3* is very close to $\mathcal{AS}$-*2* is a plus because it means existing implementations can be easily patched.

We clarify that the security of $\mathcal{AS}$-*3* is not proved in [7]. They prove secure only $\mathcal{AS}$-*1*. The security of $\mathcal{AS}$-*2* is a consequence, but the security of $\mathcal{AS}$-*3* is not. What we do instead is to *directly* analyze security in the case that signatures are on enhanced messages. Our analysis explicitly uses and exploits the presence of the prepended public keys to obtain the stronger conclusion that $\mathcal{AS}$-*3* (not just $\mathcal{AS}$-*2*) is secure.

MOTIVATION. The limitation of $\mathcal{AS}$-*2* —namely that aggregation is restricted to settings where no two enhanced messages are the same— may seem minor, because all it says is that a set of signatures to be aggregated should not contain duplicates, meaning multiple signatures by a particular signer of a particular messages. However, as we now explain, there are in fact several motivations for schemes like $\mathcal{AS}$-*3* where aggregation is possible even in the presence of duplicates.

Consider a network of sensor nodes, periodically recording temperatures in a nuclear reactor or voltage levels in a power grid and transmitting the information to a center. Ensuring integrity is crucial (if an attacker

Scheme	**Sign**	**Aggregate verification process accepts iff**
$\mathcal{AS}$-*1* [7]	$\mathrm{H}(m)^x$	$\mathbf{e}(\sigma, g) = \prod_{i=1}^n \mathbf{e}(\mathrm{H}(m_i), g^{x_i})$ and $m_1, \ldots, m_n$ all distinct
$\mathcal{AS}$-*2* [7]	$\mathrm{H}(g^x \| m)^x$	$\mathbf{e}(\sigma, g) = \prod_{i=1}^n \mathbf{e}(\mathrm{H}(g^{x_i} \| m_i), g^{x_i})$ and $g^{x_1} \| m_1, \ldots, g^{x_n} \| m_n$ all distinct
$\mathcal{AS}$-*3*	$\mathrm{H}(g^x \| m)^x$	$\mathbf{e}(\sigma, g) = \prod_{i=1}^n \mathbf{e}(\mathrm{H}(g^{x_i} \| m_i), g^{x_i})$

Table 1: The aggregate signature schemes we discuss. Here $\mathbf{e}\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map, g is a generator of $\mathbb{G}_2$ known to all parties, and $\mathrm{H}\colon \{0,1\}^* \to \mathbb{G}_1$ is a hash function. The second column shows the signature of a message m under public key g^x, generated using secret key x. In all cases, a sequence of signatures is aggregated by simply multiplying them in $\mathbb{G}_1$. The third column shows under what conditions the aggregate verification algorithm accepts σ as a valid aggregate signature of messages $m_1, \ldots, m_n$ under public keys $g^{x_1}, \ldots, g^{x_n}$ respectively.

---

can tamper with the transmitted information it can trigger a false alarm, or, even worse, prevent a true emergency situation from being detected) so each node signs its transmissions. These are aggregated by the center to save storage. (The schemes we are discussing permit on-line aggregation in which one can maintain a current aggregate and aggregate into it a received signature.) However, information such as temperatures or voltage levels can certainly repeat over time! Indeed, especially in stable conditions, we would expect frequent repeats. So we would expect to see a single signer (sensor) signing the same data (measurement) many times. In general, this can happen any time the data being signed is drawn from a small space. In any such situation, not just messages, but even enhanced messages can repeat and an unrestricted aggregate signature scheme is necessary.

Perhaps an even more important reason to prefer unrestricted schemes is that they are less likely to be misused and less likely to result in unexpected errors. An application designer contemplating using $\mathcal{AS}$-*2* must ask herself whether, in her application, enhanced messages might repeat. This may not only be hard to know in advance, but might also change with time. (Experience has repeatedly shown that once a piece of cryptography is deployed, it is used for purposes or applications beyond those originally envisaged.) With an unrestricted scheme, the application designer is freed from the need to worry about whether the setting of her application meets the restrictions, reducing the chance of error. In general, application designers and programmers have a hard enough time as it is to make error-free use of cryptography. Asking them to understand message distinctness restrictions and anticipate whether their application meets them is an added burden and one more place where things can go wrong.

POSSIBLE WORKAROUNDS. Various ways to get around message distinctness restrictions may come to mind, but these workarounds either do not work or tend to be more complex or restrictive than direct use of an unrestricted scheme. For example, one could have the verifier drop from its input list $(pk_1, m_1), \ldots, (pk_n, m_n)$, σ any pair pk_i, m_i that occured previously in the list, but then, assuming σ was a correct aggregate signature of the given list, it will not be a correct aggregate signature for the truncated list, and verification will fail. Another possibility is that the aggregator refrain from including in the aggregate any signature corresponding to a public key and message whose signature is already in the aggregate. But aggregation may be done on-line, and the aggregator may know only the current aggregate and the incoming signature, and have no way to tell whether or not it is a duplicate. Adding time-stamps or other nonces is another possibility, but complicates the application. It seems clear that being able to use $\mathcal{AS}$-*3* without any worries about signature or message replication is simpler, easier and more practical.

RESULTS FOR SASs. A sequential aggregate signature (SAS) scheme [15] permits a more restrictive kind of aggregation in which the signers must participate in the process and use their secret keys. Imagine the signers forming a chain. In step i, the i-th signer receives from the previous one a current aggregate, and, using its secret key, it aggregates into this its own signature, passing the new aggregate on to the next signer in the chain. The output of the final signer is the aggregate signature. Although clearly less powerful than general aggregate signature (GAS) schemes —following [8] we now use this term for the BGLS-type aggregate signatures discussed above in order to distinguish them from SASs— the argument of [15] is that sequential aggregation is possible for base signature schemes other than BLS or may be done more cheaply. Specifically, Lysyanskaya et al. [15] present a SAS scheme based on certified [5] claw-free trapdoor permutations.

However, the model and schemes of [15] also have some limitations. They require that no public key can appear more than once in a chain. That is, the signers who are part of a signing chain must be distinct. But in practice there are many reasons to prefer to allow aggregation even when a particular signer signs multiple messages, meaning when there are loops in the chain and public keys can repeat. Certainly, the previously discussed motivations are still true. Namely, in applications like signing in sensor nets or ad hoc networks, a particular signer (sensor) will be producing many signatures and it would be convenient to allow these to be aggregated. More importantly, an unrestricted SAS scheme is more misuse resistant because it does not require the application designer to try to predict in advance whether or not there will be repeated signers in a prospective chain. But in fact the restrictions in [15] are even greater than the ones for $\mathcal{AS}$-2, for they say not only that one cannot aggregate when a particular signer signs a particular message twice, but that one cannot aggregate even when a particular signer signs two or more messages that are distinct. This makes the number of excluded applications even larger, for it is common that a particular signer needs to sign multiple messages.

Our result —analogous to the GAS case— is a new analysis and proof that shows that the restrictions imposed by [15] on their schemes can be lifted without impacting security (that is, verification can just drop the condition that one reject if there are repeated public keys, and security is preserved), yielding unrestricted schemes. Again, that only a minor modification to the scheme is needed is convenient if existing implementations need to be updated.

RELATED WORK. Lu et. al. [14] present an SAS scheme for which they can lift the distinct signer restriction, as follows. If a signer wishes to add its signature of a message M_{new} to an aggregate S which already contains its signature S_{old} on some message M_{old}, then it removes S_{old} from S and then adds back in a signature on the message $M_{\text{new}}\|M_{\text{old}}$. However, their scheme is weaker than the others we have discussed —ours or those of [7, 15]— with regard to some security properties and also with regard to efficiency. Specifically, [14] use the certified public key model [1, 6], which reflects the assumption that signers provide strong ZK proofs of knowledge of secret keys to the CA at key-registration, an assumption that we, following [7, 15], do not make. Also, in the scheme of [14], public keys are very large, namely 162 group elements, which is particularly expensive if public keys have to be transmitted along with the signatures. In contrast, other schemes have short public keys. On the other hand, the proofs of [14] are in the standard model, while ours, following [7, 15], are in the random oracle model of [3].

Interestingly, in a survey paper, Boneh et. al. [8] present $\mathcal{AS}$-3, claiming that the results of [7] prove it secure. However, this appears to be an oversight because, as we have seen, the results of [7] prove $\mathcal{AS}$-2 secure, not $\mathcal{AS}$-3. By proving $\mathcal{AS}$-3 secure via our direct analysis, we are filling the gap and validating the claim of [8]. Shacham's Ph.D thesis [16] notes that the concrete security of the reduction of [7] can be slightly improved by replacing messages with enhanced ones, but he does not claim security of $\mathcal{AS}$-3.

2 Notation and Basic Definitions

NOTATION AND CONVENTIONS. If x is a string, then $|x|$ is the length of x. We denote by $x_1\|\cdots\|x_n$ an encoding of objects $x_1, \ldots, x_n$ as a binary string from which the constituent objects are uniquely recoverable. Since, in most cases, the objects can themselves be encoded as strings whose length is known from the context, simple concatenation will serve the purpose. If S is a finite set, then $|S|$ is its size, and $s \xleftarrow{\$} S$ means that s is chosen at random from S. We let e denote the base of the natural logarithm. An algorithm may be randomized unless otherwise indicated. An adversary is an algorithm. If A is an algorithm then $y \xleftarrow{\$} A(x_1, x_2, \ldots)$ means that y is the result of executing A on fresh random coins and inputs $x_1, x_2, \ldots$. We denote by $[A(x_1, x_2, \ldots)]$ the set of all possible outputs of A on the indicated inputs, meaning all strings that have a positive probability of being output by A on inputs $x_1, x_2, \ldots$. We let $\mathsf{Maps}(D)$ denote the set of all functions with domain $\{0,1\}^*$ and range D.

DIGITAL SIGNATURE SCHEMES. We recall definitions for (standard) signature schemes [12] in the random-oracle (RO) model [3]. A signature scheme $\mathcal{DS} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$ is specified by three algorithms, the last deterministic. Via $(pk, sk) \xleftarrow{\$} \mathsf{Kg}$, a signer generates its public key pk and matching secret key sk, where $\mathrm{H}\colon \{0,1\}^* \to D$ is a random oracle whose range D is a parameter of the scheme. Via $\sigma \xleftarrow{\$} \mathsf{Sign}^{\mathrm{H}}(sk, m)$ the signer can generate a signature σ on a message m. A verifier can run $\mathsf{Vf}^{\mathrm{H}}(pk, m, \sigma)$ to obtain a bit, with 1 indicating accept and 0 reject. The consistency (or correctness) condition is that

$$\Pr\left[\,\mathsf{Vf}^{\mathrm{H}}(pk, m, \sigma) = 1 \;\mid\; (pk, sk) \xleftarrow{\$} \mathsf{Kg}\,;\; \mathrm{H} \xleftarrow{\$} \mathsf{Maps}(D)\,;\; \sigma \xleftarrow{\$} \mathsf{Sign}^{\mathrm{H}}(sk, m)\,\right] \;=\; 1$$

for all messages $m \in \{0,1\}^*$. To capture security (unforgeability under chosen-message attack) we define the advantage of an adversary B as

$$\mathbf{Adv}^{\mathrm{uf\text{-}cma}}_{\mathcal{DS}}(\mathsf{B}) \;=\; \Pr\left[\,\mathsf{Vf}^{\mathrm{H}}(pk, m, \sigma) = 1 \;\mid\; (pk, sk) \xleftarrow{\$} \mathsf{Kg}\,;\; \mathrm{H} \xleftarrow{\$} \mathsf{Maps}(D)\,;\; (m, \sigma) \xleftarrow{\$} \mathsf{B}^{\mathsf{Sign}^{\mathrm{H}}(sk,\cdot),\mathrm{H}}\,\right] \,.$$

To make this meaningful, we only consider adversaries that are *legitimate* in the sense that they never queried the message in their output to their sign oracle. We say that $\mathcal{DS}$ is $(t, q_{\mathrm{S}}, q_{\mathrm{H}}, \epsilon)$-secure if no adversary B running in time at most t, invoking the signature oracle at most q_{S} times and the random oracle at most q_{H} times, has advantage more than ϵ.

3 Unrestricted General Aggregate Signatures

GAS SCHEMES. A general aggregate signature (GAS) scheme [7] $\mathcal{AS} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Agg}, \mathsf{AVf})$ consists of four algorithms, the last deterministic. The key generation and signing algorithms are exactly as for standard digital signatures. Via $\sigma \xleftarrow{\$} \mathsf{Agg}^{\mathrm{H}}((pk_1, m_1, \sigma_1), \ldots, (pk_n, m_n, \sigma_n))$, anyone can aggregate a sequence of public key, message, and signature triples to yield an aggregate signature. A verifier can run $\mathsf{AVf}^{\mathrm{H}}((pk_1, m_1), \ldots, (pk_n, m_n), \sigma)$ to obtain a bit, with 1 indicating accept and 0 reject.

SECURITY. The security requirement of [7] is strong, namely that an adversary find it computationally infeasible to produce an aggregate forgery involving an honest signer, even when it can play the role of all other signers, in particular choosing their public keys, and can mount a chosen-message attack on the honest signer. To capture this, we define the advantage of an adversary A as

$$\mathbf{Adv}^{\mathrm{agg\text{-}uf}}_{\mathcal{AS}}(\mathsf{A}) \;=\; \Pr\left[\,\mathsf{AVf}^{\mathrm{H}}((pk_1, m_1), \ldots, (pk_n, m_n), \sigma) = 1\,\right]$$

where the probability is over the experiment

$$(pk, sk) \xleftarrow{\$} \mathsf{Kg}\,;\; \mathrm{H} \xleftarrow{\$} \mathsf{Maps}(D)\,;\; ((pk_1, m_1), \ldots, (pk_n, m_n), \sigma) \xleftarrow{\$} \mathsf{A}^{\mathsf{Sign}^{\mathrm{H}}(sk,\cdot),\mathrm{H}}(pk)\,.$$

To make this meaningful, we only consider adversaries that are *legitimate* in the sense that pk_1 must equal pk but A cannot have queried m_1 to its sign oracle. Thus, the honest signer here is the one whose keys are pk, sk, and we are asking that the aggregate forgery include some message and signature corresponding to this honest signer, but the adversary never legitimately obtained a signature of this message. We say that A $(t, q_{\mathrm{S}}, n_{\max}, q_{\mathrm{H}}, \epsilon)$-breaks $\mathcal{AS}$ if it runs in time at most t, invokes the signature oracle at most q_{S} times, invokes the hash oracle at most q_{H} times, outputs a forgery containing at most $n_{\max}$ public-key-message pairs, and has advantage strictly greater than ϵ. We say that $\mathcal{AS}$ is $(t, q_{\mathrm{S}}, n_{\max}, q_{\mathrm{H}}, \epsilon)$-secure if there is no adversary that

$(t, q_S, n_{\max}, q_H, \epsilon)$-breaks $\mathcal{AS}$.

A significant feature of this definition, highlighted in [7], is that A can choose $pk_2, \ldots, pk_n$ as it wishes, in particular as a function of $pk_1 = pk$. Unlike [1, 6, 14], there is no requirement that the adversary "know" the secret key corresponding to a public key it produces, and this makes the system more practical since it avoids the need for strong zero-knowledge proofs of knowledge [2] of secret keys done to the CA during key-registration. Our results continue to achieve this strong notion of security.

BILINEAR MAPS AND COCDH. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be groups, all of the same prime order p. Let $\mathbf{e}$: $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a non-degenerate, efficiently computable bilinear map, also called a pairing. Let g be a generator of $\mathbb{G}_2$. For the rest of this section, we regard $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g$ as fixed, globally known parameters, and also let $t_{\exp}$ denote the time to perform an exponentiation in $\mathbb{G}_1$. Note that following [9, 7] we use the asymmetric setting ($\mathbb{G}_1, \mathbb{G}_2$ are not necessarily equal) and must also assume there exists an isomorphism ψ: $\mathbb{G}_2 \to \mathbb{G}_1$. (The first is in order to make signatures as short as possible, and the second is required for the security proofs.) We define the advantage of an adversary A in solving the coCDH problem as

$$\mathbf{Adv}^{\text{co-cdh}}(\mathsf{A}) \;=\; \Pr\left[\, \mathsf{A}(g, g^a, h) = h^a \;\;|\;\; h \xleftarrow{\$} \mathbb{G}_1 \,;\, a \xleftarrow{\$} \mathbb{Z}_p \,\right] \;.$$

We say that the coCDH problem is (t', ϵ')-hard if no algorithm A running in time at most t' has advantage strictly more than ϵ' in solving it. Note that when $\mathbb{G}_1 = \mathbb{G}_2$, the coCDH problem becomes the standard CDH problem in $\mathbb{G}_1$, whence the name.

THE $\mathcal{BLS}$ SCHEME. We recall the $\mathcal{BLS}$ standard signature scheme of [9]. The signer chooses a secret key $x \xleftarrow{\$} \mathbb{Z}_p$ and computes the corresponding public key $X \leftarrow g^x$. Let $\mathrm{H} : \{0,1\}^* \to \mathbb{G}_1$ be a random oracle. The signature of message m is $\sigma = \mathrm{H}(m)^x$, which can be verified by checking that $\mathbf{e}(\sigma, g) = \mathbf{e}(\mathrm{H}(m), X)$. Regarding security, we have the following:

Theorem 3.1 **[9]** *If the coCDH problem is (t', ϵ')-hard, then the $\mathcal{BLS}$ standard signature scheme is (t, q_S, q_H, ϵ)-secure for any t, q_S, q_H, ϵ satisfying*

$$\epsilon \;\geq\; e(q_S + 1) \cdot \epsilon' \quad \textit{and} \quad t \;\leq\; t' - t_{\exp}(q_H + 2q_S) \;. \blacksquare$$

THE GAS SCHEMES WE CONSIDER. We consider four closely related aggregate signature schemes that we denote $\mathcal{AS}$-*0*, $\mathcal{AS}$-*1*, $\mathcal{AS}$-*2*, $\mathcal{AS}$-*3*. These schemes share common key generation and aggregation algorithms, but differ in their signing and verification algorithms. All use a random oracle $\mathrm{H} : \{0,1\}^* \to \mathbb{G}_1$. Key generation is exactly as in the $\mathcal{BLS}$ scheme: the secret key is an exponent $x \xleftarrow{\$} \mathbb{Z}_p$ and the corresponding public key is $X = g^x$. For $\mathcal{AS}$-*0* and $\mathcal{AS}$-*1*, the signing algorithm is the $\mathcal{BLS}$ one, namely the signature on message m is $\sigma = \mathrm{H}(m)^x$. For $\mathcal{AS}$-*2* and $\mathcal{AS}$-*3*, a signature on m under public key X is $\sigma = \mathrm{H}(X \| m)^x$. For all schemes, aggregation is done by simply multiplying the signatures, i.e. $\sigma = \prod_{i=1}^{n} \sigma_i$ in $\mathbb{G}_1$. Verification is different for each scheme. On inputs $(X_1, m_1), \ldots, (X_n, m_n), \sigma$, the verification algorithm of $\mathcal{AS}$-*0* accepts iff $\mathbf{e}(\sigma, g) = \prod_{i=1}^{n} \mathbf{e}(\mathrm{H}(m_i), X_i)$. The verification algorithms of the other schemes are depicted in Table 1. In particular, $\mathcal{AS}$-*1* rejects if $m_1, \ldots, m_n$ are not all distinct, $\mathcal{AS}$-*2* rejects if $X_1 \| m_1, \ldots, X_n \| m_n$ are not all distinct, while $\mathcal{AS}$-*3* performs no such checks.

CONSISTENCY CONDITIONS. The consistency condition (under what conditions correctly generated aggregates are accepted by the verifier) differs from scheme to scheme, and is in fact the place where the restrictions they make surface in a formal sense. $\mathcal{AS}$-*0* and $\mathcal{AS}$-*3* meet the natural, strongest possible consistency requirement, namely that

$$\Pr\left[\, \mathsf{AVf}^{\mathrm{H}}((pk_1, m_1), \ldots, (pk_n, m_n), \sigma) = 1 \,\right] \;=\; 1$$

for all positive integers n, all messages $m_1, \ldots, m_n \in \{0,1\}^*$ and all $(pk_1, sk_1), \ldots, (pk_n, sk_n) \in [\mathsf{Kg}]$, where the probability is over the experiment $\mathrm{H} \xleftarrow{\$} \mathsf{Maps}(D)$; $\sigma_1 \xleftarrow{\$} \mathsf{Sign}^{\mathrm{H}}(sk_1, m_1)$; $\cdots$; $\sigma_n \xleftarrow{\$} \mathsf{Sign}^{\mathrm{H}}(sk_n,$

$m_n)$; $\sigma \xleftarrow{\$} \mathsf{Agg}^{\mathrm{H}}((pk_1, m_1, \sigma_1), \ldots, (pk_n, m_n, \sigma_n))$. However, $\mathcal{AS}$*-1* meets this condition only when $m_1, \ldots, m_n$ are distinct and $\mathcal{AS}$*-2* when $pk_1 \| m_1, \ldots, pk_n \| m_n$ are distinct.

DISCUSSION OF SECURITY. An attack provided in [7] shows that $\mathcal{AS}$*-0* is insecure. In this attack, however, the forgery output by the adversary contains repeated messages. To exclude the attack, [7] define $\mathcal{AS}$*-1*, where the aggregate verification process rejects when messages repeat. They are able to show this suffices to guarantee security, meaning that they prove $\mathcal{AS}$*-1* is secure if the coCDH problem is hard. This is their main result. Then they suggest to alleviate the message-distinctness restriction of $\mathcal{AS}$*-1* by having each signer prepend its public key to the message before signing. However, they appear to want to argue the security of the resulting aggregate signature scheme as a corollary of their main result on the security of $\mathcal{AS}$*-1*. If so, verification still needs to check that $X_1 \| m_1, \ldots, X_n \| m_n$ are all distinct (otherwise, the result about $\mathcal{AS}$*-1* does not apply), leading to the $\mathcal{AS}$*-2* scheme.

As we have discussed, however, for practical reasons, $\mathcal{AS}$*-3* is a preferable scheme. But the results of [7] do not prove it secure. This is a subtle point because the difference in the schemes seems to be merely in the presence of duplicates, and duplicating a signature is not a forgery. (Anyone can copy a signature.) But that is not the issue. An example may help to see what the problem is. Suppose there was an adversary A that, on input $pk = X$ and without making oracle query m, produced a forgery of the form $(X, m), (X', m'), (X', m'), \sigma$, for some $m' \neq m$ and $X' \neq X$, that was accepted by the verification procedure of $\mathcal{AS}$*-3*. Since the output of A contains repeated enhanced messages, the results of [7] do not allow us to rule out the existence of A. Yet, security requires us to do so, because this adversary does forge a signature under X. (Note this signature is not a duplicate. The duplicates are under other keys.) Our result however does rule out such an adversary.

Theorem 3.2 *If the coCDH problem is (t', ϵ')-hard, then the $\mathcal{AS}$-3 aggregate signature scheme is $(t, q_{\mathrm{S}}, n_{\max}, q_{\mathrm{H}}, \epsilon)$-secure for any $t, q_{\mathrm{S}}, n_{\max}, q_{\mathrm{H}}, \epsilon$ satisfying*

$$\epsilon \;\geq\; e(q_{\mathrm{S}}+1)\cdot\epsilon' \quad \textit{and} \quad t \;\leq\; t' - t_{\exp}(2q_{\mathrm{H}} + 2q_{\mathrm{S}} + 3n_{\max} + 1) \;\; . \blacksquare$$

Our approach to the proof is different from the one used by [7] to prove that $\mathcal{AS}$*-1* is secure if coCDH is hard. They gave a direct reduction to coCDH, meaning, given an adversary attacking $\mathcal{AS}$*-1* they construct and analyze an adversary attacking coCDH. But, in so doing, they end up duplicating a lot of the proof of the security of the $\mathcal{BLS}$ scheme as given in [9]. Instead, we reduce the security of $\mathcal{AS}$*-3* to the security of $\mathcal{BLS}$. That is, we prove the following:

Lemma 3.3 *If the $\mathcal{BLS}$ standard signature scheme is $(t', q'_{\mathrm{S}}, q'_{\mathrm{H}}, \epsilon')$-secure then the $\mathcal{AS}$-3 aggregate signature scheme is $(t, q_{\mathrm{S}}, n_{\max}, q_{\mathrm{H}}, \epsilon)$-secure for any $t, q_{\mathrm{S}}, n_{\max}, q_{\mathrm{H}}, \epsilon$ satisfying*

$$\epsilon \;\geq\; \epsilon' \quad , \quad q_{\mathrm{S}} \leq q'_{\mathrm{S}} - n_{\max} \quad , \quad q_{\mathrm{H}} \leq q'_{\mathrm{H}} \quad \textit{and} \quad t \leq t' - t_{\exp}\cdot(q_{\mathrm{H}} + n_{\max} + 1) \; . \blacksquare$$

Theorem 3.2 follows easily from Lemma 3.3 and Theorem 3.1. Our modular approach yields a simple proof even though we obtain a somewhat stronger result.

An interesting element of the proof of Lemma 3.3 is that it involves reducing the security of one random oracle model scheme to another one. We will be given a forger A against the $\mathcal{AS}$*-3* that queries a random oracle. We need to build an adversary B against $\mathcal{BLS}$. But B is itself given a random oracle. The idea is that B will answer some of A queries via its own random oracle and directly simulate the others. The full proof follows.

Proof of Lemma 3.3: Given a forger A that $(t, q_{\mathrm{S}}, n_{\max}, q_{\mathrm{H}}, \epsilon)$-breaks $\mathcal{AS}$*-3*, consider the following forger B against the $\mathcal{BLS}$ standard signature scheme. B is given public key $X^* = g^{x^*}$ as input, and has access to a random oracle $\mathrm{H}_{\mathcal{BLS}}(\cdot)$ and a signing oracle $\mathsf{Sign}_{\mathcal{BLS}}(x^*, \cdot) = \mathrm{H}_{\mathcal{BLS}}(\cdot)^{x^*}$. It runs A on input X^* and responds to its $\mathrm{H}_{\mathcal{AS}\text{-}3}(\cdot)$ and $\mathsf{Sign}_{\mathcal{AS}\text{-}3}(x^*, \cdot)$ oracle queries using the subroutines in Figure 1.

When A submits a query M to its random oracle $\mathrm{H}_{\mathcal{AS}\text{-}3}(\cdot)$, the forger B executes H-SIM(M). We note here that in some cases the subroutine H-SIM can in turn submit queries to B's random oracle $\mathrm{H}_{\mathcal{BLS}}$. When A submits a

```
Subroutine H-SIM($M$)
If ($\exists m : M = X^* \| m$) then return $\mathrm{H}_{\mathcal{BLS}}(M)$
Else If $\mathrm{HT}[M] = \perp$ then $\mathrm{y}[M] \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ ; $\mathrm{HT}[M] \leftarrow \psi(g)^{\mathrm{y}[M]}$
    Return $\mathrm{HT}[M]$

Subroutine SIGN-SIM($m$)
Return $\mathsf{Sign}_{\mathcal{BLS}}(x^*, X^* \| m)$
```

Figure 1: The subroutines for B to simulate the random oracle $\mathrm{H}_{\mathcal{AS}\text{-}3}(\cdot)$ and the sign oracle $\mathsf{Sign}_{\mathcal{AS}\text{-}3}(x^*, \cdot)$. Above, HT and y are associative arrays assumed initially to have value $\perp$ everywhere.

query m to its sign oracle $\mathsf{Sign}_{\mathcal{AS}\text{-}3}(x^*, \cdot)$, the forger B executes SIGN-SIM(m). Eventually, A halts and outputs a forgery $(X_1, m_1), \ldots, (X_n, m_n), \sigma$. Now B defines the sets:

$$\begin{array}{rcl} I & = & \{\, i \mid X_i = X^* \text{ and } m_i = m_1 \,\} \\ J & = & \{\, i \mid X_i = X^* \text{ and } m_i \neq m_1 \,\} \\ K & = & \{\, i \mid X_i \neq X^* \,\} \,. \end{array}$$

Clearly, we have that $I \cup J \cup K = \{1, \ldots, n\}$ and that I, J, K are disjoint. Since A is legitimate, we know that $X_1 = X^*$ and thus I is non-empty. We can assume without loss of generality that $n < p$, because otherwise B can trivially forge and output a $\mathcal{BLS}$ signature under X^* in time $O(nt_{\mathrm{exp}})$ via exhaustive search for x^*. This means that $|I| \in \mathbb{Z}_p^*$, and hence has an inverse modulo p that we denote by l. Now for each $i \in J$, our adversary B executes SIGN-SIM(m_i) to obtain $\sigma_i \leftarrow \mathsf{Sign}_{\mathcal{BLS}}(x^*, X^* \| m_i)$. For each $i \in K$, it calls its subroutine H-SIM($X_i \| m_i$), thereby ensuring that $\mathrm{y}[X_i \| m_i]$ is defined, lets $y_i \leftarrow \mathrm{y}[X_i \| m_i]$, and then lets $\sigma_i \leftarrow \psi(X_i)^{y_i}$, which we note is the $\mathcal{BLS}$ signature of $X_i \| m_i$ under public key X_i. Finally, B lets

$$M_1 \leftarrow X^* \| m_1 \qquad \text{and} \qquad \sigma_1 \leftarrow \left(\sigma \cdot \textstyle\prod_{i \in J \cup K} \sigma_i^{-1}\right)^l \;, \tag{1}$$

and outputs (M_1, σ_1) as its forgery.

For the analysis, we first argue that if A's forgery is valid then B's forgery is valid too. Assuming the former, the verification equation of $\mathcal{AS}\text{-}3$ tells us that

$$\begin{aligned} \mathbf{e}(\sigma, g) &= \prod_{i=1}^{n} \mathbf{e}\big(\mathrm{H}_{\mathcal{AS}\text{-}3}(X_i \| m_i) \,,\, X_i\big) \\ &= \prod_{i \in I} \mathbf{e}\big(\mathrm{H}_{\mathcal{BLS}}(X^* \| m_1) \,,\, X^*\big) \cdot \prod_{i \in J} \mathbf{e}\big(\mathrm{H}_{\mathcal{BLS}}(X^* \| m_i) \,,\, X^*\big) \cdot \prod_{i \in K} \mathbf{e}\big(\mathrm{H}_{\mathcal{AS}\text{-}3}(X_i \| m_i) \,,\, X_i\big) \\ &= \mathbf{e}\big(\mathrm{H}_{\mathcal{BLS}}(X^* \| m_1) \,,\, X^*\big)^{|I|} \cdot \prod_{i \in J} \mathbf{e}(\sigma_i \,,\, g) \cdot \prod_{i \in K} \mathbf{e}(\sigma_i \,,\, X_i) \qquad (2) \\ &= \mathbf{e}\big(\mathrm{H}_{\mathcal{BLS}}(X^* \| m_1) \,,\, X^*\big)^{|I|} \cdot \prod_{i \in J \cup K} \mathbf{e}(\sigma_i \,,\, g) \;\;. \end{aligned}$$

Above, (2) is true because σ_i, as computed above by B, is the $\mathcal{BLS}$ signature of $X_i \| m_i$ under public key X_i, for all $i \in J \cup K$. We then applied the verification equation of the $\mathcal{BLS}$ scheme. Now we see that if σ_1 is defined by (1) then, from the above and the fact that $|I| \cdot l \equiv 1 \pmod{p}$ we have

$$\begin{aligned} \mathbf{e}(\sigma_1, g) &= \mathbf{e}(\sigma, g)^l \cdot \prod_{i \in J \cup K} \mathbf{e}(\sigma_i, g)^{-l} \\ &= \mathbf{e}\big(\mathrm{H}_{\mathcal{BLS}}(X^* \| m_1) \,,\, X^*\big)^{|I| \cdot l \bmod p} \cdot \prod_{i \in J \cup K} \mathbf{e}(\sigma_i \,,\, g)^l \cdot \prod_{i \in J \cup K} \mathbf{e}(\sigma_i, g)^{-l} \\ &= \mathbf{e}\big(\mathrm{H}_{\mathcal{BLS}}(M_1) \,,\, X^*\big) \;, \end{aligned}$$

which means that σ_1 is a valid $\mathcal{BLS}$ signature of message M_1 under public key X^*.

Furthermore, it is easy to see that the answers that B provided to the oracle queries of A are distributed identically to the ones that A would have obtained from its oracles in the game defining its advantage. The last thing we need to check is that B is legitimate, meaning did not query M_1 to its $\mathsf{Sign}_{\mathcal{BLS}}(x^*, \cdot)$ oracle. But it did not do so while answering $\mathsf{Sign}_{\mathcal{AS}\text{-}3}(x^*, \cdot)$ oracle queries of B because the latter, being legitimate itself, did not query m_1 to its $\mathsf{Sign}_{\mathcal{AS}\text{-}3}(x^*, \cdot)$ oracle. Now B also called $\mathsf{Sign}_{\mathcal{BLS}}(x^*, \cdot)$ on $X^* \| m_i$ for all $i \in J$, but the definition of J implies that here $m_i \neq m_1$. Putting everything together, we get $\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{BLS}}(\mathsf{B}) \geq \mathbf{Adv}^{\text{agg-uf}}_{\mathcal{AS}\text{-}3}(\mathsf{A})$.

Finally, we analyze the resource usage of B. For q_S, we note that B makes sign queries in only two situations: (1) whenever A makes a sign query, so does B, and (2) once A outputs a forgery, B makes $|J| \leq n_{\max}$ additional sign queries. For q_H, it is easy to see that B makes at most the same number of random oracle queries to $\mathrm{H}_{\mathcal{BLS}}$ as A makes to $\mathrm{H}_{\mathcal{AS}\text{-}3}$. For t, notice that B (1) answers q_H random oracle queries, each of which results in a call to H-SIM, (2) possibly makes $n_{\max}$ more calls to H-SIM after A outputs its forgery, and (3) computes one exponentiation in the last step to convert A's forgery into its own. Thus, the claimed running time bound follows, and the proof is complete. ▮

4 Unrestricted Sequential Aggregate Signatures

SAS SCHEMES. A sequential aggregate signature (SAS) scheme [15] $\mathcal{SAS} = (\mathsf{Kg}, \mathsf{SASign}, \mathsf{SAVf})$ consists of three algorithms, the last deterministic. The key generation algorithm is exactly as for standard digital signatures. The first signer computes a signature on message m by calling $\sigma \leftarrow \mathsf{SASign}^{\mathrm{H}}(sk, m)$. Subsequent signers run $\sigma \stackrel{\$}{\leftarrow} \mathsf{SASign}^{\mathrm{H}}(sk, m, \sigma', (pk_1, m_1), \ldots, (pk_n, m_n))$, where $n > 0$, to aggregate their signature on a message m into a given sequential aggregate signature σ' corresponding to a sequence of public-key-message pairs. A verifier can run $\mathsf{SAVf}^{\mathrm{H}}((pk_1, m_1), \ldots, (pk_n, m_n), \sigma)$, where $n \geq 0$, to obtain a bit, with 1 indicating accept and 0 reject.

SECURITY. The security of a SAS scheme in the random oracle model requires that it be computationally infeasible for an adversary to produce a sequential aggregate forgery involving an honest signer, even when it can play the role of all other signers and can mount a chosen-message attack on the honest signer. Formally, the advantage of an adversary A is

$$\mathbf{Adv}^{\text{seq-agg-uf}}_{\mathcal{SAS}}(\mathsf{A}) \;=\; \Pr\left[\, \mathsf{SAVf}^{\mathrm{H}}((pk_1, m_1), \ldots, (pk_n, m_n), \sigma) = 1 \,\right]$$

where the probability is over the experiment

$$(pk, sk) \stackrel{\$}{\leftarrow} \mathsf{Kg} \;;\; \mathrm{H} \stackrel{\$}{\leftarrow} \mathsf{Maps}(D) \;;\; ((pk_1, m_1), \ldots, (pk_n, m_n), \sigma) \stackrel{\$}{\leftarrow} \mathsf{A}^{\mathsf{SASign}^{\mathrm{H}}(sk, \cdots), \mathrm{H}}(pk) \;.$$

To make this meaningful, we impose that A be *legitimate* in the sense that there exists $i \in \{1, \ldots, n\}$ such that $pk_i = pk$ and there exists no $\sigma_{i-1} \in \{0,1\}^*$ such that A submitted query $m_i, \sigma_{i-1}, (pk_1, m_1), \ldots, (pk_{i-1}, m_{i-1})$ to the signing oracle. Thus, the honest signer here is the one whose keys are pk, sk, and we are asking that the sequential aggregate forgery include some message and signature corresponding to this honest signer, but the adversary never legitimately obtained a sequential aggregate signature of this message. We say that A $(t, q_S, n_{\max}, q_H, \epsilon)$-breaks $\mathcal{SAS}$ if it runs in time at most t, invokes its signing oracle at most q_S times, invokes its random oracle at most q_H times, has advantage strictly greater than ϵ, and there are at most $n_{\max}$ public keys involved in either its forgery or any of its queries to its signing or random oracles. We say that $\mathcal{SAS}$ is $(t, q_S, n_{\max}, q_H, \epsilon)$-secure if there is no adversary that $(t, q_S, n_{\max}, q_H, \epsilon)$-breaks $\mathcal{SAS}$.

The security requirement we are making is stronger than the one of [15]. One can view their definition as asking for the same condition but for a more restricted (smaller) class of adversaries, namely ones whose output must not contain repeated public keys and who may not submit any queries containing repeating public keys

```
Alg $\mathsf{SASign}^{\mathrm{H}}((\pi, \pi^{-1}), m, \sigma_n, (\pi_1, m_1), \ldots, (\pi_n, m_n))$
If $n = 0$ then $\sigma_0 \leftarrow 1$
If $\mathsf{SAVf}^{\mathrm{H}}((\pi_1, m_1), \ldots, (\pi_n, m_n), \sigma_n) \neq 1$
    then Return $\perp$
$h \leftarrow \mathrm{H}(\pi_1 \| m_1 \| \ldots \| \pi_n \| m_n \| \pi \| m)$
$\sigma \leftarrow \pi^{-1}(h \cdot \sigma_n)$
Return $\sigma$
```

```
Alg $\mathsf{SAVf}^{\mathrm{H}}((\pi_1, m_1), \ldots, (\pi_n, m_n), \sigma)$
$\boxed{\text{If } \pi_1, \ldots, \pi_n \text{ not all different then Return } 0}$
$\sigma_n \leftarrow \sigma$
For $i = n, \ldots, 1$ do
    If $\mathsf{Test}(\pi_i) = 0$ then Return 0
    $h_i \leftarrow H(\pi_1 \| m_1 \| \ldots \| \pi_i \| m_i)$
    $\sigma_{i-1} \leftarrow \pi_i(\sigma_i) \cdot h_i^{-1}$
If $\sigma_0 = 1$ then Return 1 else Return 0
```

Figure 2: The sequential aggregation and verification algorithms of $\mathcal{SAS}$-*1*. Removing the framed text yields $\mathcal{SAS}$-*0*.

to the signing oracle. We will show that a construction proposed in [15] remains secure even under our more stringent definition.

CERTIFIED CLAW-FREE TRAPDOOR PERMUTATIONS. The schemes we consider use a family of certified claw-free trapdoor permutations over a group [15]. Let us recall the definitions. Let $\mathbb{G}$ be a multiplicative group. A *family of certified claw-free trapdoor permutations* Π over $\mathbb{G}$ is a 4-tuple $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Inv}, \mathsf{Test})$ of algorithms, all but the first being deterministic. Via $(\pi, \overline{\pi}, \pi^{-1}) \xleftarrow{\$} \mathsf{Gen}$, one can generate (the descriptions of) a pair of permutations $\pi, \overline{\pi}$ on $\mathbb{G}$ and (the trapdoor information describing) the inverse permutation π^{-1}. For all $(\pi, \overline{\pi}, \pi^{-1}) \in [\mathsf{Gen}]$ and all $x \in \mathbb{G}$, the evaluation algorithm Eval, on inputs π, x, returns $\pi(x)$ in time at most T_Π, and on inputs $\overline{\pi}, x$ returns $\overline{\pi}(x)$ in time at most T_Π, where T_Π is a number associated to Π. The inversion algorithm Inv takes input π^{-1} and $y \in \mathbb{G}$ and returns $\pi^{-1}(y)$ in time at most T_Π. The map Test takes input any string π' and, in time at most T_Π, returns a bit, this being 1 if and only if $\mathsf{Eval}(\pi', \cdot)$ is a permutation on $\mathbb{G}$ whose computation takes time at most T_Π. We assume the time for multiplication, inversion, and sampling random elements in $\mathbb{G}$ is also at most T_Π. We define the advantage of an algorithm B in finding a claw in Π as

$$\mathbf{Adv}^{\mathrm{claw}}_{\Pi}(\mathsf{B}) \;=\; \Pr\left[\, \pi(x) = \overline{\pi}(y) \;\middle|\; (\pi, \overline{\pi}, \pi^{-1}) \xleftarrow{\$} \mathsf{Gen}\,;\; (x, y) \xleftarrow{\$} \mathsf{B}(\pi, \overline{\pi}) \,\right] .$$

We say that Π is (t', ϵ')-claw-free if there is no adversary B that runs in time at most t' yet has advantage strictly more than ϵ'. From now on we will confound the permutations and their descriptions, writing $\pi(x)$ for $\mathsf{Eval}(\pi, x)$, and so on.

THE SAS SCHEMES WE CONSIDER. The $\mathcal{SAS}$-*1* scheme of [15] associated to a family of certified claw-free trapdoor permutations Π works as follows. Each signer generates a key pair (pk, sk) by computing $(\pi, \overline{\pi}, \pi^{-1}) \xleftarrow{\$} \mathsf{Gen}$; $pk \leftarrow \pi$; $sk \leftarrow (\pi, \pi^{-1})$. Let $\mathrm{H} : \{0,1\}^* \rightarrow \mathbb{G}$ be a random oracle. The sequential aggregation and verification algorithms are described in Figure 2.

The $\mathcal{SAS}$-*1* scheme was proposed and proven secure in [15]. We consider here the $\mathcal{SAS}$-*0* scheme that is identical to the $\mathcal{SAS}$-*1* scheme, except that the boxed distinctness test in the verification algorithm in Figure 2 is dropped. (Note that this also affects the signing algorithm because it invokes the verification algorithm as a subroutine.)

CONSISTENCY CONDITIONS. Similar to general aggregate signatures, the consistency condition for sequential aggregate signatures differs from scheme to scheme. $\mathcal{SAS}$-*0* meets the natural, strongest possible consistency requirement, namely that

$$\Pr\left[\, \mathsf{SAVf}^{\mathrm{H}}((pk_1, m_1), \ldots, (pk_n, m_n), \sigma_n) = 1 \,\right] \;=\; 1$$

for all positive integers n and all messages $m_1, \ldots, m_n \in \{0,1\}^*$, the probability being in the experiment where we select $\mathrm{H} \xleftarrow{\$} \mathsf{Maps}(D)$; $(pk_1, sk_1), \ldots, (pk_n, sk_n) \in [\mathsf{Kg}]$, then select $\sigma_1 \xleftarrow{\$} \mathsf{SASign}^{\mathrm{H}}(sk_1,$

$m_1)$; $\sigma_2 \xleftarrow{\$} \mathsf{SASign}^{\mathrm{H}}(sk_2, m_2, \sigma_1, (pk_1, m_1))$; $\cdots$; $\sigma_n \xleftarrow{\$} \mathsf{SASign}^{\mathrm{H}}(sk_n, m_n, \sigma_{n-1}, (pk_1, m_1), \ldots, (pk_{n-1}, m_{n-1}))$. However, $\mathcal{SAS}$-*1* meets this condition only when $pk_1, \ldots, pk_n$ are distinct.

Discussion of security. Lysyanskaya et al. [15] make no claims regarding the security of the $\mathcal{SAS}$-*0* scheme. Unlike the case of the $\mathcal{AS}$-*1* general aggregate signature scheme, there does not seem to be an easy attack when the distinctness condition is dropped. At the same time, the security proof of [15] clearly ceases to go through for the $\mathcal{SAS}$-*0* scheme, because the simulation of signatures and the way the forgery is exploited explicitly rely on all public keys being distinct. One may therefore rightfully wonder what the reason for this restriction is, and whether it is strictly necessary.

Our result. The following implies that the distinctness restriction in the $\mathcal{SAS}$-*1* scheme can safely be dropped:

Theorem 4.1 *If the family of certified claw-free trapdoor permutations* Π *is* (t', ϵ')*-claw-free, then the* $\mathcal{SAS}$-*0* *sequential aggregate signature scheme is* $(t, q_{\mathrm{S}}, n_{\max}, q_{\mathrm{H}}, \epsilon)$*-secure for any* $t, q_{\mathrm{S}}, n_{\max}, q_{\mathrm{H}}, \epsilon$ *satisfying*

$$\epsilon \;\geq\; e(q_{\mathrm{S}} + 1) \cdot \epsilon' \quad \textit{and} \quad t \leq t' - n_{\max} \cdot (q_{\mathrm{H}} + q_{\mathrm{S}} + 2) \cdot O(T_\Pi) \;. \;\blacksquare$$

We achieve this result through a number of refinements to the security proof of [15] that allow us to either simulate signatures for queries with multiple occurrences of the target public key, or to directly exploit the aggregate signature provided by the adversary in a signature query to find a claw for the underlying permutation.

Instantiating $\mathcal{SAS}$-*0* with RSA. RSA does not directly give rise to a family of certified trapdoor permutations. (Each instance comes with its own modulus, so that different instances are over different groups. Also it is not clear in general how to test that a given number is a valid RSA exponent relative to a given modulus.) An RSA-based instantiation of $\mathcal{SAS}$-*0* can however be obtained by using the family of certified claw-free trapdoor permutations from [13]. This comes at the cost of efficiency, however, as a permutation evaluation now takes two RSA exponentiations. Another option is to use techniques in [15], which either require the signers to be ordered by increasing moduli, or add one bit to the signature for each signer. To ensure the certification, they suggest that e can be chosen to be a prime larger than N. Alternatively, one can use arbitrary e such that $\gcd(e, \varphi(N)) = 1$ at the cost of longer public keys as proposed in [10]. As the group operation, one must use addition (modulo $2^{|N|}$ when using [13] or modulo N when using the techniques of [15]) since multiplication is only a group operation over $\mathbb{Z}_N^*$.

Games. Our proof of Theorem 4.1 will use code-based game-playing [4], and we begin by recalling some background from [4]. A game —look at Figure 4 for examples— has an **Initialize** procedure, procedures to respond to adversary oracle queries, and a **Finalize** procedure. A game G is executed with an adversary A, as follows. First, **Initialize** executes, and its outputs are the inputs to A. Then the latter executes, its oracle queries being answered by the corresponding procedures of G. When A terminates, its output becomes the input to the **Finalize** procedure. The output of the latter, denoted G^{A}, is called the output of the game, and we let "$G^{\mathsf{A}} \Rightarrow y$" denote the event that this game output takes value y. The boolean flag bad is assumed initialized to `false`, and GOOD will always denote the event that it is never set to `true`. (This event is defined in all games.) Games G_i, G_j are *identical until* bad if their code differs only in statements that follow the setting of bad to `true`. For examples, games G_0, G_1 of Figure 4 are identical until bad. The following is one version of the Fundamental Lemma of [4].

Lemma 4.2 [4] *Let* G_i, G_j *be identical until* bad *games, and* A *an adversary. Then*

$$\Pr\left[\, G_i^{\mathsf{A}} \Rightarrow 1 \;\wedge\; \textsc{good} \,\right] \;=\; \Pr\left[\, G_j^{\mathsf{A}} \Rightarrow 1 \;\wedge\; \textsc{good} \,\right] \;. \;\blacksquare$$

Proof of Theorem 4.1. We say that an adversary A against $\mathcal{SAS}$-*0* is *simplified* if it has the following properties, where π denotes the public key input to A and π^{-1} its inverse:

1. A never repeats a query to its H-oracle.

2. Any H-query of A has the form $\pi_1 \| m_1 \| \cdots \| \pi_n \| m_n$ for some $n \geq 1$, some $\pi_1, \ldots, \pi_n$ such that $\mathsf{Test}(\pi_i) = 1$ for all $1 \leq i \leq n$, and some $m_1, \ldots, m_n \in \{0,1\}^*$.

3. If A makes a query $m_{n+1}, \sigma_n, (\pi_1, m_1), \ldots, (\pi_n, m_n)$ to $\mathsf{SASign}^{\mathrm{H}}((\pi, \pi^{-1}), \cdots)$ then it previously made H-query $\pi_1 \| m_1 \| \cdots \| \pi_n \| m_n \| \pi \| m_{n+1}$.

4. If A outputs $((\pi_1, m_1), \ldots, (\pi_n, m_n), \sigma)$ then it previously made H-query $\pi_1 \| m_1 \| \cdots \| \pi_n \| m_n$.

5. If A makes H-query $\pi_1 \| m_1 \| \cdots \| \pi_n \| m_n$ then it previously made H-query $\pi_1 \| m_1 \| \cdots \| \pi_{n-1} \| m_{n-1}$. (And hence, inductively, has already queried $\pi_1 \| m_1 \| \cdots \| \pi_i \| m_i$ for all $1 \leq i \leq n-1$, in this order.)

6. If A makes a query $m_{n+1}, \sigma_n, (\pi_1, m_1), \ldots, (\pi_n, m_n)$ to $\mathsf{SASign}^{\mathrm{H}}((\pi, \pi^{-1}), \cdots)$ then $\mathsf{SAVf}^{\mathrm{H}}((\pi_1, m_1), \ldots, (\pi_n, m_n), \sigma_n) = 1$. i such that $\pi_i = \pi$.

7. If A makes a query $m_{n+1}, \sigma_n, (\pi_1, m_1), \ldots, (\pi_n, m_n)$ to $\mathsf{SASign}^{\mathrm{H}}((\pi, \pi^{-1}), \cdots)$ then for every $i \in \{1, \ldots, n\}$ such that $\pi_i = \pi$, it is the case that A previously queried $m_i, \sigma_{i-1}, (\pi_1, m_1), \ldots, (\pi_{i-1}, m_{i-1})$ to $\mathsf{SASign}^{\mathrm{H}}((\pi, \pi^{-1}), \cdots)$.

There are two stages in our proof. First, we transform a given adverary A′ attacking $\mathcal{SAS}$-*0* into a simplified adversary A attacking $\mathcal{SAS}$-*0* without loss in advantage although at some cost in resources:

Lemma 4.3 *Let* A′ *be an adversary that* $(t, q_{\mathrm{S}}, n_{\max}, q_{\mathrm{H}}, \epsilon)$*-breaks* $\mathcal{SAS}$-*0. Then, we can construct a simplified adversary* A *that* $(t^*, q_{\mathrm{S}}, n_{\max}, q^*_{\mathrm{H}}, \epsilon)$*-breaks* $\mathcal{SAS}$-*0 where*

$$t^* = t + n_{\max}(q_{\mathrm{H}} + q_{\mathrm{S}} + 1) \cdot O(T_\Pi) \ , \quad q^*_{\mathrm{H}} = n_{\max}(q_{\mathrm{H}} + q_{\mathrm{S}} + 1) \ . \ \blacksquare$$

Next, we construct an adversary B which enlists A's help in breaking the family Π underlying $\mathcal{SAS}$-*0*:

Lemma 4.4 *Let* A *be a simplified adversary that* $(t^*, q_{\mathrm{S}}, n_{\max}, q^*_{\mathrm{H}}, \epsilon)$*-breaks* $\mathcal{SAS}$-*0. Then, we can construct an adversary* B *attacking* Π *such that*

$$\mathbf{Adv}^{\mathrm{claw}}_{\Pi}(\mathsf{B}) \ \geq \ \frac{\mathbf{Adv}^{\mathrm{seq\text{-}agg\text{-}uf}}_{\mathcal{SAS}\text{-}0}(\mathsf{A})}{e(q_{\mathrm{S}} + 1)} \ , \qquad (3)$$

and the running time of B *is at most that of* A *plus* $(q^*_{\mathrm{H}} + 1) \cdot O(T_\Pi)$. $\blacksquare$

Theorem 4.1 follows directly from these two lemmas. We now proceed to prove the lemmas. A convention made in writing code is that all array elements are assumed to initially be $\perp$.

Proof of Lemma 4.3: The adversary A has access to oracles H and $\mathsf{SASign}^{\mathrm{H}}$. On input a public key π, it begins by initializing S to $\emptyset$. It then runs $\mathsf{A}'(\pi)$, answering hash and sign oracle queries of A′ via the subroutines H-SIM and SIGN-SIM, respectively, of Figure 3. Note that these subroutines call A's oracles. When A′ halts with some output $((\pi_1, m_1), \ldots, (\pi_n, m_n), \sigma_n)$, adversary A calls H-SIM$(\pi_1 \| m_1 \| \cdots \| \pi_n \| m_n)$, outputs $((\pi_1, m_1), \ldots, (\pi_n, m_n), \sigma_n)$, and halts. Let us now explain how this ensures that A has the properties listed above without loss in advantage as compared to A′.

Property **1** is achieved because A stores as $\mathrm{HT}[x]$ the answer to H-query x of A′, and, if this query is repeated, returns $\mathrm{HT}[x]$ without re-querying the oracle. A answers H-query x of A′ via its own H oracle only when x has the form $\pi_1 \| m_1 \| \cdots \| \pi_n \| m_n$ with $\mathsf{Test}(\pi_i) = 1$ for all $1 \leq i \leq n$, and, otherwise, itself picks a random value to play the role of the answer. This ensures property **2**, yet will not decrease the advantage of A because the algorithms of $\mathcal{SAS}$-*0* never invoke the H oracle on inputs not of the above form. Properties **3** and **4** are obtained by having A make the extra H-query if necessary. Property **5** is provided by having A query all appropriate unqueried prefixes of a H-query $\pi_1 \| m_1 \| \cdots \| \pi_n \| m_n$ before querying the latter. Algorithm $\mathsf{SASign}^{\mathrm{H}}(\pi^{-1}, \cdots)$ returns $\perp$ on input $m_{n+1}, \sigma_n, (\pi_1, m_1), \ldots, (\pi_n, m_n)$ unless $\mathsf{SAVf}^{\mathrm{H}}((\pi_1, m_1), \ldots, (\pi_n, m_n), \sigma_n) = 1$, so we

```
Subroutine H-sim($x$)
If $\exists\, n, \pi_1, \ldots, \pi_n, m_1, \ldots, m_n$ such that
  – $n \geq 1$ and $x = \pi_1 \| m_1 \| \cdots \| \pi_n \| m_n$
  – $\forall i : 1 \leq i \leq n : m_i \in \{0,1\}^*$ and $\mathsf{Test}(\pi_i) = 1$
Then
  For $i = 1, \ldots, n$ do $Q_i \leftarrow \pi_1 \| m_1 \| \cdots \| \pi_i \| m_i$ ; If $\mathrm{HT}[Q_i] = \bot$ then $\mathrm{HT}[Q_i] \leftarrow \mathrm{H}(Q_i)$
Else If $\mathrm{HT}[x] = \bot$ then $\mathrm{HT}[x] \stackrel{\$}{\leftarrow} \mathbb{G}$
Return $\mathrm{HT}[x]$

Subroutine Sign-sim($m_{n+1}, \sigma_n, (\pi_1, m_1), \ldots, (\pi_n, m_n)$)
$\pi_{n+1} \leftarrow \pi$ ; $\sigma_0 \leftarrow 1$
For $i = 1, \ldots, n+1$ do If $\mathsf{Test}(\pi_i) = 0$ then return $\bot$ ; $Q_i \leftarrow \pi_1 \| m_1 \| \cdots \| \pi_i \| m_i$
$y \leftarrow$ H-sim($Q_{n+1}$)
For $i = n, \ldots, 1$ do $\sigma_{i-1} \leftarrow \pi_i(\sigma_i) \cdot \mathrm{HT}[Q_i]^{-1}$
If $\sigma_0 \neq 1$ then return $\bot$
$S \leftarrow S \cup \{(m_{n+1}, \sigma_n, (\pi_1, m_1), \ldots, (\pi_n, m_n))\}$
For $i = 1, \ldots, n$ do
  If $\pi_i = \pi$ and $(m_i, \sigma_{i-1}, (\pi_1, m_1), \ldots, (\pi_{i-1}, m_{i-1})) \notin S$ then Output $((\pi_1, m_1), \ldots, (\pi_i, m_i), \sigma_i)$
$\sigma_{n+1} \leftarrow \mathsf{SASign}^{\mathrm{H}}((\pi, \pi^{-1}), m_{n+1}, \sigma_n, (\pi_1, m_1), \ldots, (\pi_n, m_n))$
Return $\sigma_{n+1}$
```

Figure 3: Subroutines for adversary A.

have A do this test and refrain from making the query unless the answer is one, providing property **6**. Property **7** is the most interesting, and an important element in dealing with loops in signing chains. To explain how A provides it, suppose A′ made a query $m_{n+1}, \sigma_n, (\pi_1, m_1), \ldots, (\pi_n, m_n)$ to $\mathsf{SASign}^{\mathrm{H}}((\pi, \pi^{-1}), \cdots)$ such that for some $1 \leq i \leq n$ it was the case that $\pi_i = \pi$ but A did not previously query $m_i, \sigma_{i-1}, (\pi_1, m_1), \ldots, (\pi_{i-1}, m_{i-1})$ to $\mathsf{SASign}^{\mathrm{H}}((\pi, \pi^{-1}), \cdots)$. Then A, rather than making query $m_{n+1}, \sigma_n, (\pi_1, m_1), \ldots, (\pi_n, m_n)$ to $\mathsf{SASign}^{\mathrm{H}}((\pi, \pi^{-1}), \cdots)$, outputs $((\pi_1, m_1), \ldots, (\pi_i, m_i), \sigma_i)$ as its forgery and halts. Property **6** tells us that $\mathsf{SAVf}^{\mathrm{H}}((\pi_1, m_1), \ldots, (\pi_i, m_i), \sigma_i) = 1$, and the fact that A did not previously query $m_i, \sigma_{i-1}, (\pi_1, m_1), \ldots, (\pi_{i-1}, m_{i-1})$ to $\mathsf{SASign}^{\mathrm{H}}((\pi, \pi^{-1}), \cdots)$ means that A remains legitimate. So this can only increase the advantage of A compared to that of A′. ▌

Proof of Lemma 4.4: We consider the games of Figure 4. The array entry $\mathrm{HT}[\pi_1 \| m_1 \| \cdots \| \pi_n \| m_n]$ plays the role of $\mathrm{H}(\pi_1 \| m_1 \| \cdots \| \pi_n \| m_n)$. The notation $c \stackrel{\delta}{\leftarrow} \{0,1\}$ means that c is assigned 0 with probability δ and 1 with probability $1 - \delta$, where $\delta \in [0,1]$ is a parameter whose value will be chosen later [11]. These games rely on some of the properties of A listed above. For example, when answering H-query $Q_n = \pi_1 \| m_1 \| \cdots \| \pi_n \| m_n$, property **5** allows us to assume $\mathrm{HT}[Q_{n-1}]$, and thus also $\sigma[Q_{n-1}]$, are already defined, where $Q_{n-1} = \pi_1 \| m_1 \| \cdots \| \pi_{n-1} \| m_{n-1}$. Similarly, property **3** tells us that the relevant $\mathrm{HT}[\cdot]$ entries are defined at the time a $\mathsf{SASign}^{\mathrm{H}}((\pi, \pi^{-1}), \cdots)$ query is made. Property **4** tell us that the relevant $\mathrm{HT}[\cdot]$ entries are defined at the time Finalize is run, and the legitimacy of A tells us that s computed at line 130 is well-defined in the sense that the set over which the minimum is taken is not empty. Notice that G_1 uses π^{-1} but G_0 does not.

Let us say that a H-query $\pi_1 \| m_1 \| \cdots \| \pi_n \| m_n)$ is *simulation signed* if $\mathrm{c}[Q_i] = 0$ for all $1 \leq i \leq n$, where $Q_i = \pi_1 \| m_1 \| \cdots \| \pi_i \| m_i$. Then line 112 tells us that, in both G_0 and G_1, we have:

```
Initialize                                                      Game $G_0$ / $\boxed{G_1}$
100 $(\pi, \overline{\pi}, \pi^{-1}) \xleftarrow{\$} \mathsf{Gen}$ ; $\sigma[\varepsilon] \leftarrow 1$ ; $S \leftarrow \emptyset$
101 Return $\pi$

On $\mathrm{H}$-query $\pi_1 \| m_1 \| \cdots \| \pi_n \| m_n$
110 $Q_{n-1} \leftarrow \pi_1 \| m_1 \| \cdots \| \pi_{n-1} \| m_{n-1}$ ; $Q_n \leftarrow \pi_1 \| m_1 \| \cdots \| \pi_n \| m_n$ ; $\sigma[Q_n] \xleftarrow{\$} \mathbb{G}$
111 If $\pi_n = \pi$ then $\mathrm{c}[Q_n] \xleftarrow{\delta} \{0,1\}$ else $\mathrm{c}[Q_n] \leftarrow 0$
112 If $\mathrm{c}[Q_n] = 0$ then $\mathrm{HT}[Q_n] \leftarrow \pi_n(\sigma[Q_n]) \cdot \sigma[Q_{n-1}]^{-1}$ else $\mathrm{HT}[Q_n] \leftarrow \overline{\pi}(\sigma[Q_n]) \cdot \sigma[Q_{n-1}]^{-1}$
113 Return $\mathrm{HT}[Q_n]$

On $\mathsf{SASign}^{\mathrm{H}}((\pi, \pi^{-1}), \cdots)$-query $m_{n+1}, \sigma_n, (\pi_1, m_1), \ldots, (\pi_n, m_n)$
120 $S \leftarrow S \cup \{(m_{n+1}, \sigma_n, (\pi_1, m_1), \ldots, (\pi_n, m_n))\}$ ; $\pi_{n+1} \leftarrow \pi$ ; $\sigma_0 \leftarrow 1$
121 For $i = 1, \ldots, n+1$ do $Q_i \leftarrow \pi_1 \| m_1 \| \ldots \| \pi_i \| m_i$
122 $\sigma_{n+1} \leftarrow \sigma[Q_{n+1}]$
123 If $(\exists i : 1 \le i \le n+1 : \mathrm{c}[Q_i] = 1)$ then bad $\leftarrow$ true ; $\boxed{\sigma_{n+1} \leftarrow \pi^{-1}(\sigma_n \cdot \mathrm{HT}[Q_{n+1}])}$
124 Return $\sigma_{n+1}$

Finalize$((\pi_1, m_1), \ldots, (\pi_n, m_n), \sigma)$
130 $s \leftarrow \min \{ i \mid 1 \le i \le n, \ \pi_i = \pi$ and $\forall \tau : (m_i, \tau, (\pi_1, m_1), \ldots, (\pi_{i-1}, m_{i-1})) \notin S \}$
131 $\sigma_n \leftarrow \sigma$
132 For $i = n, \ldots, 1$ do $Q_i \leftarrow \pi_1 \| m_1 \| \cdots \| \pi_i \| m_i$ ; $\sigma_{i-1} \leftarrow \pi_i(\sigma_i) \cdot \mathrm{HT}[Q_i]^{-1}$
133 If $\sigma_0 = 1$ then $d \leftarrow 1$ else $d \leftarrow 0$
134 If $(\exists i : 1 \le i \le s-1 : \mathrm{c}[Q_i] = 1)$ then bad $\leftarrow$ true
135 If $\mathrm{c}[Q_s] = 0$ then bad $\leftarrow$ true
136 $\omega \leftarrow (\sigma_s, \sigma[Q_s])$
137 Return $d$
```

Figure 4: Game G_1 includes the boxed statement, while game G_0 does not. Also, notice that G_1 uses π^{-1}, but G_0 does not.

Claim 1. Let $\pi_1 \| m_1 \| \cdots \| \pi_n \| m_n)$ be a simulation signed hash query, and let $Q_i = \pi_1 \| m_1 \| \cdots \| \pi_i \| m_i$ for $0 \le i \le n$. Then for all $1 \le i \le n$ we have $\sigma[Q_i] = \pi_n^{-1}(\sigma[Q_{i-1}] \cdot \mathrm{HT}[Q_i])$. □

Claim 2. Let $\omega = (\sigma_s, \sigma[Q_s])$ be as per line 136 and d as per line 133. Then

$$\Pr\left[\, \pi(\sigma_s) = \overline{\pi}(\sigma[Q_s]) \,\right] \ \ge \ \Pr\left[\, d = 1 \ \wedge \ \textsc{good} \,\right] ,$$

where both probabilties are over the execution of G_0 with A.

Proof. If GOOD holds then line 134 tells us that $\pi_1 \| m_1 \| \cdots \| \pi_{s-1} \| m_{s-1}$ is simulation signed. If $d = 1$ then from Claim 1 and line 132 we get $\sigma_i = \sigma[Q_i]$ for all $1 \le i \le s-1$, and, in particular, $\sigma_{s-1} = \sigma[Q_{s-1}]$. If GOOD holds then line 135 implies $\mathrm{c}[Q_s] = 1$, and then line 112 implies $\mathrm{HT}[Q_s] = \overline{\pi}(\sigma[Q_s]) \cdot \sigma[Q_{s-1}]^{-1}$. Thus we have

$$\pi(\sigma_s) \ = \ \pi_s(\sigma_s) \ = \ \mathrm{HT}[Q_s] \cdot \sigma_{s-1} \ = \ \overline{\pi}(\sigma[Q_s]) \cdot \sigma[Q_{s-1}]^{-1} \cdot \sigma_{s-1} \ = \ \overline{\pi}(\sigma[Q_s]) \ . \ \square$$

Now define adversary B against Π as follows. On inputs $\pi, \overline{\pi}$, it initializes $\sigma[\varepsilon] \leftarrow 1$ and $S \leftarrow \emptyset$. It then runs A on input π, answering its oracle queries as per the code of game G_0. When A halts, B runs the Finalize code of

```
Initialize                                                    Game $\boxed{G_2}$ / $G_3$
200 $(\pi, \overline{\pi}, \pi^{-1}) \xleftarrow{\$} \mathsf{Gen}$ ; $S \leftarrow \emptyset$
201 Return $\pi$

On H-query $\pi_1 \| m_1 \| \cdots \| \pi_n \| m_n$
210 $Q_n \leftarrow \pi_1 \| m_1 \| \cdots \| \pi_n \| m_n$
211 If $\pi_n = \pi$ then $\mathrm{c}[Q_n] \xleftarrow{\delta} \{0,1\}$ else $\mathrm{c}[Q_n] \leftarrow 0$
212 $\mathrm{HT}[Q_n] \xleftarrow{\$} \mathbb{G}$
213 Return $\mathrm{HT}[Q_n]$

On $\mathsf{SASign}^{\mathrm{H}}((\pi, \pi^{-1}), \cdots)$-query $m_{n+1}, \sigma_n, (\pi_1, m_1), \ldots, (\pi_n, m_n)$
220 $S \leftarrow S \cup \{(m_{n+1}, \sigma_n, (\pi_1, m_1), \ldots, (\pi_n, m_n))\}$ ; $\pi_{n+1} \leftarrow \pi$ ; $\sigma_0 \leftarrow 1$
221 For $i = 1, \ldots, n+1$ do $Q_i \leftarrow \pi_1 \| m_1 \| \ldots \| \pi_i \| m_i$
222 If $\mathrm{c}[Q_{n+1}] = 1$ then bad $\leftarrow$ true
223 $\boxed{\text{If } (\exists i : 1 \leq i \leq n : \mathrm{c}[Q_i] = 1) \text{ then bad} \leftarrow \texttt{true}}$
224 $\sigma_{n+1} \leftarrow \pi^{-1}(\sigma_n \cdot \mathrm{HT}[Q_{n+1}])$
225 Return $\sigma_{n+1}$

Finalize$((\pi_1, m_1), \ldots, (\pi_n, m_n), \sigma)$
230 $s \leftarrow \min \{ i \mid 1 \leq i \leq n,\ \pi_i = \pi \text{ and } \forall \tau : (m_i, \tau, (\pi_1, m_1), \ldots, (\pi_{i-1}, m_{i-1})) \notin S \}$
231 $\sigma_n \leftarrow \sigma$
232 For $i = n, \ldots, 1$ do $Q_i \leftarrow \pi_1 \| m_1 \| \cdots \| \pi_i \| m_i$ ; $\sigma_{i-1} \leftarrow \pi_i(\sigma_i) \cdot \mathrm{HT}[Q_i]^{-1}$
233 If $\sigma_0 = 1$ then $d \leftarrow 1$ else $d \leftarrow 0$
234 $\boxed{\text{If } (\exists i : 1 \leq i \leq s-1 : \mathrm{c}[Q_i] = 1) \text{ then bad} \leftarrow \texttt{true}}$
235 If $\mathrm{c}[Q_s] = 0$ then bad $\leftarrow$ true
236 Return $d$
```

Figure 5: Game G_2 includes the boxed statements while G_3 does not.

G_0 on input the output of A. It outputs ω and halts. Note that B is based on G_0 rather than G_1 and thus does not need to know π^{-1}. Then by Claim 2 we have

$$\mathbf{Adv}^{\mathrm{claw}}_{\Pi}(\mathsf{B}) \ \geq\ \Pr\left[\, G_0^{\mathsf{A}} \Rightarrow 1 \ \wedge\ \textsc{good} \,\right] . \tag{4}$$

However, G_0 and G_1 are identical-until-bad games, and so Lemma 4.2 implies that

$$\Pr\left[\, G_0^{\mathsf{A}} \Rightarrow 1 \ \wedge\ \textsc{good} \,\right] \ =\ \Pr\left[\, G_1^{\mathsf{A}} \Rightarrow 1 \ \wedge\ \textsc{good} \,\right] . \tag{5}$$

Claim 3. In the execution of G_1 with A, all oracle queries of the latter are answered correctly.

Proof. First consider a H-query $Q_n = \pi_1 \| m_1 \| \cdots \| \pi_n \| m_n$. The random choice of $\sigma[Q_n]$ at line 110, together with the fact that $\pi_n, \overline{\pi}$ are permutations, then implies that $\mathrm{HT}[Q_n]$ is uniformly distributed, meaning the answer to this query is exactly as would be given by a random oracle. Next consider a $\mathsf{SASign}^{\mathrm{H}}((\pi, \pi^{-1}), \cdots)$-query $m_{n+1}, \sigma_n, (\pi_1, m_1), \ldots, (\pi_n, m_n)$. We consider two cases. If $Q_{n+1} = \pi_1 \| m_1 \| \ldots \| \pi_n \| m_n \| \pi \| m_{n+1}$ is simulation signed then Claim 1 tells us that the value $\sigma[Q_{n+1}]$ returned is correct. Otherwise, the value σ_{n+1} returned is computed by the boxed statement at line 123 and is correct because it is computed just as in

```
Initialize                                                                      Game $G_4$
400 $(\pi, \overline{\pi}, \pi^{-1}) \xleftarrow{\$} \mathsf{Gen}$ ; $S \leftarrow \emptyset$
401 Return $\pi$

On H-query $\pi_1 \| m_1 \| \cdots \| \pi_n \| m_n$
410 $\mathrm{HT}[\pi_1 \| m_1 \| \cdots \| \pi_n \| m_n] \xleftarrow{\$} \mathbb{G}$
411 Return $\mathrm{HT}[\pi_1 \| m_1 \| \cdots \| \pi_n \| m_n]$

On $\mathsf{SASign}^{\mathrm{H}}((\pi, \pi^{-1}), \cdots)$-query $m_{n+1}, \sigma_n, (\pi_1, m_1), \ldots, (\pi_n, m_n)$
420 $S \leftarrow S \cup \{(m_{n+1}, \sigma_n, (\pi_1, m_1), \ldots, (\pi_n, m_n))\}$ ; $\sigma_0 \leftarrow 1$
421 $\sigma_{n+1} \leftarrow \pi^{-1}(\sigma_n \cdot \mathrm{HT}[\pi_1 \| m_1 \| \cdots \| \pi_n \| m_n \| \pi \| m_{n+1}])$
422 Return $\sigma_{n+1}$

Finalize$((\pi_1, m_1), \ldots, (\pi_n, m_n), \sigma)$
430 $s \leftarrow \min \{ i \mid 1 \le i \le n,\ \pi_i = \pi$ and $\forall \tau : (m_i, \tau, (\pi_1, m_1), \ldots, (\pi_{i-1}, m_{i-1})) \notin S \}$
431 $\sigma_n \leftarrow \sigma$
432 For $i = n, \ldots, 1$ do $Q_i \leftarrow \pi_1 \| m_1 \| \cdots \| \pi_i \| m_i$ ; $\sigma_{i-1} \leftarrow \pi_i(\sigma_i) \cdot \mathrm{HT}[Q_i]^{-1}$
433 If $\sigma_0 = 1$ then $d \leftarrow 1$ else $d \leftarrow 0$
434 For all $\pi'_1 \| m'_1 \| \cdots \| \pi'_l \| m'_l$ such that $\mathrm{HT}[\pi'_1 \| m'_1 \| \cdots \| \pi'_l \| m'_l] \ne \bot$ do
435     If $\pi'_l = \pi$ then $\mathrm{c}[\pi'_1 \| m'_1 \| \cdots \| \pi'_l \| m'_l] \xleftarrow{\delta} \{0,1\}$ else $\mathrm{c}[\pi'_1 \| m'_1 \| \cdots \| \pi'_l \| m'_l] \leftarrow 0$
436 For all $(m'_{l+1}, \sigma'_l, (\pi'_1, m'_1), \ldots, (\pi'_l, m'_l)) \in S$ do
437     If $\mathrm{c}[\pi'_1 \| m'_1 \| \ldots \| \pi'_l \| m'_l \| \pi \| m'_{l+1}] = 1$ then bad $\leftarrow$ true
438 If $\mathrm{c}[Q_s] = 0$ then bad $\leftarrow$ true
439 Return $d$
```

Figure 6: Game G_4.

$\mathsf{SASign}^{\mathrm{H}}((\pi, \pi^{-1}), \cdots)$. Here we use property **6**, which tells us that σ_{n-1} was correct. $\square$

Claim 3 implies that

$$\Pr\left[\, G_1^{\mathsf{A}} \Rightarrow 1 \ \wedge\ \textsc{good} \,\right] \;=\; \Pr\left[\, G_2^{\mathsf{A}} \Rightarrow 1 \ \wedge\ \textsc{good} \,\right], \tag{6}$$

where game G_2 is in Figure 5. Game G_2 directly answers all oracle queries correctly, meaning just as in the game defining the advantage of A. Additionally it splits up the setting of bad as done by line 123 of G_1 into lines 222, 223. Next we claim that

$$\Pr\left[\, G_2^{\mathsf{A}} \Rightarrow 1 \ \wedge\ \textsc{good} \,\right] \;=\; \Pr\left[\, G_3^{\mathsf{A}} \Rightarrow 1 \ \wedge\ \textsc{good} \,\right]. \tag{7}$$

To justify this, we explain why lines 223, 234 of G_2 are redundant and can simply be dropped to arrive at G_3. First consider line 223. Suppose $\mathrm{c}[Q_i] = 1$ for some $1 \le i \le n$. Then it must be that $\pi_i = \pi$, since, otherwise, due to line 211, $\mathrm{c}[Q_i]$ can only be 0. But then property **7** says that A must have previously made $\mathsf{SASign}^{\mathrm{H}}((\pi, \pi^{-1}), \cdots)$-query $m_i, \sigma_{i-1}, (\pi_1, m_1), \ldots, (\pi_{i-1}, m_{i-1})$. If so, line 222 would have set bad at the time this query was made. Now consider line 234. The definition of s and line 211 tell us that if there is a $1 \le i \le s-1$ such that $\mathrm{c}[Q_i] = 1$ then it must be that $\pi_i = \pi$ and $(m_i, \tau, (\pi_1, m_1), \ldots, (\pi_{i-1}, m_{i-1})) \in S$ for some τ. But then, again, bad would have been set by line 222 when $\mathsf{SASign}^{\mathrm{H}}((\pi, \pi^{-1}), \cdots)$-query $m_i, \tau, (\pi_1, m_1), \ldots, (\pi_{i-1}, m_{i-1})$ was made.

In Game G_3, the responses to oracle queries do not depend on the value of the flag bad. Thus, the choices of $\mathrm{c}[\cdot]$ and the setting of bad can be postponed, meaning G_3 is equivalent to game G_4 of Figure 6. In this game, the random choices of $\mathrm{c}[\cdot]$ are made after the game output d is determined, so clearly the events "$G_4^{\mathsf{A}} \Rightarrow 1$" and GOOD are independent. Thus we have

$$\begin{aligned}
\Pr\left[G_3^{\mathsf{A}} \Rightarrow 1 \ \wedge\ \textsc{good}\right] &= \Pr\left[G_4^{\mathsf{A}} \Rightarrow 1 \ \wedge\ \textsc{good}\right] \\
&= \Pr\left[G_4^{\mathsf{A}} \Rightarrow 1\right] \cdot \Pr\left[\,\textsc{good}\,\right] \\
&= \Pr\left[G_4^{\mathsf{A}} \Rightarrow 1\right] \cdot \Pr\left[G_4^{\mathsf{A}} \text{ doesn't set bad}\right] . \qquad (8)
\end{aligned}$$

The output d of G_4^{A} is 1 exactly when A succeeds in forgery, meaning

$$\Pr\left[G_4^{\mathsf{A}} \Rightarrow 1\right] = \mathbf{Adv}^{\text{seq-agg-uf}}_{\mathcal{SAS}\text{-}\mathcal{O}}(\mathsf{A}) . \qquad (9)$$

On the other hand

$$\Pr\left[G_4^{\mathsf{A}} \text{ doesn't set bad}\right] = \delta^{q_{\mathrm{S}}} \cdot (1-\delta) . \qquad (10)$$

We now select $\delta \in [0,1]$ to maximize the function $f(\delta) = \delta^{q_{\mathrm{S}}}(1-\delta)$, which yields $\delta = 1 - 1/(q_{\mathrm{S}}+1)$ and we have

$$\delta^{q_{\mathrm{S}}} \cdot (1-\delta) = \left(1 - \frac{1}{q_{\mathrm{S}}+1}\right)^{q_{\mathrm{S}}} \cdot \frac{1}{q_{\mathrm{S}}+1} > \frac{1}{e(q_{\mathrm{S}}+1)} . \qquad (11)$$

Putting together (4), (5), (6), (7), (8), (9), (10), and (11), we get (3). The running time of B is that of A plus an overhead of $(q_{\mathrm{H}}^* + 1) \cdot O(T_\Pi)$ due to line 112. ▮

Acknowledgments

Mihir Bellare was supported by NSF grant CNS-0524765 and a gift from Intel Corporation. Gregory Neven is a Postdoctoral Fellow of the Research Foundation Flanders (FWO-Vlaanderen), and was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government and in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

References

[1] M. Bellare, A. Boldyreva, and J. Staddon. Randomness re-use in multi-recipient encryption schemeas. In Y. Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 85–99. Springer-Verlag, Jan. 2003. (Cited on pages 3 and 5.)

[2] M. Bellare and O. Goldreich. On defining proofs of knowledge. In E. F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer-Verlag, Aug. 1992. (Cited on page 5.)

[3] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, Nov. 1993. (Cited on pages 3 and 4.)

[4] M. Bellare and P. Rogaway. Code-based game-playing proofs and the security of triple encryption. In S. Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*. Springer-Verlag, May 2006. Available as Cryptology ePrint Report 2005/334. (Cited on page 10.)

[5] M. Bellare and M. Yung. Certifying permutations: Noninteractive zero-knowledge based on any trapdoor permutation. *Journal of Cryptology*, 9(3):149–166, 1996. (Cited on page 3.)

[6] A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Y. Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46. Springer-Verlag, Jan. 2003. (Cited on pages 3 and 5.)

[7] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer-Verlag, May 2003. (Cited on pages 1, 2, 3, 4, 5 and 6.)

[8] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. A survey of two signature aggregation techniques. *RSA's CryptoBytes*, 6(2), Summer 2003. (Cited on page 3.)

[9] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer-Verlag, Dec. 2001. (Cited on pages 1, 5 and 6.)

[10] D. Catalano, D. Pointcheval, and T. Pornin. Trapdoor hard-to-invert group isomorphisms and their application to password-based authentication. *Journal of Cryptology*, 2006. To appear, available from `http://www.di.ens.fr/~pointche/`. (Cited on page 10.)

[11] J.-S. Coron. On the exact security of full domain hash. In M. Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer-Verlag, Aug. 2000. (Cited on page 12.)

[12] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, Apr. 1988. (Cited on page 4.)

[13] R. Hayashi, T. Okamoto, and K. Tanaka. An RSA family of trap-door permutations with a common domain and its applications. In F. Bao, R. Deng, and J. Zhou, editors, *PKC 2004: 7th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 291–304. Springer-Verlag, Mar. 2004. (Cited on page 10.)

[14] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters. Sequential aggregate signatures and multisignatures without random oracles. In S. Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*. Springer-Verlag, May 2006. Available as Cryptology ePrint Report 2006/096. (Cited on pages 3 and 5.)

[15] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 74–90. Springer-Verlag, May 2004. (Cited on pages 1, 2, 3, 8, 9 and 10.)

[16] H. Shacham. *New Paradigms in Signature Schemes*. PhD thesis, Stanford University, 2005. (Cited on page 3.)