Aggregate and Verifiably Encrypted Signatures from Multilinear Maps Without Random Oracles

Markus Rückert[1]* and Dominique Schröder[2]**

[1] Cryptography and Computeralgebra
[2] Minicrypt
TU Darmstadt, Germany
rueckert@cdc.informatik.tu-darmstadt.de schroeder@me.com

Abstract. Aggregate signatures provide bandwidth-saving aggregation of ordinary signatures. We present the first unrestricted instantiation without random oracles, based on the Boneh-Silverberg signature scheme. Moreover, our construction yields a multisignature scheme where a single message is signed by a number of signers. Our second result is an application to verifiably encrypted signatures. There, signers encrypt their signature under the public key of a trusted third party and output a proof that the signature is inside. Upon dispute between signer and verifier, the trusted third party is able to recover the signature. These schemes are provably secure in the standard model.

1 Introduction

Boneh et al. present aggregate signature schemes (AS) and verifiably encrypted signature schemes (VES) in [2]. In essence, an AS allows any party to combine q signatures $\sigma_1, \ldots, \sigma_q$ on q messages $m_1, \ldots, m_q$ of q signers into a single aggregate signature $\mathbf{S}$, which has roughly the same size as an ordinary signature. VES schemes serve a different purpose, typically in fair exchange protocols [1]. Signers encrypt their signatures under the public key of a trusted party, the adjudicator, while preserving signature verifiability. The adjudicator is able to extract a regular signature from a verifiably encrypted signature in the case of a dispute. Note that we use the revised security model from [15].

A variant of AS, called sequential aggregate signature (SAS), is organized like a chain. The i-th signer receives an "aggregate-so-far", adds its own signature and sends the new aggregate to the $(i+1)$-th signer. Thus, SAS are a slightly restricted variant of AS as they do not support simultaneous aggregation. Basically, security of AS is formalized in the chosen-key model, where an adversary gets as input a challenge public key and is allowed to choose all other user keys. The adversary has access to a signing oracle for the challenge key and is successful if it outputs a valid aggregate signature containing a signature that verifies under the challenge public key.

* This work was supported by CASED (www.cased.de).
** Dominique Schröder was supported by the Emmy Noether Program Fi 940/2-1 of the German Research Foundation (DFG).

Typical applications for aggregate signatures are, e.g., secure routing [10] or certificate chain compression [2]. The main advantage of AS is that it saves bandwidth, which makes it an optimal solution for networks of small, battery-powered devices that communicate over energy-consuming wireless channels [14]. Other important applications, mentioned by Bellare et al. [3], are sensor networks such as the "Tsunami early warning system" located the Indian Ocean. There, each sensor node collects data from the environment and sends it — digitally signed — to a monitoring station. Typically, a sensor node does not send its data directly to the station, but rather forwards it over several nodes to the station, which is why there is a need to ensure authenticity.

Random Oracle Model vs. Registered-Key Model. The random oracle model [5], due to Bellare and Rogaway, treats hash function like truly random function. This approach allows researchers to give confidence about their construction as long as the hash function is ideal. However, Canetti, Goldreich, and Halevi [7] disputed the soundness of the random oracle model and encourage researchers to find efficient schemes, provably secure without them. In the context of multi-message-signatures, such as multisignatures or aggregate signatures, the known schemes either rely on the random oracle model or on a different proof model, known as certified-key model. There, each user has to prove knowledge of the secret key during a key registration protocol. We follow this idea and model this by forcing the adversary to certify all key pairs. Note that this methodology follows the work of, e.g., Boldyreva [4] or Lu et al. [12].

Related Work. After Boneh et al. [2] proposed the notion of aggregate signatures schemes and presented a construction, there were, to the best of our knowledge, only sequential aggregate signature schemes such as [2,11,12,3,14]. As for VES schemes, we refer the reader to the instantiations in, e.g., [2,12].

Our Contribution. The results of this paper are twofold. As the first result, we present an aggregate signature scheme in the certified-key model without random oracles. The scheme is based on the Boneh-Silverberg signature scheme [6], which in turn is a variant of the signature scheme by Lysyanskaya [13]. In fact, our scheme provides the first (unrestricted) aggregate signature in the standard model. Moreover, our scheme can be used as a multisignature scheme without any modifications. We prove its security in the standard model while maintaining an optimal signature size and reasonable efficiency. The construction is based on multilinear maps, whose practicability is yet unknown. Boneh and Silverberg show that multilinear maps cannot be build from Weil or Tate pairings, however, they mention that such maps might come from algebraic geometry but much rather from other areas. Nevertheless, our work widens the range of possible applications for those maps, and we are sure that this will encourage researchers to explore possible realizations.

The second result covers a different area, namely verifiably encrypted signatures. In [2], the authors observe that AS schemes give rise to VES schemes if the extraction of individual signatures from an aggregate signature is hard. Therefore, based on our aggregate signature scheme, we construct a VES scheme that is secure in the standard model.

Organization. Some basic notations and the necessary assumptions are briefly recalled in Section 2. In Section 3, we review the security model for aggregate signature schemes and subsequently present our construction in Section 3.2. Finally, in Section 4, we deal with an application to verifiably encrypted signatures along with the according security proofs.

2 Preliminaries

In this section, we recall some background on multilinear (n-linear) maps, following the notation of [6]. By $(G_1, *)$ and $(G_{\mathrm{T}}, *)$, we denote two groups of prime order ℓ such that: all group actions can be computed efficiently; if $a_1, \ldots, a_n \in \mathbb{Z}$ and $x_1, \ldots, x_n \in G_1$ then $e\left(x_1^{a_1}, x_2^{a_2}, \ldots, x_n^{a_n}\right) = e\left(x_1, x_2, \ldots, x_n\right)^{a_1\, a_2 \cdots a_n}$; if $g \in G_1$ is a generator of G_1then $e\left(g, g, \ldots, g\right)$ generates G_{T}.

In the rest of this paper, we assume that G_1, G_{T}, g, and e are fixed and public parameters. The security parameter is 1^k and the length of the message is n. Let $|S|$ denote the size of the finite set S. An adversary is an efficient algorithm. A function, which is not negligible, is noticeable.

2.1 Complexity Assumption

This section deals with the complexity assumption for the proposed aggregate signature scheme and for the novel verifiably encrypted signature scheme. The following definition can be found in [6].

Generalized Diffie-Hellman Assumption. An algorithm $\mathcal{A}$, breaking the generalized Diffie-Hellman problem, takes as input $(g^{a_1}, \ldots, g^{a_n}) \in G_1$ for randomly chosen $(a_1, \ldots, a_n) \in \mathbb{Z}_\ell^*$ and has access to an oracle $\mathsf{O}_{\mathsf{GDH}}((a_1, \ldots, a_n), \cdot)$ which, when queried with any strict subset $S \subset \{1, \ldots, n\}$, responds with $\mathsf{O}_{\mathsf{GDH}}((a_1, \ldots, a_n), S) = g^{\prod_{i \in S} a_i} \in G_1$.

The probability that $\mathcal{A}$ returns $g^{a_1\, a_2 \cdots a_n}$ is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{GDH}} := \mathsf{Prob}[\mathcal{A}^{\mathsf{O}_{\mathsf{GDH}}}(g, g^{a_1}, \ldots, g^{a_n}) = g^{a_1\, a_2 \ldots a_n} \ : \ (a_1, \ldots, a_n) \in \mathbb{Z}_\ell^*] \ .$$

The GDH problem is $(t, q_{\mathsf{GDH}}, \epsilon)$-*hard* if for any algorithm $\mathcal{A}$, running in time at most t, querying $\mathsf{O}_{\mathsf{GDH}}$ no more than q_{GDH} times, the probability $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{GDH}}$ is less than ϵ.

Secure Signature Schemes. We briefly recall the Boneh-Silverberg ($\mathcal{BS}$) signature scheme [6] and the definition of secure signature schemes. Roughly speaking, a digital signature scheme is secure if there is no adversary which (adaptively) queries a signing oracle and outputs a valid message-signature pair (m^*, σ^*) such that it never queried m^* to the signing oracle. The formal definition can be found in [9].

$\mathcal{BS}$ Signature Scheme. The signature scheme of Boneh and Silverberg, which builds upon the unique signature scheme proposed by Lysyanskaya [13], is defined through the following algorithms. *KeyGen:* The key generation algorithm

$\mathsf{Kg}(1^k)$ takes as input the security parameter 1^k. It randomly selects $2n$ elements $a_{1,0}, a_{1,1}, \ldots, a_{n,0}, a_{n,1} \in \{1, \ldots, \ell - 1\}$. The algorithm computes $u_{1,0} \leftarrow g^{a_{1,0}}, u_{1,1} \leftarrow g^{a_{1,1}}, \ldots, u_{n,0} \leftarrow g^{a_{n,0}}, u_{n,1} \leftarrow g^{a_{n,1}}$ and returns the private key $sk = (a_{1,0}, a_{1,1}, \ldots, a_{n,0}, a_{n,1})$ and the public key $pk = (u_{1,0}, u_{1,1}, \ldots, u_{n,0}, u_{n,1})$. *Signing:* $\mathsf{Sign}(sk, m)$ accepts as input a message $m = (m_1, \ldots, m_n) \in \{0,1\}^n$ as well as a signing key $sk = (a_{1,0}, a_{1,1}, \ldots, a_{n,0}, a_{n,1})$ and computes the signature $\sigma \leftarrow g^{\prod_{i=1}^{n} a_{i,m_i}} \in G_1$. Note that the message space $\{0,1\}^n$ can always be extended to $\{0,1\}^*$ by hashing the messages first. *Verification:* $\mathsf{Vf}(pk, \sigma, m)$ returns 1 iff $e(\sigma, g, \ldots, g) = e(u_{1,m_1}, u_{2,m_2}, \ldots, u_{n,m_n})$. Boneh and Silverberg follow the proof of Lysyanskaya ([13,6]) and show that their scheme is unforgeable under chosen message attacks.

3 Unrestricted Aggregate Signatures

Basically, an aggregate signature [2] is a signature of q different signers on q different messages such that the signature has roughly the same size as a single signature. In such a scheme, the aggregation algorithm can be executed by anyone, including untrusted parties. Bellare, Namprempre, and Neven [3] generalize aggregate signatures to unrestricted aggregate signatures, removing the restriction that all messages as well as all signers have to be distinct. For brevity, we just write aggregate signature instead of unrestricted aggregate signature.

An aggregate signature scheme is a tuple of algorithms $\mathsf{AS} = (\mathsf{AggKGen}, \mathsf{Sign}, \mathsf{Agg}, \mathsf{Vf}, \mathsf{AggVf})$, where

Key Generation. $\mathsf{AggKGen}(1^k)$ generates a key pair (sk, pk) for each user independently.

Signature Issue. The signing algorithm $\mathsf{Sign}(sk, m)$ takes as input the secret key sk as well as a message $m \in \{0,1\}^n$ and outputs a signature σ.

Signature Aggregation. $\mathsf{Agg}((pk^{(1)}, m^{(1)}, \sigma^{(1)}), \ldots, (pk^{(q)}, m^{(q)}, \sigma^{(q)}))$ builds an aggregate $\mathbf{S}$ on messages $\mathbf{M} = (m^{(1)}, \ldots, m^{(q)})$ under public keys $\mathbf{pk} = (pk^{(1)}, \ldots, pk^{(q)})$ and outputs the triple $(\mathbf{pk}, \mathbf{M}, \mathbf{S})$.

Signature Verification. The signature verification algorithm $\mathsf{Vf}(pk, m, \sigma)$ accepts as input a public key pk, the message m, and a signature σ. It outputs a bit, indicating the validity of σ.

Aggregate Verification. The algorithm $\mathsf{AggVf}(\mathbf{pk}, \mathbf{M}, \mathbf{S})$ takes as input a set of public keys $\mathbf{pk} = (pk^{(1)}, \ldots, pk^{(q)})$, a set of messages $\mathbf{M} = (m^{(1)}, \ldots, m^{(q)})$ as well as an aggregate $\mathbf{S}$. It returns 1 iff $\mathbf{S}$ is a valid aggregate signature on messages $(m^{(1)}, \ldots, m^{(q)})$ under public keys $(pk^{(1)}, \ldots, pk^{(q)})$.

The scheme is complete if for any set of q key-pairs $(sk^{(i)}, pk^{(i)}) \leftarrow \mathsf{AggKGen}(1^k)$, for any set of q messages $m^{(i)} \in \{0,1\}^n$, for any honestly generated set of q signatures $\sigma^{(i)} \leftarrow \mathsf{Sign}(sk, m^{(i)})$, and for any aggregate $\mathbf{S}$ returned by $\mathsf{Agg}((pk^{(1)}, m^{(1)}, \sigma^{(1)}), \ldots, (pk^{(q)}, m^{(q)}, \sigma^{(q)}))$ we have that for all $i \in \{1, 2, \ldots, q\}$: $\mathsf{Vf}(pk^{(i)}, m^{(i)}, \sigma^{(i)}) = 1$ and $\mathsf{AggVf}(\mathbf{pk}, \mathbf{M}, \mathbf{S}) = 1$.

3.1 Security

The security of aggregate signatures is formalized in the aggregate certified-key model, which combines the chosen-key model of [2] with the certified-key approach presented in [4] and [12]. Informally, an adversary is given a challenge public key and tries to forge an aggregate signature on messages of its choice and users (keys) of its choice. This adversary has access to a signing oracle O for the challenge key and wins if it is able to output an aggregate signature including a signature σ' on m' under the challenge key, without querying the signing oracle with m'. In addition, the adversary has to certify all signature keys, using a method that allows secret key extraction. To keep it simple, we avoid rewinding the adversary during complex proof protocols and simply force it to provide the secret key during the key certification process.

An aggregate signature scheme AS is *secure* in the certified-key model if the probability that following game evaluates to 1 is negligible (as a function of k).

Setup. Choose the challenge key pair $(pk, sk) \leftarrow \mathsf{AggKGen}(1^k)$, initialize the set of certified keys with $C \leftarrow \emptyset$, and execute algorithm $\mathcal{A}$ on input pk.

Certification Queries. Algorithm $\mathcal{A}$ provides a key pair (sk', pk') in order to certify pk'. If (sk', pk') is a valid key pair, add (sk', pk') to C.

Signing Queries. Algorithm $\mathcal{A}$ adaptively queries q_{O} messages. For any message m, it receives the signature $\sigma \leftarrow \mathsf{Sign}(sk, m)$ under the private key sk.

Output. $\mathcal{A}$ stops, outputting a triple $(\mathbf{pk}, \mathbf{M}, \mathbf{S})$. This triple consists of a set of public keys $\mathbf{pk}$, a set of messages $\mathbf{M}$, and a forged aggregate signature $\mathbf{S}$. The game outputs 1 iff $\mathsf{AggVf}(\mathbf{pk}, \mathbf{M}, \mathbf{S}) = 1$, all keys in $\mathbf{pk}$ (except for pk) are in C, $\exists i : pk^{(i)} = pk$, and $\mathcal{A}$ never invoked $\mathsf{Sign}(sk, \cdot)$ on $m^{(i)}$.

An aggregate signature scheme is $(t, q_{\mathsf{O}}, q_{\max}, \epsilon)$-secure if for any adversary $\mathcal{A}$, running in time at most t, querying the sign oracle at most q_{O} times, the probability that it outputs a valid forgery using at most $q_{\max}$ public key-message pairs, is less than ϵ.

3.2 Our Construction

The proposed aggregate signature scheme AS is defined as follows.

Key Generation, Signature Issue, Signature Verification. Same as in the Boneh-Silverberg signature scheme.

Signature Aggregation. The algorithm $\mathsf{Agg}((pk^{(1)}, m^{(1)}, \sigma^{(1)}), \ldots, (pk^{(q)}, m^{(q)}, \sigma^{(q)}))$ sets $\mathbf{pk} \leftarrow (pk^{(1)}, pk^{(2)}, \ldots, pk^{(q)})$, $\mathbf{M} \leftarrow (m^{(1)}, m^{(2)}, \ldots, m^{(q)})$, computes $\mathbf{S} \leftarrow \prod_{i=1}^{q} \sigma^{(i)}$, and outputs the triple $(\mathbf{pk}, \mathbf{M}, \mathbf{S})$.

Aggregate Verification. $\mathsf{AggVf}(\mathbf{pk}, \mathbf{M}, \mathbf{S})$ returns 1 iff

$$\prod_{i=1}^{q} e\left(u^{(i)}_{1,m_1^{(i)}}, u^{(i)}_{2,m_2^{(i)}}, \ldots, u^{(i)}_{n,m_n^{(i)}}\right) = e\left(\mathbf{S}, g, \ldots, g\right).$$

In the following we show that our aggregate signature scheme is complete. Let $(sk^{(1)}, pk^{(1)}), \ldots, (sk^{(q)}, pk^{(q)})$ be the honestly generated key pairs of all participating users and let $(m^{(1)}, \sigma^{(1)}), \ldots, (m^{(q)}, \sigma^{(q)})$ be the corresponding message-signature pairs that verify under Vf. Now, let $\mathbf{S}$ be the output of Agg under those messages, signatures, and keys. AggVf evaluates

$$\prod_{i=1}^{q} e\left(u^{(i)}_{1,m_1^{(i)}}, u^{(i)}_{2,m_2^{(i)}}, \ldots, u^{(i)}_{n,m_n^{(i)}}\right) = \prod_{i=1}^{q} e\left(\sigma^{(i)}, g, \ldots, g\right) = e\left(g, g, \ldots, g\right)^{\sum_{i=1}^{q}\prod_{j=1}^{n} a^{(i)}_{j,m_j^{(i)}}}$$
$$= e\left(g^{\sum_{i=1}^{q}\prod_{j=1}^{n} a^{(i)}_{j,m_j^{(i)}}}, g, \ldots, g\right) = e\left(\prod_{i=1}^{q}\sigma^{(i)}, g, \ldots, g\right) = e\left(\mathbf{S}, g, \ldots, g\right).$$

Aggregate Security. We prove that our scheme is secure in the chosen-key model as long as the $\mathcal{BS}$ signature scheme is unforgeable.

Theorem 1. *Let T_{Sign} be the cost function for $\mathcal{BS}$ signature generation. Our scheme is $(t, q_{\mathsf{O}}, q_{max}, \epsilon)$-secure if the $\mathcal{BS}$ signature scheme is $(t', q'_{\mathsf{O}}, \epsilon')$-unforgeable with $t' = t + (q_{max} - 1)\, T_{\mathsf{Sign}}$, $q'_{\mathsf{O}} = q_{\mathsf{O}}$, and $\epsilon' = \epsilon$.*

The proof is a black-box reduction. We build an algorithm $\mathcal{B}$ against unforgeability of the underlying signature scheme. $\mathcal{B}$ gets as input a public key pk and has access to a signing oracle $\mathsf{Sign}(sk, \cdot)$. The adversary $\mathcal{A}$, which is a forger against the aggregate signature scheme, is given the challenge public key pk and, if successful, outputs a forged aggregate signature $(\mathbf{pk}, \mathbf{M}, \mathbf{S})$ containing a forged (ordinary) signature under the key pair (sk, pk). Algorithm $\mathcal{B}$ can extract this signature and thus returns a forgery against the underlying signature scheme.

Proof. Assuming there exists a successful adversary $\mathcal{A}$ against the aggregate signature scheme, we construct an algorithm $\mathcal{B}$ via a black-box simulation, which is an equally successful forger against the $\mathcal{BS}$ signature scheme.

Setup. $\mathcal{B}$ receives the challenge key pk, initializes the set of certified keys ($C \leftarrow \emptyset$), and runs $\mathcal{A}$ on input pk.

Certification Queries. $\mathcal{A}$ wants to certify a key pk' and hands over the pair (sk', pk'). If sk' is the corresponding secret key to pk', then add the pair to the list $C \leftarrow C \cup \{(sk, pk)\}$, otherwise reject.

Signature queries. $\mathcal{B}$ answers $\mathcal{A}$'s queries on a message m by querying its own signature oracle $\sigma \leftarrow \mathsf{Sign}(sk, m)$ and returns σ.

Output. $\mathcal{A}$ halts, outputting an aggregate signature triple $(\mathbf{pk}, \mathbf{M}, \mathbf{S})$. If $\mathcal{A}$ is successful, all public keys (except the challenge key) are registered. $\mathcal{B}$ extracts the message-signature pair (m^*, σ^*), corresponding to the public key pk, outputs (m^*, σ^*), and stops.

W.l.o.g., let (sk_1, pk_1) be the key pair of $\mathcal{B}$'s signing oracle, i.e. $pk_1 = pk$. The extraction of the corresponding message-signature pair from $(\mathbf{pk}, \mathbf{M}, \mathbf{S})$, with $|\mathbf{pk}| = q$, works as follows:

1. Let $\mathbf{sk}$ be the sequence of secret keys $(a^{(i)}_{1,0}, a^{(i)}_{1,1}, \ldots, a^{(i)}_{n,0}, a^{(i)}_{n,1})$ for $i = 2, \ldots, q$, obtained from the certified key store C, which were chosen by $\mathcal{A}$.

2. Let $m^* \leftarrow m^{(1)}$ and $\sigma^* \leftarrow \mathbf{S}\left(\prod_{i=2}^{q} g^{\prod_{j=1}^{n} a^{(i)}_{j,m_j^{(i)}}}\right)^{-1}$.
3. Output the forged signature (m^*, σ^*).

Analysis. We first show that the extraction yields a valid signature. We have

$$e\left(\sigma^*, g, \ldots, g\right) = e\left(\mathbf{S}\left(\prod_{i=2}^{q} g^{\prod_{j=1}^{n} a^{(i)}_{j,m_j^{(i)}}}\right)^{-1}, g, \ldots, g\right)$$
$$= e\left(g^{\sum_{i=1}^{q}\prod_{j=1}^{n} a^{(i)}_{j,m_j^{(i)}} - \sum_{i=2}^{q}\prod_{j=1}^{n} a^{(i)}_{j,m_j^{(i)}}}, g, \ldots, g\right) = e(u_{1,m_1}, u_{2,m_2}, \ldots, u_{n,m_n}).$$

Therefore, σ^* verifies under pk. Moreover, algorithm $\mathcal{B}$ answers all of $\mathcal{A}$'s queries as expected and therefore simulates the environment perfectly. Whenever $\mathcal{A}$ invokes the signing oracle, algorithm $\mathcal{B}$ queries its signing oracle. Since $\mathcal{B}$ has access to the secret signing keys, which $\mathcal{A}$ generates prior to outputting a corresponding forgery, $\mathcal{B}$ can always extract a valid forged signature. Thus, $\mathcal{B}$ is successful whenever $\mathcal{A}$ is. The overhead of the extraction algorithm is $(q_{max} - 1)\, T_{\mathsf{Sign}}$ (we omit the list processing cost induced by C). □

4 Verifiably Encrypted Signatures

Boneh et al. [2] introduced verifiably encrypted signature schemes (VES) along with a security model that is extended in [15]. In a VES, a signer encrypts its signature under the public key of a trusted third party (the adjudicator) and proves that the encrypted value contains a valid signature.

A verifiably encrypted signature scheme $\mathsf{VES} = (\mathsf{VesKGen}, \mathsf{VesAdjKGen}, \mathsf{Sign}, \mathsf{Vf}, \mathsf{Create}, \mathsf{VesVf}, \mathsf{Adj})$ consists of the following seven algorithms.

Key Generation, Signing, Verification. Defined as in a standard digital signature scheme.

Adjudicator Key Generation. $\mathsf{VesAdjKGen}(1^k)$ outputs (ask, apk), where ask is private and apk is public.

VES **Creation.** $\mathsf{Create}(sk, apk, m)$ takes as input a secret signature key sk, the adjudicator's public key apk, and a message $m \in \mathcal{M}$. It returns a verifiably encrypted signature ω on m.

VES **Verification.** $\mathsf{VesVf}(apk, pk, \omega, m)$ takes as input the adjudicator's public key apk, a public verification key pk, a verifiably encrypted signature ω, and a message m. It returns a bit.

Adjudication. $\mathsf{Adj}(ask, apk, pk, \omega, m)$ accepts as input the key pair (ask, apk) of the adjudicator, the public key pk of the signer, a verifiably encrypted signature ω, and a message m. It extracts an ordinary signature σ on m.

For brevity, we sometimes omit the key parameters. A verifiably encrypted signature scheme is *complete* if for all honestly generated keys $(ask, apk) \leftarrow \mathsf{VesAdjKGen}(1^k)$ and $(sk, pk) \leftarrow \mathsf{VesKGen}(1^k)$, and for all $m \in \mathcal{M}$ we have $\mathsf{VesVf}(\mathsf{Create}(m), m) = 1 \wedge \mathsf{Vf}(\mathsf{Adj}(\mathsf{Create}(m)), m) = 1$.

4.1 Security

VES schemes must satisfy unforgeability, opacity [2], extractability, and abuse-freeness [15]. Unforgeability requires that it is hard to forge a verifiably encrypted signature. The adversary is given access to two oracles: oracle VESig gets a message m as input and returns a verifiably encrypted signature ω, and oracle VEAdj, which extracts a signature σ from a given ω. The adversary is successful if it outputs a pair (m^*, ω^*) such that it never queried m^* to VESig or VEAdj.

A scheme is $(t, q_{\mathsf{VESig}}, q_{\mathsf{VEAdj}}, \epsilon)$-unforgeable if there is no adversary $\mathcal{A}$ that runs in time at most t, makes at most q_{VESig} queries to the VESig, at most q_{VEAdj} queries to VEAdj, and $\mathsf{AdvVSigF}_{\mathcal{A}}$ is at least ϵ, where $\mathsf{AdvVSigF}_{\mathcal{A}} =$

$$\mathsf{Prob}\left[\begin{array}{c} \mathsf{VesVf}(apk, pk, \omega^*, m^*) = 1 \ : \\ (apk, ask) \leftarrow \mathsf{VesAdjKGen}(1^n) \\ (pk, sk) \leftarrow \mathsf{VesKGen}(1^n) \\ (m^*, \omega^*) \leftarrow \mathcal{A}^{\mathsf{VESig},\mathsf{VEAdj}}(apk, pk) \end{array}\right].$$

Opacity states that it is difficult to extract an ordinary signature from ω. Again, the adversary is given access to the oracles VESig and VEAdj. A scheme is $(t, q_{\mathsf{VESig}}, q_{\mathsf{VEAdj}}, \epsilon)$-opaque if there is no adversary $\mathcal{B}$ that runs in time at most t, makes at most q_{VESig} queries to the VESig, at most q_{VEAdj} queries to VEAdj, and $\mathsf{AdvVSigO}_{\mathcal{A}}$ is at least ϵ, where $\mathsf{AdvVSigO}_{\mathcal{A}} =$

$$\mathsf{Prob}\left[\begin{array}{c} \mathsf{Vf}(pk, \sigma^*, m^*) = 1 \ : \\ (apk, ask) \leftarrow \mathsf{VesAdjKGen}(1^n) \\ (pk, sk) \leftarrow \mathsf{VesKGen}(1^n) \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{VESig},\mathsf{VEAdj}}(apk, pk) \end{array}\right].$$

An additional requirement is *weak-extractability*, stating that for all $(ask, apk) \leftarrow \mathsf{VesAdjKGen}(1^k)$, $(sk, pk) \leftarrow \mathsf{VesKGen}(1^k)$, and all verifiably encrypted signatures ω on some message m, we have $\mathsf{VesVf}(\omega, m) = 1 \implies \mathsf{Vf}(\mathsf{Adj}(\omega, m), m) = 1$, i.e. if ω is valid then the adjudicator can always extract a valid ordinary signature. This can be improved to *extractability* by a standard transformation in [15]. Moreover, VES schemes have to satisfy *abuse-freeness*, which ensures that signer and adjudicator cannot collude in order to forge verifiably encrypted signatures on bahalf of a third party.

4.2 Our Construction

Our verifiably encrypted signature scheme VES is defined as follows.

Key Generation, Signing, Verification. As in $\mathcal{BS}$.

Adjudicator Key Generation. Pick a random element $\beta \leftarrow \mathbb{Z}_p^*$ and return the secret key $ask \leftarrow \beta$ along with the public key $apk \leftarrow g^\beta$.

VES **Creation.** $\mathsf{Create}(sk, apk, m)$ takes as input a signing key sk, the adjudicator's public key apk, as well as a message $m \in \{0,1\}^n$. It selects a random value $r \in \mathbb{Z}_p^*$ and computes the verifiably encrypted signature as $\sigma \leftarrow g^{a_{1,m_1} \cdots a_{n,m_n}}$. The algorithm sets $K \leftarrow (apk)^r$, $\mu \leftarrow g^r$, and calculates $\omega \leftarrow \sigma \cdot K$. It returns (ω, μ).

VES Verification. $\mathsf{VesVf}(apk, pk, (\omega, \mu), m)$ returns 1 iff $e\left(u_{1,m_1}, u_{2,m_2}, \ldots, u_{n,m_n}\right) = e\left(\omega, g, \ldots, g\right) \cdot e\left(\mu, apk, g, \ldots, g\right)^{-1}$.

Adjudication. $\mathsf{Adj}(ask, pk, (\omega, \mu), m)$ extracts the signature $\sigma \leftarrow \omega \cdot \mu^{-\beta}$ if (ω, μ) is valid.

It is easy to see that the scheme is complete. For the following security proofs, let T_{VesKGen}, $T_{\mathsf{VesAdjKGen}}$, T_{Create}, T_{Adj} be the cost functions for user key generation, adjudication key generation, verifiably encrypted signature generation, and adjudication.

Theorem 2. *If the $\mathcal{BS}$-signature is $(t', q'_{\mathsf{O}}, \epsilon')$-unforgeable then our scheme is $(t, q_{\mathsf{VESig}}, q_{\mathsf{VEAdj}}, \epsilon)$-unforgeable, where $t' = t + q_{\mathsf{VESig}}\, T_{\mathsf{Create}} + T_{\mathsf{VesAdjKGen}} + (q_{\mathsf{VEAdj}} + 1)\, T_{\mathsf{Adj}}$, $q'_{\mathsf{O}} = q_{\mathsf{VESig}}$, and $\epsilon' = \epsilon$.*

The proof is a straightforward black-box reduction against the $\mathcal{BS}$ signature.

The presented verifiably encrypted signature scheme is opaque. The reduction is somewhat different to the one of Boneh et al. [2] as well as Lu et al. [12]. Both require the aggregate extraction problem to be hard, for which Coron and Naccache [8] showed that it is equivalent to the Diffie-Hellmann problem. In our case, however, the security of the underlying signature scheme is not based on the Diffie-Hellmann assumption and we therefore cannot base any security property on the aggregate extraction problem. Instead, we show that it is opaque under the GDH assumption.

Theorem 3. *If the GDH problem is (t', ϵ')-hard, our scheme is $(t, q_{\mathsf{VESig}}, q_{\mathsf{VEAdj}}, \epsilon)$-opaque with $t' = t + T_{\mathsf{VesAdjKGen}} + T_{\mathsf{VesKGen}} + (q_{\mathsf{VESig}} + q_{\mathsf{VEAdj}})\, T_{\mathsf{Create}}$ and $\epsilon' = \epsilon / O(t)$.*

The reduction immediately follows the idea of the proof of Lysyanskaya [13, Theorem 1] but differs in an important aspect. We first explain the main idea and then the difference. Due to space restrictions, the full proof is omitted. The main idea is to guess s-bits, along with their positions, of the adversary's output message. Once the reduction has guessed s bits $B = (b_1, \ldots, b_s)$, it puts the values $(Y_1, \ldots, Y_s)$ (which the reduction gets as input from the GDH instance) into the corresponding parts of the public verification key and selects all other keys on its own.

The difference to the proof in [13] is that, here, the adversary is allowed to invoke VESig on the message it outputs, but *not* VEAdj. Thus, the adversary is allowed to invoke VESig on a message, which contains the bits B, each at the guessed position. In order to simulate VESig upon this query, we query the GDH oracle on the index set $(0, 1, \ldots, 1)$ and set ω and μ such that they pass the VesVf algorithm. By correctly guessing some bits of $\mathcal{A}$'s output message during the setup phase, the reduction forces $\mathcal{A}$ to output a signature that contains a valid answer to the GDH problem.

Extractability. The scheme is weakly-extractable because $e\left(\omega\, \mu^{-\beta}, g, \ldots, g\right) = e\left(u_{1,m_1}, u_{2,m_2}, \ldots, u_{n,m_n}\right)$ for all valid ω (on m) for honestly chosen keys.

Abuse-freeness. Since our scheme employs the signature and encryption algorithms independently of each other, a theorem in [15] applies, which states that extractability is sufficient for abuse-freeness in this case.

References

1. N. Asokan, Victor Shoup, and Michael Waidner. *Optimistic Fair Exchange of Digital Signatures. IEEE Journal on Selected Areas in Communications*, 18(4):593–610, 2000.
2. Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. *Aggregate and Verifiably Encrypted Signatures from Bilinear Maps.* Advances in Cryptology — Eurocrypt'03, Lecture Notes in Computer Science, pages 416–432. Springer-Verlag, 2003.
3. Mihir Bellare, Chanathip Namprempre, and Gregory Neven. *Unrestricted Aggregate Signatures.* ICALP 2007, Volume 4596 of Lecture Notes in Computer Science, pages 411–422. Springer-Verlag, 2007.
4. Alexandra Boldyreva. *Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme.* Public-Key Cryptography (PKC)'03, Volume 2567 of Lecture Notes in Computer Science, pages 31–46. Springer-Verlag, 2003.
5. Mihir Bellare and Pil Rogaway. *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols.* Proceedings of the Annual Conference on Computer and Communications Security (CCS). ACM Press, 1993.
6. Dan Boneh and Alice Silverberg. *Applications of Multilinear Forms to Cryptography.* Topics in Algebraic and Noncommutative Geometry, Contemporary Mathematics 324, pages 71–90, , 2003.
7. Ran Canetti, Oded Goldreich, and Shai Halevi. *The random oracle methodology, revisited. J. ACM*, 51(4):557–594, 2004.
8. Jean-Sébastien Coron and David Naccache. *Boneh et al.'s k-Element Aggregate Extraction Assumption Is Equivalent to the Diffie-Hellman Assumption.* Advances in Cryptology — Asiacrypt'03, Lecture Notes in Computer Science, pages 392–397. Springer-Verlag, 2003.
9. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. *A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM J. Comput.*, 17(2):281–308, 1988.
10. Stephen T. Kent, Charles Lynn, Joanne Mikkelson, and Karen Seo. *Secure Border Gateway Protocol (S-BGP) - Real World Performance and Deployment Issues.* NDSS. Internet Society, 2000.
11. Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. *Sequential Aggregate Signatures from Trapdoor Permutations.* Advances in Cryptology — Eurocrypt'04, Lecture Notes in Computer Science, pages 74–90. Springer-Verlag, 2004.
12. Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. *Sequential Aggregate Signatures and Multisignatures Without Random Oracles.* Advances in Cryptology — Eurocrypt'06, Lecture Notes in Computer Science, pages 465–485. Springer-Verlag, 2006.
13. Anna Lysyanskaya. *Unique Signatures and Verifiable Random Functions from the DH-DDH Separation.* Advances in Cryptology — Crypto 2002, Lecture Notes in Computer Science, pages 597–612. Springer-Verlag, 2002.
14. Gregory Neven. *Efficient Sequential Aggregate Signed Data.* Advances in Cryptology — Eurocrypt'08, Lecture Notes in Computer Science, pages 52–69. Springer-Verlag, 2008.
15. Markus Rückert and Dominique Schröder. *Security of Verifiably Encrypted Signatures.* Number 2009/027 in Cryptology eprint archive. `eprint.iacr.org`, 2009.