Security of Verifiably Encrypted Signatures

Markus Rückert* Dominique Schröder†

Department of Computer Science
TU Darmstadt

February 22, 2009

Abstract. In a verifiably encrypted signature scheme, signers encrypt their signature under the public key of a trusted third party and prove that they did so correctly. The security properties are unforgeability and opacity. Unforgeability states that a malicious signer should not be able to forge verifiably encrypted signatures and opacity prevents extraction from an encrypted signature.

This paper proposes two novel fundamental requirements for verifiably encrypted signatures, called *extractability* and *abuse-freeness*, and analyze its effects on the security model of Boneh et al. Extractability ensures that the trusted third party is always able to extract a valid signature from a valid verifiably encrypted signature and abuse-freeness guarantees that a malicious signer, who cooperates with the trusted party, is not able to forge a verifiably encrypted signature. We further show that both properties are not covered by the model of Boneh et al., introduced at Eurocrypt 2003.

Keywords. Verifiably encrypted signatures, Online contract signing, Security Model

1 Introduction

The concept of verifiably encrypted signature (VES) schemes was proposed by Boneh, Gentry, Lynn, and Shacham [BGLS03] at Eurocrypt 2003. There, a signer encrypts its signature under the public key of a trusted third party, the adjudicator, and attaches a proof about its content. One popular application for verifiably encrypted signature is online contract signing, which is a type of optimistic fair exchange protocol [ASW00, BDM98]. Suppose Alice and Bob wish to sign the same contract. Both want to be sure that the other party will also produce a signature before revealing their own. Following the protocol, Alice and Bob exchange verifiably encrypted signatures. After they have ascertained themselves of the correctness of the encrypted signature, they reveal the corresponding ordinary signature. If, for example, Alice is not willing to disclose her signature then Bob can take her verifiably encrypted signature together with the transcript to the adjudicator, who uncovers Alice's ordinary signature. This fail-safe mechanism prevents Alice from misusing this one-sided

*rueckert@cdc.informatik.tu-darmstadt.de

†schroeder@me.com – The author was supported by the Emmy Noether Programme Fi 940/2 of the German Research Foundation (DFG).

commitment to a contract for legal actions, blackmail, or simply negotiating a better deal elsewhere.

Security of verifiably encrypted signatures is defined via unforgeability and opacity [BGLS03]. Roughly speaking, unforgeability assures that a malicious user cannot produce signatures on behalf of another party. Opacity guarantees that only the adjudicator and the signer can disclose a signature from a verifiably encrypted signature.

Surprisingly, the original security model does *not* guarantee that the adjudicator is always able to extract a valid signature from a valid verifiably encrypted signature. We show that every VES can easily be turned into a scheme which remains secure, but where a malicious signer can output a verifiably encrypted signature such that the ordinary signature is hidden irrecoverably. This is disastrous for fair exchange protocols, because it implies that a VES, which does not support extractability, is not suitable for such protocols. Thus, as our first result, we extend the model of [BGLS03] to ensure *extractability*. Afterwards, we study the effect of extractability on the security model, showing that the unforgeability of the underlying signature scheme often implies unforgeability of the verifiably encrypted signature scheme as long as the scheme is extractable. Though there is no explicit proof of extractability in the known constructions [BGLS03] and [LOS+06], they already support the property due to the similarity of the signature verification algorithm and the verification of verifiably encrypted signatures in their schemes.

We further propose the definition of *abuse-freeness*. Basically, an abuse-free VES guarantees that an adversary who cooperates with the adjudicator is not able to derive a verifiably encrypted signature on behalf of an honest signer. We show that for a "natural" class of VES schemes, abuse-freeness is already implied. Since the instantiation of [BGLS03] and [LOS+06] fall into this class, our results give more confidence about the security of their schemes.

Garay, Jakobsson, and MacKenzie already considered abuse-freeness in the context of optimistic contract signing [GJM99]. Their definition demands that no single signer should be able to prove to any third party that he can determine the outcome of the protocol. Since verifiably encrypted signature schemes are typically non-interactive, and because the verification equation assures that the contained signature is valid, this definition seems not to be applicable to the scenario of VES.

Organization. We start out by introducing our notation and some basic definitions in Section 2. In Section 3, we recall the model for verifiably encrypted signatures along with the security definitions. In the subsequent section, Section 4, we introduce a new property, *extractability*, to the model of Boneh et al. and argue why this is necessary. Finally, Section 5 deals with abuse-freeness in the context of verifiably encrypted signatures and we discuss the influence of extractability on this additional requirement.

2 Notation and Basic Definitions

By $a_1 \| \ldots \| a_\ell$ we denote the encoding of $a_1, \ldots, a_\ell$ such that $a_1, \ldots, a_\ell$ are uniquely recoverable. Furthermore, n always denotes the security parameter.

2.1 Secure Signature Schemes

Recall that a digital signature scheme DSig is defined as:

Definition 2.1 *A signature scheme consists of a triple of efficient algorithms* $\mathsf{DSig} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$*, where*

Key Generation: $\mathsf{Kg}(1^n)$ *outputs a private signing key sk and a public verification key pk.*

Signature Generation: $\mathsf{Sign}(sk, m)$ *outputs a signature* σ *on a message* m *from the message space* $\mathcal{M}$ *under sk.*

Signature Verification: *The algorithm* $\mathsf{Vf}(pk, \sigma, m)$ *outputs 1 iff* σ *is a valid signature on* m *under pk.*

Signature schemes are complete if for any $(sk, pk) \leftarrow \mathsf{Kg}(1^n)$*, any message* $m \in \mathcal{M}$*, and any* $\sigma \leftarrow \mathsf{Sign}(sk, m)$*, we have* $\mathsf{Vf}(pk, \sigma, m) = 1$*.*

Security of signature schemes is proven against existential forgery under chosen message attacks (EU-CMA) [GMR88]. In this model, an adversary adaptively invokes a signing oracle and is successful if it outputs a signature on a *new* message.

Definition 2.2 *A signature scheme* $\mathsf{DSig} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$ *is called* unforgeable under chosen message attacks *(EU-CMA) if for any efficient algorithm* $\mathcal{A}$ *the probability that the experiment* $\mathsf{sForge}_{\mathcal{A}}^{\mathsf{DSig}}$ *evaluates to 1 is negligible.*

Experiment $\mathsf{sForge}_{\mathcal{A}}^{\mathsf{DSig}}(n)$
$(sk, pk) \leftarrow \mathsf{Kg}(1^n)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(sk,\cdot)}(pk)$
Return 1 *iff* $\mathsf{Vf}(pk, m^*, \sigma^*) = 1$ *and* $\mathcal{A}$ *never queried* $\mathsf{Sign}(sk, \cdot)$ *about* m^**.*

A signature scheme DSig *is* (t, q_S, ϵ)-secure *if no adversary running in time at most* t*, invoking the signing oracle at most* q_S *times, outputs a valid forgery* (m^*, σ^*) *with probability larger than* ϵ*.*

2.2 Trapdoor Functions

We briefly recall the definition of trapdoor functions. Let D be a group and let a family Π of trapdoor functions over D is defined as a triple of algorithms *Generate*, *Evaluate*, and *Invert*. The randomized algorithm *Generate* outputs the description s of a trapdoor function along with the corresponding trapdoor t. The evaluation algorithm *Evaluate* takes as input the description s of the function and a value $x \in D$. It outputs the image $a \in D$ under the trapdoor function. The input of the inversion algorithm *Invert* is a description of the function s, the trapdoor t, and a value $a \in D$. It returns the preimage of a under the function.

We require that $\mathit{Invert}(s, t, \mathit{Evaluate}(s, x)) = x$ holds for all $(s, t) \leftarrow \mathit{Generate}$ and that all algorithms are efficient. In the following, we define the advantage of an algorithm $\mathcal{A}$ in inverting a trapdoor.

Definition 2.3 *The advantage of an algorithm* $\mathcal{A}$ *in inverting a trapdoor family is*

$$\mathsf{Adv}\,\mathsf{Invert}_{\mathcal{A}} := \mathrm{Prob}[\, x = \mathcal{A}(s, \mathit{Evaluate}(s, x)) \,:\, (s, t) \leftarrow \mathit{Generate},\, x \leftarrow D] \;.$$

The probability is taken over the coin tosses of Generate and of $\mathcal{A}$*. An algorithm* $\mathcal{A}$ (t, ϵ)*-inverts a trapdoor function family if* $\mathcal{A}$ *runs in time at most* t *and the advantage* $\mathsf{Adv}\,\mathsf{Invert}_{\mathcal{A}}$ *is at least* ϵ*. We say that a trapdoor function is family is* (t, ϵ)*-one-way if no algorithm* (t, ϵ)*-inverts the trapdoor function family.*

3 Verifiably Encrypted Signatures

Verifiably encrypted signature schemes support the encryption of signatures under the public key of a trusted third party, while simultaneously proving that the encryption contains the signature. More precisely:

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

Key Generation, Signing, and Verification: Same as in a digital signature scheme.

Adjudicator Key Generation: $\mathsf{AdjKg}(1^n)$ outputs a key pair (ask, apk), where ask is the private key and apk the corresponding public key.

VES Creation: $\mathsf{Create}(sk, apk, m)$ takes as input a secret key sk, the adjudicator's public key apk, and a message $m \in \{0,1\}^n$. It returns a verifiably encrypted signature ω on m.

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

Adjudication: The algorithm $\mathsf{Adj}(ask, apk, pk, \omega, m)$ accepts as input the key pair (ask, apk) of the adjudicator, the public key of the signer pk, a verifiably encrypted signature ω, and a message m. It extracts an ordinary signature[1] σ on m and returns σ.

Definition 3.1 *A verifiably encrypted signature scheme is* complete[2]*, i.e. for all adjudication key pairs* $(ask, apk) \leftarrow \mathsf{AdjKg}(1^n)$ *and for all signature key pairs* $(sk, pk) \leftarrow \mathsf{Kg}(1^n)$ *the following holds:*

$$\mathsf{VesVf}(apk, pk, \mathsf{Create}(sk, apk, m), m) = 1 \qquad \textit{and}$$

$$\mathsf{Vf}(pk, \mathsf{Adj}(ask, apk, pk, \mathsf{Create}(sk, apk, m)), m) = 1 \ \textit{ for all } m \in \mathcal{M}.$$

Security Model. Security of verifiably encrypted signatures is defined through unforgeability and opacity [BGLS03]. *Unforgeability* requires that it is hard to forge a verifiably encrypted signature and *opacity* implies that it is difficult to extract an ordinary signature from a verifiably encrypted signature.

Both intuitions are formalized in experiments, where the adversary is given the public keys of the signer and of the adjudicator. Moreover, the adversary has access to two oracles: oracle C returns verifiably encrypted signatures for a given message; oracle A extracts a verifiably encrypted signature and returns a corresponding ordinary signature.

Definition 3.2 *A verifiably encrypted signature* $\mathsf{VES} = (\mathsf{Kg}, \mathsf{AdjKg}, \mathsf{Sign}, \mathsf{Vf}, \mathsf{Create}, \mathsf{VesVf}, \mathsf{Adj})$ *scheme is called* secure *if the following holds:*

Unforgeability: *For any efficient algorithm $\mathcal{A}$, the probability that the following experiment evaluates to 1 is negligible.*

[1] Not necessarily the same signature, cf. [LOS+06].

[2] Note that in [BGLS03] this condition is called validity.

Experiment $\mathsf{VesForge}^{\mathsf{VES}}_{\mathcal{A}}(n)$
$(ask, apk) \leftarrow \mathsf{AdjKg}(1^n)$
$(sk, pk) \leftarrow \mathsf{Kg}(1^n)$
$(m^*, \omega^*) \leftarrow \mathcal{A}^{\mathsf{C}(sk,apk,\cdot),\mathsf{A}(ask,apk,pk,\cdot,\cdot)}(pk, apk)$
Return 1 *iff* $\mathsf{VesVf}(apk, pk, \omega^*, m^*) = 1$ *and*
$\mathcal{A}$ *has never queried* $\mathsf{C}(sk, apk, \cdot)$ *or* $\mathsf{A}(ask, apk, pk, \cdot, \cdot)$ *about* m^*.

Opacity: *For any efficient algorithm* $\mathcal{A}$*, the probability that the following experiment evaluates to 1 is negligible.*

Experiment $\mathsf{Opac}^{\mathsf{VES}}_{\mathcal{A}}(n)$
$(ask, apk) \leftarrow \mathsf{AdjKg}(1^n)$
$(sk, pk) \leftarrow \mathsf{Kg}(1^n)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{C}(sk,apk,\cdot),\mathsf{A}(ask,apk,pk,\cdot,\cdot)}(pk, apk)$
Return 1 *iff* $\mathsf{Vf}(pk, \sigma^*, m^*) = 1$ *and*
$\mathcal{A}$ *has never queried* $\mathsf{A}(ask, apk, pk, \cdot, \cdot)$ *about* m^*.

A scheme is called $(t, q_{\mathsf{C}}, q_{\mathsf{A}}, \epsilon)$*-unforgeable (-opaque), if no adversary, running in time at most* t*, making at most* q_{C} *verifiably encrypted signature oracle queries, and at most* q_{A} *adjudication oracle queries, can succeed with probability at least* ϵ *in the* VesForge (Opac) *experiment.*

4 The Need for Extractability

In the following, we propose a fundamental requirement of verifiably encrypted signatures, called *extractability*. It states that if a verifiably encrypted signature ω is valid then the adjudicator is always able to extract a valid signature σ. Note that this property is essential for protocols like optimistic fair exchange. We formalize extractability as follows.

Definition 4.1 *A verifiably encrypted signature scheme* VES *is* extractable *if for any efficient algorithm* $\mathcal{A}$*, the probability that the following experiment evaluates to 1 is negligible.*

Experiment $\mathsf{Extract}^{\mathsf{VES}}_{\mathcal{A}}(n)$
$(ask, apk) \leftarrow \mathsf{AdjKg}(1^n)$
$(m^*, \omega^*, pk^*) \leftarrow \mathcal{A}^{\mathsf{A}(ask,apk,\cdot,\cdot,\cdot)}(apk)$
Let $\sigma^* \leftarrow \mathsf{Adj}(ask, apk, pk^*, \omega^*, m^*)$
Return 1 *iff* $\mathsf{VesVf}(apk, pk^*, \omega^*, m^*) = 1$ *and* $\mathsf{Vf}(pk^*, \sigma^*, m^*) = 0$.

Observe that, in this case, the adjudication oracle A takes as input a tuple (pk, ω, m) which consists of a public key pk, a verifiably encrypted signature ω, and a message m. Thus, extractability must hold for all pairs (m, ω), even for those not properly generated and even in case pk^* is not chosen honestly. Similarly, we define *weak-extractability*, where the adversary is not allowed to choose its public key dishonestly. Note that a scheme that satisfies weak-extractability can always be turned into an extractable scheme by having the signer prove the correct form of its public key to the (universally trusted) adjudicator. The adjudicator may then sign the public key or otherwise vouch for its validity. We motivate the need for this new property, showing that every verifiably encrypted signature scheme, secure in the model of [BGLS03], can simply be turned into one which is not extractable.

Theorem 4.2 *If there exists a secure scheme* VES*, then there exists a scheme* VES' *which is* not *extractable.*

Proof. The basic idea is that the verifiably encrypted signature may consist of two *independent* parts. One part is the encrypted signature and the other part forms the proof. As both parts are independent of each other, a malicious signer can easily set the encrypted signature to an empty string while computing the proof honestly. We assume that the bit length of a verifiably encrypted signature is $\mathsf{out}(n)$.

Key Generation, Signing, Verification: Same as in VES.

VES Creation: Given a message $m \in \{0,1\}^n$, a signing key sk, and the public key of the adjudicator apk. Create' computes $\omega' \leftarrow \mathsf{Create}(sk, apk, m)$ and outputs $(\omega_1 \| \omega_2) \leftarrow (\omega' \| \omega') \in \{0,1\}^{2\,\mathsf{out}(n)}$.

VES Verification: Given a verifiably encrypted signature $\omega_1 \| \omega_2$ on m, algorithm VesVf' outputs 1 iff $\mathsf{VesVf}(apk, pk, \omega_1, m)$ evaluates to 1.

Adjudication: $\mathsf{Adj}'(ask, apk, pk, \omega_1 \| \omega_2, m)$ outputs $\sigma \leftarrow \mathsf{Adj}(ask, apk, pk, \omega_2, m)$.

Obviously, if VES is complete, unforgeable, and opaque, so is VES'. Though, now, the following adversary $\mathcal{A}$ contradicts extractability.

Setup: $\mathcal{A}$ receives the adjudicator's public key apk and (honestly) generates its signature key $(sk, pk) \leftarrow \mathsf{Kg}(1^n)$.

VES Creation: When $\mathcal{A}$ signs a message m, it calls $\omega_1 \| \omega_2 \leftarrow \mathsf{Create}'(sk, apk, m)$ and outputs $(m^*, \omega^*, pk^*) \leftarrow (m, \omega_1 \| 0^{\mathsf{out(n)}}, pk)$.

Since ω_1 remains unchanged in Create', VesVf' always returns 1. The algorithm Adj', however, cannot extract a valid (ordinary) signature out of the second part of the verifiably encrypted signature because it is 0. Thus, $\mathcal{A}$ breaks extractability with probability 1. □

To conclude, we have shown that security and completeness in the model of Boneh et al. *do not imply extractability* and therefore do not suffice.

4.1 Implications

In this section, we show that the unforgeability of the underlying signature scheme already implies unforgeability for all verifiably encrypted signature schemes with a common property. Furthermore, we remove a restriction from the definition of unforgeability. We prove that the condition that the adversary is not allowed to output a forgery m^* for a message already queried to the adjudication oracle (and *not* to the creation oracle), is unnecessary.

In order to prove these theorems, we need the following property which we call key-independence. It states that computing the encrypted signature can be performed by two algorithms, independently. The first one computes the signature σ as in DSig and the second algorithm the verifiable encryption of it. More precisely:

Definition 4.3 (Key-Independence) *A verifiably encrypted signature scheme* VES *is called* key-independent *if there exists an efficient algorithm* Enc *such that*

$$\mathsf{Enc}(apk, \mathsf{DSig.Sign}(sk, m), m) \equiv \mathsf{Create}(sk, apk, m) \quad \textit{for all } m \in \mathcal{M}\,.$$

Note that the encryption algorithm Enc may also depend on the a second secret sk' that is not part of the private signing key sk. We stress that most schemes are key-independent, such as the scheme of Boneh et al. [BGLS03] or the scheme of Lu et al. [LOS+06].

Theorem 4.4 *Let* VES *be an extractable, key-independent verifiably encrypted signature scheme defined through the following algorithms* $(\mathsf{Kg}, \mathsf{AdjKg}, \mathsf{Sign}, \mathsf{Vf}, \mathsf{Create}, \mathsf{VesVf}, \mathsf{Adj})$*. The scheme* VES *is unforgeable if and only if the underlying signature scheme* $\mathsf{DSig} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$ *is unforgeable.*

Proof. In order to prove this theorem, we have to show two directions. We begin with the (interesting) direction, showing that the existence of and algorithm $\mathcal{A}_1$, forging the verifiably encrypted signature, implies the existence of an adversary $\mathcal{B}$, breaking the underlying signature scheme.

Algorithm $\mathcal{B}$ gets as input the public key pk of the underlying signature scheme DSig, picks a key-pair for the simulation of the adjudicator $(ask, apk) \leftarrow \mathsf{AdjKg}(1^n)$, and simulates $\mathcal{A}_1$ in a black-box way on input (apk, pk).

During the simulation, $\mathcal{A}_1$ may invoke its creation oracle C on a message m. Algorithm $\mathcal{B}$ answers this query as follows. It first generates the signature $\sigma \leftarrow \mathsf{Sign}(sk, m)$ with the help of its external signing oracle $\mathsf{Sign}(sk, \cdot)$ and computes the verifiably encrypted signature $\omega \leftarrow \mathsf{Enc}(apk, \sigma)$. Whenever $\mathcal{A}_1$ invokes its adjudication oracle A on a (valid) tuple (m, ω), then algorithm $\mathcal{B}$ returns $\sigma \leftarrow \mathsf{Adj}(ask, apk, pk, \omega, m)$. Eventually, $\mathcal{A}_1$ stops, outputting a tuple (m^*, ω^*), then $\mathcal{B}$ computes $\sigma^* \leftarrow \mathsf{Adj}(ask, apk, pk, \omega^*, m^*)$ and outputs (m^*, σ^*).

For the analysis, it is assumed that $\mathcal{A}_1$ succeeds with non-negligible probability $\epsilon(n)$. Observe that $\mathcal{B}$ performs a perfect simulation from $\mathcal{A}_1$'s point of view because the scheme VES is key-independent. Note that $\mathcal{A}_1$ succeeds if its outputs a "fresh" tuple (m^*, ω^*), i.e., $\mathcal{A}_1$ has neither queried its creation oracle nor the adjudication oracle about m^*. But if $\mathcal{A}_1$ has never sent m^* to one of the oracles, then $\mathcal{B}$ has never queried its singing oracle about m^*. Since the scheme VES is extractable, $\mathcal{B}$ always outputs a valid signature whenever $\mathcal{A}_1$ generates a valid verifiably encrypted signature. This, however, contradicts the assumption that the signature scheme is unforgeable.

The other direction shows how break unforgeability of the verifiably encrypted signature scheme with the help of an adversary $\mathcal{A}_2$ that forges the underlying signature scheme. The idea of the proof is to encrypt the forgery of $\mathcal{A}_2$ by executing the algorithm Enc (which is possible because VES is key-independent). We omit the proof. □

Roughly speaking, the next theorem states that the adjudication oracle does not help to forge verifiably encrypted signatures. This means that the restriction in the unforgeability experiment, which requires that the adversary must output a "new" message m^* (that has never been sent to the adjudication oracle), is superfluous. One might think that the adversary against the unforgeability of the VES scheme succeeds because it manages to modify some "ciphertext" such that the adjudicator extracts a "fresh" message-signature pair. The adversary then simply "encrypts" this signature and wins the game. Here, however, we show that this intuition is wrong. If the adversary invokes the adjudication oracle on a "fresh" and valid tuple (m, ω) (i.e., it has never queried its creation oracle about m), then we can already forge the verifiably encrypted signature scheme.

To prove this theorem formally, let *adjudication-free unforgeability* be defined as unforgeability of verifiably encrypted signature schemes, except that the adversary wins the experiment even if it queries its adjudication oracle with the message m^*.

Theorem 4.5 *A verifiably encrypted signature scheme is unforgeable if and only if it is adjudication-free unforgeable.*

Proof (sketch). The first direction is to prove that an adversary, which breaks unforgeability can be used to break adjudication-free unforgeability. We omit this part because it straightforward. In the second part of the proof, we construct an algorithm $\mathcal{B}$ against unforgeability, which runs an adversary $\mathcal{A}$ that succeeds in the adjudication-free unforgeability game. Algorithm $\mathcal{B}$ answers all oracle queries with its own oracles. Whenever $\mathcal{A}$ invokes the adjudication oracle A on a "fresh" and valid pair (m^*, ω^*) (i.e., the adversary has never sent m^* to the creation oracle), then $\mathcal{B}$ stops, outputting this pair as its forgery. Otherwise, if $\mathcal{A}$ never performs such queries, $\mathcal{B}$ forwards the final output of $\mathcal{A}$. □

5 The Need for Abuse-Freeness

Roughly speaking, abuse-freeness means that an adversary, who may cooperate with the (possibly malicious) adjudicator, is not able to compute a verifiably encrypted signature on behalf of the other signer. We model this intuition in an experiment where the malicious signer $\mathcal{A}$ receives the private key of the adjudicator and the public key of the honest signer. It succeeds if it outputs a "fresh" tuple (m^*, ω^*), i.e., the attacker $\mathcal{A}$ has never invoked its creation oracle about m^*. Observe that given the algorithm $\mathcal{A}$ access to an adjudication oracle is redundant since $\mathcal{A}$ can simulate this oracle with the private key *ask*.

Definition 5.1 *A verifiably encrypted signature scheme* VES *is called* abuse-free *if for any efficient algorithm* $\mathcal{A}$ *the probability that experiment* $\mathsf{Abuse}_{\mathcal{A}}^{\mathsf{VES}}$ *evaluates to 1 is negligible, where*

Experiment $\mathsf{Abuse}_{\mathcal{A}}^{\mathsf{VES}}(n)$
$(apk, ask) \leftarrow \mathsf{AdjKg}(1^n)$
$(sk, pk) \leftarrow \mathsf{Kg}(1^n)$
$(m^*, \omega^*) \leftarrow \mathcal{A}^{\mathsf{C}(sk, apk, \cdot)}(apk, ask, pk)$
Return 1 *iff* $\mathsf{VesVf}(apk, pk, \omega^*, m^*) = 1$ *and*
$\mathcal{A}$ *has never queried* $\mathsf{C}(sk, apk, \cdot)$ *about* m^*.

This definition can easily strengthened in the sense that the adversary $\mathcal{A}$ may choose the public of the adjudicator *apk*. We call schemes satisfying the stronger notion *strongly abuse-free*.

5.1 Relation to the Security Model

We study in this section the relation between abuse-freeness and the other definitions. The interesting point is that for a "natural" class of verifiably encrypted signature scheme (which we call key-independent) abuse-freeness is already guaranteed as long as the scheme is extractable.

Theorem 5.2 *If there exist secure verifiably encrypted signature schemes, and if there exist trapdoor functions, then there exists a secure verifiably encrypted signature scheme which is* not *abuse-free.*

Proof. The idea of the proof is as follows. We build a verifiably encrypted signature scheme VES' out of a secure scheme VES such that VES' remains secure but such a malicious adjudicator is able to reveal the private signing key.

Construction 5.3 *Let* $\mathsf{VES} = (\mathsf{Kg}, \mathsf{AdjKg}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Create}, \mathsf{VesVf}, \mathsf{Adj})$ *be a verifiably encrypted signature scheme and let* F *be a one-way trapdoor function. Define the following algorithms:*

Key Generation: *The algorithms* Kg' *is identical to* Kg*. The key generation algorithm for the adjudicator* AdjKg' *first generates* $(ask, apk) \leftarrow \mathsf{AdjKg}$ *and runs the generation algorithm of the trapdoor one-way function* $(s,t) \leftarrow$ *Generate. It sets* $(ask', apk') \leftarrow ((ask,t),(apk,s))$*.*

VES **Creation:** *The input of the algorithm* $\mathsf{Create}'(sk, apk', m)$ *is a private signing key* sk*, the public key of the adjudicator* $apk' = (apk, s)$ *and a message* m*. It first executes the underlying creation algorithm* $\omega' \leftarrow \mathsf{Create}(sk, apk', m)$ *and then the evaluation algorithm* $a \leftarrow$ *Evaluate*(s, sk)*. It outputs the verifiably encrypted signature* $\omega' \leftarrow (\omega, a)$*.*

VES **Verification:** *The algorithm* $\mathsf{VesVf}'(apk', pk, \omega', m)$ *takes as input the public key of the adjudicator* $apk' = (apk, s)$*, the public key of the signer* pk*, a verifiably encrypted signature* $\omega' = (\omega, a)$ *together with a message* m*. It outputs the result of the underlying verification algorithm* $\mathsf{VesVf}(apk', pk, \omega, m)$*.*

Adjudication: *The algorithm* $\mathsf{Adj}'(ask', apk', pk, \omega', m)$ *accepts as input the private key of the adjudicator* $ask' = (ask, t)$*, the signer's public key* pk*, a verifiably encrypted signature* $\omega' = (\omega, a)$ *and the corresponding message* m*. It outputs the result of the underlying adjudication algorithm* $\mathsf{Adj}(ask, pk, \omega, m)$*.*

Completeness, unforgeability, and opacity of VES' follow easily form the underlying scheme VES. Observe that the one-wayness of the trapdoor function guarantees that the adversary cannot extract the signing key in order to forge a verifiably encrypted signature. Such an adversary could be used to invert the one-way function. Next, we show that the resulting scheme is not abuse-free.

The adversary $\mathcal{A}$ breaking abuse-freeness gets as input an adjudicator key-pair $(ask, apk) \leftarrow \mathsf{AdjKg}'(1^n)$ together with a public key pk from the honest signer. It then selects two arbitrary messages m_1, m_2 and invokes the creation oracle on m_1 obtaining the verifiably encrypted signature $\omega = (\omega', a)$. Algorithm $\mathcal{A}$ then extracts the private singing key with the help of the trapdoor $sk \leftarrow$ *Invert*(t, a). Next, $\mathcal{A}$ uses this private key in order to compute the verifiably encrypted signature $\omega^* \leftarrow \mathsf{Create}(sk, apk, m_2)$ for the message m_2 and outputs (ω^*, m_2) as its forgery.

A straightforward analysis shows that the algorithm $\mathcal{A}$ is efficient and succeeds with probability 1. □

In the following, we prove an interesting result, namely, that a key-indepent verifiably encrypted signature scheme which is unforgeable and extractable, already achieves abuse-freenes.

Theorem 5.4 *A key-independent and extractable verifiably encrypted signature scheme is abuse-free, if the underlying signature scheme is unforgeable.*

Proof. Suppose that there exists an adversary $\mathcal{A}$ against abuse-freeness that succeeds with noticeable probability. We then show how to forge the underlying signature scheme. More precisely, let $\mathsf{VES} = (\mathsf{Kg}, \mathsf{AdjKg}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Create}, \mathsf{VesVf}, \mathsf{Adj})$ be a key-independent and extractable VES that is unforgeable but which is *not* abuse-free. Let $\mathsf{DSig} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Verify})$

be the underlying signature scheme. Algorithm $\mathcal{B}$ against unforgeability of the underlying signature scheme gets as input a public key pk. It generates a key-pair for the adjudicator $(apk, ask) \leftarrow \mathsf{AdjKg}(1^n)$ and runs a black-box simulation of $\mathcal{A}$ on input (apk, ask, pk). Whenever $\mathcal{A}$ queries its creation oracle C about a message m, then $\mathcal{B}$ answers this query with the help of its external signing oracle $\sigma \leftarrow \mathsf{Sign}(sk, \cdot)$ and with the key-independent encryption algorithm $\omega \leftarrow \mathsf{Enc}(apk, \sigma)$. Finally, $\mathcal{A}_1$ stops, outputting a pair (m^*, ω^*). Algorithm $\mathcal{B}$ derives the signature executing the adjudication algorithm $\sigma^* \leftarrow \mathsf{Adj}(ask, apk, pk, \omega^*, m^*)$ and returns (m^*, σ^*).

We assume for the analysis that $\mathcal{A}$ succeeds with noticeable probability $\epsilon(n)$ and observe that $\mathcal{B}$ is efficient. According to our assumption that VES is key-independent, we conclude that $\mathcal{B}$ performs a perfect simulation from $\mathcal{A}$'s point of view. Furthermore, we know that the output (m^*, ω^*) from $\mathcal{A}$ is extractable, thus $\mathcal{B}$ succeeds whenever $\mathcal{A}$ does. This, however, contradicts the assumption that DSig is unforgeable.

□

References

[ASW00] 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.

[BDM98] Bao, Deng, and Mao. *Efficient and Practical Fair Exchange Protocols with Off-Line TTP.* RSP: 19th IEEE Computer Society Symposium on Research in Security and Privacy. IEEE Computer Society Press, 1998.

[BGLS03] 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.

[GJM99] Juan A. Garay, Markus Jakobsson, and Philip D. MacKenzie. *Abuse-Free Optimistic Contract Signing.* Advances in Cryptology — Crypto'99, Lecture Notes in Computer Science, pages 449–466. Springer-Verlag, 1999.

[GMR88] 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.

[LOS+06] 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.