

Verifiably Encrypted Signatures from RSA without NIZKs

Markus Rückert*

rueckert@cdc.informatik.tu-darmstadt.de

Cryptography and Computeralgebra
Department of Computer Science
TU Darmstadt

Abstract. Verifiably encrypted signature (VES) schemes allow a signer to encrypt a signature under the public key of a trusted party, the adjudicator, while maintaining public signature verifiability *without* interactive proofs. A popular application for this concept is fair online contract signing.

This paper answers the question of whether it is possible to implement a VES without pairings and zero-knowledge proofs. Our construction is based on RSA signatures and a Merkle hash tree. Hence, the scheme is stateful but relies on relatively mild assumptions in the random oracle model. Thus, we provide an alternative that does not rely on pairing-based assumptions.

The advantage of our approach over previous schemes is that widespread efficient hard- and software implementations of hash functions and RSA signatures can be easily reused for VES, i.e., we can avoid costly re-development. Furthermore, in contrast to using non-interactive zero-knowledge proofs, we only need a constant, small number of modular exponentiations.

Keywords Online contract signing, Merkle hash trees, RSA

* This work was supported by CASED (www.cased.de).

1 Introduction

Verifiably encrypted signature (VES) schemes were introduced by Boneh, Gentry, Lynn, and Shacham [7]. They are built upon regular, i.e., non-encrypted, signature schemes and preserve their public verifiability, while hiding the signature itself from the public. There are three parties involved: the signer, the receiver, and a passive adjudicator. The signer computes a signature σ on a document m and outputs a masked signature ϖ . The general public can verify that ϖ contains a signature on m , but is unable to extract a valid regular signature. This also holds for the designated receiver. A popular application of this concept is *online contract signing* — an optimistic fair exchange protocol [3,1]. We assume two signers, who do not unconditionally trust each other, want to sign a contract. However, neither of them wants to sign first because each fears that the other party might back out at the last possible moment with a partly signed contract. Such a one-sided commitment to a contract may be used for blackmail or simply for negotiating a better deal elsewhere. In order to avoid this situation, both parties agree on the following protocol: 1. Exchange verifiably encrypted signatures; 2. Verify the encrypted signatures; 3. Exchange regular signatures.

In the case of a dispute, either party can appeal to the adjudicator, who is able to disclose both regular signatures. In contrast to other fair exchange protocols, VES schemes perform the initial signature exchange step more efficiently, in a single move.

Security of verifiably encrypted signatures comprises unforgeability and opacity [7], as well as extractability and abuse-freeness [18]. In [15], Huang et al. state that such protocols should be “ambiguous”, which is not guaranteed by VES.

Along with the first security model, Boneh et al. proposed the BGLS construction that is provably secure in the random oracle model. Later on, Zhang et al. described a more efficient solution (ZSNS) [22] that seems to be provably secure in the random oracle model but lacks a formal proof of opacity. Lu et al. presented the first verifiably encrypted signature scheme (LOSSW) [16] that is secure in the standard model. The LOSSW scheme is based on the Waters signature scheme [21], which has a fairly large public key. In [18], Rückert and Schröder (RS) show a construction in the standard model with short keys. Lu et al. also sketch a generic construction using non-interactive zero-knowledge proofs (NIZKs). Since such constructions are typically very inefficient with respect to computational cost (esp. the number of modular exponentiations) and signature size, our work strictly focuses on direct and practical instantiations.

There is a second line of work on “verifiable encryption” in a more general scenario as described in, e.g., [11] or [2]. However, their objectives differ from the one in [7] as Boneh et al.’s work demands that transmitting and verifying an encrypted signature can be done in a single move. In particular, the verification process does not involve interactive zero-knowledge proofs as required in both [11] and [2]. Therefore, we focus on the line of research coming from Boneh et al.’s model.

To the best of our knowledge, all previous schemes use bilinear maps that yield very elegant constructions at the cost of new cryptographic assumptions,

i.e., the bilinear Diffie-Hellman assumptions discussed in [6,13,19]. All these papers suggest that correctly applying and fully understanding these assumptions involves some pitfalls and it is safe to assume that this also applies to hardware and software implementations of the respective schemes. Furthermore, compared to hash functions and RSA signatures, the field of pairing based cryptography is young, which makes the newly developed complexity assumptions questionable.

Our contribution. Until now, it has been an open question whether efficient VES schemes in the sense of Boneh et al. can be realized without bilinear maps by using well-understood assumptions instead. We positively answer this question with our construction using full-domain hash RSA signatures [5] and a Merkle-style hash tree [17], which makes our scheme stateful. We also suggest a change to the signer’s key generation algorithm that involves an initial registration with the adjudicator, which we believe to be a sensible and practical extension.

When instantiated correctly, our construction is efficient and can even be implemented on smartcards, having RSA and hash co-processors. Compared to the previous constructions, our approach yields competitive performance for creation and verification of verifiably encrypted signatures. The key generation process in our construction, however, is costly and should be done on a more powerful device. This recommendation does not contradict the common requirement that private signing keys are not allowed to leave the signing device, such as a smartcard, as the involved tree can be computed independently of the secret, security-sensitive RSA key.

Table 1 shows that only ZSNS has a faster `Create` method when omitting the constant (independent of the security parameter) number of hash evaluations in our scheme. NIZK constructions for RSA are not considered here because they involve many inefficient modular exponentiations and the number of ring elements in the signature grows with the security parameter. Moreover, using the secret key that often might increase the chance of a successful side-channel attack. Performance of the verification functions is hardly comparable but previous works suggest that pairing evaluations are costly. We, however, only need a constant number of typically very fast hash evaluations here.

As for the storage requirements, we state that they are slightly elevated in our scheme because of the authentication tree. The secret key can be compressed into a single random seed (for the secret leaves of the tree) and a secret RSA exponent. The public key comprises the public RSA key, an RSA signature, and one additional hash value. Note that a certain part of the user’s key has to be trusted and will therefore be generated by the adjudicator. However, this does not limit the security of the signature scheme but merely allows the adjudicator to work correctly in case of a dispute.

As for the signature size, our scheme requires an overhead due to the Merkle authentication path. The overhead, however, is constant and does not grow with the security parameter. For instance, the tree depth ℓ can be set as small as 12, even if we assume that the signer works 365 days a year and signs 11 documents per day, where the key is also valid for one year. By increasing ℓ , the signature capability (2^ℓ) quickly becomes virtually unlimited. In the Merkle signature

VES	R.O.	Keys (sk/pk)	Signature	Create	VesVf
BGLS	YES	G / G	$2G$	$2 \text{ ex} + \text{ mul}$	3 pair
ZSNS	YES	$2G / 2G$	G	ex	2 pair
LOSSW	NO	$G / > 160G$ (*)	$3G$	$4 \text{ ex} + (\text{ham}(m) + 3) \text{ mul}$	3 pair
RS	NO	$2G / 4G$	$3G$	$4 \text{ ex} + 1 \text{ mul}$	2 pair
Section 3	YES	$R + S / 3R + H$	$3R + \ell H$	$2 \text{ ex} + \text{ mul} + \mathcal{O}(\ell) \text{ hash}$	$2 \text{ ex} + (\ell + 1) \text{ hash}$

Table 1. Comparison of the different verifiably encrypted signature schemes. The column “R.O.” states whether security is proven in the random oracle model. Let $\text{ham}(m)$ be the hamming weight of a message m , mul a multiplication, and ex an exponentiation in the respective group or ring. Let pair be the cost for a pairing evaluation and hash be the cost for a hash function evaluation. The sizes are measured in group elements (G), ring elements (R), and hash values (H). S is the random seed to a pseudo random number generator. The parameter ℓ is a tree constant, e.g., $\ell = 20$ for 2^{20} verifiably encrypted signatures.

(*) Depends on the size of the message space.

context, one typically sets $\ell = 20$, which allows for over a million signatures at manageable computational cost. In our case, we can manage even larger ℓ .

Finally, we state that it remains unclear whether it is possible to avoid both pairings and the random oracle methodology, which is encouraged by the famous work of Canetti, Goldreich, and Halevi [12].

Organization. We start by introducing our notation and some basic definitions in Section 2 before presenting our instantiation of verifiably encrypted signatures and proving its security in Section 3.

2 Preliminaries

In this section, we recall the specifications of digital signature schemes and of verifiably encrypted signature schemes along with their respective security model as well as Merkle authentication trees. Throughout the paper, n always denotes the security parameter.

2.1 Digital signatures

The well-known definition of digital signatures along with its security model [14] and a specification of full-domain hash RSA signatures [5] can be found in Appendix A.1.

2.2 Verifiably encrypted signature schemes

Verifiably encrypted signatures are specified in [7], which we slightly modify at the clearly marked spots below. A verifiably encrypted signature scheme $\text{VES} = (\text{AdjKg}, \text{Kg}, \text{Sign}, \text{Vf}, \text{Create}, \text{VesVf}, \text{Adj})$ consists of the following seven efficient algorithms.

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

Key Generation, Signature Verification. Signature verification Vf is defined as with a digital signature scheme DSig (cf. Appendix A.1). The key generation algorithm of the signer is $\text{Kg}(1^n, \text{apk})$, which yields the secret key sk' and the public key pk' . In [7], it is $\text{Kg}(1^n)$. In addition, we need the signer to register with the adjudicator once, sending pk and receiving an additional key pair $(\text{sk}'', \text{pk}'')$. The output is $\text{sk} = (\text{sk}', \text{sk}'')$ and $\text{pk} = (\text{pk}', \text{pk}'')$.

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

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

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

A verifiably encrypted signature scheme is *complete* if for all adjudication key pairs $(\text{ask}, \text{apk}) \leftarrow \text{AdjKg}(1^n)$ and for all signature key pairs $(\text{sk}, \text{pk}) \leftarrow \text{Kg}(1^n, \text{apk})$ the following holds:

$$\begin{aligned} \text{VesVf}(\text{apk}, \text{pk}, \text{Create}(\text{sk}, \text{apk}, m), m) &= 1 && \text{and} \\ \text{Vf}(\text{pk}, \text{Adj}(\text{ask}, \text{apk}, \text{pk}, \text{Create}(\text{sk}, \text{apk}, m)), m) &= 1 && \text{for all } m \in \mathcal{M}. \end{aligned}$$

Security of verifiably encrypted signatures is defined via *unforgeability* and *opacity* as described in [7], as well as *extractability* and *abuse-freeness* as defined in [18]. 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 without the secret adjudication key. Extractability requires that the adjudicator can always extract valid regular signatures from valid verifiably encrypted signatures, even if the signer's key is not chosen honestly. The weaker definition, weak-extractability, assumes that the signer key is chosen correctly. As described in [18], weak-extractability can be improved to extractability by common key registration techniques. In Section 3, however, we follow the above modified model and let the adjudicator choose a non-critical part of the user's key that does not limit the security of the signature scheme.

The first two 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 Create returns a verifiably encrypted signature for a given message; oracle Adj extracts a regular signature from a valid verifiably encrypted signature.

A verifiably encrypted signature scheme VES is *secure* if the following holds:

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

Experiment $\text{Exp}_{\mathcal{A}, \text{VES}}^{\text{ves-forge}}(n)$
 $(\text{ask}, \text{apk}) \leftarrow \text{AdjKg}(1^n)$
 $(\text{sk}, \text{pk}) \leftarrow \text{Kg}(1^n, \text{apk})$
 $(m^*, \varpi^*) \leftarrow \mathcal{A}^{\text{Create}(\text{sk}, \text{apk}, \cdot), \text{Adj}(\text{ask}, \text{apk}, \text{pk}, \cdot, \cdot)}(\text{pk}, \text{apk})$
Return 1 iff $\text{VesVf}(\text{apk}, \text{pk}, \varpi^*, m^*) = 1$ and
 \mathcal{A} has never queried `Create` or `Adj` on m^* .

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

Experiment $\text{Exp}_{\mathcal{A}, \text{VES}}^{\text{ves-opac}}(n)$
 $(\text{ask}, \text{apk}) \leftarrow \text{AdjKg}(1^n)$
 $(\text{sk}, \text{pk}) \leftarrow \text{Kg}(1^n, \text{apk})$
 $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\text{Create}(\text{sk}, \text{apk}, \cdot), \text{Adj}(\text{ask}, \text{apk}, \text{pk}, \cdot, \cdot)}(\text{pk}, \text{apk})$
Return 1 iff $\text{Vf}(\text{pk}, \sigma^*, m^*) = 1$ and
 \mathcal{A} has never queried `Adj`(ask, apk, pk, ·, ·) on m^* .

A scheme is called $(t, q_{\text{Create}}, q_{\text{Adj}}, \epsilon)$ -unforgeable (-opaque), if no adversary \mathcal{A} , running in time at most t , making at most q_{Create} verifiably encrypted signature oracle queries, and at most q_{Adj} adjudication oracle queries, can succeed with probability at least ϵ in the $\text{Exp}_{\mathcal{A}, \text{VES}}^{\text{ves-forge}}$ ($\text{Exp}_{\mathcal{A}, \text{VES}}^{\text{ves-opac}}$) experiment.

The scheme is *extractable* if for all adjudication keys $(\text{ask}, \text{apk}) \leftarrow \text{AdjKg}(1^n)$, for all signing key pairs (sk, pk) , and for all verifiably encrypted signatures ϖ on some message m the following holds: $\text{VesVf}(\text{apk}, \text{pk}, \varpi, m) = 1 \implies \text{Vf}(\text{pk}, \text{Adj}(\text{ask}, \text{apk}, \text{pk}, \varpi, m), m) = 1$. In [18], the authors introduce *abuse-freeness* as an additional property. It guarantees that even if signer and adjudicator collude, they cannot forge verifiably encrypted signatures on behalf of a third party. Due to a theorem in [18], we do not have to deal with abuse-freeness explicitly if a VES scheme is unforgeable, extractable, and key-independent. Key-independence is a property of the `Create` algorithm and states that it can be separated into a signing algorithm and an encryption algorithm, where the encryption algorithm is independent of the secret signature key. This is the case in Section 3.

2.3 Merkle authentication trees

A discussion of Merkle trees [17] and their efficient implementation can be found in Appendix A.2.

3 Our construction

We present an efficient verifiably encrypted signature scheme based on RSA signatures in the random oracle model. Whereas previous constructions exploit the special properties of pairings in order to verifiably encrypt regular signatures, we use a Merkle hash tree to verifiably link encrypted signatures with regular ones. The core idea is that the signer masks the signature using a secret value x , encrypts x under the public key of the adjudicator, and attaches the encryption

to the verifiably encrypted signature. Then, the signer uses authentication paths in a special Merkle tree to prove that x has been honestly encrypted. Observe that we make a sensible extension to the model of Boneh et al. and let the adjudicator generate a part of the user's key. This is a practical assumption because in real-world contract signing applications, the users simply register with the escrow beforehand. In the unlikely scenario, where there is more than one adjudicator, the signer has to create one public key per adjudicator.

Let $c = 2^\ell$ for an $\ell \in \mathbb{N}$ be the maximum number of messages to be signed under a single key and let $G : \{0, 1\}^* \rightarrow \{0, 1\}^{k(n)}$, $k(n) = \omega(\log(n))$, and $H : \{0, 1\}^* \rightarrow \mathbb{Z}_{N_S}$ be collision resistant, one-way hash functions. The proposed verifiably encrypted signature scheme **VERSA** is a tuple $(\text{AdjKg}, \text{Kg}, \text{Sign}, \text{Vf}, \text{Create}, \text{VesVf}, \text{Adj})$, which is defined as follows.

Adjudicator Key Generation. $\text{AdjKg}(1^n)$ uses $\text{RSA.Kg}(1^n)$ to compute (N_E, e, d) and a second set $(\text{authsk}, \text{authpk})$ of RSA keys for key authenticity. It outputs (N_E, e, authpk) as the public key and (N_E, d, authsk) as the secret key.

Key Generation. $\text{Kg}(1^n, (N_E, e, \text{authpk}))$ uses $\text{RSA.Kg}(1^n)$ to compute (N_S, v, s) , such that $N_S > N_E$. Upon registration with the adjudicator, the authority randomly chooses c secret values $x_1, \dots, x_c \leftarrow \mathbb{Z}_{N_E}$, such that they are invertible in \mathbb{Z}_{N_S} , and builds a binary tree T . The leaves of T are

$$G(x_1^e \bmod N_E \| x_1^v \bmod N_S), \dots, G(x_c^e \bmod N_E \| x_c^v \bmod N_S).$$

T can be represented by a single random seed to a pseudo random number generator that was used to generate the values x_1, \dots, x_c .

Each inner node η has two children, $\text{left}(\eta)$ and $\text{right}(\eta)$. The value of η is formed recursively: $\eta \leftarrow G(\text{left}(\eta) \| \text{right}(\eta))$. Let ρ be the root of T . The adjudicator computes a signature: $\sigma_\rho \leftarrow \text{RSA.Sign}(\text{authsk}, \rho)$ and sends it to the user. The user's output is the private key (N_S, s, T) and the public key $(N_S, v, \rho, \sigma_\rho)$. The algorithm sets up a state comprising a signature counter $\iota \leftarrow 0$ and a small cache of $O(\ell)$ tree nodes in order to speed up the path computation (cf. Appendix A.2).

Signing, Signature Verification. As in RSA.

VES Creation. $\text{Create}((N_S, s, T), (N_E, e, \text{authpk}), m)$ increments the signature counter ι and computes

$$\begin{aligned} \sigma &\leftarrow \text{RSA.Sign}((N_S, s), m), \\ \alpha &\leftarrow \sigma x_\iota \bmod N_S, \\ \beta &\leftarrow x_\iota^e \bmod N_E, \\ \text{and } \gamma &\leftarrow x_\iota^v \bmod N_S. \end{aligned}$$

Then, it generates the authentication path π for x_ι in T , which is used to prove the relation between β and γ via the collision-resistance of G . The output is $\varpi = (\alpha, \beta, \gamma, \pi)$.

VES Verification. $\text{VesVf}((N_E, e, \text{authpk}), (N_S, v, \rho, \sigma_\rho), (\alpha, \beta, \gamma, \pi), m)$ is 1 iff

1. $0 \leq \alpha < N_S$,
2. $\alpha^v \equiv H(m) \gamma \pmod{N_S}$,
3. π correctly authenticates the leaf $G(\beta||\gamma)$ for the given root ρ .
4. $\text{RSA.Vf}(\text{authpk}, \sigma_\rho, \rho) = 1$.

Adjudication. $\text{Adj}((N_E, d, \text{authsk}), (N_E, e, \text{authpk}), (N_S, v, \rho, \sigma_\rho), (\alpha, \beta, \gamma, \pi), m)$ checks that the presented verifiably encrypted signature is valid. If so, it computes $x' \leftarrow \beta^d \pmod{N_E}$, $\sigma \leftarrow \alpha/x' \pmod{N_S}$, and returns σ .

See Appendix B for the proof of completeness. In the following paragraphs, we show that **VERSA** satisfies extractability, unforgeability, and opacity according to the model of Boneh et al. and its modifications in [18]. Due to the fact that **Create** works in two key-independent steps — sign with **RSA.Sign** and verifiably encrypt with additional secret values — our scheme is abuse-free according to [18, Theorem 5.4].

Extractability. We show that our scheme is extractable. For

$$\begin{aligned} \text{all adjudication keys:} & \quad (N_E, e, d, \text{authsk}, \text{authpk}) \leftarrow \text{AdjKg}(1^n), \\ \text{all signing keys:} & \quad (N_S, v, \rho, s, T, \sigma_\rho), \\ \text{and all encrypted signatures:} & \quad \varpi = (\alpha, \beta, \gamma, \pi) \text{ on some message } m, \end{aligned}$$

$\text{VesVf}((N_E, e, \text{authpk}), (N_S, v, \rho, \sigma_\rho), \varpi, m) = 1$ guarantees the correctness of β and γ by conditions 3 and 4, i.e., they were computed from the same secret value x as long as G is collision resistant and **RSA** is unforgeable. In consequence, the adjudicator can always unmask α with the decryption of β . The resulting regular signature σ is always valid because of conditions 1 and 2 in **VesVf**.

For the following proofs, let $T_{\text{AdjKg}}(n)$ and $T_{\text{Kg}}(n)$ be the cost functions for adjudication and signature key generation and let $T_{\text{Create}}(n)$, $T_{\text{VesVf}}(n)$, and $T_{\text{Adj}}(n)$ be the cost functions for creation, verification, and adjudication of verifiably encrypted signatures.

Unforgeability. We show that our scheme is unforgeable, provided that the underlying signature scheme is unforgeable.

Theorem 1 (Unforgeability). *VERSA is $(t, q_{\text{Create}}, q_{\text{Adj}}, \epsilon)$ -unforgeable if the RSA signature scheme is $(t', q_{\text{Create}}, \epsilon - \delta)$ -unforgeable with $t' = T_{\text{AdjKg}}(n) + T_{\text{Kg}}(n) + q_{\text{Create}} T_{\text{Create}}(n) + q_{\text{Adj}} T_{\text{Adj}}(n)$ and there is no polynomial-time adversary that can find collisions under G with probability $\geq \delta$.*

Proof. Towards contradiction, let's assume that there exists a successful polynomial time adversary \mathcal{A} against the unforgeability of **VERSA**, running in time t and with success probability ϵ . Furthermore, assume that \mathcal{A} makes q_{Create} verifiably encrypted signature queries and q_{Adj} adjudication queries. Via a black box simulation of \mathcal{A} , we construct an efficient and equally successful adversary \mathcal{B} against the underlying signature scheme in the $\text{Exp}_{\mathcal{A}, \text{RSA}}^{\text{eu-cma}}$ experiment.

Setup. \mathcal{B} gets as input the public verification key (N_S, v) and has access to a signing oracle $\text{RSA.Sign}((N_S, s), \cdot)$. It generates its own adjudication key $(N_E, e, d, \text{authsk}, \text{authpk})$ and executes Kg in order to generate the secret values (x_1, \dots, x_c) as well as ρ and σ_ρ , accordingly. \mathcal{B} simulates \mathcal{A} with the input $(N_S, v, \rho, \sigma_\rho), (N_E, e, \text{authpk})$.

VES Queries. Whenever \mathcal{A} queries the verifiably encrypted signature oracle Create on some message m , algorithm \mathcal{B} invokes its signing oracle $\sigma \leftarrow \text{RSA.Sign}((N_S, s), \cdot)$ on m , masks the signature as described in Create , and outputs $(\alpha, \beta, \gamma, \pi)$.

Adjudication Queries. If \mathcal{A} invokes the adjudication oracle Adj on a message m and on a verifiably encrypted signature $(\alpha, \beta, \gamma, \pi)$, \mathcal{B} verifies it. If it is invalid, \mathcal{B} returns fail . Otherwise, it extracts the signature by computing $x' \leftarrow \beta^d \pmod{N_E}$, $\sigma \leftarrow \alpha/x' \pmod{N_S}$, and returns σ .

Output. Finally, \mathcal{A} halts, outputting a forged verifiably encrypted signature (m^*, ϖ^*) , such that $\text{VesVf}((N_E, e, \text{authpk}), (N_S, v, \rho, \sigma_\rho), \varpi^*, m^*) = 1$ with $\varpi^* = (\alpha^*, \beta^*, \gamma^*, \pi^*)$. \mathcal{B} extracts $\sigma^* \leftarrow \text{Adj}((N_E, d, \text{authsk}), (N_E, e, \text{authpk}), (N_S, v, \rho, \sigma_\rho), \varpi^*, m^*)$ and stops, outputting (m^*, σ^*) .

Analysis. Since \mathcal{B} acts honestly on all queries and all keys are generated honestly, the environment of \mathcal{A} is perfectly simulated. According to the security model, \mathcal{A} is only successful if it returns a verifiably encrypted signature for a message which has never been signed before. In addition, extractability ensures that every verifiably encrypted signature yields a regular signature via adjudication. This step only fails if \mathcal{A} can find collisions under \mathbf{G} . Thus, \mathcal{B} 's success probability is at least $\epsilon - \delta$. As for the detailed complexity analysis, note that \mathcal{B} queries the signature oracle whenever \mathcal{A} queries the verifiably encrypted signature oracle. When \mathcal{A} queries the adjudication oracle, \mathcal{B} decrypts the query if it is valid. The overall overhead, including key generation, encryption, and adjudication, is at most $T_{\text{AdjKg}}(n) + T_{\text{Kg}}(n) + q_{\text{Create}} T_{\text{Create}}(n) + q_{\text{Adj}} T_{\text{Adj}}(n)$, which completes the proof. \square

Opacity. We prove that breaking opacity of VERSA implies being able to invert the RSA trapdoor. In order to emphasize the interesting case in the proof of opacity, where the adversary extracts a regular signature from an encrypted one, we omit the straightforward proof for the type of adversary that simply forges the underlying signature scheme. Inverting the RSA trapdoor with non-negligible probability, is assumed to be infeasible for all polynomially bounded algorithms, i.e., the problem is (t, ϵ) -hard for every non-negligible ϵ and any $t = \text{poly}(n)$.

Theorem 2 (Opacity). *The VERSA scheme is $(t, q_{\text{Create}}, q_{\text{Adj}}, \epsilon)$ -opaque if the RSA signature scheme is $(t', q_{\text{Create}}, \epsilon - \delta)$ -unforgeable, inverting the RSA trapdoor is $(t', q_{\text{Create}}, \epsilon')$ -hard with $t' = t + T_{\text{Kg}}(n) + T_{\text{AdjKg}}(n) + q_{\text{Create}} T_{\text{Create}}(n) + q_{\text{Adj}} T_{\text{Adj}}(n)$ and $\epsilon' = \epsilon/q_{\text{Create}}$, and there is no polynomial-time adversary that can find collisions under \mathbf{G} with probability $\geq \delta$.*

Proof. Assume there is a successful polynomial time adversary \mathcal{A} against opacity of **VERSA**, running in time t and with success probability ϵ , which makes q_{Create} verifiably encrypted signature queries and q_{Adj} adjudication queries. Using \mathcal{A} , we construct an efficient algorithm \mathcal{B} that is able to invert an RSA challenge $y \in \mathbb{Z}_{N_S}^*$. Thus, the goal of \mathcal{B} is to find a value $x \in \mathbb{Z}_{N_S}$ with $x^v \equiv y \pmod{N_S}$ — see [4] for a formal definition.

Setup. \mathcal{B} gets as input the public RSA key (N_S, v) and a challenge y . The algorithm executes **AdjKg** and **Kg** in order to generate $(N_E, e, d, \text{authsk}, \text{authpk})$ and $(x_1, \dots, x_{j-1}, \square, x_{j+1}, \dots, x_c)$, where $j \leftarrow \{1, \dots, q_{\text{Create}}\}$, $q_{\text{Create}} \leq c$, is chosen uniformly at random. Then, it chooses $z \leftarrow \mathbb{Z}_{N_S}$ uniformly at random. Root ρ and signature σ_ρ of T is generated from

$$(\mathsf{G}(x_1^e || x_1^v), \dots, \mathsf{G}(x_{j-1}^e || x_{j-1}^v), \mathsf{G}(y || z^v / y), \mathsf{G}(x_{j+1}^e || x_{j+1}^v), \dots, \mathsf{G}(x_c^e || x_c^v)),$$

where $\mathsf{G}(a || b)$ is shorthand for $\mathsf{G}(a \bmod N_E || b \bmod N_S)$. Algorithm \mathcal{B} runs \mathcal{A} on input $(N_S, v, \rho, \sigma_\rho), (N_E, e, \text{authpk})$ in a black box simulation. \mathcal{B} initializes a signature counter $i \leftarrow 0$. Furthermore, \mathcal{B} maintains a list L_{H} of triples (m, r, σ) in order to simulate a consistent random oracle and the signature oracle.

Random Oracle Queries. On input m , algorithm \mathcal{B} searches an entry (m, r, \square) in L_{H} . If it does not exist, \mathcal{B} randomly chooses an $\sigma \leftarrow \mathbb{Z}_{N_S}^*$ and adds $(m, r \leftarrow \sigma^v \bmod N_S, \sigma)$ to L_{H} . The oracle returns r .

VES Queries. When \mathcal{A} queries the oracle on a message m , algorithm \mathcal{B} increments j .

– Case $i = j$: \mathcal{B} adds $(m, y, 0)$ to L_{H} and sets

$$\begin{aligned} \alpha &\leftarrow z \bmod N_S, \\ \beta &\leftarrow y \bmod N_E, \\ \gamma &\leftarrow z^v / y \bmod N_S. \end{aligned}$$

– Case $i \neq j$: \mathcal{B} executes $\mathsf{H}(m)$, yielding a triple $(m, r, \sigma) \in L_{\text{H}}$, and sets

$$\begin{aligned} \alpha &\leftarrow \sigma x_i \bmod N_S, \\ \beta &\leftarrow x_i^e \bmod N_E, \\ \gamma &\leftarrow x_i^v \bmod N_S. \end{aligned}$$

In both cases, the outputs is $(\alpha, \beta, \gamma, \pi)$, where π authenticates the i -th leaf.

Adjudication Queries. Whenever \mathcal{A} invokes the adjudication oracle **Adj** on a message m and on a verifiable encrypted signature $\varpi = (\alpha, \beta, \gamma, \pi)$, algorithm \mathcal{B} verifies its validity. If it is invalid, \mathcal{B} answers with **fail**. If $\mathsf{H}(m) = y$, \mathcal{B} stops and returns **fail**. Otherwise, it searches the list L_{H} for an entry (m, r, σ) and outputs σ . If it does not exist, \mathcal{B} extracts the signature σ^* from ϖ . It stops and outputs the forgery (m, σ^*) .

Output. When \mathcal{A} halts, outputting a forged signature (m^*, σ^*) . If m^* was never queried to **Create**, algorithm \mathcal{B} outputs the forgery (m^*, σ^*) . If m^* is in L_{H} , algorithm \mathcal{B} returns σ^* as the v -th root of y modulo N_S .

Analysis. First, we have to show that \mathcal{A} 's queries are correctly answered by \mathcal{B} . The keys are chosen honestly except for the j -th leaf of T . However, this deviation is not efficiently recognizable as the distribution of the arguments under G stays uniform. The random oracle and adjudication queries are perfectly simulated because \mathcal{B} returns random hash values and valid signatures.

As for verifiably encrypted signature queries, we deal with the two cases separately. For $i \neq j$, we use a standard random oracle technique for RSA signature simulation. The returned encrypted signatures are always valid. In case $i = j$, algorithm \mathcal{A} is given a tuple $(\alpha, \beta, \gamma, \pi)$ that is valid for m because $\alpha^v \equiv z^v \equiv z^v y/y \equiv \mathsf{H}(m) \gamma \pmod{N_S}$. Therefore, \mathcal{A} 's environment is simulated as expected. The computational overhead $T_{\mathsf{Kg}}(n) + T_{\mathsf{AdjKg}}(n) + q_{\mathsf{Create}} T_{\mathsf{Create}}(n) + q_{\mathsf{Adj}} T_{\mathsf{Adj}}(n)$ of the reduction is induced by the fact that \mathcal{B} initially executes parts of Kg and AdjKg and that the simulation of Create and Adj can be performed at least as efficient as in the original scheme. As for Adj , the list-based simulation is even more efficient. Note that for brevity, our overhead analysis does not take the simulation of the random oracle into account. The success probability of \mathcal{B} depends on the type of adversary \mathcal{A} . It may be an adversary that never queried m^* to Create . In this case, however, we have a direct forgery against the RSA signature scheme. The same holds in case that \mathcal{A} queries Adj with a fresh message m because VERSA is extractable but with probability δ . In the interesting case, i.e., \mathcal{A} queried m^* to Create , the success probability of \mathcal{B} depends on the correct guess of the index j . If the guess is correct, \mathcal{A} outputs (m^*, σ^*) with $\sigma^{*v} \equiv \mathsf{H}(m^*) \equiv y \pmod{N_S}$.

In consequence, if \mathcal{A} is successful with noticeable probability ϵ then \mathcal{B} is successful with probability $\epsilon/q_{\mathsf{Create}}$, which is still noticeable. \square

4 Conclusions

With our construction, we have shown that verifiably encrypted signatures can be constructed without pairings by using simple and efficient primitives instead. The main benefit of our result is that existing widespread hard- and software implementations of hash functions and RSA signatures can be easily reused in the VES setting. From a theoretical point of view, we introduced a new construction principle at the expense of making the scheme stateful. The remaining open question is whether the scheme can be made stateless and whether we can avoid the random oracle methodology.

Acknowledgments

The author would like to thank Dominique Schröder and Michael Schneider for many insightful discussions on the subject. Furthermore, the author thanks the anonymous reviewers of Indocrypt 2009 for their valuable comments.

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. Giuseppe Ateniese. Verifiable encryption of digital signatures and applications. *ACM Trans. Inf. Syst. Secur.*, 7(1):1–20, 2004.
3. Feng Bao, Robert H. Deng, and Wenbo Mao. Efficient and practical fair exchange protocols with off-line ttp. In *IEEE Symposium on Security and Privacy*, pages 77–85. IEEE Computer Society, 1998.
4. Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-rsa-inversion problems and the security of chaum’s blind signature scheme. *J. Cryptology*, 16(3):185–215, 2003.
5. Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - how to sign with rsa and rabin. In *EUROCRYPT*, pages 399–416, 1996.
6. Dan Boneh. The decision diffie-hellman problem. In Joe Buhler, editor, *ANTS*, volume 1423 of *LNCS*, pages 48–63. Springer, 1998.
7. Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *LNCS*, pages 416–432. Springer, 2003.
8. Johannes Buchmann, Erik Dahmen, Elena Klintsevich, Katsuyuki Okeya, and Camille Vuillaume. Merkle signatures with virtually unlimited signature capacity. In Jonathan Katz and Moti Yung, editors, *ACNS*, volume 4521 of *LNCS*, pages 31–45. Springer, 2007.
9. Johannes Buchmann, Erik Dahmen, and Michael Schneider. Merkle tree traversal revisited. In Johannes Buchmann and Jintai Ding, editors, *PQCrypto*, volume 5299 of *LNCS*, pages 63–78. Springer, 2008.
10. Johannes Buchmann, Luis Carlos Coronado García, Erik Dahmen, Martin Döring, and Elena Klintsevich. Cmss - an improved merkle signature scheme. In Rana Barua and Tanja Lange, editors, *INDOCRYPT*, volume 4329 of *LNCS*, pages 349–363. Springer, 2006.
11. Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *LNCS*, pages 126–144. Springer, 2003.
12. Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.
13. S.D. Galbraith, K.G. Paterson, and N.P. Smart. Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165, 2006. <http://eprint.iacr.org/>.
14. 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.
15. Qiong Huang, Guomin Yang, Duncan S. Wong, and Willy Susilo. Ambiguous optimistic fair exchange. In Josef Pieprzyk, editor, *ASIACRYPT*, volume 5350 of *LNCS*, pages 74–89. Springer, 2008.
16. Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *LNCS*, pages 465–485. Springer, 2006.
17. Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *LNCS*, pages 218–238. Springer, 1989.

18. Markus Rückert and Dominique Schröder. Security of verifiably encrypted signatures and a construction without random oracles. In Hovav Shacham and Brent Waters, editors, *Pairing*, volume 5671 of *LNCS*, pages 17–34. Springer, 2009. Full version: ePrint 2009/027.
19. Nigel P. Smart and Frederik Vercauteren. On computable isomorphisms in efficient asymmetric pairing-based systems. *Discrete Applied Mathematics*, 155(4):538–547, 2007.
20. Michael Szydło. Merkle tree traversal in log space and time. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *LNCS*, pages 541–554. Springer, 2004.
21. Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 114–127. Springer, 2005.
22. Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. Efficient verifiably encrypted signature and partially blind signature from bilinear pairings. In Thomas Johansson and Subhamoy Maitra, editors, *INDOCRYPT*, volume 2904 of *LNCS*, pages 191–204. Springer, 2003.

A Preliminaries

A.1 Digital signatures

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

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

Signing. $\text{Sign}(\text{sk}, m)$ outputs a signature σ on a message m from the message space \mathcal{M} under sk .

Verification. The algorithm $\text{Vf}(\text{pk}, \sigma, m)$ outputs 1 if σ is a valid signature on m under pk , otherwise 0.

Signature schemes are complete if for any $(\text{sk}, \text{pk}) \leftarrow \text{Kg}(1^n)$, any message $m \in \mathcal{M}$, and any $\sigma \leftarrow \text{Sign}(\text{sk}, m)$, we have $\text{Vf}(\text{pk}, \sigma, m) = 1$. Security of signature schemes is proven against existential forgery under chosen message attacks [14]. In this model, an adversary adaptively invokes a signing oracle and is successful if he outputs a valid signature on a *new* message.

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

Experiment $\text{Exp}_{\mathcal{A}, \text{DSig}}^{\text{eu-cma}}(n)$

$(\text{sk}, \text{pk}) \leftarrow \text{Kg}(1^n)$

$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\text{Sign}(\text{sk}, \cdot)}(\text{pk})$

Let (m_i, σ_i) be the answer of $\text{Sign}(\text{sk}, \cdot)$ on input m_i , for $i = 1, \dots, k$.

Return 1 iff $\text{Vf}(\text{pk}, \sigma^*, m^*) = 1$ and $m^* \notin \{m_1, \dots, m_k\}$.

A signature scheme DSig is $(t, q_{\text{sig}}, \epsilon)$ -unforgeable if no adversary running in time at most t , invoking the signing oracle at most q_{sig} times, outputs a valid forgery (m^*, σ^*) with probability at least ϵ .

Recall the definition of secure RSA signatures. Let $\text{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_N^*$ be a collision resistant hash function for a given modulus N . The full-domain hash RSA signature scheme RSA [5] is a 3-tuple $(\text{Kg}, \text{Sign}, \text{Vf})$, which is defined as follows.

Key Generation. $\text{Kg}(1^n)$ receives the security parameter n and generates two primes p, q of bit length $n/2$. Let $N = pq$ be the public modulus. It chooses v relatively prime to $(p-1)(q-1)$, $s \leftarrow v^{-1} \pmod{(p-1)(q-1)}$, and returns the secret key (N, s) and the public key (N, v) .

Signing. $\text{Sign}((N, s), m)$ receives the modulus N , the private key s and a message $m \in \{0, 1\}^*$. It computes $\sigma \leftarrow \text{H}(m)^s \pmod N$ and returns σ .

Verification. $\text{Vf}((N, v), \sigma, m)$ receives the public key (N, v) , a signature σ and a message m . It returns 1 iff $0 \leq \sigma < N$ and $\sigma^v \equiv \text{H}(m) \pmod N$.

It is well-known that RSA is complete and unforgeable in the random oracle model, when following the “hash-then-sign” paradigm [5].

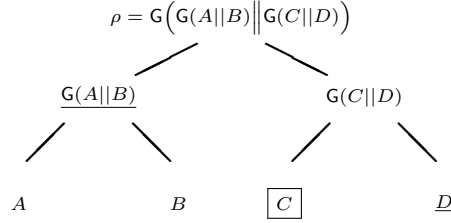
A.2 Merkle authentication trees

In [17], Merkle describes a tree-based construction for digital signature schemes based on the collision-resistance of hash functions. Recall collision-resistance as a property of a hash function $\text{G} : D \rightarrow R$, which states that it is computationally infeasible to find distinct $x_1, x_2 \in D$, such that $\text{G}(x_1) = \text{G}(x_2)$.

At its core, Merkle’s scheme relies on authenticating a large amount of data, the 2^ℓ leaves of a binary tree of depth ℓ , with a single public hash value, the root ρ of the binary tree, by providing authentication paths. The authentication path comprises all inner nodes, along with the relative position w.r.t. to their parent nodes, that are necessary to compute the value ρ . The path can be written as $\pi = [(I_1, E_1), \dots, (I_\ell, E_\ell)]$, where $I_i \in \{\text{left}, \text{right}\}$ and $E_i \in R$ for $i = 1 \dots, \ell$. For an example, see Figure 1. The verification of the authentication path for a leaf X can be done by checking whether $\eta_\ell = \rho$, where

$$\begin{aligned} \eta_0 &\leftarrow X, \\ \eta_i &\leftarrow \begin{cases} \text{G}(E_i || \eta_{i-1}) & \text{if } I_i = \text{left} \\ \text{G}(\eta_{i-1} || E_i) & \text{if } I_i = \text{right} \end{cases} \quad \text{for } i = 1, \dots, \ell. \end{aligned}$$

While authentication path creation and verification can be implemented using $O(\ell)$ hash evaluations by using a small cache of $O(\ell)$ hash values [17,20,9], the initial tree generation is rather costly. Setting $\ell = 20$, however, has proven to be a reasonable trade-off between signature capability and efficiency in the Merkle signature scheme [10] but there generalized constructions that allow for 2^{40} or even 2^{80} signatures without having to compute 2^{40} respectively 2^{80} hash values during the set-up phase [8].



The authentication path for the leaf C : in order to compute ρ , it is necessary to publish D and $G(A||B)$ along with their relative positions. The resulting path is $\pi = [(\text{right}, D), (\text{left}, G(A||B))]$. Note that instead of publishing the relative position (left or right), it is sufficient to publish the zero-based index (2) of C .

Fig. 1. Merkle authentication tree.

In order to keep it more accessible, we do not apply these generalizations to our construction and state that leaf computation in our case is far less expensive than leaf computation in a Merkle signature scheme. Therefore, one can expect that the upper bound for the number of verifiably encrypted signatures can be raised to 2^{30} . On a 2.4 GHz Opteron CPU, the overhead for tree generation with height $\ell = 30$ is about 30 minutes without further optimization or parallelization. Taking into account that hardly anyone will ever issue that many signatures in a lifetime, the computational overhead of our tree construction can be significantly reduced to 1 *minute* ($\ell = 25$), 1.5 *seconds* ($\ell = 20$), and 50 *milliseconds* ($\ell = 15$). Observe that, like the Merkle signature scheme, our construction scales well with the individual requirements of different application scenarios.

B Completeness of our construction

For

- all adjudication keys $(N_E, e, d, \text{authsk}, \text{authpk}) \leftarrow \text{AdjKg}(1^n)$,
- all signing keys $(N_S, v, \rho, s, T, \sigma_\rho) \leftarrow \text{Kg}(1^n, (N_E, e, \text{authpk}))$,
- all messages $m \in \{0, 1\}^*$,
- and encrypted signatures $\varpi \leftarrow \text{Create}((N_S, s, T), (N_E, e, \text{authpk}), m)$,

we have

$$\begin{aligned}
 \varpi &= (\alpha, \beta, \gamma, \pi), \\
 \alpha &\equiv \text{H}(m)^s x \pmod{N_S}, \\
 \beta &\equiv x^e \pmod{N_E}, \\
 \gamma &\equiv x^v \pmod{N_S}, \\
 \pi &= [(I_1, E_1), \dots, (I_\ell, E_\ell)].
 \end{aligned}$$

As for VesVf , condition 2 is satisfied because $\alpha^v \equiv \sigma^v x_i^v \equiv \text{H}(m)\gamma \pmod{N_S}$ and condition 1 holds trivially. Conditions 3 and 4 are satisfied by construction of ρ and σ_ρ . Thus, $\text{VesVf}((N_E, e, \text{authpk}), (N_S, v, \rho, \sigma_\rho), (\alpha, \beta, \gamma, \pi), m) = 1$. As for Vf and Adj , we state that Adj computes

$$\begin{aligned} x' &\leftarrow \beta^d \pmod{N_E}, x' \equiv x^{e d} \equiv x \pmod{N_E}, x' \in \mathbb{Z}_{N_S}^*, \\ \sigma' &\leftarrow \alpha/x' \pmod{N_S}, \sigma' \equiv \text{H}(m)^s x/x' \equiv \text{H}(m)^s \pmod{N_S} \end{aligned}$$

and that, therefore, $\text{Vf}((N_S, v), \sigma', m) = 1$. Thus, VERSA is complete.