Rethinking Verifiably Encrypted Signatures: A Gap in Functionality and Potential Solutions

Theresa Calderon[1] and Sarah Meiklejohn[1] and Hovav Shacham[1] and Brent Waters[2]

[1] UC San Diego
{tcaldero, smeiklej, hovav}@cs.ucsd.edu
[2] UT Austin
bwaters@cs.utexas.edu

Abstract. Verifiably encrypted signatures were introduced by Boneh, Gentry, Lynn, and Shacham in 2003, as a non-interactive analogue to interactive protocols for verifiable encryption of signatures. As their name suggests, verifiably encrypted signatures were intended to capture a notion of encryption, and constructions in the literature use public-key encryption as a building block.
In this paper, we show that previous definitions for verifiably encrypted signatures do not capture the intuition that encryption is necessary, by presenting a generic construction of verifiably encrypted signatures from any signature scheme. We then argue that signatures extracted by the arbiter from a verifiably encrypted signature object should be distributed identically to ordinary signatures produced by the original signer, a property that we call resolution independence. Our generic construction of verifiably encrypted signatures does not satisfy resolution independence, whereas all previous constructions do. Finally, we introduce a stronger but less general version of resolution independence, which we call resolution duplication. We show that verifiably encrypted signatures that satisfy resolution duplication generically imply public-key encryption.

Keywords: Verifiably encrypted signatures, signatures, public-key encryption

1 Introduction

Verifiably encrypted signatures were introduced by Boneh, Gentry, Lynn, and Shacham in 2003 [5] as a non-interactive analogue to interactive protocols for verifiable encryption of signatures [3,4] and of other cryptographic objects [7]. As their name suggests, verifiably encrypted signatures were intended to incorporate a notion of encryption: the signer encrypts her signature in such a way that a special trusted party, called the arbiter, can later decrypt and reveal the underlying ordinary signature. Indeed, ElGamal encryption of BLS signatures [6] was at the heart of the original verifiably encrypted signature construction.

In this paper, we show that this intention — incorporating a notion of encryption — is not actually achieved by the definitions given in previous papers. To demonstrate this, we first in Section 3 give a generic construction of verifiably encrypted signatures from any existentially unforgeable signature scheme. The intuition behind our construction is fairly straightforward: to form a verifiably encrypted signature, the signer includes an annotation with her signature asking users not to verify it; later, the arbiter can append to this signature another annotation, under his own key, telling users that verification is now allowed. Our generic construction satisfies both the original security model for verifiably encrypted signatures and the tweaked definitions later given by Hess [11], Rückert and Schröder [16], and Rückert, Schneider, and Schröder [15].

Given that our generic construction therefore yields a secure verifiably encrypted signature yet makes no use of encryption, thus seemingly contradicting the spirit of the primitive, we are left with a number of different ways to interpret this result. One possible interpretation is that the current definitions are fine as is, and verifiably encrypted signatures are simply misnamed. They remain equally useful as a building block in larger protocols such as optimistic fair exchange [2] whether or not they involve encryption.

A second interpretation is that previous definitions have failed to capture something fundamental about verifiably encrypted signatures. If the signer escrows her ordinary signature and the arbiter is later meant to recover that signature, then the start and end points of that process — the signer's signature, and the signature obtained by the arbiter — should look the same. Previous definitions do not model this requirement, and in fact our generic construction does not satisfy it. In Section 4, we therefore formalize this notion, which we call *resolution independence*. We then provide a "separation" of sorts between our generic construction and existing ones (which, again, do use some form of public-key encryption) by arguing that all previous constructions of verifiably encrypted signatures do satisfy it.

A third, perhaps more extreme, interpretation is that a verifiably encrypted signature should not merely be "encryption-like" in facilitating the transfer of a signature from signer to arbiter, but should actually make use of public-key encryption in a fundamental way. To this end, we introduce in Section 5 a stronger version of resolution independence that requires the signer to be able to produce a signature that is *identical* to the one that the arbiter will output. We show that verifiably encrypted signatures that satisfy this property, which we call *resolution duplication*, generically imply the existence of public-key encryption; this approach is inspired by Abdalla and Warinschi [1], who showed that group signatures generically imply public-key encryption. Although resolution duplication is a less general property than resolution independence, all previous constructions of verifiably encrypted signatures except one — that of Lu et al. [12] — satisfy resolution duplication.

2 Definitions and Notation

In this section we provide the basic definitions for verifiably encrypted signatures as defined by Boneh et al. [5] and Hess [11]. Formally, a *verifiably encrypted signature* (VES) consists of seven algorithms. The first three, KeyGen, Sign, and Verify, comprise an ordinary signature scheme. The fourth, AKeyGen, generates a keypair (apk, ask) to be used by the arbiter (previously called the adjudicator); the fifth, VESign, takes as input (sk, apk, m) and outputs a verifiable encrypted signature ω; the sixth, VEVerify, takes as input (pk, apk, ω, m) and outputs 1 if ω is a valid verifiably encrypted signature on m and 0 otherwise; and finally the seventh, Resolve, takes as input (ask, pk, ω, m) and outputs a valid regular signature σ on m under pk (i.e., a value σ such that $\mathsf{Verify}(pk, \sigma, m) = 1$).

In order to say that the scheme is complete, we would like to ensure that an honestly computed VES will indeed verify as such, and also that once this VES is honestly resolved it will produce a valid signature as desired. This can be summarized formally as follows:

Definition 2.1. **[5]** *A VES* (KeyGen, Sign, Verify, AKeyGen, VESign, VEVerify, Resolve) *is* complete *if for all* $(apk, ask) \xleftarrow{\$} \mathsf{AKeyGen}(1^k)$, $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$, *and* $m \in \mathcal{M}$, *for* $\omega \xleftarrow{\$} \mathsf{VESign}(sk, apk, m)$ *it holds that* $\mathsf{VEVerify}(pk, apk, \omega, m) = 1$ *and* $\mathsf{Verify}(pk, \mathsf{Resolve}(ask, pk, \omega, m), m) = 1$.

As we show in Section 3, completeness, as well as all the security properties below, can be satisfied by a construction based solely on signatures. We therefore define in Section 4 a new notion for verifiably encrypted signatures intended to capture the "verifiable encryption" functionality.

Briefly, there are three main security properties we consider for VES schemes: *unforgeability*, *opacity*, and *extractability*. The first of these says that, for a given public key pk, no one except the signer in possession of the corresponding sk should be able to form a verifiably encrypted signature under pk. We alter slightly the original definition; as completeness does not guarantee that signatures produced by Sign and Resolve look the same (although our new definition in Section 4 does), we additionally provide the adversary with access to the Sign oracle, as otherwise the underlying signature scheme could be completely broken and the VES would still be considered unforgeable.

Definition 2.2. *For a VES* (KeyGen, Sign, Verify, AKeyGen, VESign, VEVerify, Resolve) *and an adversary* $\mathcal{A}$, *define the following game:*

- *Step 1.* $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$, $(apk, ask) \xleftarrow{\$} \mathsf{AKeyGen}(1^k)$.
- *Step 2.* $(m, \omega) \xleftarrow{\$} \mathcal{A}^{\mathsf{Sign}(sk,\cdot),\mathsf{VESign}(sk,apk,\cdot),\mathsf{Resolve}(ask,pk,\cdot,\cdot)}(pk, apk)$.

Then the verifiably encrypted signature scheme is unforgeable *(more precisely, secure against existential forgeries) if for all PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that the probability (taken over the choices of* KeyGen, AKeyGen, Sign, VESign, Resolve, *and* $\mathcal{A}$*) that* $\mathsf{VEVerify}(pk, apk, \omega, m) = 1$ *but* m *was not queried to any of the three oracles is at most* $\nu(k)$.

The next property, opacity, says that a user given just the verifiably encrypted signature should not be able to pull out the underlying signature without help from the arbiter.

Definition 2.3. **[5]** *For a VES* (KeyGen, Sign, Verify, AKeyGen, VESign, VEVerify, Resolve) *and an adversary* $\mathcal{A}$*, define the following game:*

- *Step 1.* $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$, $(apk, ask) \xleftarrow{\$} \mathsf{AKeyGen}(1^k)$.
- *Step 2.* $(m, \sigma) \xleftarrow{\$} \mathcal{A}^{\mathsf{VESign}(sk, apk, \cdot), \mathsf{Resolve}(ask, pk, \cdot, \cdot)}(pk, apk)$.

Then the verifiably encrypted signature scheme is opaque *if for all PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that the probability (taken over the choices of* KeyGen, AKeyGen, VESign, Resolve, *and* $\mathcal{A}$*) that* $\mathsf{Verify}(pk, \sigma, m) = 1$ *but* $(m, \cdot)$ *was not queried to the* Resolve *oracle is at most* $\nu(k)$.

While opacity promises that no user can pull out the underlying signature given just the verifiable encrypted signature, we still need to guarantee that the arbiter can in fact do just this if necessary. While such a guarantee was not defined in the original BGLS paper, this property was proposed shortly thereafter by Hess [11] (and later formalized by Rückert and Schröder [16], who called it extractability).

Definition 2.4. **[11]** *For a VES* (KeyGen, Sign, Verify, AKeyGen, VESign, VEVerify, Resolve) *and an adversary* $\mathcal{A}$*, define the following game:*

- *Step 1.* $(apk, ask) \xleftarrow{\$} \mathsf{AKeyGen}(1^k)$.
- *Step 2.* $(pk, \omega, m) \xleftarrow{\$} \mathcal{A}^{\mathsf{Resolve}(ask, \cdot, \cdot, \cdot)}(apk)$.

Then the verifiably encrypted signature scheme is extractable *if for all PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that the probability (taken over the choices of* AKeyGen, Resolve, *and* $\mathcal{A}$*) that, for* $\sigma \xleftarrow{\$} \mathsf{Resolve}(ask, pk, \omega, m)$, $\mathsf{VEVerify}(pk, apk, \omega, m) = 1$ *but* $\mathsf{Verify}(pk, \sigma, m) = 0$ *is at most* $\nu(k)$.

In addition to these three basic properties, there is another property we might consider called *abuse freeness*, as defined by Rückert and Schröder [16]; briefly, this says that even an adversary colluding with the arbiter cannot forge verifiably encrypted signatures for another user. Although we limit our focus here and do not consider this definition formally, we note that our signature-based construction in the next section does satisfy abuse freeness.

3 A Signature-Based Verifiably Encrypted Signature

In this section, we show how to generically construct a secure verifiable encrypted signature using just an unforgeable signature scheme. As mentioned in the introduction, our scheme works intuitively as follows: to run VESign, a user will sign the message, but then stamp, or annotate, the signed message to say "Do

Not Verify." To verify that this is a valid VES, $\mathsf{VEVerify}$ will ensure that it is a signed message with this stamp. To resolve this VES, $\mathsf{Resolve}$ will simply add its own stamp "Yes Do Verify." The verification algorithm will then check for these cases: if the signed message has a "Do Not Verify" stamp then it will output 0 (i.e., it will not verify), unless the signed message also has a "Yes Do Verify" stamp, in which case it will output 1 (i.e., it will verify).

More formally, let $(\mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Verify}')$ be an unforgeable (i.e., EUF-CMA secure) signature scheme with message space $\mathcal{M}'$. To construct a VES with message space $\mathcal{M}$, let T be a function that takes a tuple of four elements: a message $M \in \mathcal{M}$, an arbiter public key $apk \in \{0,1\}^*$, a bit $b \in \{0,1\}$, and a verifiably encrypted signature $\omega \in \{0,1\}^*$, and encodes it into a binary string $M' \in \mathcal{M}'$. We will use $b = 0$ to indicate "Do Not Verify" and $b = 1$ to indicate "Yes Do Verify," and will use $\perp$ to indicate that the given field is being left empty. Furthermore, to avoid possible collisions, we assume that T encodes inputs uniquely; i.e., $T(M, apk, b, \omega) \neq T(M', apk', b', \omega')$ unless all these values are equal. Then we can define our VES as follows:

- $\mathsf{KeyGen}(1^k)$: Output $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}'(1^k)$.
- $\mathsf{Sign}(sk, M)$: Output $\sigma \xleftarrow{\$} \mathsf{Sign}'(sk, T(M, \perp, \perp, \perp))$.
- $\mathsf{Verify}(pk, \sigma, M)$: If σ is of the form (apk, ω, ω') then check that $\mathsf{VEVerify}(pk, apk, \omega, M) = 1$ and $\mathsf{Verify}'(apk, \omega', T(M, apk, 1, \omega)) = 1$; output 1 if and only if both of these checks pass. Otherwise, if σ is a single element then check that $\mathsf{Verify}'(pk, \sigma, T(M, \perp, \perp, \perp)) = 1$ and output 1 if and only if this check passes; in all other cases, output 0.
- $\mathsf{AKeyGen}(1^k)$. Compute $(apk', ask') \xleftarrow{\$} \mathsf{KeyGen}'(1^k)$ and output $(apk := apk', ask := (ask', apk))$.
- $\mathsf{VESign}(sk, apk, M)$: Output $\omega \xleftarrow{\$} \mathsf{Sign}'(sk, T(M, apk, 0, \perp))$.
- $\mathsf{VEVerify}(pk, apk, \omega, M)$: Output $\mathsf{Verify}'(pk, \omega, T(M, apk, 0, \perp))$.
- $\mathsf{Resolve}(ask, pk, \omega, M)$: If $\mathsf{VEVerify}(pk, apk, \omega, M) = 0$ output $\perp$. Otherwise, compute $\omega' \xleftarrow{\$} \mathsf{Sign}'(ask, T(M, apk, 1, \omega))$ and output $\sigma := (apk, \omega, \omega')$.

Essentially then, signing the message $T(M, apk, 0, \perp)$ corresponds to signing the message and then applying the "Do Not Verify" stamp (but indicating that the arbiter corresponding to apk may resolve if necessary), while signing the message $T(M, apk, 1, \omega)$ corresponds to applying the "Yes Do Verify" stamp. To show that this is a secure VES, we first prove that it satisfies completeness.

Theorem 3.1. *If the signature scheme* $(\mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Verify}')$ *is complete, then the VES construction is complete as well.*

Proof. By definition, for any $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$, $(apk, ask) \xleftarrow{\$} \mathsf{AKeyGen}(1^k)$, and $M \in \mathcal{M}$, an honestly computed VES ω looks like $\mathsf{Sign}'(sk, apk, T(M, apk, 0, \perp))$. As $\mathsf{VEVerify}(pk, apk, \omega, M) = \mathsf{Verify}'(pk, \omega, T(M, apk, 0, \perp))$, the completeness of the underlying signature scheme guarantees that this check will pass. As for resolution, by definition $\mathsf{Resolve}(ask, pk, \omega, M) = (apk, \omega, \omega') \xleftarrow{\$} \mathsf{Sign}'(ask,$

$T(M, apk, 1, \omega)))$, and Verify, on input (apk, ω, ω'), checks that $\mathsf{VEVerify}(pk, apk, \omega, M) = 1$ and $\mathsf{Verify}'(apk, \omega', T(M, apk, 1, \omega)) = 1$. As we've already argued that this first of these checks will pass, and the second will pass again by completeness of the signature scheme, the entire check will pass and $\mathsf{Verify}(pk, (apk, \omega, \omega'), M) = 1$. □

We now prove that our construction also satisfies the three security properties defined in Section 2, beginning with unforgeability (as defined in Definition 2.2).

Theorem 3.2. *If the signature scheme* $(\mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Verify}')$ *is EUF-CMA secure, then the VES construction is unforgeable.*

Proof. To prove this, we show that if there exists an adversary $\mathcal{A}$ that breaks the unforgeability of the VES scheme with some non-negligible probability ϵ, then there exists an adversary $\mathcal{B}$ that breaks the unforgeability of the underlying signature scheme with the same probability. To start, $\mathcal{B}$ will take as input a public key pk. It then proceeds as follows:

1. $\mathcal{B}$ generates $(apk, ask) \xleftarrow{\$} \mathsf{AKeyGen}(1^k)$ and gives pk and apk as inputs to $\mathcal{A}$. Because apk was generated honestly and pk is assumed to be the output of KeyGen' and thus KeyGen, both of these keys will be distributed identically to what $\mathcal{A}$ expects.
2. When $\mathcal{A}$ queries its Sign oracle on a message M, $\mathcal{B}$ creates a new message $M' := T(M, \perp, \perp, \perp)$ and queries its own Sign' oracle on M' to get back a signature σ that it then returns to $\mathcal{A}$. By definition, $\sigma \xleftarrow{\$} \mathsf{Sign}'(sk, T(M, \perp, \perp, \perp)) = \mathsf{Sign}(sk, M)$, so the σ returned to $\mathcal{A}$ will be distributed identically to what it expects.
3. When $\mathcal{A}$ queries its VESign oracle on a message M, $\mathcal{B}$ creates a new message $M' := T(M, apk, 0, \perp)$ and queries its own Sign' oracle on M' to get back a signature σ that it then returns to $\mathcal{A}$. By definition, we have that $\sigma \xleftarrow{\$} \mathsf{Sign}'(sk, T(M, apk, 0, \perp)) = \mathsf{VESign}(sk, apk, M)$, so the σ returned to $\mathcal{A}$ will be distributed identically to the one that it expects.
4. When $\mathcal{A}$ queries its $\mathsf{Resolve}$ oracle on a message M and a verifiably encrypted signature ω, $\mathcal{B}$ will use its knowledge of ask to execute the code of $\mathsf{Resolve}$ honestly to obtain a tuple of the form $\sigma := (apk, \omega, \omega')$ that it returns to $\mathcal{A}$. As $\mathcal{B}$ is behaving completely honestly, this will again be distributed identically to what $\mathcal{A}$ expects.
5. At some point $\mathcal{A}$ will output a message-signature pair (M, ω) such that M was not queried to any of the oracles but $\mathsf{VEVerify}(pk, apk, \omega, M) = 1$; $\mathcal{B}$ will then output $(T(M, apk, 0, \perp), \omega)$. By definition of $\mathsf{VEVerify}$, if $\mathsf{VEVerify}(pk, apk, \omega, M) = 1$ then $\mathsf{Verify}'(pk, T(M, apk, 0, \perp), \omega) = 1$ and thus $\mathcal{B}$'s output will pass verification; similarly, if $\mathcal{A}$ did not query its VESign oracle on M then, by definition of $\mathcal{B}$, we know that $\mathcal{B}$ did not query $T(M, apk, 0, \perp)$ to its Sign' oracle, and its output will therefore be a valid forgery.

As $\mathcal{B}$ therefore succeeds whenever $\mathcal{A}$ does, and the interaction with $\mathcal{B}$ is furthermore identical to the interaction that $\mathcal{A}$ expects, $\mathcal{B}$ will succeed with the same non-negligible probability ϵ as $\mathcal{A}$. □

Next, we prove that our construction is opaque, as defined in Definition 2.3.

Theorem 3.3. *If the signature scheme* $(\mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Verify}')$ *is EUF-CMA secure, then the VES construction is opaque.*

Proof. To prove this, we show that if there exists an adversary $\mathcal{A}$ that breaks the opacity of the VES scheme with some non-negligible probability ϵ, then there exists an adversary $\mathcal{B}$ that breaks the unforgeability of the signature scheme with probability $\epsilon/2$. To start, $\mathcal{B}$ will take as input a public key pk'. It then picks a random bit $b \xleftarrow{\$} \{0,1\}$ to decide which path it thinks $\mathcal{A}$ will pursue: if $b = 0$ then it assumes $\mathcal{A}$ will produce a forgery of the form (apk, ω, ω'), and if $b = 1$ then it assumes $\mathcal{A}$ will produce a forgery of the form σ. We discuss both of these paths as follows:

1. If $b = 0$ then $\mathcal{B}$ will generate $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$. It will then set $apk := pk'$ and give pk and apk to $\mathcal{A}$. As $\mathsf{AKeyGen}$ calls KeyGen' and pk' is assumed to be output by KeyGen', this will be distributed identically to what $\mathcal{A}$ expects.

   If instead $b = 1$ then $\mathcal{B}$ will generate $(apk, ask) \xleftarrow{\$} \mathsf{AKeyGen}(1^k)$. It will then set $pk := pk'$ and give pk and apk to $\mathcal{A}$. Again, as KeyGen calls KeyGen' and pk' is assumed to be output by KeyGen', this will be distributed identically to what $\mathcal{A}$ expects.
2. When $\mathcal{A}$ queries its VESign oracle on a message M, $\mathcal{B}$ again has two choices. If $b = 0$ then $\mathcal{B}$ can use its knowledge of the signing key sk to honestly execute the code of VESign and return the resulting ω; the distribution here is by definition identical to the one that $\mathcal{A}$ expects.
   If instead $b = 1$ then $\mathcal{B}$ sets $M' := T(M, apk, 0, \perp)$ and queries its own Sign' oracle on M' to get back a signature σ that it then returns to $\mathcal{A}$. By definition, $\sigma \xleftarrow{\$} \mathsf{Sign}'(sk, T(M, apk, 0, \perp)) = \mathsf{VESign}(sk, apk, M)$, so the σ returned to $\mathcal{A}$ will be distributed identically to what it expects.
3. When $\mathcal{A}$ queries its $\mathsf{Resolve}$ oracle on a message M and a verifiably encrypted signature ω, $\mathcal{B}$ can first check that $\mathsf{VEVerify}(pk, apk, \omega, M) = 1$ and abort if not; then, it again has two choices. If $b = 0$ then it sets $M' := T(M, apk, 1, \omega)$ and queries its own Sign' oracle on M' to get back a signature σ; it will then return (apk, ω, σ) to $\mathcal{A}$. By definition, $\sigma \xleftarrow{\$} \mathsf{Sign}'(sk, T(M, apk, 1, \omega))$ and so the resulting (apk, ω, σ) will again be distributed identically to what $\mathcal{A}$ expects.
   If instead $b = 1$ then $\mathcal{B}$ will use its knowledge of the secret key ask to execute the code of $\mathsf{Resolve}$ honestly and return the resulting (apk, ω, ω'); the distribution here is then by definition identical to the one that $\mathcal{A}$ expects.
4. At some point, $\mathcal{A}$ will output a message-signature pair (M, σ) such that $\mathsf{Verify}(pk, \sigma, M) = 1$ but $(M, \cdot)$ was not queried to the $\mathsf{Resolve}$ oracle. If $b = 0$ then $\mathcal{B}$ will check that σ is of the form (apk, ω, ω'); it it is not, then $\mathcal{B}$ must abort. If it is then, looking at the definition of Verify, we see it must be the case that $\mathsf{Verify}'(pk, \omega, T(M, apk, 0, \perp)) = 1$ and $\mathsf{Verify}'(apk, \omega', T(M, apk, 1,$

$\omega)) = 1$. As we know that $\mathcal{A}$ never queried its $\mathsf{Resolve}$ oracle on ω we also know that $\mathcal{B}$ never queried its Sign' oracle on $T(M, apk, 1, \omega)$ and therefore $\mathcal{B}$ can output $(T(M, apk, 1, \omega), \omega')$ to win its game.
Otherwise, if $b = 1$ then $\mathcal{B}$ will once again check if σ is of the form (apk, ω, ω'). If it is, then it is once again the case that $\mathsf{Verify}'(pk, \omega, T(M, apk, 0, \perp)) = 1$; if $\mathcal{A}$ never queried its VESign oracle on M, then $\mathcal{B}$ never queried its Sign' oracle on $T(M, apk, 0, \perp)$ and it can output $(T(M, apk, 0, \perp), \omega)$ to win its game. Otherwise, it can check if σ is a single element. If it is, then $\mathcal{B}$ can output $(T(M, \perp, \perp, \perp), \sigma)$ to once again win its game.

As $\mathcal{B}$ succeeds whenever $\mathcal{A}$ does and it correctly guesses which key $\mathcal{A}$ will use (which it will with probability $1/2$, as it guesses randomly), and interactions with $\mathcal{B}$ (in either execution) are furthermore identical to those that $\mathcal{A}$ expects, $\mathcal{B}$ will succeed with probability $\epsilon/2$ in providing a forgery for the signature scheme. □

Finally, we prove that our construction is extractable, as defined in Definition 2.4. In fact, it is not just the case that it should be hard to produce a VES that verifies but cannot be resolved to a valid signature; by how $\mathsf{Resolve}$ and Verify are defined, this is actually impossible.

Theorem 3.4. *The VES construction is unconditionally extractable.*

Proof. To prove this, we show that for all $(apk, ask) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$, $M \in \mathcal{M}$, ω, and pk, every time $\mathsf{VEVerify}(pk, apk, \omega, M) = 1$ it must be the case that $\mathsf{Verify}(pk, apk, \mathsf{Resolve}(ask, pk, \omega, M), M) = 1$ as well; this implies that the probability that any (even unbounded) adversary $\mathcal{A}$ can output (pk, ω, M) such that $\mathsf{VEVerify}(pk, apk, \omega, M) = 1$ but $\mathsf{Verify}(pk, apk, \mathsf{Resolve}(ask, pk, \omega, M), M) = 0$ is equal to 0 and thus the scheme is unconditionally extractable.

To therefore show that $\mathsf{VEVerify}(pk, apk, \omega, M) = 1$ implies $\mathsf{Verify}(pk, \mathsf{Resolve}(ask, pk, \omega, M)) = 1$, define $(apk, \omega, \omega') \xleftarrow{\$} \mathsf{Resolve}(ask, pk, \omega, M)$. Then we observe that, by the definition of the scheme, $\mathsf{Verify}(pk, apk, (apk, \omega, \omega'), M) = \mathsf{VEVerify}(pk, apk, \omega, M) \wedge \mathsf{Verify}'(apk, \omega', T(M, apk, 1, \omega))$. As $\mathsf{Resolve}$ guarantees that the second condition is satisfied (i.e., $\mathsf{Verify}'(apk, \omega', T(M, apk, 1, \omega)) = 1$), this reduces to $\mathsf{Verify}(pk, apk, (apk, \omega, \omega'), M) = \mathsf{VEVerify}(pk, apk, \omega, M)$ and thus the two values must always agree. □

4 Resolution Independence

As we've demonstrated in the previous section, the existing definitions for verifiably encrypted signatures do not seem to fully capture their desired functionality, as in particular we constructed a secure VES using only signatures. Furthermore, in our scheme the signatures returned by the arbiter look completely different from the regular signatures produced by Sign. In this section, we attempt to close this functional gap by proposing a new notion, *resolution independence*, that requires that the signatures returned by the arbiter and by the signer look the same. We then prove that our signature-based construction does not satisfy resolution independence whereas, to the best of our knowledge, all previous VES constructions do.

4.1 Resolution independence

Informally, we want that the values output by the Resolve algorithm look like regular signatures. More formally, we have the following definition:

Definition 4.1. *A VES* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{AKeyGen}, \mathsf{VESign}, \mathsf{VEVerify}, \mathsf{Resolve})$ *is* resolution independent *if for all* $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$, $(apk, ask) \xleftarrow{\$} \mathsf{AKeyGen}(1^k)$, *and* $m \in \mathcal{M}$, *the distributions* $\{\mathsf{Sign}(sk, m)\}$ *and* $\{\mathsf{Resolve}(ask, pk, \mathsf{VESign}(sk, apk, m), m)\}$ *are identical.*

To begin motivating why resolution independence is the "right" definition to capture the desired VES functionality, we first observe that our signature-based construction from Section 3 cannot be resolution independent, as regular signatures and signatures output by the arbiter have completely different forms.

Theorem 4.1. *The VES construction in Section 3 is not resolution independent.*

Proof. Recall that signatures output by the signer are in Σ; i.e., the space of all possible signatures. Signatures output by the arbiter, however, consist of a public key and two signatures, meaning that if the space of all possible apk values is A, then they are in the space (A, Σ, Σ); the distributions over the two types of signatures are therefore not identical. □

4.2 Existing schemes satisfy resolution independence

In order to further separate our signature-based construction from existing VES schemes, we also demonstrate that, to the best of our knowledge, all existing VES schemes are in fact resolution independent. As there are too many VES constructions in the literature to enumerate here, we focus on three (which we picked to demonstrate different types of schemes): the original BGLS construction [5], which is based on pairings and the BLS signature scheme [6], a construction due to Lu et al. [12] that is also based on pairings but uses the Waters signature [17], and a construction due to Rückert [14] that is based on the RSA signature scheme.

BGLS [5]. The BGLS scheme works over a prime-order bilinear group G with a generator g and a hash function $H : \{0,1\}^* \to G$. The arbiter's keys are $ask = x' \xleftarrow{\$} \mathbb{F}_p$ and $apk = v' = g^{x'} \in G$, and the user's keys are $sk = x \xleftarrow{\$} \mathbb{F}_p$ and $pk = v = g^x \in G$. As we can see in the algorithm descriptions below, Sign forms a BLS signature, while VESign runs Sign and then encrypts the signature using ElGamal encryption. The Resolve algorithm then decrypts and pulls out the original signature.

- $\mathsf{Sign}(sk, M)$: Parse $sk = x$ and return $\sigma := H(M)^x$.
- $\mathsf{VESign}(sk, apk, M)$: Parse $sk = x$ and $apk = v'$ and compute $\sigma := H(M)^x$. Pick $r \xleftarrow{\$} \mathbb{F}_p$ and set $\mu := g^r$ and $\sigma' := (v')^r$. Finally, compute $\omega' := \sigma\sigma'$ and output (ω', μ).

– $\mathsf{Resolve}(ask, pk, \omega, M)$: Parse $ask = x'$ and $\omega = (\omega', \mu)$ and output $\sigma := \omega/\mu^{x'}$.

To see that the signatures output by Sign and $\mathsf{Resolve}$ are in fact identical, we observe that

$$\begin{aligned}
\mathsf{Resolve}(ask, pk, \mathsf{VESign}(sk, apk, M; r), M) &= \frac{\omega}{\mu^{x'}} \\
&= \frac{\sigma\sigma'}{\mu^{x'}} \\
&= \frac{H(M)^x (v')^r}{(g^r)^{x'}} \\
&= \frac{H(M)^x ((g^{x'})^r)}{(g^{rx'})} \\
&= H(M)^x \\
&= \mathsf{Sign}(sk, M),
\end{aligned}$$

and thus the scheme satisfies resolution independence.

Lu et al. [12]. The Lu et al. scheme also works in a prime-order bilinear group G with generator g. It builds off of the Waters signature [17], which we briefly recall uses a secret key $sk = \alpha \xleftarrow{\$} \mathbb{F}_p$ (corresponding to a public key $pk = A = e(g,g)^\alpha$, where e is the bilinear map) to create signatures of the form (S_1, S_2), where $S_1 := g^\alpha (u' \prod_i u_i^{b_i})^r$, $S_2 := g^r$ for $u', u_1, \ldots, u_k \xleftarrow{\$} G$, $r \xleftarrow{\$} \mathbb{F}_p$, and where b_i is the i-th bit of the message M; i.e., $M = b_1 \ldots b_k$. We denote the Waters signing algorithm as $\mathsf{WSign}(sk, M)$.

As we see in the algorithm descriptions below, Sign is equivalent to WSign. VESign will first run Sign and then blind the resulting signature; this means users' keys will just be keys for the Waters signature, and the arbiter's keys will be $sk = \beta \xleftarrow{\$} \mathbb{F}_p$ and $pk = v = g^\beta$. The $\mathsf{Resolve}$ algorithm first pulls out the underlying signature, and then re-randomizes it.

– $\mathsf{Sign}(sk, M)$: Output $(S_1, S_2) \xleftarrow{\$} \mathsf{WSign}(sk, M)$.
– $\mathsf{VESign}(sk, apk, M)$: Parse $apk = v$. Compute $(S_1, S_2) \xleftarrow{\$} \mathsf{WSign}(sk, M)$, pick a random $s \xleftarrow{\$} \mathbb{F}_p$, and compute $K_1 := S_1 \cdot v^s$, $K_2 := S_2$, and $K_3 := g^s$. Output (K_1, K_2, K_3).
– $\mathsf{Resolve}(ask, pk, \omega, M)$: Parse $ask = \beta$, $\omega = (K_1, K_2, K_3)$, and $M = b_1 \ldots b_k$. Check first that ω is a valid VES on M, and then unblind the signature by computing $S_1 := K_1 K_3^{-\beta}$ and $S_2 := K_2$. Now, re-randomize the signature by picking $s \xleftarrow{\$} \mathbb{F}_p$ and computing $S_1' := S_1 (u' \prod_i u_i^{b_i})^s$ and $S_2' := S_2 \cdot g^s$. Output (S_1', S_2').

To see that the outputs of Sign and $\mathsf{Resolve}$ are distributed identically, we observe that

$$\begin{aligned}
\mathsf{Resolve}(ask, pk, \mathsf{VESign}(sk, apk, M), M) &= (K_1 \cdot K_3^{-\beta} \cdot (u' \prod_i u_i^{b_i})^{r'},\ K_2 \cdot g^{r'}) \\
&= (S_1 \cdot v^s \cdot g^{-\beta s} \cdot (u' \prod_i u_i^{b_i})^{r'},\ S_2 \cdot g^{r'}) \\
&= (S_1 \cdot (g^{\beta s} g^{-\beta s}) \cdot (u' \prod_i u_i^{b_i})^{r'},\ g^{r+r'}) \\
&= (g^{\alpha} \cdot (u' \prod_i u_i^{b_i})^{r} \cdot (u' \prod_i u_i^{b_i})^{r'},\ g^{r+r'}) \\
&= (g^{\alpha} \cdot (u' \prod_i u_i^{b_i})^{r+r'},\ g^{r+r'}) \\
&= \mathsf{WSign}(sk, M; r + r')
\end{aligned}$$

for random $r, r' \xleftarrow{\$} \mathbb{F}_p$. The signature is therefore a random signature on M and thus has the same distribution as the signature output by $\mathsf{Sign}(sk, M)$ and the scheme is resolution independent.

Rückert [14]. Rückert's construction is a stateful VES based on the RSA signature scheme, which we recall works as follows: keys are of the form $pk := (N, e)$ and $sk := (pk, d)$, where $N = pq$ and e and d are values such that $ed \equiv 1 \bmod \phi(N)$. To form a signature, $\mathsf{RSASign}$ computes $\sigma := H(M)^d \bmod N$, which can be verified by checking that $H(M) \equiv \sigma^e \bmod N$. Briefly, in Rückert's construction, when forming the i-th VES, the RSA signature is blinded using a secret value x_i, which is then encrypted under the arbiter's public key. To ensure that this ciphertext contains the appropriate blinding factor, the signer will form an authentication path in a particular Merkle tree. This means that the keys for the arbiter will look like $apk = (N_e, e, \mathsf{authpk})$ and $ask = (apk, d, \mathsf{authsk})$, where (N_e, e, d) are RSA keys and authpk and authsk are used for the Merkle authentication. The user's keys, on the other hand, will look like $pk = (N_u, u, \rho, \sigma_\rho)$ and $sk = (pk, v, T)$, where (N_u, u, v) are RSA keys, T is the Merkle tree (and also contains information about the blinding factors $\{x_i\}$ by providing the seed used to generate them), ρ is the root of the tree, and σ_ρ is a RSA signature on ρ.

- $\mathsf{Sign}(sk, M)$: Output $\sigma := \mathsf{RSASign}(sk, M)$.
- $\mathsf{VESign}(sk, apk, M)$: Parse $sk = (pk = (N_u, u, \rho, \sigma_\rho), v, T)$ and $apk = (N_e, e, \mathsf{authpk})$. First form the signature $\sigma := \mathsf{RSASign}(sk, M)$. Now, increment the counter i, blind the signature by forming $\alpha := \sigma x_i \bmod N_s$, and encrypt x_i by forming $\beta := x_i^e \bmod N_e$, and $\gamma := x_i^u \bmod N_u$. Finally, generate the authentication path π for x_i in the Merkle tree T, and output $\omega := (\alpha, \beta, \gamma, \pi)$.
- $\mathsf{Resolve}(ask, pk, \omega, M)$: Parse $ask = (apk = (N_e, e, \mathsf{authpk}), d, \mathsf{authsk})$, $pk = (N_u, u, \rho, \sigma_\rho)$, and $\omega = (\alpha, \beta, \gamma, \pi)$. First check that ω is a valid VES on M, and then compute $x' := \beta^d \bmod N_e$ and output $\sigma := \alpha / x' \bmod N_u$.

To see this that the signatures output by Resolve and Sign are identical, we observe that

$$\begin{aligned}\mathsf{Resolve}(ask, pk, \mathsf{VESign}(sk, apk, M), M) &= \alpha/x' \bmod N_u \\ &= \alpha/\beta^d \bmod N_u \\ &= (\sigma x_i)/(x_i^e)^d \bmod N_u \\ &= (\sigma x_i)/x_i \bmod N_u \\ &= \sigma \bmod N_u,\end{aligned}$$

which is the same as the signature output by $\mathsf{Sign}(sk, M)$ and the scheme is therefore resolution independent.

5 Resolution Duplication and Public-Key Encryption

While resolution independence, as we saw in the previous section, can be used to separate our particular signature-based VES construction from existing constructions, it still does not require that more than just signatures are required to construct a verifiably encrypted signature scheme, although the name would suggest otherwise. We therefore propose in this section a stronger notion of resolution independence, *resolution duplication*, in which the signer must be able to output a signature that is *identical* to that of the arbiter. This definition is less general than resolution independence (yet still met by some existing VES constructions), but we show that it implies the existence of public key encryption.

5.1 Resolution duplication

In spirit, resolution independence requires a functionality similar to that of encryption: the VES ω contains a signature σ, yet should not reveal this σ to anyone in possession of just ω (by opacity). The exception to this rule is the arbiter who, according to completeness, should be able to pull out from ω a signature σ' that, according to resolution independence, has the same distribution as σ. While this comes close to encryption, the fact that σ' and σ might be identically distributed but not identical means the functionality is not exactly the same.

In Section 4.2, however, we saw that in fact two out of the three schemes presented did in fact meet this exact requirement (and this was not an accident; indeed most VES schemes meet this requirement); in particular, because both the BGLS and Rückert schemes were based on *unique signatures* [10,13], any two signatures on the same message with the same distribution must be identical. To formally capture this stronger property, we have the following definition:

Definition 5.1. *A VES* (KeyGen, Sign, Verify, AKeyGen, VESign, VEVerify, Resolve) *is* resolution duplicate *if (1) it is resolution independent, (2)* Resolve

is deterministic, and (3) there exists an additional PPT algorithm $\mathsf{Extract}(\cdot,\cdot,\cdot)$ *such that for all* $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$*,* $(apk, ask) \xleftarrow{\$} \mathsf{AKeyGen}(1^k)$*,* $m \in \mathcal{M}$*, and random tapes* $r \in \{0,1\}^*$*, it is the case that* $\mathsf{Extract}(sk, m, r) = \mathsf{Resolve}(ask, pk, \mathsf{VESign}(sk, apk, m; r), m)$*.*

While this strengthened definition can no longer be met by *all* existing VES constructions (e.g., any that use a randomized resolution algorithm, such as the Lu et al. one above), we will see below that any secure VES satisfying resolution duplication can be used to construct public key encryption. In this respect then, VES constructions meeting resolution duplication guarantee that some kind of encryption really is taking place, whereas we cannot make the same guarantees about ones that meet only resolution independence.

Finally, we note that the existence of Extract is not a particularly strong requirement; for a unique signature, for example, Extract can simply run Sign. Furthermore, in our usage in the next section, VESign and Extract will be run by the same party, so the randomness used in VESign can simply be remembered and given to Extract.

5.2 Constructing public key encryption

Using resolution duplication, our construction of public key encryption is fairly straightforward. Recall first our intuitive outline above: the signature σ can be thought of as the plaintext and the VES ω as the ciphertext encrypted under the public key of the arbiter; running Resolve and pulling out the underlying σ is therefore how the arbiter decrypts. Because we want to encrypt arbitrary bits rather than signatures, however, we instead use the Goldreich-Levin trick [9] and the fact that ω should not reveal σ to treat $\langle \sigma, r \rangle$ as a *hard-core predicate* for VESign; i.e., given ω and r, it should be hard to predict the value of $\langle \sigma, r \rangle$ (where $r \xleftarrow{\$} \{0,1\}^{|\sigma|}$ and $\langle \sigma, r \rangle$ denotes the inner product of σ and r modulo 2).[3] To construct our encryption scheme, we therefore prove first that this property holds:

Theorem 5.1. *Let* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{AKeyGen}, \mathsf{VESign}, \mathsf{VEVerify}, \mathsf{Resolve})$ *be a verifiably encrypted signature scheme, and let* $b(x, r) := \langle x, r \rangle \bmod 2$ *for any* x *and* r *such that* $|x| = |r|$*. Then, if the VES is opaque for all messages* $m \in \mathcal{M}$*,* $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$*, and* $(apk, ask) \xleftarrow{\$} \mathsf{AKeyGen}(1^k)$*, it is hard to compute* $b(\sigma, r)$ *given* m*,* apk*,* pk*,* $\omega \xleftarrow{\$} \mathsf{VESign}(sk, apk, m)$*, and* $r \xleftarrow{\$} \{0,1\}^{|\sigma|}$*, where* $\sigma := \mathsf{Resolve}(ask, pk, \omega, M)$*.*

Our proof strategy for this theorem closely follows that of Goldreich [8]. First, we describe how an adversary $\mathcal{B}$ attempting to break opacity can, by using specific values of r, meaningfully use an adversary $\mathcal{A}$ that can predict the value of $b(\sigma, r)$ to recover the value of σ from the verifiably encrypted signature.

[3] This isn't a hard-core bit in the usual sense, since VESign is randomized and therefore not a function, but we can nevertheless argue that it should be hard to predict.

Then, following Goldreich's exact argument, we argue how these specific values of r can be chosen to ensure that $\mathcal{B}$'s success probability will be appropriately correlated with that of $\mathcal{A}$.

At a high level, to use $\mathcal{A}$ to recover σ, $\mathcal{B}$ will first receive as input public keys pk and apk. To now prepare an input for $\mathcal{A}$, $\mathcal{B}$ can first pick a random message $m \xleftarrow{\$} \mathcal{M}$ and query its VESign oracle on m to get back a value ω. It now picks a random value $r \xleftarrow{\$} \{0,1\}^{|\sigma|}$ (note that, while $\mathcal{B}$ does not know σ, it might still know its length, for example if σ is encrypted) and gives (ω, r) to $\mathcal{A}$; this causes $\mathcal{A}$ to return its guess b' for the bit $b(\sigma, r)$. We could then also have $\mathcal{B}$ give to $\mathcal{A}$ $(\omega, r \oplus e_i)$ for all i, where e_i has a 1 in the i-th place and a 0 everywhere else, and get back in return guess bits b_i. If $\mathcal{A}$ guesses b' and b_i correctly for each i, then $\mathcal{B}$ can recover σ as follows: first, observe that $b(x, r) \oplus b(x, s) = b(x, r \oplus s)$. Then, if $b' = b(\sigma, r)$ and $b_i = b(\sigma, r \oplus e_i)$, it must be the case that

$$b' \oplus b_i = b(\sigma, r) \oplus b(\sigma, r \oplus e_i) = b(\sigma, r \oplus (r \oplus e_i)) = b(\sigma, e_i) = \sigma_i;$$

that is, that $b' \oplus b_i$ is the i-th bit of σ. Repeating this process for each i, $\mathcal{B}$ can therefore recover $\sigma_i := b' \oplus b_i$ and $\sigma := \sigma_1 \dots \sigma_n$.

As observed by Goldreich, however, this process of using $r \oplus e_i$ might significantly blow up $\mathcal{B}$'s error probability, to the point where we cannot argue that if $\mathcal{A}$ has some non-negligible success probability then so does $\mathcal{B}$. We therefore follow Goldreich's exact argument to pick more clever choices for the randomness r and thus guarantee a non-negligible success probability for $\mathcal{B}$.

Proof. To show this, we assume that there exists an adversary $\mathcal{A}$ that, given (pk, apk, m, ω, r) such that $\omega \xleftarrow{\$} \mathsf{VESign}(sk, apk, m)$, $\sigma := \mathsf{Resolve}(ask, pk, \omega, m)$, and $|r| = |\sigma| = n$, can predict the value of $b(\sigma, r)$ with some non-negligible advantage ϵ and use it to construct an adversary $\mathcal{B}$ that can recover the signature σ from ω (i.e., break opacity), with related non-negligible probability ϵ'. First, we observe that if ϵ is non-negligible then by definition, $\mathcal{A}$'s advantage must be $\epsilon(n) > 1/p(n)$ for some polynomial $p(\cdot)$, and that furthermore this must hold for infinitely many n (i.e., there must exist an infinite set N such that $\epsilon(n) > 1/p(n)$ for $n \in N$). We furthermore establish the following two claims, both due to Goldreich [8]:

Claim. [8] There exists a set $S_n \subseteq \{0,1\}^n$ of cardinality at least $2^n \cdot (\epsilon(n)/2)$ such that for every $\sigma \in S_n$, it holds that

$$s(x) := \Pr[\mathcal{A}(pk, apk, m, \omega, R_n) = b(\sigma, R_n)] \geq \frac{1}{2} + \frac{\epsilon(n)}{2},$$

where the probability is taken over all possible values of R_n and internal coin tosses of $\mathcal{A}$.

Claim. [8] For every $\sigma \in S_n$ and $i \in \{1, \dots, n\}$, it holds that

$$\Pr\left[|\{J : b(x, r_J) \oplus \mathcal{A}(pk, apk, m, \omega, r_J \oplus e_i) = \sigma_i\}| > \frac{1}{2} \cdot (2^\ell - 1)\right] > 1 - \frac{1}{2n} \tag{1}$$

where $r_J := \oplus_{j \in J} s_j$ and the s_j values are chosen independently and uniformly from $\{0,1\}^n$.

To prepare inputs for $\mathcal{A}$ given pk and apk, $\mathcal{B}$ first picks a random message $m \xleftarrow{\$} \mathcal{M}$ and queries its VESign oracle on m to get back a value ω. It now sets $\ell := \lceil \log_2(2n \cdot p(n)^2 + 1) \rceil$, where we recall $n := |\sigma|$ and $p(\cdot)$ is such that $\epsilon(n) > 1/p(n)$. It now samples $s_1, \ldots, s_\ell \xleftarrow{\$} \{0,1\}^n$ and $t_1, \ldots, t_\ell \xleftarrow{\$} \{0,1\}$, where t_i acts as $\mathcal{B}$'s guess for the value $b(\sigma, s_i)$. Next, for every non-empty set $J \subseteq \{1, 2, \ldots, \ell\}$, $\mathcal{B}$ computes $r_J := \oplus_{j \in J} s_j$ and $\rho_J := \oplus_{j \in J} t_j$. $\mathcal{B}$ now gives to $\mathcal{A}$, for all $i \in \{1, \ldots, n\}$ and non-empty $J \subseteq \{1, \ldots, \ell\}$, the tuple $(apk, pk, m, \omega, r_J \oplus e_i)$, for which it will get back a guess bit b_{iJ}. $\mathcal{B}$ then sets $z_{iJ} := \rho_J \oplus b_{iJ}$; now, for every i, it sets z_i to be the majority of the z_{iJ} values, and outputs $z := z_1 \ldots z_n$.

We first observe that the $r_J \oplus e_i$ values given to $\mathcal{A}$ will be uniformly random and pairwise independent and thus distributed identically to the input that $\mathcal{A}$ expects (and, as all the other values are chosen honestly, its entire input will be identical to what it expects). To see this, we observe that the s_i values are chosen uniformly at random, and each r_J value is set as $\oplus s_j$, which will itself be uniformly random, and thus so will $r_J \oplus e_i$. Furthermore, because each subset J is distinct, the values will be pairwise independent as well.

To determine the success probability of $\mathcal{B}$, our proof now follows exactly the proof of Goldreich. In particular, we first observe that, by Claim 5.2,

$$s(x) \geq \frac{1}{2} + \frac{\epsilon(n)}{2} > \frac{1}{2} + \frac{1}{2p(n)}.$$

Furthermore, as the values s_i were chosen uniformly at random, the probability that our guesses were correct and $t_i = b(\sigma, s_i)$ for all i is

$$2^{-\ell} = \frac{1}{2n \cdot p(n)^2 + 1} = \frac{1}{\mathrm{poly}(n)},$$

which is non-negligible. Furthermore, if our guesses are indeed correct then

$$\rho_J = \oplus_{j \in J} t_j = \oplus_{j \in J} b(\sigma, s_j) = b(\sigma, \oplus_{j \in J} s_j) = b(\sigma, r_J)$$

for all non-empty sets J. In this case, we have

$$z_{iJ} = \rho_J \oplus b_{iJ} = b(\sigma, r_J) \oplus b_{iJ},$$

which we know is equal to σ_i with probability greater than $1 - 1/2n$ by Claim 5.2, meaning the overall probability that $z = \sigma$ is at least $1/2$. Putting everything together, we therefore know that $\mathcal{B}$ will succeed with probability at least $1/4p(n)$ for $\sigma \in S_n$; recalling further by Claim 5.2 that $|S_n| > 2^n/2p(n)$, we conclude that for random σ, $\mathcal{B}$ succeeds with probability at least $1/8p(n)^2$, or $\epsilon(n)^2/8$. $\square$

Now, armed with this theorem, we can construct public key encryption. To start, assume we have a VES ($\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{AKeyGen}, \mathsf{VESign}, \mathsf{VEVerify}, \mathsf{Resolve}$) with the extra algorithm $\mathsf{Extract}$ required by Definition 5.1. Then we can construct an IND-CPA secure public key encryption scheme ($\mathsf{EKeyGen}, \mathsf{Enc}, \mathsf{Dec}$) as follows:

– $\mathsf{EKeyGen}(1^k)$: Output $(pk, sk) \xleftarrow{\$} \mathsf{AKeyGen}(1^k)$.
– $\mathsf{Enc}(pk, m)$: Generate signing keys $(spk, ssk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$ and set $c_1 := spk$. Now pick a random tape r, compute $\omega := \mathsf{VESign}(ssk, pk, 0; r)$ and set $c_2 := \omega$. Next, compute $\sigma \xleftarrow{\$} \mathsf{Extract}(sk, m, r)$; finally, pick $r_\sigma \xleftarrow{\$} \{0,1\}^{|\sigma|}$, set $c_3 := r_\sigma$, and set $c_4 := m \oplus \langle \sigma, r_\sigma \rangle$. Output $c := (c_1, c_2, c_3, c_4)$.
– $\mathsf{Dec}(sk, c)$: Parse $c = (c_1, c_2, c_3, c_4)$. Check first that $\mathsf{VEVerify}(c_1, pk, c_2, 0) = 1$; if this check fails then output $\perp$. Otherwise, if it passes, compute $\sigma := \mathsf{Resolve}(sk, c_1, c_2, 0)$, and output $m := c_4 \oplus \langle \sigma, c_3 \rangle$.

Theorem 5.2. *If the verifiably encrypted signature is resolution duplicate (according to Definition 5.1), the above encryption scheme is correct.*

Proof. If the ciphertext c is formed as $c \xleftarrow{\$} \mathsf{Enc}(pk, m)$, then $c_1 = spk$ and $c_2 = \omega$, which allows decryption to compute $\sigma := \mathsf{Resolve}(sk, spk, \omega, 0)$. Additionally, we have $c_3 = r_\sigma$ and $c_4 = m \oplus \langle \sigma, r_\sigma \rangle$, where by resolution duplication the σ used to form c_4 is the same as the one produced by $\mathsf{Resolve}$. We therefore have that $c_4 \oplus \langle \sigma, c_3 \rangle = (m \oplus \langle \sigma, r_\sigma \rangle) \oplus \langle \sigma, r_\sigma \rangle = m$, so decryption really will produce the message. □

Theorem 5.3. *If the verifiably encrypted signature is opaque, then the above encryption scheme is IND-CPA secure.*

Proof. By Theorem 5.1, we know that if the VES is opaque then $\langle \sigma, r \rangle$ will be hard to predict given only ω; furthermore, if it is opaque for all messages then in particular this must hold for the message $m = 0$. Thus, to prove the theorem, we can show that if there exists an adversary $\mathcal{A}$ that breaks IND-CPA security with some non-negligible advantage ϵ then there exists an adversary $\mathcal{B}$ that can predict the value of $\langle \sigma, r \rangle$ for the message $m = 0$ with the same advantage.

To start, $\mathcal{B}$ will receive as input $(pk, apk, 0, \omega, r)$, where $\omega \xleftarrow{\$} \mathsf{VESign}(sk, apk, 0)$ and $r \xleftarrow{\$} \{0,1\}^{|\sigma|}$ for $\sigma := \mathsf{Resolve}(ask, pk, \omega, 0)$. $\mathcal{B}$ will now give $\mathcal{A}$ the public key apk and at some point will receive back a challenge query (m_0, m_1). To compute c^*, $\mathcal{B}$ will set $c_1^* := pk$, $c_2^* := \omega$, and $c_3^* := r$. It then picks random bits $b, b^* \xleftarrow{\$} \{0,1\}$ and sets $c_4^* := m_b \oplus b^*$, and returns $c^* := (c_1^*, c_2^*, c_3^*, c_4^*)$ to $\mathcal{A}$. When $\mathcal{A}$ outputs its guess bit b', $\mathcal{B}$ guesses b^* if $b = b'$ and $1 - b^*$ otherwise.

To see that interactions with $\mathcal{B}$ are indistinguishable from those that $\mathcal{A}$ expects, we observe that the apk given to $\mathcal{A}$ is distributed identically to what $\mathcal{A}$ expects from $\mathsf{EKeyGen}$. As for c^*, all the values except c_4^* are again distributed identically to what $\mathcal{A}$ expects: c_1^* is a random user public key, c_2^* is a valid VES, and c_3^* is a random string of the same length as σ. As for c_4^*, if $\mathcal{B}$ has correctly guessed the value of b^* (i.e., $b^* = \langle \sigma, r \rangle$), then $c_4^* = m_b \oplus \langle \sigma, r \rangle$, c^* is a valid encryption of m_b, and thus $\mathcal{A}$ should behave just as it does in the honest interaction (i.e., it should guess b with its usual non-negligible advantage ϵ). In this case, if $\mathcal{A}$ guesses b correctly, then $\mathcal{B}$ will assume that it guessed b^* correctly and thus output b^*. In the other case, if $\mathcal{B}$ did not guess b^* correctly, then c_4^* is just a random bit, meaning all information about m will be obscured and $\mathcal{A}$

will have no advantage. As $\mathcal{B}$ therefore succeeds at least whenever $\mathcal{A}$ succeeds and it correctly guesses b^*, which it will with probability $1/2$, $\mathcal{B}$ will succeed in predicting the value of $\langle \sigma, r \rangle$ with overall advantage at least $\epsilon/2$. □

Acknowledgments

The first three authors were supported by the MURI program under AFOSR Grant No. FA9550-08-1-0352. Brent Waters was supported by NSF CNS-0915361 and CNS-0952692, CNS-1228599 DARPA through the U.S. Office of Naval Research under Contract N00014-11-1-0382, DARPA N11AP20006, Google Faculty Research award, the Alfred P. Sloan Fellowship, Microsoft Faculty Fellowship, and Packard Foundation Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of Defense or the U.S. Government.

References

1. M. Abdalla and B. Warinschi. On the minimal assumptions of group signature schemes. In J. Lopez, S. Qing, and E. Okamoto, editors, *Proceedings of ICICS 2004*, volume 3269 of *LNCS*, pages 1–13. Springer-Verlag, Oct. 2004.
2. N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In L. Gong and C. Neuman, editors, *Proceedings of CCS 1997*, pages 7–17. ACM Press, Apr. 1997.
3. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4):593–610, Apr. 2000.
4. G. Ateniese. Verifiable encryption of digital signatures and applications. *Journal of Cryptology*, 7(1):1–20, Feb. 2004.
5. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proceedings of Eurocrypt 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer-Verlag, 2003.
6. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In *Proceedings of Asiacrypt 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer-Verlag, 2001.
7. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Proceedings of Crypto 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144. Springer-Verlag, 2003.
8. O. Goldreich. Three XOR-lemmas – an exposition, 1991. `http://www.wisdom.weizmann.ac.il/~oded/COL/xor.pdf`.
9. O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proceedings of STOC 1989*, pages 25–32, 1989.
10. S. Goldwasser and R. Ostrovsky. Invariant signatures and noninteractive zero-knowledge proofs are equivalent. In *Proceedings of Crypto 1992*, volume 740 of *Lecture Notes in Computer Science*, pages 228–245. Springer-Verlag, 1992.
11. F. Hess. On the security of the verifiably-encrypted signature scheme of Boneh, Gentry, Lynn, and Shacham. *Information Processing Letters*, 89(3):111–114, 2004.

12. S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters. Sequential aggregate signatures and multisignatures without random oracles. *Journal of Cryptology*, 2012.
13. A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In *Proceedings of Crypto 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 597–612. Springer-Verlag, 2002.
14. M. Rückert. Verifiably encrypted signatures from RSA without NIZKs. In *Proceedings of Indocrypt 2009*, pages 363–377, 2009.
15. M. Rückert, M. Schneider, and D. Schröder. Generic constructions for verifiably encrypted signatures without random oracles or NIZKs. In *Proceedings of ACNS 2010*, pages 69–86, 2010.
16. M. Rückert and D. Schröder. Security of verifiably encrypted signatures and a construction without random oracles. In *Proceedings of Pairing 2009*, pages 17–34, 2009.
17. B. Waters. Efficient identity-based encryption without random oracles. In *Proceedings of Eurocrypt 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer-Verlag, 2005.