One-Time Verifiably Encrypted Signatures A.K.A. Adaptor Signatures

Lloyd Fournier*

October 2019

Abstract

On Bitcoin-like ledgers, smart contract functionality can be realised without using the ledger's native smart contract language through the "adaptor signature" technique[**?**]. An adaptor signature is a kind of partial signature that, if completed, will reveal valuable information to the signer. In this work, we conduct the first formal analysis of the adaptor signature as an isolated primitive. We find that it offers similar functionality to the already well established concept of *verifiably encrypted signatures* (VES) with one notable difference: the decryption key can be recovered from the ciphertext and the decrypted signature. To capture this property, we formally introduce the notion of *one-time verifiably encrypted signatures*. To properly define their security in the Bitcoin layer-2 setting we revisit the original VES definitions and modify them to remove the ingrained assumption of a trusted third party.

After the extending our VES definitions to the one-time VES, we attempt to prove that the existing Schnorr and ECDSA adaptor signature schemes satisfy them. In the case of Schnorr, we succeed unconditionally but our definitions expose a (non-fatal) flaw in the ECDSA scheme. Nevertheless, we show how to use the ECDSA scheme to realise functionality in Bitcoin that was previously thought to be out of reach without Schnorr signatures or complex two-party ECDSA protocols.

1 Introduction

In 2017, Andrew Poelstra posted a message to the Mimblewimble mailing list[**?**] demonstrating an interesting feature of the Schnorr signature scheme. He showed that through a small tweak to the signing algorithm the core lock construction of the lightning payment channel network[**?**] could be realised without using Bitcoin's *smart contract* language called *script*. This was a remarkable discovery as the existing proposal relied heavily on enforcing the lock logic using script based hash locks. His explanation ended with "I'm very excited about this" and in retrospect his excitement seems justified. This breakthrough has been used to create "scriptless" variants of most Bitcoin layer-2[1] protocols (we give a fairly comprehensive list in Section **??**).

Scriptless scripts. Achieving smart contract logic without script has been termed *scriptless scripts* within the Bitcoin research community. It is a significant challenge to the usual conception of smart contracts. Instead of being expressed in a programming language, a smart contract becomes a kind of multi-party computation that outputs transaction signatures according to the desired contract logic. In other words, if according to a contract Alice is meant to receive funds when event X occurs, instead of expressing event X in some way using the smart contract language, the contract is set up such that Alice is only able to compute a signature on a transaction that claims the funds due to her when event X occurs. This is clearly beneficial to Alice because no passive observer of the blockchain can tell that the transaction depended on event X, rather it (optimistically) looks just like a normal payment transaction.

Verifiably Encrypted Signatures. In this work we frame the adaptor signature, the key building block of scriptless protocols, as a kind of signature encryption. In particular, we find it almost fits the definition of *verifiably encrypted signatures*[**?**] (VES) introduced for the BLS signature scheme[**?**]. A VES scheme enables a signer to produce an encrypted signature whose validity can be verified non-interactively. The adaptor signature is a VES where the encryption is *one-time* in a similar sense to a one-time pad: the decryption key can easily be recovered from the ciphertext and its plaintext (the signature).

*lloyd.fourn@gmail.com

[1] In this work we will use layer-2 to refer to any protocol that requires a direct communication channel between participants not just "off-chain" payment channel type protocols

Fair Exchange of Signatures Without a Trusted Party. This "one-time" property may seem like a bug, but in many settings it is actually a powerful feature. Consider the motivating example for the original VES scheme[**?**]: two parties, Alice and Bob, wish to fairly exchange signatures with the assistance of a trusted third party named the *adjudicator*. First, Alice sends to Bob her signature encrypted under the adjudicator's public key. Bob verifies the validity of the signature encryption and then sends his signature to Alice. If Alice refuses to reciprocate, Bob may ask the adjudicator to intervene by decrypting the encryption of Alice's signature he already has. If we assume the adjudicator is able to correctly assess the situation (i.e. will not be tricked by a malicious Bob) then this a secure optimistic fair signature exchange protocol.

With a one-time VES we can achieve the fair exchange of signatures *without* a trusted party under the assumption that the signatures need to be published publicly to be useful. The one-time property gives the Bitcoin layer-2 protocol designer leverage over malicious parties by forcing them to leak a secret key when they publish a decrypted signature. This leaked key can then be used by honest parties to carry on the protocol and claim the funds due to them. A simple one-time VES fair signature exchange protocol can be described as follows: Alice generates her signature encryption under a public encryption key she herself generates and sends the key and the ciphertext to Bob. Bob responds by generating a signature encrypted under the same key. Alice decrypts this signature and publishes it. Due to the one-time property of the VES, Bob is able to recover the Alice's decryption key and decrypt the signature from the ciphertext given to him by Alice.

1.1 Survey of Scriptless Protocols

In order to motivate our formalisation, we first review how one-time VES schemes (i.e. adaptor signatures) are used in existing scriptless layer-2 protocol proposals. We give a simplified explanation of each protocol below focusing on the role the VES plays in the construction. Interestingly not every proposal requires the VES to have the one-time property; the "Discreet Log Contracts" and lottery protocols only need an ordinary VES. In practice, each protocol requires a VES scheme with a two-party encrypted signing algorithm but we leave this out to simplify the description (Section **??** focuses on this point). In general, each scriptless protocol is more efficient in terms of transaction footprint and more confidential than its script based counterpart.

Atomic swaps. In an atomic swap, two parties, Alice and Bob, exchange the ownership of two assets, A and B, on two different ledgers, α and β. As initially conceived[**?**], Alice generates a random secret y and shares the hash of it $Y \leftarrow H(y)$ with Bob. Alice then locks A in a smart contract on α that will only release A if the Bob activates it with y' such that $Y = H(y')$. Bob locks B into a similar contract on β. Alice then claims B, by activating the contract on β with y. Bob then learns y and claims A with it on α.

The scriptless atomic swap protocol[**?**] is the classic example of how to apply adaptor signatures to create a practically identical structure without requiring smart contract based hash operations. Rather than a hash pre-image, Alice generates an encryption key-pair (y, Y). On α Alice gives Bob a one-time VES under Y on a transaction giving Bob A should he be able to decrypt it. Bob then sends Alice a one-time VES under Y on a transaction giving Alice B. Alice decrypts he signature with y and broadcasts it claiming B. Bob then sees the decrypted signature, and due to the one-time property of the VES, he is able to extract y from it and decrypts his own signature to claim A. Note that unlike the hash based protocol, the secret y is never placed into any transaction on either ledger making it much harder to link the transactions on α and β with each other.

Payment channels. Poelstra's original mailing list post applies the same transformation used in the scriptless atomic swap to the Lightning Network[**?**]. Payment channels gain an additional benefit from having a group element (the encryption key) as the lock, rather than a hash. Using the homomorphic properties of the group, the lock can be randomised at each hop making it difficult to link a payments traveling through the network just based on them having the same lock. This was formalised in [**?**]. Informal discussions on the lightning network development mailing list also suggest other benefits of using group elements, rather than hashes as the lock[**?**].

Discreet Log Contracts. In layer-2 protocols, "oracles" are parties who are trusted to cryptographically attest to the outcome of real world events. In his work, "Discreet Log Contracts"[**?**], Dryja proposes a model where rather than interacting with smart contracts directly, oracles publicly reveal secret information (usually a signature) depending on the outcome of the real world event. Two parties who wish to engage in

a bet can construct a set of jointly signed transactions where only one of them becomes valid determined by the signature the oracle reveals. Importantly, the oracle remains oblivious to the existence of the bet.

The protocol in the original work required three on-chain transactions to settle a bet. We observe that by using a one-time VES we can construct a more efficient two transaction protocol. The oracle announces it will reveal the decryption key corresponding to public encryption key A or B depending on the outcome of some event. Alice and Bob wish to bet 1 BTC on outcomes A and B respectively with even odds. To securely set up this contract, they both pay their 1 BTC into a single joint 2 BTC output and Alice gives a VES on a transaction paying Bob the 2 BTC encrypted by B and Bob gives a VES on a transaction paying Alice the 2 BTC encrypted by A. When the oracle releases the decryption key for one of A or B, the winner can decrypt their transaction signature and claim the 2 BTC with 1 BTC profit. Note that this protocol does not require the VES to be one-time.

Tumblers. Inspired by Chaumian eCash[**?**], the first tumbler protocol, TumbleBit[**?**] enabled Bitcoin payments through an untrusted intermediary called the *tumbler* to be mixed so it is hard (even for the tumbler) to link the incoming and outgoing payments. TumbleBit requires script in the un-cooperative case to verify that the tumbler correctly releases blinded RSA decryptions.

Tairi et al. recently proposed a more efficient scriptless tumbler called A^{2}L[**?**]. The tumbler first locks coins up and gives a VES on a transaction releasing the coins to the *receiver* encrypted under $A = g^{\alpha_i}$. The tumbler also gives the sender a homomorphic encryption of the decryption key α_i. The *sender* then attempts to pay the tumbler for α_i without letting the tumbler know which α she paid for (the tumbler is presumed to be executing many such protocols at once each with a different α). The sender purchases α_i by giving the tumbler a one-time VES on a payment transaction encrypted under a blinded version of the encryption key e.g. $g^{\alpha_i+\beta}$ where the sender knows β. When the tumbler takes the payment $\alpha_i + \beta$ is revealed to the sender allowing them to decrypt receiver's signature.

In addition, a scriptless tumbler protocol based on blind Schnorr signatures is proposed in [**?**].

Lotteries. The possibility of Bitcoin lotteries without a trusted party[**?**] was one of the first ideas academic Bitcoin researchers formally explored[**?**][**?**]. In its simplest form two parties both bet 1 Bitcoin and at the end of the protocol, the randomly chosen winner has 2 Bitcoin and the loser has 0. Most proposals, including more recent ones[**?**][**?**] are based on hash commitment coin tossing to produce the random outcome. The protocols use script to check the commitment openings on-chain.

A scriptless lottery was recently proposed[**?**] which uses the *oblivious signatures* from [**?**] rather than hash commitment coin tossing. The oblivious signature scheme is essentially built on a VES: The receiver first sends a Pedersen commitment $T = g^x h^c$ where $c \in \{0, 1\}$, to the signer. The singer then sends a VES for each of m_0 and m_1 encrypted under T and Th^{-1} respectively. Due to the binding property of the commitment, the receiver only knows the decryption key x for the signature on m_c but the sender never learns c. The proposed protocol uses the adaptor signature style one-time VES but in principle the idea works with any VES as long as there is an algorithm that allows the receiver to verifiably produce a set of public keys for which it only knows one of the private keys (without anyone else knowing which one).

Pay for commitment opening. A script based smart contract to pay for an opening hash commitment is straightforward to construct. A scriptless alternative for paying for the opening of a Pedersen commitment has been proposed in [**?**].

1.2 Our Contribution

In Section **??**, we contribute to theory of verifiably encrypted signatures in general. The original security definitions[**?**] only required the scheme to be secure if the encryption key-pair was chosen by the trusted adjudicator. VES schemes secure by these definitions are generally inappropriate for use in layer-2 protocols where a trusted party is rarely assumed. We propose new security definitions without reference to trusted parties and show that all existing schemes satisfy our new definitions.

In Section **??** we formally introduce one-time verifiably encrypted signatures. We then frame the existing Schnorr and ECDSA "adaptor signature" schemes as one-time VES schemes. We prove the Schnorr scheme secure but find that the ECDSA one-time VES unintentionally leaks a Diffie-Hellman tuple for the signing key and the encryption key. We incorporate this weakness and prove that this is at least the only problem with the scheme.

Finally in Section **??** we introduce the practical "semi-scriptless" paradigm where protocols can only have scripts with a single `OP_CHECKMULTISIG` script opcode. We show how to generically transform any scriptless protocol into a semi-scriptless protocol using the ECDSA one-time VES. This allows any scriptless protocol to be practically realised on Bitcoin as it is today without a complex two-party ECDSA multi-signature scheme.

1.3 Previous Work with Adaptor Signatures In Security Proofs

As far as we are aware, this work is the first attempt to formalise the adaptor signature as a primitive on its own. However, the works of [**?**] and [**?**] use the idea of the adaptor signature and formally argue their security in the *Universal Composability* (UC) model[**?**]. Their UC simulation ensures that corrupted parties may learn nothing but the signature itself from any adaptor signatures they receive. In general, the UC simulation is achieved through efficient *non-interactive zero knowledge proofs of knowledge* for discrete logarithms. This allows the simulator in the ideal world to extract the secret keys of the corrupted party and use them to synthetically create adaptor signatures to simulate the view of the adversary in the real world.

In our work, we use a weaker game-based model of security. This ensure that the corrupted party learns nothing *useful*, rather than nothing at all, from a signature encryption other than the signature itself. In particular, we will define security for VES and one-time VES schemes such that an adversary cannot learn anything from a signature encryption that will help them forge a signature on another message. We believe this is a viable approach in general, but it is especially reasonable in the context of Bitcoin style ledgers. In layer-2 Bitcoin protocols, the keys that own coins are not re-used between protocol executions so it is easy to ensure that a key is only used in the particular ways our game-based definitions can ensure is secure.

1.4 Future Work

Multi-party VES. All proposed scriptless protocols require signature encryptions generated through a two-party encrypted signing protocol. The security of Schnorr two-party one-time VES protocols deserves attention since it is being put forward as the main primitive for the protocols listed in Section **??** above in the long-term. We limit ourselves to focusing on defining formal security for single singer VES schemes and leave this analysis for future work.

Schnorr vs BLS. This work suggests a significant trade-off between the choice of Schnorr and BLS as the main signature scheme for Bitcoin-like systems. BLS admits an efficient VES scheme[**?**] and extremely attractive non-interactive signature aggregation [**?**]. On the other hand, Schnorr admits a less elegant interactive multi-signature scheme[**?**] but a very efficient one-time VES scheme. Additionally, it should be possible to create a Schnorr (not one-time) VES scheme with general zero-knowledge proof techniques where as a one-time VES for BLS looks much less likely. This trade off deserves further exploration including the possibility of a signature scheme that admits both an efficient VES and one-time VES scheme.

2 Preliminaries

2.1 Bitcoin

We assume a fair amount of familiarity with the Bitcoin transaction structure in Section **??**. A Bitcoin transaction has the following elements:

- A set of inputs, each of which refer to a previous output.
- For each input, a witness that satisfies the spending constraints specified in the output it references.
- For each input, a *relative* time-lock which prevents the transaction from being included in the ledger until enough time has passed since the output it references was included in the ledger.
- A set of new outputs, each with a value and spending constraint.
- An *absolute* time-lock, which prevents the transaction from being included in the ledger until a specific time.

Whenever a transaction is included in the ledger, its outputs are considered "spent" and their value is transferred to the new outputs created by the transaction (and a small fee to the miner who included them).

When we refer to a "layer-2" protocol we mean any protocol that is composed of messages sent between the participants and transactions sent to the ledger. A scriptless protocol means any protocol where the transactions only use a basic public key spending constraint which can only be satisfied by a signature on the spending transaction under that public key. It's important to note that a scriptless protocol can still use the time-lock constraints (and most do).

2.2 Notation

We analyse security asymptotically with our security parameter as k. We assume that the algorithms for each scheme have been generated by some $\mathsf{Setup}(1^k)$ algorithm which generates the a concrete instance of the scheme according to k. By $\mathsf{negl}(k)$ we denote any negligible function of k i.e. $\mathsf{negl}(k) < 1/p(k)$ for all positive polynomials $p(k)$ and all sufficiently large values of k. We say a probability is "negligible" if it can be expressed as $\mathsf{negl}(k)$ or overwhelming if it can be expressed as $1 - \mathsf{negl}(k)$. A polynomial time adversary runs in time that can be upper bounded by some polynomial of k. A *probabilistic* algorithm is also implicitly given a long string of random bits in addition to its other arguments. We denote invoking a probabilistic algorithm with $\leftarrow_{\$}$. If an algorithm is probabilistic and polynomial time we say it is a PPT algorithm.

2.3 The Discrete Logarithm Problem

The concrete signature schemes we deal with (Schnorr and ECDSA) are based on the discrete logarithm problem (DL). Written multiplicatively, the DL problem is to find $x \in \mathbb{Z}_q$ given (X, g) such that $g^x = X$, in a group $\mathbb{G}$ of prime order by q. We denote the fixed generator of a DL based scheme by g. In reality $(\mathbb{G}, q, g)$ are fixed by the ledger i.e. the Secp256k1 elliptic curve group for Bitcoin, but we reason about them as if the they were generated by the $\mathsf{Setup}(1^k)$ algorithm relative to k. Note that when we describe schemes based on the DL problem, all scalar operations are implicitly done modulo q.

3 Verifiably Encrypted Signatures Without a Trusted Third Party

Verifiably encrypted signatures (VES) were introduced by Boneh, Gentry, Lynn and Shacham (BGLS) [**?**] in 2003 for the BLS signature scheme[**?**]. A VES scheme lets a signer create a signature encryption that can be non-interactively verified. Just by knowing the message and public signing key, a verifier can tell that a ciphertext contains a valid signature encrypted by a particular encryption key. This separates it from the earlier notion of *signcryption*[**?**] where the message is also encrypted and the verification is often interactive.

This idea is somewhat unintuitive. If I can verify that the signer has indeed signed the message then what's the point of decrypting the ciphertext? A VES is only useful in settings where the signature itself is what has value rather than the fact that someone has signed it. In layer-2 protocols we have exactly this situation. Having a verifiably encrypted transaction signature is not enough to make the transaction valid — it must first be decrypted and attached to the transaction. Skipping ahead a bit, with our new definitions we do not even require that a valid encrypted signature was created by the signer for it to be a secure VES scheme.

The original definitions of BGLS were made relative to a trusted third party, the *adjudicator*. The original idea was that two parties could optimistically exchange signatures, by first exchanging verifiably encrypted signatures, then should one of them fail to provide their signature, the other could go to the adjudicator to have it decrypted. This is not appropriate for our setting. In most of the protocols described in Section **??**, one of the possibly malicious parties generates the encryption key-pair. We take our first step towards a trusted party free definition by removing references to the "adjudicator" from the definition of VES:

Definition 3.1 (Verifiably Encrypted Signature Scheme). A verifiably encrypted signature scheme (VES) $\hat{\Sigma}$ is defined with an ordinary *underlying* signature scheme $\Sigma := (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ and four additional algorithms:

- $\mathsf{EncGen} \xrightarrow{\$} (sk_E, pk_E)$: A probabilistic encryption key generation algorithm which outputs an encryption key pk_E and a decryption key sk_E. There should also exist an efficient predicate $\mathsf{valid}(sk_E, pk_E)$ that returns 1 when (sk_E, pk_E) is a valid key-pair.

- $\mathsf{EncSign}(sk_S, pk_E, m) \xrightarrow{\$} \hat{\sigma}$: A possibly probabilistic encrypted signing algorithm, which on input of a secret signing key sk_S, a public encryption key pk_E and a message m outputs a ciphertext $\hat{\sigma}$.
- $\mathsf{EncVrfy}(pk_S, pk_E, m, \hat{\sigma}) \rightarrow \{0, 1\}$: A deterministic encrypted signature verification algorithm which on input of a public signing key pk_S, a public encryption key pk_E, a message m and a ciphertext $\hat{\sigma}$ outputs 1 only if $\hat{\sigma}$ is a valid encryption of a signature on m for pk_S under pk_E.
- $\mathsf{DecSig}(sk_E, \hat{\sigma}) \rightarrow \sigma$: A (usually) deterministic signature decryption algorithm which on input of a decryption key sk_E and a valid ciphertext $\hat{\sigma}$ under that encryption key outputs a valid signature σ.

Any coherent VES should satisfy a basic notion of completeness such that for all messages m, valid encryption and signing key pairs (sk_E, pk_E) and (sk_S, pk_S) and for all coin tosses of $\mathsf{EncSign}$ the following always holds:

$$\hat{\sigma} = \mathsf{EncSign}(sk_S, pk_E, m) \implies \mathsf{EncVrfy}(pk_S, pk_E, m, \hat{\sigma}) = 1 \wedge \mathsf{Vrfy}(pk_S, m, \mathsf{DecSig}(sk_E, \hat{\sigma})) = 1$$

The original work BGLS proposed three security properties: *validity*, *unforgeability*, and *opacity*. To meet the requirements of our setting, we will restate validity without reference to a trusted party and replace unforgeability with a new *existential unforgeability under chosen message attack* ($\mathsf{EUF\text{-}CMA}[\mathsf{VES}]$) property. We keep the original definition of opacity. Therefore our VES requirements are informally as follows:

- **Validity:** It is infeasible to generate a valid looking ciphertext that does not yield a valid signature upon decryption.
- $\mathsf{EUF\text{-}CMA}[\mathsf{VES}]$: A VES ciphertext does not help an adversary forge signatures. In other words, an adversary cannot learn anything useful from a VES ciphertext other than the signature that is encrypted.
- **Opacity:** The encrypted signature cannot be extracted from the ciphertext without the decryption key.

We formally present our new definitions for validity and introduce $\mathsf{EUF\text{-}CMA}[\mathsf{VES}]$ after summarizing previous work on improving VES definitions. We do not formally describe opacity as the original definitions are not applicable the one-time VES (see Section **??**).

3.1 Previous Revisions of the VES Security Definitions

Rückert et al. [**?**] noticed that *key independent* schemes (which we call Sign then Encrypt (StE)) whose underlying signature schemes are $\mathsf{EUF\text{-}CMA}$ secure can be proved unforgeable generically. This is remarkable as the original BGLS scheme, which is StE, had AN involved proof for unforgeability that spanned several pages. We use a similar idea to prove any StE scheme satisfies $\mathsf{EUF\text{-}CMA}[\mathsf{VES}]$. Hanser et al. [**?**] also identified the importance of the $\mathsf{EUF\text{-}CMA}$ security of the underlying signature scheme. They proved that the underlying signature scheme is unforgeable if the VES scheme is unforgeable based on a different property.

Calderon et al. [**?**] discuss the original definitions in detail. They show a pathological VES scheme which is secure according to the original definitions but can be constructed only using a signature scheme i.e. without any encryption. They introduce two additional properties to address this. Our definitions allow for the same pathological constructions as we see no reason to prevent them in our setting.

Shao [**?**] addresses the assumption of trust in the adjudicator by providing a stronger definition of unforgeability which prevents forging a VES even if the adjudicator is corrupted. Unfortunately, the original and highly practical VES scheme of BGLS does not meet this stronger requirement. Our approach to removing trust in a third party is to simply replace the concept of unforgeability with $\mathsf{EUF\text{-}CMA}[\mathsf{VES}]$. The ability of malicious parties to forge signature encryptions is not a concern in our setting as long as they cannot forge signatures (which is already ensured by the $\mathsf{EUF\text{-}CMA}$ security of the signature scheme).

3.2 Validity

Validity protects against an adversary who attempts to create a valid looking ciphertext that when decrypted will not yield a valid signature. The original BGLS definition of validity was inadequate as it only guaranteed a valid signature upon decryption if the ciphertext was generated by the $\mathsf{EncSign}$ algorithm. Rückert et al.

[?] noticed this problem and introduced the additional property of *extractability* which ensured the adversary could not find a malicious ciphertext against the trusted adjudicator's encryption key.

This definition of extractability is not appropriate for our setting as we have no trusted party. In layer-2 protocols, the encryption key and the signing key may be generated by the same malicious party. For example, imagine an adversary who maliciously generates a VES ciphertext on some transaction signature and attempts to sell the decryption key to someone who wishes to know the signature. If they are successful they can get paid for the decryption key but even after obtaining the decryption key the buyer will be unable to get their desired signature. Thus in our setting the adversary must be free to choose the signing and encryption keys. We choose to use the original name of validity to capture this idea as it ensures that the validity of a ciphertext carries through to the validity of the encrypted signature.

Definition 3.2 (Validity). A VES scheme is valid if, for all PPT algorithms $\mathcal{A}$, the VES-Validity experiment outputs 1 except with negligible probability over coin tosses of $\mathcal{A}$ and Setup:

$\mathsf{VES\text{-}Validity}^{\mathcal{A}}_{\mathsf{Setup}}$

$\hat{\Sigma} \leftarrow\!\!\$\ \mathsf{Setup}(1^k)$

$(pk_S, (sk_E, pk_E), m, \hat{\sigma}) \leftarrow\!\!\$\ \mathcal{A}(\hat{\Sigma})$

$\sigma \leftarrow \hat{\Sigma}.\mathsf{DecSig}(sk_E, \hat{\sigma})$

return $\hat{\Sigma}.\mathsf{valid}(sk_E, pk_E) \implies \hat{\Sigma}.\mathsf{EncVrfy}(pk_S, pk_E, m, \hat{\sigma}) = \hat{\Sigma}.\mathsf{Vrfy}(pk_S, m, \sigma)$

3.3 EUF-CMA[VES]

Our goal in this section is to formally capture the requirement that from a VES ciphertext nothing can be learned except the signature encrypted within. We start with the typical EUF-CMA security definition for signature schemes which ensures that from a signature, nothing can be learned about a signature on any other message under the same signing key. The experiment tests this by giving the forger a signature oracle S from which it can query signatures under the signing key on messages of its choosing. For a secure scheme, no algorithm should exist that is able to forge signatures even with access to S. By implication, the forger learns nothing useful from the signatures other than the signatures themselves.

To show that a ciphertext equally offers no extra information to the forger, we modify the experiment so the forger also has access to a signature encryption oracle E. The forger is allowed to query this oracle under any encryption key and message it wants. This allows, for example, the forger to request a ciphertext where the encryption key is the signing key they are trying to forge against in the hope that will leak something about the signing key. We call denote this modified experiment EUF-CMA[VES] and define it as follows.

Definition 3.3 (EUF-CMA[VES]). A VES scheme $\hat{\Sigma}$ is *existentially unforgeable under a chosen message attack* (EUF-CMA[VES]) if its underlying EUF-CMA signature scheme remains unforgeable in a modified experiment where in addition to the signature oracle S the forger has access to a signature encryption oracle E.

$\mathsf{EUF\text{-}CMA[VES]}^{\mathcal{F}}_{\hat{\Sigma}}$

$Q := \emptyset$

$(sk_S, pk_S) \leftarrow\!\!\$\ \mathsf{Gen}$

$(m^*, \sigma) \leftarrow\!\!\$\ \mathcal{F}^{E,S}(pk_S)$

return $\mathsf{Vrfy}(pk_S, m^*, \sigma) \wedge m^* \notin Q$

Oracle $E(pk_E, m)$

$Q := Q \cup \{m\}$

return $\mathsf{EncSign}(sk_S, pk_E, m)$

Oracle $S(m)$

$Q := Q \cup \{m\}$

return $\mathsf{Sign}(sk_S, m)$

EUF-CMA[VES] effectively replaces the original definition of *unforgeability* so we now discuss and prove the relationship between the two. The original unforgeability property referred to signature encryptions being unforgeable under the trusted adjudicator's encryption key. EUF-CMA[VES] says nothing about the unforgeability of signature encryptions. In fact, an adversary who can produce valid VES ciphertexts without the secret signing key is perfectly compatible. Of course, they will never be able to forge a VES ciphertext under a particular encryption key. If they could do that, then they could trivially forge an encrypted signature under a key for which they know the decryption key and decrypt it. The original definitions seem to have missed this intuition: it is not the security of the VES scheme that prevents there being a successful

forger of signature encryptions under a particular key, the EUF-CMA security of the underlying signature scheme already ensures that no such algorithm can exist. After recalling the original BGLS definition of unforgeability, we use this intuition to prove that any EUF-CMA[VES] scheme is also unforgeable.

Definition 3.4 (VES Unforgeability [?]). A VES scheme $\hat{\Sigma}$ is unforgeable if for every PPT algorithm $\mathcal{F}$, with access to an encryption oracle $\tilde{E}$ and a decryption oracle $\tilde{D}$ the VES-Forge experiment outputs 1 with at most negligible probability over the coin tosses of EncGen, Gen, EncSign and $\mathcal{F}$.

$\mathsf{VES\text{-}Forge}^{\mathcal{F}}_{\hat{\Sigma}}$	Oracle $\tilde{E}(m)$	Oracle $\tilde{D}(\hat{\sigma})$
$Q \leftarrow \emptyset$	$Q := Q \cup \{m\}$	$\mathsf{EncVrfy}(pk_S, pk_E, \hat{\sigma}) \stackrel{?}{=} 1$
$(sk_S, pk_S) \leftarrow\!\!\$\ \mathsf{Gen}$	$\hat{\sigma} \leftarrow\!\!\$\ \mathsf{EncSign}(sk_S, pk_E, m)$	**return** $\mathsf{DecSig}(sk_E, \hat{\sigma})$
$(sk_E, pk_E) \leftarrow\!\!\$\ \mathsf{EncGen}$	**return** $\hat{\sigma}$	
$(m^*, \hat{\sigma}^*) \leftarrow\!\!\$\ \mathcal{F}^{\tilde{E},\tilde{D}}(pk_S, pk_E)$		
return $\mathsf{EncVrfy}(pk_S, pk_E, m^*, \hat{\sigma}^*) \wedge m^* \notin Q$		

Note that the oracles in VES-Forge only provide encryptions and decryption on a particular key, representing the trusted adjudicator's key-pair.

Theorem 3.1 (EUF-CMA[VES] $\implies$ Unforgeability). *If a VES scheme is EUF-CMA[VES] secure and has the* validity *property then it also has the original BGLS unforgeability property of Definition* **??**.

Proof. Let $\hat{\Sigma}$ be a VES scheme and let $\mathcal{F}$ be PPT a forger with non-negligible advantage ϵ in forging signature encryptions in the original $\mathsf{VES\text{-}Forge}^{\mathcal{F}}_{\hat{\Sigma}}$ experiment. We can construct a reduction $\mathcal{R}$ with non-negligible advantage ϵ' in forging signatures in the $\mathsf{EUF\text{-}CMA[VES]}^{\mathcal{R}}_{\hat{\Sigma}}$ experiment just given black-box access to $\mathcal{F}$ as follows.

$\mathcal{R}^{S,E}_{\mathsf{EUF\text{-}CMA[VES]}}(pk_S)$	Simulate $\tilde{E}(m)$	Simulate $\tilde{D}(\hat{\sigma})$
$(sk_E, pk_E) \leftarrow\!\!\$\ \mathsf{EncGen}$	**return** $E(pk_E, m)$	$\mathsf{EncVrfy}(pk_S, pk_E, \hat{\sigma}) \stackrel{?}{=} 1$
$(m^*, \hat{\sigma}^*) \leftarrow\!\!\$\ \mathcal{F}^{\tilde{E},\tilde{D}}_{\mathsf{VES\text{-}Forge}}(pk_S, pk_E)$		**return** $\mathsf{DecSig}(sk_E, \hat{\sigma})$
$\sigma^* \leftarrow \mathsf{DecSig}(sk_E, \hat{\sigma}^*)$		
return (m^*, σ^*)		

Clearly $\mathcal{R}$ perfectly simulates $\tilde{E}$ and $\tilde{D}$ and runs in polynomial time. Denote $\mathsf{EncVrfy}(pk_S, pk_E, m, \hat{\sigma})$ by EV, $\mathsf{Vrfy}(pk_S, m, \sigma)$ by V and start with the following inequality:

$$\Pr[V] \geq \Pr[V \mid EV] \Pr[EV]$$

Since validity ensures that $\Pr[V \mid EV] = 1 - \mathsf{negl}(k)$, we have

$$\Pr[V] \geq (1 - \mathsf{negl}(k)) \Pr[EV] = \Pr[EV] - \mathsf{negl}(k)$$

By definition, $\epsilon = \Pr[EV]$ and $\epsilon' = \Pr[V]$

$$\epsilon' \geq \epsilon - \mathsf{negl}(k)$$

Which shows ϵ' is non-negligible if ϵ is non-negligible, completing the proof. □

As we have already argued, EUF-CMA[VES] captures our requirement that the scheme be secure without reference to a trusted party. We now propose using EUF-CMA[VES] as the standard VES security definition even in the trusted party setting since it implies BGLS unforgeability and it is much easier to prove for the schemes that have been developed so far. We can prove all existing schemes we are aware of [**?**, **?**, **?**, **?**, **?**] EUF-CMA[VES] secure just by the *Sign then Encrypt* (StE) structure they all share. That is, internally the EncSign algorithm first generates a normal signature using Sign and then encrypts the result. Formally:

Definition 3.5 (Sign then Encrypt VES). A *Sign then Encrypt* (StE) VES scheme is defined with an ordinary *underlying* signature scheme $\Sigma := (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ an *associated* public key encryption scheme $\Pi = (\mathsf{EncGen}, \mathsf{Enc}, \mathsf{Dec})$ and a VES verification algorithm $\mathsf{EncVrfy}$ such that if we define $\mathsf{DecSig} := \mathsf{Dec}$ and $\mathsf{EncSign}(sk_S, pk_E, m) := \mathsf{Enc}(pk_E, \mathsf{Sign}(pk_S, m))$ then $\hat{\Sigma} := (\mathsf{EncGen}, \mathsf{EncSign}, \mathsf{EncVrfy}, \mathsf{DecSig})$ is a VES scheme according to Definition **??**.

It is easy to see that any valid StE scheme must be EUF-CMA[VES] secure. The extra encryption oracle E the forger has access to in the EUF-CMA[VES] experiment can be simulated just by encrypting the result of a query to the signature oracle S that the EUF-CMA experiment provides.

Theorem 3.2 (StE $\implies$ EUF-CMA[VES]). *Let $\hat{\Sigma}$ be a StE VES scheme, Σ be its underlying signature scheme and $\Pi = (\mathsf{EncGen}, \mathsf{Enc}, \mathsf{Dec})$ be its associated public key encryption scheme. If there is reduction $\mathcal{R}$ from some hard problem $\mathcal{H}$ to the EUF-CMA security of Σ, then there is a reduction $\hat{\mathcal{R}}$ from $\mathcal{H}$ to the EUF-CMA[VES] security of $\hat{\Sigma}$.*

Proof. The only difference between the EUF-CMA and EUF-CMA[VES] experiments is that the forger is provided an extra signature encryption oracle E. Since $\hat{\Sigma}$ is StE, we can easily construct $\hat{\mathcal{R}}$ by having it run $\mathcal{R}$ and forwarding queries to E to the signing oracle S of $\mathcal{R}$ and encrypting the result. Specifically, when $\hat{\mathcal{R}}$ receives an E oracle query (m, pk_E) it makes a query (m) to S and receives σ. It then sets $\hat{\sigma} \leftarrow \mathsf{Enc}(pk_E, m)$ and returns $\hat{\sigma}$ to the forger. This does not interfere with $\mathcal{R}$ at all so $\hat{\mathcal{R}}$ is also a reduction from $\mathcal{H}$. □

4 One-Time Verifiably Encrypted Signatures

We introduce the *one-time verifiably encrypted signature scheme* to abstract the functionality of the "adaptor signature"[**?**]. A one-time VES is a VES with an additional property: given the ciphertext and the decrypted signature, the decryption key is easily recoverable. Obviously, this property makes it useless for the optimistic fair exchange of signatures with a trusted party envisioned for ordinary VES schemes, as the trusted adjudicator would leak their decryption key after its first use. Despite this, the property turns out to be incredibly useful to layer-2 protocol designers as it allows them to force the release of a secret key whenever a party broadcasts a decrypted transaction signature. We define by extending the original VES definition:

Definition 4.1 (One-Time Verifiably Encrypted Signatures). A One-Time Verifiably Encrypted Signature scheme (one-time VES) is a Verifiably Encrypted Signature scheme (Gen, Sign, Vrfy, EncGen, $\mathsf{EncSign}$, $\mathsf{EncVrfy}$, DecSig) with the following additional algorithms:

- $\mathsf{RecKey}(pk_E, \hat{\sigma}) \to \delta$: A deterministic recovery key extraction algorithm which extracts a recovery key δ from the ciphertext $\hat{\sigma}$ and the public encryption key pk_E.
- $\mathsf{Rec}(\sigma, \delta) \to sk_E$: A deterministic decryption key recovery algorithm which when given a decrypted signature σ and the recovery key δ associated with the original ciphertext, returns the secret decryption key sk_E.

Technically, Rec takes a "recovery key" δ rather than the ciphertext $\hat{\sigma}$ but they should usually be thought of as equivalent. The distinction only really becomes necessary in Section **??** when we construct a simple two party encrypted signing protocol where one party is not able to output the whole ciphertext but is able to output the recovery key.

To be considered secure, we require a one-time VES scheme to have the *completeness*, *validity* and EUF-CMA[VES] properties of an ordinary VES along with *recoverability* which we define as follows.

Definition 4.2 (Recoverability). A one-time VES scheme is recoverable if for all PPT algorithms $\mathcal{A}$ the VES-Recover experiment outputs 1 except with negligible probability over the coin tosses of $\mathcal{A}$ and Setup:

$\mathsf{VES\text{-}Recover}^{\mathcal{A}}_{\mathsf{Setup}}$

$\hat{\Sigma} \leftarrow\!\!\$\ \mathsf{Setup}(1^k)$

$(pk_S, (sk_E, pk_E), m, \hat{\sigma}) \leftarrow\!\!\$\ \mathcal{A}(\hat{\Sigma})$

$\delta \leftarrow \hat{\Sigma}.\mathsf{RecKey}(pk_E, \hat{\sigma}); \sigma \leftarrow \hat{\Sigma}.\mathsf{DecSig}(sk_E, \hat{\sigma})$

return $(\hat{\Sigma}.\mathsf{valid}(sk_E, pk_E) \wedge \hat{\Sigma}.\mathsf{EncVrfy}(pk_S, pk_E, m, \hat{\sigma}) \wedge \hat{\Sigma}.\mathsf{Vrfy}(pk_S, m, \sigma)) \implies \hat{\Sigma}.\mathsf{Rec}(\sigma, \delta) = sk_E$

Note that we no longer have need of the *opacity* property as defined for an ordinary VES scheme. Recall that the informal purpose of opacity is to ensure that an encrypted signature cannot be accessed without the decryption key. Recoverability makes this is trivial because if you are able to access the signature you can recover the decryption key anyway. In other words, obtaining a signature from a ciphertext without the decryption key is no easier than obtaining the decryption key just given the public encryption key (which is hopefully hard). Crucially, this also means that even the signer themselves cannot produce the signature from the ciphertext without the decryption key. This necessarily implies that one-time VES schemes only exist for probabilistic signature schemes where there is an exponential number of possible signatures under a public key for any message.

4.1 Schnorr One-Time VES Scheme

What we call the Schnorr one-time VES was introduced by Poelstra [**?**] which he termed an "adaptor signature"[**?**] with the encryption key being termed an "auxiliary point"[**?**] or sometimes just "T". A major benefit of our one-time VES concept is to be able to explain this useful idea with intuitive encryption/decryption semantics. It is typically described with a two-party signing protocol as this is where it is most useful. We describe the single singer scheme in Figure **??** including its underlying Schnorr signature scheme. The Schnorr signature scheme was introduced by its namesake in [**?**] and our description resembles the key-prefixed scheme described in the Schnorr Bitcoin Improvement Proposal[**?**] currently under consideration.

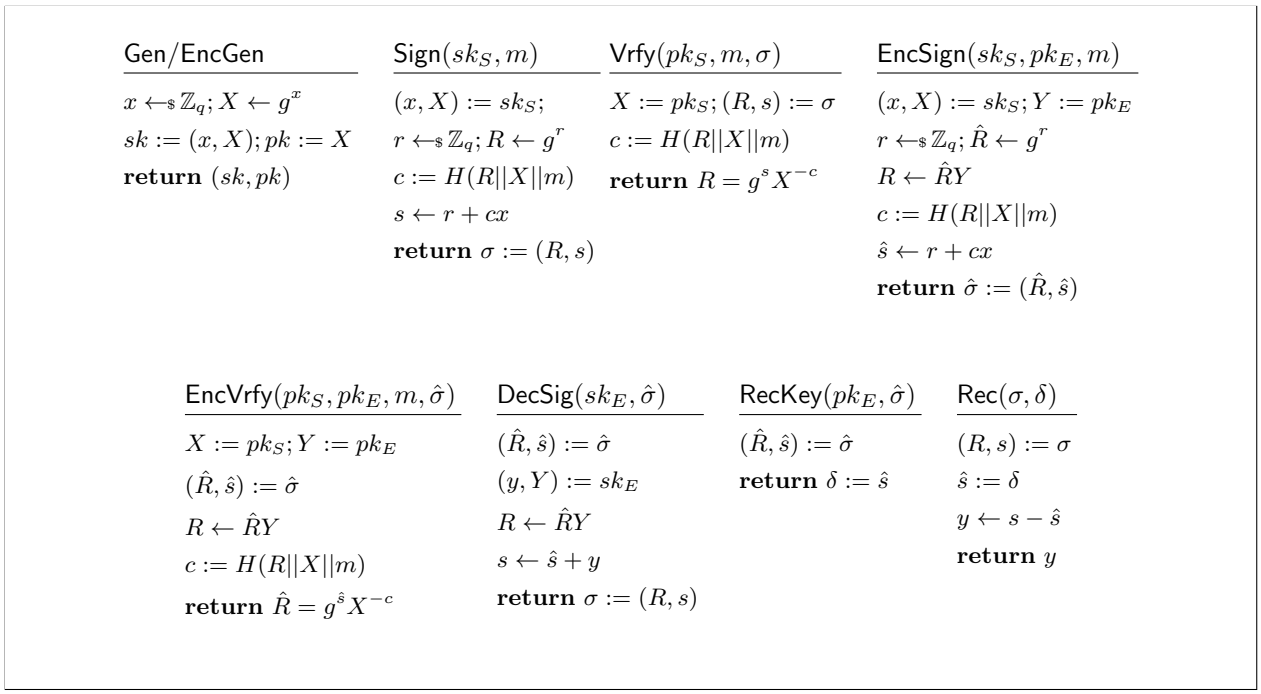

Fig. 1. The algorithms of the Schnorr one-time VES scheme with a hash algorithm H

Since the EncSign algorithm is a simple tweak of the sign algorithm it is easy to get an intuition for the security of the scheme. Unsurprisingly, we find that the scheme is unconditionally valid and recoverable and is EUF-CMA[VES] secure. We now formally prove these properties.

Lemma 4.1. *The Schnorr one-time VES is unconditionally valid and recoverable.*

Proof. Since $\mathsf{EncVrfy}(X, Y, m, (\hat{R}, \hat{s})) = 1$ implies $\hat{R} = g^{\hat{s}} X^{-c}$ where $c := H(\hat{R}Y||X||m)$, by the one-way homomorphism between $\mathbb{Z}_q$ and $\mathbb{G}$ we know that $\hat{\sigma}$ encrypts some valid signature $(R, s) := (\hat{R}Y, \hat{s} + y)$ where $g^y = Y$ and is therefore valid because:

$$\hat{R}Y = Y g^{\hat{s}} X^{-c} = g^{\hat{s}+y} X^{-c}$$

By the same token, the scheme is recoverable because $\mathsf{Rec}((\hat{R}Y, \hat{s}+y)), \hat{s})$ always returns the decryption key $y = s - \hat{s}$. □

Theorem 4.1. *The Schnorr one-time VES is EUF-CMA[VES] secure.*

Proof. We transform the standard Schnorr reduction from the DL problem to the EUF-CMA security of Schnorr signatures in the random oracle model by Pointcheval and Stern[**?**] into a EUF-CMA[VES] reduction by simulating the additional signature encryption oracle E. Note the original reduction was not did not key-prefixed but the same reduction can be applied to key-prefixed Schnorr. Our reduction $\mathcal{R}_{\mathsf{DL}}$ simulates E by programming the random oracle H in a similar way to how it usually simulates the signing oracle S as shown below:

$\mathcal{R}_{\mathsf{DL}}(X)$	Simulate $S(m)$	Simulate $E(pk_E, m)$
// Run the reduction from [**?**]	$s, c \leftarrow\!\!\$\ \mathbb{Z}_q$	$Y := pk_E$
$\ldots \leftarrow \mathcal{F}^{E,S}_{\mathsf{EUF\text{-}CMA[VES]}}(X)$	$R \leftarrow g^s X^{-c}$	$\hat{s}, c \leftarrow\!\!\$\ \mathbb{Z}_q$
	if $H(R\|\|X\|\|m) = \perp$ **then**	$\hat{R} \leftarrow g^{\hat{s}} X^{-c}; R \leftarrow \hat{R}Y$
	$\quad H(R\|\|X\|\|m) := c$	**if** $H(R\|\|X\|\|m) = \perp$ **then**
	$\quad$ **return** (R, s)	$\quad H(R\|\|X\|\|m) := c$
	else abort	$\quad$ **return** $(\hat{R}, \hat{s})$
		else abort

The fact that we can efficiently simulate E without changing the internals of $\mathcal{R}_{\mathsf{DL}}$ completes the proof. □

4.2 ECDSA One-Time VES Scheme

In order circumvent the need for Schnorr signatures, which are not included in the Bitcoin protocol today, Moreno-Sanchez et al. developed an adaptor signature construction for the standard ECDSA signature scheme already used in Bitcoin[**?**]. It was later applied to achieve a payment channel construction with better privacy in [**?**] and an efficient tumbler in [**?**]. The scheme is built upon the two-party ECDSA protocol from [**?**]. We translate it into a single signer scheme in Figure **??**.

As the protocol is single signer it avoids all the complexities that come with two-party ECDSA protocols. It only requires requires a simple non-interactive zero knowledge proof of discrete logarithm equality, whose proving and verification algorithms we denote as $(\mathsf{P}_{\mathsf{DLEQ}}, \mathsf{V}_{\mathsf{DLEQ}})$. When invoked as $\mathsf{P}_{\mathsf{DLEQ}}((g, A), (h, B), w)$, it generates a proof of membership of the language:

$$L_{\mathsf{DLEQ}} = \{(g, h, A, B) \in \mathbb{G}^4 \mid \exists w \in \mathbb{Z}_q : A = g^w \wedge B = h^w\}$$

We instantiate this proof with the Fiat-Shamir transform applied to the Sigma protocol for the relation originally from [**?**].

We now attempt to prove the scheme secure.

Lemma 4.2. *The ECDSA one-time VES is valid and unconditionally recoverable.*

Proof. If $\mathsf{EncVrfy}(X, Y, m, (R, \hat{R}, \hat{s}, \pi)) = 1$ then $\hat{R} = (g^{H(m)} X^{R_{\mathsf{x}}})^{\hat{s}^{-1}}$ and $\hat{R}^y = R$ if π is sound. Which means $\hat{R}^y = R = (g^{H(m)} X^{R_{\mathsf{x}}})^{\hat{s}^{-1} y}$. Therefore DecSig produces a valid signature $\sigma := (R, \hat{s}y^{-1})$ whenever $\hat{\sigma}$ is valid except with the negligible probability that π is unsound. If σ is valid then Rec will always recover $\tilde{y} \leftarrow s^{-1}\hat{s} = (\hat{s}y^{-1})^{-1}\hat{s}$, which is either equal to y or $-y$ (due to Vrfy only checking the x-coordinate of R). □

Unfortunately, while trying to prove the scheme EUF-CMA[VES] secure we found a problem: the ciphertext surreptitiously leaks the Diffie-Hellman key for the signing and encryption key. $Y^x = X^y$ can be computed as $(R^{\hat{s}} Y^{-H(m)})^{R_{\mathsf{x}}^{-1}}$. This clearly violates the premise of EUF-CMA[VES] which is that nothing useful should be learned from the ciphertext, other than the signature (if it can be decrypted). Furthermore, it is not difficult to imagine a composability attack where an adversary, with an ElGamal encryption $(g^r, X^r m)$ of some message $m \in \mathbb{G}$ is able to decrypt it by requesting a signature encryption under g^r from the signer, learning X^r and thereby illegally learning m. We discuss the impact of this further after proving that the scheme cannot be EUF-CMA[VES] secure.

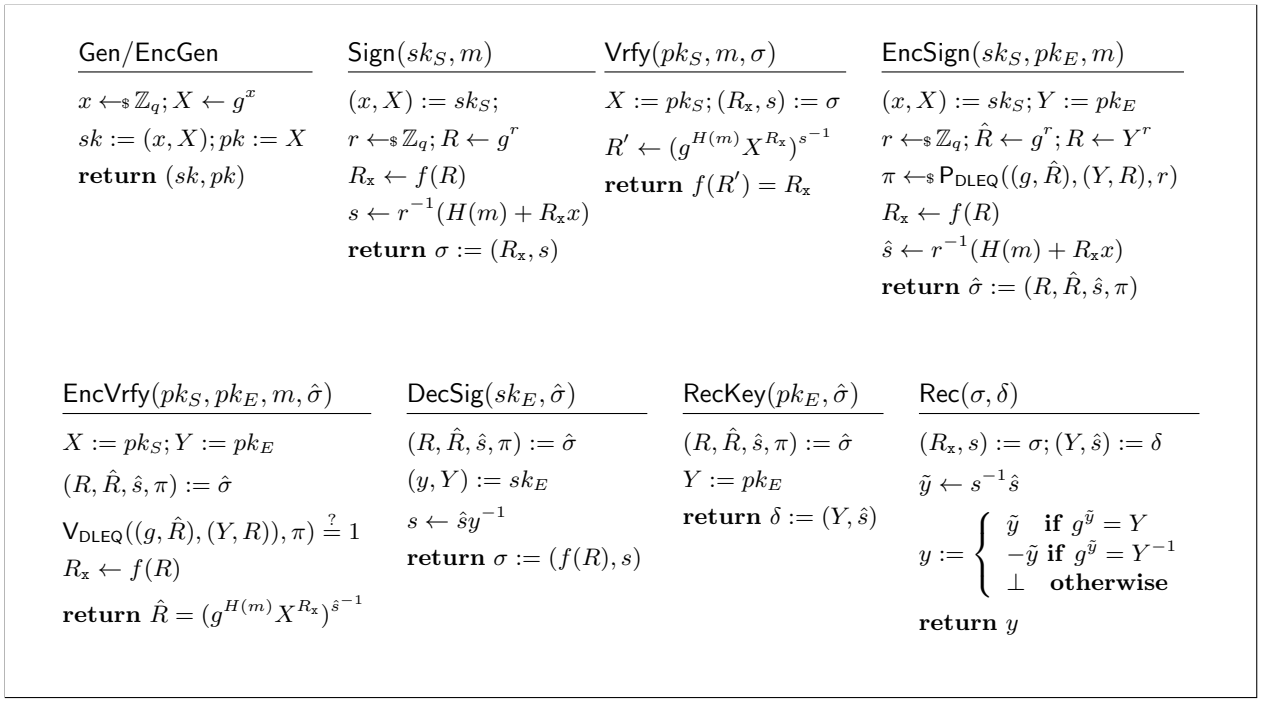

Fig. 2. The algorithms of the ECDSA one-time VES scheme. $f : \mathbb{G} \to \mathbb{Z}_q$ converts a an elliptic curve point to its x-coordinate mod q.

Lemma 4.3. *There exists no* key-preserving *EUF-CMA[VES] reduction for the ECDSA one-time VES scheme if the Computational Diffie-Hellman Problem (CDH) is hard.*

Proof. Let $\mathsf{CDH}(X, Y)$ for $(X, Y) \leftarrow_\$ \mathbb{G}^2$ denote an instance of the CDH problem with respect to the generator g of the ECDSA one-time VES. We will prove our statement by solving $\mathsf{CDH}(X, Y)$ with only black-box access to a simulator $\mathcal{S}$ for the encryption oracle E that must be provided to the forger in a key-preserving EUF-CMA[VES] reduction. This contradicts the existence of $\mathcal{S}$ and therefore no key-preserving EUF-CMA[VES] reduction scheme can exist if the CDH problem is hard. To solve $\mathsf{CDH}(X, Y)$ we run the simulator for $\mathcal{S}$ with X as the signing key and then query it with an arbitrary message m: $(R, \hat{R}, \hat{s}, \pi) \leftarrow_\$ \mathcal{S}(m, Y)$.

First, note that $(\hat{R}, Y, R)$ must be a valid Diffie-Hellman tuple if $\mathcal{S}$ properly simulates the an encrypted signing oracle. Even though $\mathcal{S}$ can simulate DLEQ proofs (by their zero knowledge property) for invalid tuples, this is not enough to simulate the view of a real forger. Regardless of validity of π, a forger can make a query to $E(Y', m')$ with an encryption key-pair it has generated itself (y', Y') and check that $\hat{R}^{y'} = R$ in the response. If the check fails with non-negligible probability then $\mathcal{S}$ can be distinguished from the real oracle E in the EUF-CMA[VES] experiment.

Given $(\hat{R}, Y, R)$ is a valid Diffie-Hellman tuple and $\hat{s} = r^{-1}(H(m) + R_\mathrm{x}x)$, $R = Y^r$ and $X = g^x$ are true since $(R, \hat{R}, \hat{s}, \pi)$ must be a valid signature encryption, we can compute $\mathsf{CDH}(X, Y)$ as $(R^{\hat{s}} Y^{-H(m)})^{R_\mathrm{x}^{-1}}$ since:

$$\begin{aligned} R^{\hat{s}} &= Y^{H(m)+R_\mathrm{x}x} \\ R^{\hat{s}} Y^{-H(m)} &= Y^{R_\mathrm{x}x} \\ (R^{\hat{s}} Y^{-H(m)})^{R_\mathrm{x}^{-1}} &= Y^x \end{aligned}$$

□

The fact that this scheme leaks a Diffie-Hellman key does not necessarily mean it is useless. None of the schemes presented in Section **??** rely on the CDH problem being hard. Furthermore, on a Bitcoin like ledger, signing keys for transactions are usually generated randomly within the protocol and are not shared between executions, so learning a CDH tuple in one execution cannot help an adversary break the security of another.

Thus, as long as the protocol designer does not rely on the CDH problem being hard within their protocol the ECDSA one-time VES is practically secure.

It remains to be shown that a CDH solution is the only extra information the adversary learns from the ciphertext. To prove this we make the CDH problem easy by giving the reduction access to a CDH oracle and show the scheme secure under this condition. We choose the EUF-CMA reduction to the DL problem for ECDSA by Fersch et al.[**?**] in the *bijective random oracle model* as our starting point. In this model, the conversation function f which converts an elliptic curve point R to its x-coordinate $R_{\mathbf{x}}$ is modelled as a bijective random oracle and signatures are simulated by programming it. We prove the following theorem in Appendix **??**.

Theorem 4.2. *The reduction from the discrete logarithm problem to the EUFCMA security of ECDSA in the bijective random oracle model from [**?**] can be transformed into a EUF-CMA[VES] reduction for the ECDSA one-time VES if the CDH problem is easy.*

5 Semi-Scriptless Protocols

The essential function of a smart contract on a Bitcoin-like ledger is to lock coins in an output such that they can only be spent to certain parties under certain conditions. With a smart contract language like Bitcoin script, the conditions can be expressed in the language and enforced by the ledger's transaction validation rules. In the scriptless model, we can only constrain spending through time-locks and by setting a public key, for which the spender must provide a signature. Clearly in order to stop one party from arbitrarily spending the coins the corresponding private key cannot be exclusively known to one party. Thus we require the parties have a joint ownership of the public key and cooperatively use a multi-signature protocol to sign transactions spending from the joint output. Multi-signature protocols exist for ECDSA[**?**, **?**], Schnorr[**?**], BLS[**?**] and most prominent signature schemes.

To realise most of the scriptless protocols from Section **??**, the multi-signature scheme also needs to admit a two-party one-time VES encrypted signing protocol to emulate EncSign on a joint signing key. It is relatively simple to build this on top of a Schnorr multi-signature scheme, but since Schnorr signatures have not been included in the Bitcoin protocol yet, this tool is out of reach for now. Both the two-party ECDSA schemes in [**?**, **?**] admit a two-party EncSign protocol as described in [**?**]. Unfortunately, these schemes are complex and rely additional exotic computational hardness assumptions. As a result, the consensus that came out of the 2018 Lightning Developer Summit was to postpone updating the lightning specification to include "payment points" until Schnorr becomes viable.[**?**]

We present a workaround that allows protocol designers to realise many of the benefits of scriptless protocols in Bitcoin as it is today. To do so, we relax the scriptless model slightly to what we call the "semi-scriptless" model where protocols are allowed to use a single `OP_CHECKMULTISIG` script opcode but no others. `OP_CHECKMULTISIG` acts as a naive ECDSA multi-signature scheme, where the public key for the scheme is the concatenation of each party's public keys and a valid signature is the concatenation of valid signatures under each public key. When locking coins with `OP_CHECKMULTISIG`, a set of public keys is specified along with how many of those keys must authorize any transaction spending from it. We will only use "2-of-2" outputs which require two signatures on two out of two of the specified public keys.

We can transform any existing scriptless protocol into a semi-scriptless protocol by locking funds to an `OP_CHECKMULTISIG` 2-of-2 on two distinct public keys and using the single signer ECDSA one-time VES from Section **??**. The simple two-party signing and encrypted signing protocols are presented in Figure **??**.

The downsides of semi-scriptless protocols are readily apparent. First, the transactions are larger because they require two public keys and two signatures to spend them (in addition the overhead that comes from using script). Secondly, it is easy to distinguish `OP_CHECKMULTISIG` outputs from a regular payment transactions (but not from other uses of `OP_CHECKMULTISIG` 2-of-2).

Having said this, semi-scriptless protocols are a practical alternative to two-party ECDSA to developers who wish to attempt to realise many of the benefits of scriptless protocols prior to the Schnorr upgrade. In general, semi-scriptless enjoy better confidentiality than their script based counterparts. Although script is used, `OP_CHECKMULTISIG` is not particular to any protocol making it at more confidential than protocols with a unique script structure. For the following protocols we note the following particular benefits:

- **Payment Channels [?]:** The typical hash lock can be replaced with a discrete logarithm based lock which enables the privacy benefits from [**?**] other conjectured improvements[**?**].

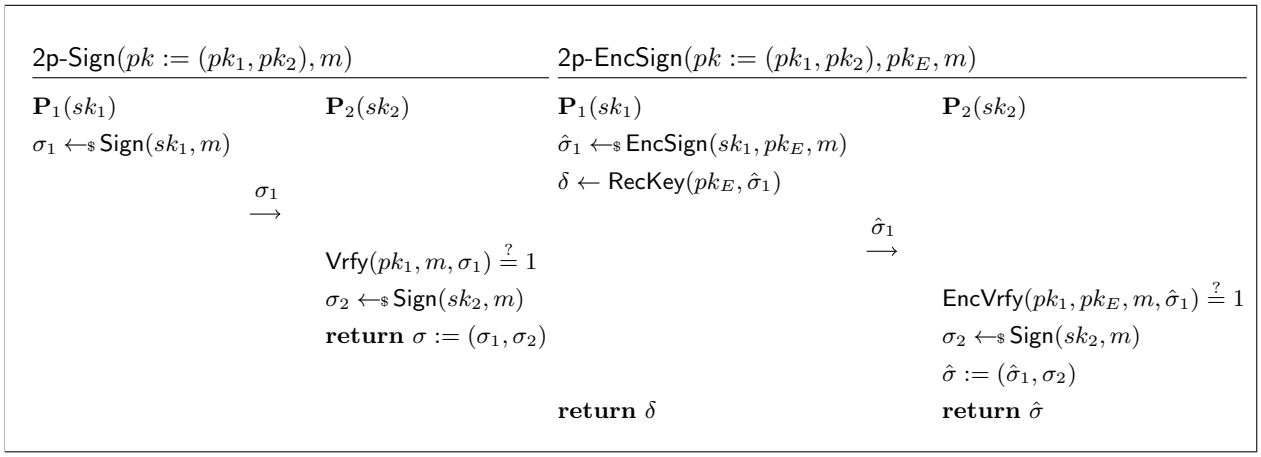

Fig. 3. The two-party signing and encrypted signing algorithms for an 2-of-2 `OP_CHECKMULTISIG` output.

- **Atomic swaps [?]:** The secret that releases funds from the escrow transactions never appears on the ledger, unlike the existing hash constructions which make it easy to associate assets changing hands as the contracts on the ledgers share the same hash.
- **Discreet Log Contracts [?]:** The protocol can be completed in two transactions rather than three.

A Proof for Theorem ??

As we have proved, the ECDSA one-time VES is not $\mathsf{EUF\text{-}CMA}[\mathsf{VES}]$ secure if the CDH problem is hard. We now wish to show that if the CDH problem were easy then it would satisfy $\mathsf{EUF\text{-}CMA}[\mathsf{VES}]$ i.e. leak no useful information to a forger. Thus we give our reduction access to an $\mathcal{O}_{\mathsf{CDH}}$ oracle which when queried with $\mathcal{O}_{\mathsf{CDH}}(X, Y)$ returns Z such that (X,Y,Z) is a Diffie-Hellman tuple with respect to g. Additionally, in the reduction we simulate the NIZK DLEQ proof π with $\mathsf{S}_{\mathsf{DLEQ}}$.

As our starting point, we take the $\mathsf{EUF\text{-}CMA}$ reduction for ECDSA by Fersch et al.[?] to the discrete logarithm problem. In this work, the conversion function $f : \mathbb{G} \to \mathbb{Z}_q$ that maps a group element to its x-coordinate mod q is decomposed and idealised as an oracle. The forger must query this oracle to produce a valid forgery. During the reduction, this oracle is programmable. In more detail, f is decomposed as $f = \psi \circ \Pi \circ \varphi$ where:

1. φ is an invertible 2-to-1 function mapping curve points to the domain of Π reflecting the fact that every x-coordinate belongs to two possible group elements.
2. Π is the bijective random oracle that the reduction is able to program.
3. ψ is an invertible function that maps the range of Π (denoted as $\mathbb{B}$) to $\mathbb{Z}_q$.

We sketch the simulation of the signature encryption oracle E in this model below. Note that we leave out and simplify some important details for the sake of clarity and encourage the reader to review the original proof in [?] to get a full understanding of the reduction.

$\mathcal{R}_{\mathsf{DL}}(X)$	Simulate $S(m)$	Simulate $E(Y, m)$
$Q \leftarrow \emptyset; \Pi \leftarrow \emptyset$	$\beta \leftarrow\!\!\$\, \mathbb{B}$	$\beta \leftarrow\!\!\$\, \mathbb{B}$
// Run the reduction from [?]	**if** $(\cdot, \beta) \in \Pi$: **abort**	**if** $(\cdot, \beta) \in \Pi$: **abort**
$\ldots \leftarrow \mathcal{F}^{E,S}_{\mathsf{EUF\text{-}CMA[VES]}}(X)$	$R_{\mathtt{x}} \leftarrow \psi(\beta)$	$R_{\mathtt{x}} \leftarrow \psi(\beta)$
	$s \leftarrow\!\!\$\, \mathbb{Z}_q$	$\hat{s} \leftarrow\!\!\$\, \mathbb{Z}_q$
	$R \leftarrow (g^{H(m)} X^{R_{\mathtt{x}}})^{s^{-1}}$	$\hat{R} \leftarrow (g^{H(m)} X^{R_{\mathtt{x}}})^{\hat{s}^{-1}}$
	$\alpha \leftarrow \varphi(R)$	$R \leftarrow \mathcal{O}_{\mathsf{CDH}}(\hat{R}, Y)$
	if $(\alpha, \cdot) \in \Pi$: **abort**	$\pi \leftarrow \mathsf{S}_{\mathsf{DLEQ}}((g, \hat{R}), (Y, R))$
	$\Pi \leftarrow \Pi \cup \{(\alpha, \beta)\}$	$\alpha \leftarrow \varphi(R)$
	$Q \leftarrow Q \cup \{m\}$	**if** $(\alpha, \cdot) \in \Pi$: **abort**
	return $(s, R_{\mathtt{x}})$	$\Pi \leftarrow \Pi \cup \{(\alpha, \beta)\}$
		$Q \leftarrow Q \cup \{m\}$
		return $(R, \hat{R}, \hat{s}, \pi)$

The fact that we can simulate E with access to $\mathcal{O}_{\mathsf{CDH}}$ without modifying the internals of $\mathcal{R}_{\mathsf{DL}}$ completes the proof.