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[1]. 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[2] 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[3] 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 1.1).

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*[4] (VES) introduced for the BLS signature scheme[5]. 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[4]: 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 6 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[6], 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[7] 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[3]. 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 [8]. Informal discussions on the lightning network development mailing list also suggest other benefits of using group elements, rather than hashes as the lock[9].

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"[10], 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[11], the first tumbler protocol, TumbleBit[12] 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[13]. 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 [14].

Lotteries. The possibility of Bitcoin lotteries without a trusted party[15] was one of the first ideas academic Bitcoin researchers formally explored[16][17]. 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[18][19] 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[20] which uses the *oblivious signatures* from [21] 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 [22].

1.2 Our Contribution

In Section 2, we contribute to theory of verifiably encrypted signatures in general. The original security definitions[4] 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 3 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 6 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 [8] and [13] use the idea of the adaptor signature and formally argue their security in the *Universal Composability* (UC) model[23]. 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 1.1 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[4] and extremely attractive non-interactive signature aggregation [24]. On the other hand, Schnorr admits a less elegant interactive multi-signature scheme[25] 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 Verifiably Encrypted Signatures Without a Trusted Third Party

Verifiably encrypted signatures (VES) were introduced by Boneh, Gentry, Lynn and Shacham (BGLS) [4] in 2003 for the BLS signature scheme[5]. 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*[26] 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 1.1, 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 2.1 (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:

- $\mathsf{EncGen} \overset{\$}{\rightarrow} (sk_E, pk_E)$: 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.

- $\mathsf{EncSign}(sk_S, pk_E, m) \overset{\$}{\rightarrow} \hat{\sigma}$: 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* (EUF-CMA[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.

- EUF-CMA[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 EUF-CMA[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).

2.1 Previous Revisions of the VES Security Definitions

Rückert et al. [27] noticed that *key independent* schemes (which we call Sign then Encrypt (StE)) whose underlying signature schemes are EUF-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 EUF-CMA[VES] . Hanser et al. [28] also identified the importance of the EUF-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. [29] 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. For simplicity, our definitions exclude the pathological constructions but could be easily modified to account for them.

Shao [30] 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 $\mathsf{EUF\text{-}CMA[VES]}$. 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).

2.2 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 they only guaranteed a valid signature upon decryption if the ciphertext was generated by the $\mathsf{EncSign}$ algorithm. Rückert et al. [27] 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. Therefore, 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 decrypted signature.

Definition 2.2 (Validity). A VES scheme $\hat{\Sigma}$ is (τ, ϵ)-valid if $\Pr\left[\mathsf{VES\text{-}Validity}_{\hat{\Sigma}}^{\mathcal{A}} = 1\right] \leq \epsilon$, for all algorithms $\mathcal{A}$ running it at most time τ.

$\underline{\mathsf{VES\text{-}Validity}_{\hat{\Sigma}}^{\mathcal{A}}}$

$(pk_S, (sk_E, pk_E), m, \hat{\sigma}) \leftarrow_{\$} \mathcal{A}$

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

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

2.3 EUF-CMA[VES]

Our goal in this section is to formally capture the requirement that nothing can be learned from a VES ciphertext except the signature encrypted within. We start with the typical $\mathsf{EUF\text{-}CMA}$ 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 VES ciphertext equally offers no extra information to the forger, we modify the experiment so the forger instead has access to a signature encryption oracle E. The forger may query this oracle under any encryption key and message it wants and receives back a valid signature on the message encrypted under that key. This allows, for example, the forger to request a ciphertext where the encryption key is a functions of the signing key they are trying to forge against in the hope that this will leak something about the signing key. This is crucial because as we will see later, due to a problem in our ECDSA one-time VES scheme, the forger can do exactly this to speed up an attempt to recover the secret signing key. Note that we provide the encryption oracle E instead of S, rather than in addition to, because we assume S can be simulated with E by simply requesting an encryption for which the forger knows the decryption key. This assumption holds as long as decrypted signatures are indistinguishable from ordinary signatures, which is true except for some pathological constructions[29]. We call denote this modified experiment $\mathsf{EUF\text{-}CMA[VES]}$ and define it as follows.

Definition 2.3 ($\mathsf{EUF\text{-}CMA[VES]}$). A VES scheme $\hat{\Sigma}$ is (ϵ, τ, Q_E)-$\mathsf{EUF\text{-}CMA[VES]}$ secure if $\Pr\left[\mathsf{EUF\text{-}CMA[VES]}_{\hat{\Sigma}}^{\mathcal{F}} = 1\right] \leq \epsilon$ for all forgers $\mathcal{F}$ making at most Q_E signature encryption queries and running in at most time τ.

$$
\begin{array}{l|l}
\underline{\mathsf{EUF\text{-}CMA[VES]}_{\hat{\Sigma}}^{\mathcal{F}}} & \underline{\text{Oracle } E(pk_E, m)} \\
Q := \emptyset & Q := Q \cup \{m\} \\
(sk_S, pk_S) \leftarrow\!\$\ \mathsf{Gen} & \textbf{return } \mathsf{EncSign}(sk_S, pk_E, m) \\
(m^*, \sigma) \leftarrow\!\$\ \mathcal{F}^{E}(pk_S) & \\
\textbf{return } \mathsf{Vrfy}(pk_S, m^*, \sigma) \wedge m^* \notin Q &
\end{array}
$$

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 and is therefore inadequate for our setting. 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 2.4 (BGLS VES Unforgeability [4]). We say a VES scheme $\hat{\Sigma}$ is $(\tau, \epsilon, Q_E, Q_D)$-BGLS unforgeable if $\Pr[\mathsf{VES\text{-}Forge} = 1] < \epsilon$ for all algorithms $\mathcal{F}$ running in time τ making Q_E and Q_D signature encryption and decryption oracle queries respectively. Note that unlike in the EUF-CMA[VES] experiment the oracles in VES-Forge only provide signature encryptions and decryption on a static key chosen by the experiment (this represents the trusted adjudicator's key).

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

Theorem 2.1 (EUF-CMA[VES] + Validity $\implies$ BGLS Unforgeability). *If a VES scheme is $(\tau_f, \epsilon_f, Q_E)$-EUF-CMA[VES] secure and (τ_v, ϵ_v)-valid then it is $(\tau, \epsilon, Q_E, Q_D)$-BGLS unforgeable where $\epsilon \leq \epsilon_f + \epsilon_v$, $\tau = \tau_f - O(Q_E + Q_D)$ and $\tau_v \approx \tau$.*

Proof. We can bound the advantage of a BGLS forger by constructing an algorithm $\mathcal{R}_{\mathsf{EUF\text{-}CMA[VES]}}$ which forge a signature in the EUF-CMA[VES] experiment and an algorithm $\mathcal{R}_{\text{valid}}$ that outputs a valid ciphertext that does not decrypt to a valid signature. $\mathcal{R}_{\mathsf{EUF\text{-}CMA[VES]}}$ is depicted below.

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

Clearly $\mathcal{R}$ perfectly simulates the view of $\mathcal{F}$ in the real experiment. If $\mathcal{F}$ outputs a valid ciphertext that yields a valid signature upon decryption $\mathcal{R}_{\mathsf{EUF\text{-}CMA[VES]}}$ successfully forges a signature in the EUF-CMA[VES] experiment. In the other case where $\mathcal{F}$ outputs a valid ciphertext that does not yield a valid signature we have broken validity. Therefore we can bound ϵ (the success probability of $\mathcal{F}$) by $\epsilon_f + \epsilon_v$. $\mathcal{R}$ takes only a constant factor longer than $\mathcal{F}$ to run in addition to answering its oracle queries so $\tau' = \tau - O(Q_E + Q_D)$.

□

We now make the case for 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 [4, 27, 31, 30, 28] 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 2.5 (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 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 2.1.

It is easy to see that any valid StE scheme must be EUF-CMA[VES] secure. The 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 2.2 (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. To solve $\mathcal{H}$, $\hat{\mathcal{R}}$ runs R internally, outputs whatever it outputs and lets it answer all oracle queries from the VES forger except for those directed to E. 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. Clearly, $\hat{\mathcal{R}}$ is also a valid EUF-CMA[VES] reduction from $\mathcal{H}$ to $\mathcal{R}$ is a valid EUF-CMA reduction from $\mathcal{H}$. □

3 One-Time Verifiably Encrypted Signatures

We introduce the *one-time verifiably encrypted signature scheme* to abstract the functionality of the "adaptor signature"[1]. 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 3.1 (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, EncSign, 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 6 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 3.2 (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:

$$
\begin{array}{l}
\underline{\mathsf{VES\text{-}Recover}^{\mathcal{A}}_{\mathsf{Setup}}} \\
\hat{\Sigma} \leftarrow_{\$} \mathsf{Setup}(1^k) \\
(pk_S, (sk_E, pk_E), m, \hat{\sigma}) \leftarrow_{\$} \mathcal{A}(\hat{\Sigma}) \\
\delta \leftarrow \hat{\Sigma}.\mathsf{RecKey}(pk_E, \hat{\sigma}); \sigma \leftarrow \hat{\Sigma}.\mathsf{DecSig}(sk_E, \hat{\sigma}) \\
\textbf{return } (\hat{\Sigma}.\mathsf{valid}(sk_E, pk_E) \wedge \hat{\Sigma}.\mathsf{EncVrfy}(pk_S, pk_E, m, \hat{\sigma}) \wedge \hat{\Sigma}.\mathsf{Vrfy}(pk_S, m, \sigma)) \implies \hat{\Sigma}.\mathsf{Rec}(\sigma, \delta) = sk_E
\end{array}
$$

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 Schnorr One-Time VES Scheme

What we call the Schnorr one-time VES was introduced by Poelstra [32] which he termed an "adaptor signature"[1] with the encryption key being termed an "auxiliary point"[14] 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 1 including its underlying Schnorr signature scheme. The Schnorr signature scheme was introduced by its namesake in [33] and our description resembles the key-prefixed scheme described in the Schnorr Bitcoin Improvement Proposal[34] currently under consideration.

$$
\begin{array}{llll}
\underline{\mathsf{Gen}/\mathsf{EncGen}} & \underline{\mathsf{Sign}(sk_S, m)} & \underline{\mathsf{Vrfy}(pk_S, m, \sigma)} & \underline{\mathsf{EncSign}(sk_S, pk_E, m)} \\
x \leftarrow_{\$} \mathbb{Z}_q; X \leftarrow g^x & (x, X) := sk_S; & X := pk_S; (R, s) := \sigma & (x, X) := sk_S; Y := pk_E \\
sk := (x, X); pk := X & r \leftarrow_{\$} \mathbb{Z}_q; R \leftarrow g^r & c := H(R||X||m) & r \leftarrow_{\$} \mathbb{Z}_q; \hat{R} \leftarrow g^r \\
\textbf{return } (sk, pk) & c := H(R||X||m) & \textbf{return } R = g^s X^{-c} & R \leftarrow \hat{R}Y \\
 & s \leftarrow r + cx & & c := H(R||X||m) \\
 & \textbf{return } \sigma := (R, s) & & \hat{s} \leftarrow r + cx \\
 & & & \textbf{return } \hat{\sigma} := (\hat{R}, \hat{s})
\end{array}
$$

$$
\begin{array}{llll}
\underline{\mathsf{EncVrfy}(pk_S, pk_E, m, \hat{\sigma})} & \underline{\mathsf{DecSig}(sk_E, \hat{\sigma})} & \underline{\mathsf{RecKey}(pk_E, \hat{\sigma})} & \underline{\mathsf{Rec}(\sigma, \delta)} \\
X := pk_S; Y := pk_E & (\hat{R}, \hat{s}) := \hat{\sigma} & (\hat{R}, \hat{s}) := \hat{\sigma} & (R, s) := \sigma \\
(\hat{R}, \hat{s}) := \hat{\sigma} & (y, Y) := sk_E & \textbf{return } \delta := \hat{s} & \hat{s} := \delta \\
R \leftarrow \hat{R}Y & R \leftarrow \hat{R}Y & & y \leftarrow s - \hat{s} \\
c := H(R||X||m) & s \leftarrow \hat{s} + y & & \textbf{return } y \\
\textbf{return } \hat{R} = g^{\hat{s}} X^{-c} & \textbf{return } \sigma := (R, s) & &
\end{array}
$$

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. *If the Schnorr signature scheme is $(\tau, \epsilon, Q_H + Q_E)$-EUF-KO secure in the random oracle model then the Schnorr one-time VES is $(\tau', \epsilon', Q_H, Q_E)$-EUF-CMA[VES] secure where*

$$\epsilon' \leq 4\epsilon + \frac{Q_H Q_E}{|\mathbb{G}|}, \tau' \approx \tau$$

and Q_H, Q_E is the upper bound on the random oracle queries and signature encryption queries respectively in the EUF-CMA[VES] experiment.

Proof. The bound stated above identical the one presented by Kiltz et al. [?] in Lemma 3.10 for EUF-CMA except with the number of signature queries Q_S to S replaced with the number of signature encryption queries Q_E to E. This is the correct bound since we can simulate E in the same way as S is usually simulated.

Simulate $S(m)$	Simulate $E(pk_E, m)$
$s, c \leftarrow_\$ \mathbb{Z}_q$	$Y := pk_E$
$R \leftarrow g^s X^{-c}$	$\hat{s}, c \leftarrow_\$ \mathbb{Z}_q$
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

Observe that the probability of the E aborting is identical to that of S. Although R is calculated differently it is distributed identically. E runs in the same time as S except for one extra group operation. Thus, assuming the bound from Kiltz et al. is correct, replacing S with E yields the same bound on the advantage of a EUF-CMA[VES] forger except with Q_S replaced with Q_E. □

5 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[36]. It was later applied to achieve a payment channel construction with better privacy in [8] and an efficient tumbler in [13]. The scheme is built upon the two-party ECDSA protocol from [37]. In Figure 2 we distill it into a single signer scheme which avoids all the complexities that come with two-party ECDSA protocols.

The construction works in a similar way to the Schnorr scheme: the public randomness R is mutated independently of its private counterpart r to include the encryption key Y. This offsets the resulting signature by the same factor. Unfortunately, due to the non-linear structure of ECDSA it needs a non-interactive zero knowledge proof of discrete logarithm equality so the verifier can confirm that s is offset by the correct amount. we denote the proof generation and verification algorithms as follows $(\mathsf{P_{DLEQ}}, \mathsf{V_{DLEQ}})$. Formally, When invoked as $\mathsf{P_{DLEQ}}((g, A), (h, B), w)$, 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 described in [38].

$\mathsf{Gen}/\mathsf{EncGen}$	$\mathsf{Sign}(sk_S, m)$	$\mathsf{Vrfy}(pk_S, m, \sigma)$	$\mathsf{EncSign}(sk_S, pk_E, m)$
$x \leftarrow_\$ \mathbb{Z}_q; X \leftarrow g^x$ $sk := (x, X); pk := X$ **return** (sk, pk)	$(x, X) := sk_S;$ $r \leftarrow_\$ \mathbb{Z}_q; R \leftarrow g^r$ $R_\mathsf{x} \leftarrow f(R)$ $s \leftarrow r^{-1}(H(m) + R_\mathsf{x} x)$ **return** $\sigma := (R_\mathsf{x}, s)$	$X := pk_S; (R_\mathsf{x}, s) := \sigma$ $R' \leftarrow (g^{H(m)} X^{R_\mathsf{x}})^{s^{-1}}$ **return** $f(R') = R_\mathsf{x}$	$(x, X) := sk_S; Y := pk_E$ $r \leftarrow_\$ \mathbb{Z}_q; \hat{R} \leftarrow g^r; R \leftarrow Y^r$ $\pi \leftarrow_\$ \mathsf{P}_{\mathsf{DLEQ}}((g, \hat{R}), (Y, R), r)$ $R_\mathsf{x} \leftarrow f(R)$ $\hat{s} \leftarrow r^{-1}(H(m) + R_\mathsf{x} x)$ **return** $\hat{\sigma} := (R, \hat{R}, \hat{s}, \pi)$

$\mathsf{EncVrfy}(pk_S, pk_E, m, \hat{\sigma})$	$\mathsf{DecSig}(sk_E, \hat{\sigma})$	$\mathsf{RecKey}(pk_E, \hat{\sigma})$	$\mathsf{Rec}(\sigma, \delta)$
$X := pk_S; Y := pk_E$ $(R, \hat{R}, \hat{s}, \pi) := \hat{\sigma}$ $\mathsf{V}_{\mathsf{DLEQ}}((g, \hat{R}), (Y, R)), \pi) \stackrel{?}{=} 1$ $R_\mathsf{x} \leftarrow f(R)$ **return** $\hat{R} = (g^{H(m)} X^{R_\mathsf{x}})^{\hat{s}^{-1}}$	$(R, \hat{R}, \hat{s}, \pi) := \hat{\sigma}$ $(y, Y) := sk_E$ $s \leftarrow \hat{s} y^{-1}$ **return** $\sigma := (f(R), s)$	$(R, \hat{R}, \hat{s}, \pi) := \hat{\sigma}$ $Y := pk_E$ **return** $\delta := (Y, \hat{s})$	$(R_\mathsf{x}, s) := \sigma; (Y, \hat{s}) := \delta$ $\tilde{y} \leftarrow s^{-1}\hat{s}$ $y := \begin{cases} \tilde{y} & \textbf{if } g^{\tilde{y}} = Y \\ -\tilde{y} & \textbf{if } g^{\tilde{y}} = Y^{-1} \\ \perp & \textbf{otherwise} \end{cases}$ **return** y

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.

We now prove the scheme secure by showing it meets our requirements of validity, recoverability and EUF-CMA[VES]. In the latter case, the proof is not straightforward because we must account for a weakness in the scheme. First we prove validity and recoverability.

Lemma 5.1. *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). □

5.1 Analysis of EUF-CMA[VES] security

The flaw in the scheme is that each valid ciphertext surreptitiously leaks the Diffie-Hellman key between the public signing key and the encryption key i.e. allows the receiver to compute $X^y = Y^x$. At a high level, the problem arises because the value s in the signature is computed through the product of r^{-1} and x and then π reveals the product of Y and r. The adversary can then cancel out r from the picture and is left with Y^x. In detail we start with a valid ciphertext $(R, \hat{R}, \hat{s}, \pi)$ on some message with signing key X and encryption key Y.

Lemma 5.2. *Let $(R, \hat{R}, \hat{s}, \pi)$ be a ECDSA one-time VES ciphertext. If $\mathsf{EncVrfy}(X, Y, m, (R, \hat{R}, \hat{s}, \pi)) = 1$, for some message m and encryption key Y, then $\mathsf{CDH}(X, Y) = R^{\hat{s}} Y^{-H(m)})^{R_\mathsf{x}^{-1}}$.*

Proof. Since $\hat{s} = r^{-1}(H(m) + R_\mathsf{x} x)$ and $R = Y^r$ we can compute Y^x as follows: o

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

□

This clearly violates the spirit of $\mathsf{EUF\text{-}CMA[VES]}$ which is that nothing useful should be learned from the ciphertext, other than the signature (if it can be decrypted). To demonstrate the fidelity of our formal definitions to this idea we show there is no $\mathsf{EUF\text{-}CMA[VES]}$ reduction for it

Lemma 5.3. *There is no* key-preserving *reduction from* DL *to the* $\mathsf{EUF\text{-}CMA[VES]}$ *security of the ECDSA one-time VES if the* CDH *problem is hard.*

Proof Sketch. Observe that any key preserving DL reduction must simulate the encryption oracle E without the secret key x. Note that it is not enough for this simulator to just return two random group elements $(\hat{R}, R)$ and simulate the proof π to make them appear valid with respect to a query for an encryption under Y. We can easily catch this behaviour by querying the simulator on key Y' such that we know the secret key y' and checking that $\hat{R}^y = R$. Thus since the simulator must return valid ciphertexts and from valid ciphertexts the Diffie-Hellman key can be extracted (as shown above) the simulator must not exist if CDH problem is hard. If no simulator exists then no key-preserving reduction can exist. $\square$

Since we believe this scheme is useful we will now attempt to salvage it by formally capturing the flaw and prove it secure in a weaker model. The problem for the reduction above is that by providing the signature encryption oracle you are also providing what is referred to by Brown et al. as a *static Diffie-Hellman* oracle[39] i.e. an oracle that will return Y^x for some fixed x on a query for Y. Such an oracle can obviously not be simulated if the CDH problem is hard. Our solution to this conundrum is to give our reduction access to such an oracle, denoted as $\mathcal{O}_{\mathsf{SDH}}$, and to show a reduction from DL in this model. Formally, our reduction is from the following weaker version of the DL problem. Note we do not give $\mathcal{A}$ a the group element for which is must find the discrete logarithm as in the typical DL experiment since it may simply query $\mathcal{O}_{\mathsf{SDH}}(g)$ to get any particular instance of a discrete logarithm problem with respect to any g.

Definition 5.1 (Discrete log with static Diffie-Hellman oracle problem). An algorithm $\mathcal{A}$ solves the u-$\mathsf{DL}_{\mathcal{O}_{\mathsf{SDH}}}$ problem in a group $\mathbb{G}$ if the following experiment outputs 1 and $\mathcal{A}$ makes u or less queries to $\mathcal{O}_{\mathsf{SDH}}$.

$\mathsf{DL}_{\mathcal{O}_{\mathsf{SDH}}}$	$\mathcal{O}_{\mathsf{SDH}}(Y)$
$x \leftarrow_\$ \mathbb{Z}_q$	**return** Y^x
$x^* \leftarrow_\$ \mathcal{A}^{\mathcal{O}_{\mathsf{SDH}}}$	
return $x^* \stackrel{?}{=} x$	

Using u-$\mathsf{DL}_{\mathcal{O}_{\mathsf{SDH}}}$ instead of DL in our $\mathsf{EUF\text{-}CMA[VES]}$ reduction means it no longer proves that ciphertexts contain no useful information for an adversary, but instead, that the Diffie-Hellman key Y^x is the only extra thing that can be extracted from a ciphertext. Note that the existing protocols that use ECDSA adaptor signatures sidestep this issue in their simulation based proofs by making any receiver of the signature encryption prove knowledge of the decryption key. Obviously if they already know the decryption key y then they can compute $X^y = Y^x$ without help from the signer. We seek to prove the scheme secure without proofs of knowledge since it may not always be practical to provide such a proofs and it complicates the application of the scheme. This security model allows us to prove the scheme secure but means we must make the following considerations whenever it is employed.

Firstly, when proving a protocol secure that uses the scheme, any time the adversary learns a ciphertext they should also be given the Diffie-Hellman key Y^x. This prevents accidentally proving a scheme secure that, for example, uses a key as both an ElGamal encryption key and to create ECDSA encrypted signatures. Practically, this issue is not as severe as it seems as none of protocols presented in Section 1.1 are based on Diffie-Hellman problems. 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 Diffie-Hellman key in one execution cannot help an adversary break the security of another.

Secondly, the scheme is only as secure as $\mathsf{DL}_{\mathcal{O}_{\mathsf{SDH}}}$ which is a strictly easier problem than DL. The reason that $\mathsf{DL}_{\mathcal{O}_{\mathsf{SDH}}}$ is easier is that the $\mathcal{O}_{\mathsf{SDH}}$ oracle actually helps the adversary break DL in a very subtle and unexpected way. As shown by Brown et al.[39] it is possible to use $\mathcal{O}_{\mathsf{SDH}}$ to assist in computing the discrete logarithm of the static key (i.e the public signing key). The attack works by querying $\mathcal{O}_{\mathsf{SDH}}$ with the static key itself and thereby learning non-linear functions of the static secret key *in the exponent* e.g. a query for the static key $X = g^x$ returns g^{x^2} and then querying with that result returns g^{x^3} and so on. In the next section, we give a fuller description of the attack and estimate how hard the u-$\mathsf{DL}_{\mathcal{O}_{\mathsf{SDH}}}$ problem is for different values u.

To make our security claim we bound the adversary against the ECDSA one-time VES by the difficulty of solving u-$\mathsf{DL}_{\mathcal{O}_{\mathsf{SDH}}}$. As in the case of our Schnorr proof we use an existing EUF-CMA reduction as our starting point. The work of Fersch et al.[40] meticulously capture the EUF-CMA security of ECDSA in a *bijective random oracle model* by showing a reduction from the DL. Their reduction also shows the security of ECDSA relies on particular properties of the message hash function H. We ignore this aspect of the reduction to focus on the relative advantage against DL but they can be thought of as implicit in the bounds function B below. In the ECDSA bijective random oracle model, the conversation function f which converts an elliptic curve point R to its modulo q reduced x-coordinate R_{x} is modelled as a bijective random oracle and signatures are simulated by programming it. As in the case of Schnorr, our approach is to modify the signature simulator to be a signature encryption simulator except that in this case we require the assistance of $\mathcal{O}_{\mathsf{SDH}}$. We provide more details of the reduction prove the following theorem in Appendix A.

Theorem 5.1. *Let $\mathcal{F}$ $(\tau, Q_E, Q_\Pi, \epsilon)$-break the EUF-CMA[VES] security of ECDSA. Then if Π is modelled as a random oracle there exists an adversary that $(\tau_{\psi dr}, \epsilon_{\psi dr})$-breaks the ψ-relative division resistance of H, an adversary that $(\tau_{cr}, \epsilon_{cr})$-breaks collision resistance of H and inverters that $(\tau', \epsilon', Q_E + 1)$-break and $(\tau'', \epsilon'', Q_E + 1)$-break, respectively, $\mathsf{DL}_{\mathcal{O}_{\mathsf{SDH}}}$ in $\mathbb{G}$ such that*

$$\epsilon \leq \sqrt{2Q_\Pi^2 \epsilon_{\psi dr} + 2Q_\Pi \epsilon'} + \epsilon'' + Q_\Pi^2/2^L + \epsilon_{cr} + \frac{3QQ_E}{(q-1)/2 - Q}$$

and $\tau_{\psi dr} = \tau' = 2\tau + O(Q_E) + O(Q_\Pi)$, $\tau'' = \tau + O(Q_E) + O(Q_\Pi)$, $\tau_{cr} = \tau + O(Q_E)$ and $u = Q_E + 1$.

Simply stated, in the bijective random oracle model, one can break u-$\mathsf{DL}_{\mathcal{O}_{\mathsf{SDH}}}$ with a EUF-CMA[VES] forger if one can break DL with a EUF-CMA forger. The reduction of Fersh et al. and therefore our transformed reduction, loses a factor of Q_Π and the square of the advantage i.e ignoring other terms $B(\epsilon', Q_S, Q_\Pi) \approx \sqrt{\epsilon' Q_\Pi}$.

5.2 Hardness of u-$\mathsf{DL}_{\mathcal{O}_{\mathsf{SDH}}}$ in secp256k1

To show the scheme is practical we now give concrete estimates of how difficult $\mathsf{DL}_{\mathcal{O}_{\mathsf{SDH}}}$ problem should is within the secp256k1 elliptic curve group used for ECDSA on Bitcoin. The basis for our estimates is the work of Brown et al.[39] who show the best known algorithm for solving u-$\mathsf{DL}_{\mathcal{O}_{\mathsf{SDH}}}$. It does so in $O(\sqrt{q/u})$ time where q is the order of the group. The algorithm is based on the idea that if we suppose the order of Z_q^* is divisible by u (or some value less than u) and so $uv = q - 1$ for some v, then there are subgroups of Z_q^* order u and v since they both divide the order. Simplifying a bit, the attack splits the problem of finding the discrete logarithm x into finding the discrete logarithm of two smaller components of x in the subgroups of order u and v. To this end the attacker queries the oracle u times to compute g^{x^u} which allows it to work in the subgroup of order v where it solves the smaller DL problem using a modified baby-step-giant-step algorithm. From there, they do the same for the subgroup of order u and combine the results to finally produce the discrete logarithm.

In addition to u oracle queries the algorithm requires $n = 2(\sqrt{u} + \sqrt{v})$ scalar multiplications in $\mathbb{G}$ to finally output the discrete logarithm with certainty. Treating the oracle queries and scalar multiplications as equal, the algorithm has an optimal running time of approximately $3\sqrt[3]{q}$ where $u = \sqrt[3]{q}$. Since the main and likely only application of this scheme will be to Bitcoin's ECDSA signature we provide concrete estimates for the amount of computation needed to run this algorithm in secp256k1 in Figure 3. As expected, the table shows the optimal value for u (treating scalar multiplications between the attacker and the oracle as equal) is $2^{84} \approx \sqrt[3]{q}$ which gives $n \approx 2^{86}$. Smaller and more plausible values for u, e.g. $u \approx 2^{24} \approx 16$ million give $n \approx 2^{116}$ which means it is only a minor improvement on the generic algorithms for solving DL in elliptic curve groups which takes $\sqrt{q} \approx 2^{128}$ group operations.

To be confident that the ECDSA one-time VES can be used in practice we provide support for two final claims: (i) the above algorithm approximates the optimal algorithm for recovering a discrete logarithm with acess to $\mathcal{O}_{\mathsf{SDH}}$ and (ii) there is no ancillary advantage a ECDSA forger can get from $\mathcal{O}_{\mathsf{SDH}}$ other than using it to recover the discrete logarithm of the signing key. To support the first claim we refer the reader to the work of Boneh and Boyen [41] that shows no generic algorithm can improve upon the complexity of the above algoirthm. Specifically they prove a lower time bound of $\Omega(\sqrt{q/u})$ where $u < \sqrt[3]{q}$ for any generic adversary against the *Strong Diffie-Hellan* problem which implies the same lower bound for $\mathsf{DL}_{\mathcal{O}_{\mathsf{SDH}}}$. The second claim is likely to be true since no analysis of ECDSA thus far has shown any relationship to the difficulty of forging

$q = 2^6 \times 3 \times 149 \times 631 \times 107361793816595537 \times 174723607534414371449 \times 341948486974166000522343609283189$

$\geq \log_2 u$	$\lfloor \log_2 u \rfloor$	$\lfloor \log_2 n \rfloor$	Factorization of u
*	84	86	$2 \times 149 \times 631 \times 174723607534414371449$
80	80	88	$2^2 \times 3 \times 631 \times 174723607534414371449$
70	70	93	$2^3 \times 174723607534414371449$
60	60	98	$2^2 \times 3 \times 107361793816595537$
50	24	116	$2^6 \times 3 \times 149 \times 631$

Fig. 3. The optimal number of scalar multiplications n required in secp256k1 for each upper bound on the bit length of u static Diffie-Hellman oracle queries to solve the $u\text{-}\mathsf{DL}_{\mathcal{O}_{\mathsf{SDH}}}$ problem

signatures to any Diffie-Hellman type problem. If we are accept these two propositions, then the scheme is secure for use in Bitcoin as it is today. For perspective on the security loss, currently the only practical way to use Bitcoin script is to use *pay-to-script-hash* type outputs which commit to script based spending rules with a 160-bit hash and thus only provide 80 bits of collision resistance. Furthermore, We stress that we propose this scheme as a short term solution until Schnorr signatures are included in the Bitcoin protocol which also enables the Schnorr one-time VES.

6 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[37, 42], Schnorr[25], BLS[24] and most prominent signature schemes.

To realise most of the scriptless protocols from Section 1.1, the multi-signature scheme also needs to admit a two-party one-time VES encrypted signing protocol to emulate $\mathsf{EncSign}$ 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 [37, 42] admit a two-party $\mathsf{EncSign}$ protocol as described in [36]. 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.[43]

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 5. The simple two-party signing and encrypted signing protocols are presented in Figure 4.

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).

$\mathsf{2p\text{-}Sign}(pk := (pk_1, pk_2), m)$		
$\mathbf{P}_1(sk_1)$		$\mathbf{P}_2(sk_2)$
$\sigma_1 \leftarrow\!\!\$\ \mathsf{Sign}(sk_1, m)$		
	$\xrightarrow{\sigma_1}$	
		$\mathsf{Vrfy}(pk_1, m, \sigma_1) \stackrel{?}{=} 1$
		$\sigma_2 \leftarrow\!\!\$\ \mathsf{Sign}(sk_2, m)$
		return $\sigma := (\sigma_1, \sigma_2)$

$\mathsf{2p\text{-}EncSign}(pk := (pk_1, pk_2), pk_E, m)$		
$\mathbf{P}_1(sk_1)$		$\mathbf{P}_2(sk_2)$
$\hat{\sigma}_1 \leftarrow\!\!\$\ \mathsf{EncSign}(sk_1, pk_E, m)$		
$\delta \leftarrow \mathsf{RecKey}(pk_E, \hat{\sigma}_1)$		
	$\xrightarrow{\hat{\sigma}_1}$	
		$\mathsf{EncVrfy}(pk_1, pk_E, m, \hat{\sigma}_1) \stackrel{?}{=} 1$
		$\sigma_2 \leftarrow\!\!\$\ \mathsf{Sign}(sk_2, m)$
		$\hat{\sigma} := (\hat{\sigma}_1, \sigma_2)$
return δ		**return** $\hat{\sigma}$

Fig. 4. The two-party signing and encrypted signing algorithms for an 2-of-2 `OP_CHECKMULTISIG` output.

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 [3]:** The typical hash lock can be replaced with a discrete logarithm based lock which enables the privacy benefits from [8] other conjectured improvements[9].
- **Atomic swaps [7]:** 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 [10]:** The protocol can be completed in two transactions rather than three.

References

[1] Andrew Poelstra. Scriptless scripts. `https://download.wpsoftware.net/bitcoin/wizardry/mw-slides/2017-05-milan-meetup/slides.pdf`, 2017.

[2] Andrew Poelstra. Lightning in scriptless scripts. `https://lists.launchpad.net/mimblewimble/msg00086.html`, 2017.

[3] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. `https://lightning.network/lightning-network-paper.pdf`, 2016.

[4] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proceedings of the 22Nd International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'03, pages 416–432, Berlin, Heidelberg, 2003. Springer-Verlag.

[5] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '01, pages 514–532, Berlin, Heidelberg, 2001. Springer-Verlag.

[6] Tier Nolan. Alt chains and atomic transfers. `https://bitcointalk.org/index.php?topic=193281.msg2224949#msg2224949`.

[7] Andrew Poelstra. Adaptor signatures and atomic swaps from scriptless scripts. `https://github.com/ElementsProject/scriptless-scripts/blob/master/md/atomic-swap.md`, 2017.

[8] Giulio Malavolta, Pedro Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. Anonymous multi-hop locks for blockchain scalability and interoperability. Cryptology ePrint Archive, Report 2018/472, 2018. https://eprint.iacr.org/2018/472.

[9] Rusty Russel. Bolt11 in the world of scriptless scripts. https://lists.linuxfoundation.org/pipermail/lightning-dev/2018-November/001489.html, 2018.

[10] Thaddeus Dryja. Discreet log contracts. https://adiabat.github.io/dlc.pdf, 2017.

[11] David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer, 1983.

[12] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. Tumblebit: An untrusted bitcoin-compatible anonymous payment hub. Cryptology ePrint Archive, Report 2016/575, 2016. https://eprint.iacr.org/2016/575.

[13] Erkan Tairi, Pedro Moreno-Sanchez, and Matteo Maffei. A^2l: Anonymous atomic locks for scalability and interoperability in payment channel hubs. Cryptology ePrint Archive, Report 2019/589, 2019. https://eprint.iacr.org/2019/589.

[14] Jonas Nick. Partially blind atomic swap using adaptor signatures. https://github.com/ElementsProject/scriptless-scripts/blob/master/md/partially-blind-swap.md, 2017.

[15] fair coin toss with no extortion and no need to trust a third party. https://bitcointalk.org/index.php?topic=277048.0, 2013.

[16] Iddo Bentov and Ranjit Kumaresan. How to use bitcoin to design fair protocols. Cryptology ePrint Archive, Report 2014/129, 2014. https://eprint.iacr.org/2014/129.

[17] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Łukasz Mazurek. Secure multiparty computations on bitcoin. Cryptology ePrint Archive, Report 2013/784, 2013. https://eprint.iacr.org/2013/784.

[18] Massimo Bartoletti and Roberto Zunino. Constant-deposit multiparty lotteries on bitcoin. Cryptology ePrint Archive, Report 2016/955, 2016. https://eprint.iacr.org/2016/955.

[19] Andrew Miller and Iddo Bentov. Zero-collateral lotteries in bitcoin and ethereum. *CoRR*, abs/1612.05390, 2016.

[20] Lloyd Fournier. Scriptless bitcoin lotteries from oblivious transfer. https://telaviv2019.scalingbitcoin.org/files/scriptless-lotteries-on-bitcoin-from-oblivious-transfer.pdf, 2016.

[21] Raylin Tso, Takeshi Okamoto, and Eiji Okamoto. 1outtoffnn oblivious signatures. In *Proceedings of the 4th International Conference on Information Security Practice and Experience*, ISPEC'08, pages 45–55, Berlin, Heidelberg, 2008. Springer-Verlag.

[22] Jonas Nick. Atomic pedersen swap using adaptor signatures. https://github.com/ElementsProject/scriptless-scripts/blob/master/md/pedersen-swap.md, 2017.

[23] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. https://eprint.iacr.org/2000/067.

[24] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. Cryptology ePrint Archive, Report 2018/483, 2018. https://eprint.iacr.org/2018/483.

[25] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple schnorr multi-signatures with applications to bitcoin. Cryptology ePrint Archive, Report 2018/068, 2018. https://eprint.iacr.org/2018/068.

[26] Laila El Aimani. *Verifiable Composition of Signature and Encryption*. 12 2017.

[27] Markus Rückert and Dominique Schröder. Security of verifiably encrypted signatures and a construction without random oracles. In *Proceedings of the 3rd International Conference Palo Alto on Pairing-Based Cryptography*, Pairing '09, pages 17–34, Berlin, Heidelberg, 2009. Springer-Verlag.

[28] Christian Hanser, Max Rabkin, and Dominique Schröder. Verifiably encrypted signatures: Security revisited and a new construction. In *Proceedings, Part I, of the 20th European Symposium on Computer Security – ESORICS 2015 - Volume 9326*, pages 146–164, Berlin, Heidelberg, 2015. Springer-Verlag.

[29] Theresa Calderon, Sarah Meiklejohn, Hovav Shacham, and Brent Waters. Rethinking verifiably encrypted signatures: a gap in functionality and potential solutions. In *Cryptographers' Track at the RSA Conference*, pages 349–366. Springer, 2014.

[30] Zuhua Shao. Fair exchange protocol of signatures based on aggregate signatures. *Computer Communications*, 31(10):1961 – 1969, 2008.

[31] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. Cryptology ePrint Archive, Report 2006/096, 2006. `https://eprint.iacr.org/2006/096`.

[32] Andrew Poelstra. Scriptless scripts. `https://download.wpsoftware.net/bitcoin/wizardry/mw-slides/2017-03-mit-bitcoin-expo/slides.pdf`, 2017.

[33] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '89, pages 239–252, Berlin, Heidelberg, 1990. Springer-Verlag.

[34] Pieter Wuille. Taproot: Segwit version 1 output spending rules. `https://github.com/sipa/bips/blob/bip-schnorr/bip-schnorr.mediawiki`. Accessed 2019-06-09.

[35] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[36] Aniket Kate Pedro Moreno-Sanchez. Scriptless scripts with ecdsa. https://lists.linuxfoundation.org/pipermail/lightning-dev/attachments/20180426/fe978423/attachment-0001.pdf, 2018.

[37] Yehuda Lindell. Fast secure two-party ecdsa signing. Cryptology ePrint Archive, Report 2017/552, 2017. `https://eprint.iacr.org/2017/552`.

[38] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '92, pages 89–105, London, UK, UK, 1993. Springer-Verlag.

[39] Daniel Brown and Robert Gallant. The static diffie-hellman problem. *IACR Cryptology ePrint Archive*, 2004:306, 01 2004.

[40] Manuel Fersch, Eike Kiltz, and Bertram Poettering. On the provable security of (ec)dsa signatures. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1651–1662, New York, NY, USA, 2016. ACM.

[41] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *Journal of cryptology*, 21(2):149–177, 2008.

[42] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Two-party ecdsa from hash proof systems and efficient instantiations. Cryptology ePrint Archive, Report 2019/503, 2019. `https://eprint.iacr.org/2019/503`.

[43] ZmnSCPxj. Payment point+scalar. `https://lists.linuxfoundation.org/pipermail/lightning-dev/2019-October/002211.html`, 2019.

[44] Prabhat Kushwaha. Improved lower bound on DHP: towards the equivalence of DHP and DLP for important elliptic curves used for implementation. *CoRR*, abs/1610.01354, 2016.

A Proof for Theorem ??

To prove our theorem, we show that when the signature oracle S from the reduction by Fersh et alis replaced by a signature encryption oracle E it becomes a $\mathsf{EUF\text{-}CMA}[\mathsf{VES}]$ reduction from the u-$\mathsf{DL}_{\mathcal{O}_{\mathsf{SDH}}}$ problem where u is the number of queries to E plus 1.

First we give more thorough description of the original reduction. Fersh et aldecomposed and idealize the conversion function $f : \mathbb{G} \to \mathbb{Z}_q$ that maps a group element to its x-coordinate mod q as a random oracle. The forgery queries the oracle and eventually outputs a forgery with some probability. To solve a an instance of DL the reduction programs the oracle and rewinds the successful forger to attempt to get it to produce two signatures with different values for $R_{\mathsf{x}} = f(R)$ but the same value for R in the typical "forking lemma". In more detail, f is decomposed as $f = \psi \circ \Pi \circ \varphi$ where:

1. $\varphi : \mathbb{G} \to \mathbb{A}$ 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. $\Pi : \mathbb{A} \to \mathbb{B}$ is the bijective random oracle that the reduction is able to program.

3. $\psi : \mathbb{B} \to \mathbb{Z}_q$ is an invertible function that maps the range of Π to $\mathbb{Z}_q$.

Theorem A.1. *Let $\mathcal{F}$ $(\tau, Q_s, Q_\Pi, \epsilon)$-break the $\mathsf{EUF\text{-}CMA}$ security of ECDSA. Then if Π is modelled as a random oracle there exists an adversary that $(\tau_{\psi\mathsf{dr}}, \epsilon_{\psi\mathsf{dr}})$-breaks the ψ-relative division resistance of H, an adversary that $(\tau_{\mathsf{cr}}, \epsilon_{\mathsf{cr}})$-breaks collision resistance of H and inverters that (τ', ϵ')-break and (τ'', ϵ'')-break, respectively, DL in $\mathbb{G}$ such that*

$$\epsilon \leq \sqrt{2Q_\Pi^2 \epsilon_{\psi\mathsf{dr}} + 2Q_\Pi \epsilon'} + \epsilon'' + Q_\Pi^2/2^L + \epsilon_{\mathsf{cr}} + \frac{3QQ_s}{(q-1)/2 - Q}$$

and $\tau_{\psi\mathsf{dr}} = \tau' = 2\tau + O(Q_s) + O(Q_\Pi)$, $\tau'' = \tau + O(Q_s) + O(Q_\Pi)$ and $\tau_{\mathsf{cr}} = \tau + O(Q_s)$.

Thus our theorem begins as follows

Theorem A.2. *Let $\mathcal{F}$ $(\tau, Q_E, Q_\Pi, \epsilon)$-break the $\mathsf{EUF\text{-}CMA}[\mathsf{VES}]$ security of ECDSA. Then if Π is modelled as a random oracle there exists an adversary that $(\tau_{\psi\mathsf{dr}}, \epsilon_{\psi\mathsf{dr}})$-breaks the ψ-relative division resistance of H, an adversary that $(\tau_{\mathsf{cr}}, \epsilon_{\mathsf{cr}})$-breaks collision resistance of H and inverters that (τ', ϵ')-break and (τ'', ϵ'')-break, respectively, u-$\mathsf{DL}_{\mathcal{O}_{\mathsf{SDH}}}$ in $\mathbb{G}$ such that*

$$\epsilon \leq \sqrt{2Q_\Pi^2 \epsilon_{\psi\mathsf{dr}} + 2Q_\Pi \epsilon'} + \epsilon'' + Q_\Pi^2/2^L + \epsilon_{\mathsf{cr}} + \frac{3QQ_E}{(q-1)/2 - Q}$$

and $\tau_{\psi\mathsf{dr}} = \tau' = 2\tau + O(Q_E) + O(Q_\Pi)$, $\tau'' = \tau + O(Q_E) + O(Q_\Pi)$, $\tau_{\mathsf{cr}} = \tau + O(Q_E)$ and $u = Q_E + 1$.

To begin our reduction to u-$\mathsf{DL}_{\mathcal{O}_{\mathsf{SDH}}}$ we first query $X \leftarrow \mathcal{O}_{\mathsf{SDH}}(g)$, which gives us the notional public key X that the signature forger would see in a real experiment. The reduction then carries on exactly as in the original by activating the forger with X and responding to its oracle queries except that the signature oracle S is replaced with an signature encryption oracle E.

For each query to $E(Y, m)$ we must return $(R, \hat{R}, \hat{s}, \pi)$ such that the ciphertext is valid and $\hat{R}^y = R$ where $Y = g^y$. The latter requirement is not straightforward since the simulator does not know the discrete logarithms of $\hat{R}$ and R and does not know y. The trick here is to query $\mathcal{O}_{\mathsf{SDH}}$ with Y to figure out X^y reverse the operations from Lemma 5.2 to compute R and $\hat{R}$ correctly. Once we have a well formed $(\hat{R}, R)$ we run the simulator for $\mathsf{P}_{\mathsf{DLEQ}}$ which we denote $\mathcal{S}_{\mathsf{DLEQ}}$ to create π without the witness y (which we have access to due to its zero knowledge property). We depict the simulation of the signature encryption oracle E along side the signature oracle S of the original reduction below.

Simulate $S(m)$	Simulate $E(Y, m)$
$\beta \leftarrow_{\$} \mathbb{B}$	$\beta \leftarrow_{\$} \mathbb{B}$
if $(\cdot, \beta) \in \Pi$: **abort**	**if** $(\cdot, \beta) \in \Pi$: **abort**
$R_{\mathsf{x}} \leftarrow \psi(\beta)$	$R_{\mathsf{x}} \leftarrow \psi(\beta)$
$s \leftarrow_{\$} \mathbb{Z}_q$	$\hat{s} \leftarrow_{\$} \mathbb{Z}_q$
$u_1 \leftarrow H(m)s^{-1}$	$Z \leftarrow \mathcal{O}_{\mathsf{SDH}}(Y)$
$u_2 \leftarrow R_{\mathsf{x}}s^{-1}$	$u_1 \leftarrow H(m)\hat{s}^{-1}$
$R \leftarrow g_1^u X^{u_2}$	$u_2 \leftarrow R_{\mathsf{x}}\hat{s}^{-1}$
$\alpha \leftarrow \varphi(R)$	$R \leftarrow Z^{u_1} Y^{u_2}$
if $(\alpha, \cdot) \in \Pi$: **abort**	$\hat{R} \leftarrow X^{u_1} g^{u_2}$
$\Pi \leftarrow \Pi \cup \{(\alpha, \beta)\}$	$\pi \leftarrow \mathcal{S}_{\mathsf{DLEQ}}((g, \hat{R}), (Y, R))$
$Q \leftarrow Q \cup \{m\}$	$\alpha \leftarrow \varphi(R)$
return (s, R_{x})	**if** $(\alpha, \cdot) \in \Pi$: **abort**
	$\Pi \leftarrow \Pi \cup \{(\alpha, \beta)\}$
	$Q \leftarrow Q \cup \{m\}$
	return $(R, \hat{R}, \hat{s}, \pi)$

Observe that the probability of the simulator for E aborting per query is the same as for the simulator for S.

Note that we leave out and several details to focus on the simulation of E which is the only relevant part for proving our theorem.

for the sake of clarity and encourage the reader to review the original proof in [40] to get a full understanding of the reduction.