Cryptographic Oracle-Based Conditional Payments

Varun Madathil
North Carolina State University
vrmadath@ncsu.edu

Sri AravindaKrishnan Thyagarajan
NTT Research
t.srikrishnan@gmail.com

Dimitrios Vasilopoulos
IMDEA Software Institute
dimitrios.vasilopoulos@imdea.org

Lloyd Fournier
Independent Researcher
lloyd.fourn@gmail.com

Giulio Malavolta
Max Planck Institute for Security and Privacy
giulio.malavolta@hotmail.it

Pedro Moreno-Sanchez
IMDEA Software Institute
pedro.moreno@imdea.org

***Abstract*—We consider a scenario where two mutually distrustful parties, Alice and Bob, want to perform a payment conditioned on the outcome of some real-world event. A semi-trusted oracle (or a threshold number of oracles, in a distributed trust setting) is entrusted to attest that such an outcome indeed occurred, and only then the payment is successfully made. Such *oracle-based conditional (ObC) payments* are ubiquitous in many real-world applications, like financial adjudication, pre-scheduled payments or trading, and are a necessary building block to introduce information about real-world events into blockchains.**

In this work we show how to realize ObC payments with provable security guarantees and efficient instantiations. To do this, we propose a new cryptographic primitive that we call *verifiable witness encryption based on threshold signatures (VweTS)*: Users can encrypt signatures on payments that can be decrypted if a threshold number of signers (e.g., oracles) sign another message (e.g., the description of an event outcome). We require two security notions: (1) *one-wayness* that guarantees that without the threshold number of signatures, the ciphertext hides the encrypted signature, and (2) *verifiability*, that guarantees that a ciphertext that correctly verifies can be successfully decrypted to reveal the underlying signature.

We present provably secure and efficient instantiations of VweTS where the encrypted signature can be some of the widely used schemes like Schnorr, ECDSA or BLS signatures. Our main technical innovation is a new batching technique for cut-and-choose, inspired by the work of Lindell-Riva on garbled circuits. Our VweTS instantiations can be readily used to realize ObC payments on virtually all cryptocurrencies of today in a fungible, cost-efficient, and scalable manner. The resulting ObC payments are the first to support distributed trust (i.e., multiple oracles) without requiring any form of synchrony or coordination among the users and the oracles. To demonstrate the practicality of our scheme, we present a prototype implementation and our benchmarks in commodity hardware show that the computation overhead is less than 25 seconds even for a threshold of 4-of-7 and a payment conditioned on 1024 different real-world event outcomes, while the communication overhead is below 2.3 MB.

I. Introduction

Many Decentralized Finance (DeFi) applications that offer a complex financial architecture for scenarios like money lending, decentralized exchange of assets, markets of derivatives, etc. [37], [27], make use of services of *oracles* or external *data feeds* to input information that is external to the blockchain. For example, many Ethereum-based DeFi applications [5] rely on such oracles at their core, and there exist companies such as Chainlink [1] whose business model consists on offering oracle services to current and future smart contracts. However, conditioning a blockchain payment on a real-world event (certified by some oracle), turns out to be a non-trivial problem.

To illustrate the obstacles, consider the toy example where Alice wants to make a payment (denoted by m) to Bob provided an oracle (Olivia) attests to the occurrence of some external outcome (denoted by $\overline{m}$). As the first step, we require Alice to lock some funds into a shared address with Bob, for a pre-determined amount of time.[1] In blockchain-based cryptocurrencies, this is a standard procedure that can be realized, e.g., in the form of *2-out-of-2 multisig addresses* [33]. To complete the transfer, Bob needs Alice's signature on a transaction from the locked address to Bob's address. However, we are now faced with a conundrum:

- Alice cannot send Bob the signed transaction *before* Olivia's attestation, since Bob may decide to cash it in regardless of the external outcome.
- Bob cannot rely on Alice to send him the signed transaction *after* Olivia's attestation, since Alice may decide to go offline and never perform the agreed-upon transaction.

How can we design a secure exchange if Alice and Bob do not trust each other?

In this work, we advocate for a *cryptographic solution* to this problem. We design a special-purpose encryption scheme that allows Alice to send Bob an encrypted version of her signature σ_m, that Bob can decrypt using only the attestation of Olivia (namely, a signature on $\overline{m}$). More precisely, we require two main properties from this special-purpose encryption scheme: (1) Bob wants to be ensured that, if Olivia indeed attests the outcome $\overline{m}$, he would indeed obtain the signature σ_m and therefore redeem the payment m (*verifiability*). (2) Alice wants to ensure that in the absence of the attestation on $\overline{m}$, Bob cannot obtain the signature σ_m (*one-wayness*).

[1]This is necessary to ensure that Alice does not quit the protocol prematurely, in case of an unfavorable outcome. The funds are locked for a pre-specified time T, which determines the maximum duration of the protocol.

Network and Distributed System Security (NDSS) Symposium 2023
27 February - 3 March 2023, San Diego, CA, USA
ISBN 1-891562-83-5
https://dx.doi.org/10.14722/ndss.2023.24024
www.ndss-symposium.org

As an additional property, our scheme will also enable Alice and Bob to condition the decryption on the output of *multiple oracles*. This is relevant for the case where Alice and Bob may not trust a single Olivia and may instead avail the service of N different Olivia(s) with the promise that, if a large enough fraction of Olivia(s) agrees on an outcome $\overline{m}$, then the payment is made. The natural requirement for this scenario is that corrupting a small fraction of Olivia(s) does not jeopardize the security of the overall system (*threshold security*).

We refer to these conditional payments between Alice and Bob as *oracle-based conditional (ObC) payments*. Somewhat surprisingly, ObC payments lack a thorough investigation in the literature and there are no formal models and efficient instantiations. The focus of this work is to place ObC payments on a firm cryptographic foundation and propose *practical* protocols based on well-studied cryptographic assumptions.

A. Applications

While ObC payments may appear to be an abstract problem, we argue that this is in fact a recurrent scenario in many real-world applications, even beyond cryptocurrencies. To see this, observe that ObC payments are not tied to any particular functionality of a cryptocurrency: As long as the payment system supports locking funds for a pre-determined amount of time (e.g., authorization credit card holds[2]) then one can use ObC on top of more traditional centralized banking systems. We discuss a few representative example applications below.

Financial Adjudication. Companies and business firms often get involved in mergers and acquisitions. In such cases, they have a clearly worded and binding terms of agreement. Violating the agreement can result in a financial settlement after adjudication by a designated entity like a court of law [3]. Here the trusted entity acts as an oracle, and the adjudication by the court acts as an attestation upon which a financial payment is made to the grieving party from the other one that is involved.

Pre-Scheduled Payments. Companies hire contractors for services and schedule payments upon different stages of project completion [4]. All payments are set at the beginning of the contract and a third party (i.e., oracle) attests the completion of a stage upon which the corresponding payment is made to the contractor. A similar scenario is that of a monthly subscription fee for Netflix or some news feed, where subscribers can schedule payments for the entire duration of the subscription in one shot. Payments can be made with an oracle attesting the start of a month as the event outcome. Other examples are monthly utility bills, salaries to employees or shipping.

Trading. Betting on events like a football match, trade prices, etc., are facilitated by oracles who attest such event outcomes [6]. Users can avail the services of such oracles and make bets with each other, without even requiring the oracles to learn any information about the users or their bet.

Beyond Payments. Apart from ObC payments, our special-purpose encryption scheme can be used for timed encryption of messages [20]. More specifically, users can encrypt messages which can only be decrypted when some consensus committee (oracles in our earlier examples) signs some fixed messages in the future. For instance, these messages could be encoding block numbers of a blockchain that ought to appear in the future. Recently, the DRAND network [9] implemented a timelock encryption based on similar ideas. Our solution can be used to generalize their scheme on different levels: (1) It allows the oracles to generate their keys independently and without interaction, (2) it adds verifiability for the encrypted message, and (3) it enables larger outcome spaces, allowing for more fine-grained timestamp values.

B. Our Contributions

Our contributions can be summarized as follows:

(1) New Cryptographic Primitive. We present a new cryptographic primitive, *verifiable witness encryption based on threshold signatures* (VweTS) (Section IV). On an intuitive level, VweTS allows a user (Alice) to encrypt signatures $(\sigma_1, \ldots, \sigma_M)$ on payment messages $(m_1, \ldots, m_M)$, computed under the public keys of different oracles (Olivias). Anyone (Bob) can recover the signature σ_i on m_i, if they possess ρ-many of valid signatures (attestations) on another message $\overline{m}_i$. Here, the ρ signatures on $\overline{m}_i$ are computed under the public keys of ρ different oracles. We require that VweTS satisfies *one-wayness*, which guarantees that Bob cannot recover Alice's signatures unless he collects enough ρ valid signatures from the oracles, and *verifiability*, which guarantees that Bob can efficiently verify if Alice's ciphertext c is well-formed, and consequently contains valid signatures $(\sigma_1, \ldots, \sigma_M)$. We also propose a formal model for ObC payments in the full version[32] and we show how VweTS are the cryptographic cornerstone to realize ObC payments.

(2) Practically Efficient Constructions. We give two practically efficient constructions for VweTS: In the first construction (Section IV-B), the signatures $(\sigma_1, \ldots, \sigma_M)$ that are encrypted are either Schnorr or ECDSA signatures, and in the second construction (Appendix B) the signatures that are encrypted are BLS signatures [13]. In both constructions, the oracle attestations are BLS signatures on outcome-encoded messages. Our constructions support a polynomial number of oracles (N) and a polynomial number of outcomes (M). The main ingredient for high practical efficiency of all our VweTS constructions is a new technique to batch cut-and-choose proofs of well-formedness of ciphertexts, inspired by the batching technique of Lindell and Riva [30], originally developed to optimize garbled circuit computations over many executions. We also present an amortized version of our VweTS constructions (Section IV-C), referred to as VweTS-*extension*,[3] that can efficiently support a large outcome space. Here the computationally intense work is performed only for a handful of instances (logarithmic in M) while the security for all other instances comes almost for free, with a minimal amount of work required by both parties.

[2]For example, when a car rental company blocks an amount on a customer's credit card as a security deposit.

[3]This terminology is similar in spirit to the terminology used to refer to the amortization of *Oblivious Transfer (OT)* as *OT-extension* [25].

(3) Implementation. We provide a prototype implementation (Section V) of VweTS and evaluate how the running time and the communication overhead is affected by the system parameters such as the oracle threshold setting and the number of possible outcomes of an event. Our evaluation shows that, for a security parameter of 128 bits, our construction imposes a computation overhead less than 25 seconds even for a threshold of 4 out of 7 and a payment conditioned on up to 1024 different real-world event outcomes, while the communication overhead is below 2.3 MB. Moreover, our approach scales better with the number of oracles and outcomes compared to current solutions and is practical to be executed even in commodity hardware.

C. *ObC Payments:* VweTS *vs. Smart Contracts*

Many blockchain-related applications of ObC payments can alternatively be realized using smart contracts [37], [31], [27] or scripts specific to cryptocurrencies. Compared to the *cryptographic* variant that we study in this paper, a smart contract-based solution suffers from several drawbacks: (1) it is tailored to the characteristics offered by a *restricted* set of currencies (e.g., those supporting Turing-complete scripting languages); (2) it hinders *scalability* since the complete event-outcome information as well as attestation data is stored on the blockchain; (3) it hampers *fungibility* since tokens involved in such an oracle-based smart contract are trivially distinguishable from the ones of other contracts by a blockchain observer; and (4) finally, it also results in high on-chain costs as attestation data needs to be stored and interpreted, which requires high transaction fees or gas cost in case of Ethereum.

In contrast, VweTS solves all of the above issues in the case of ObC payments over blockchains. To set up the payment, Alice will transfer her coins into a shared address with Bob (e.g., a 2-out-of-2 multisig address, or an address whose secret key is shared between Alice and Bob), such that after time T, Alice can refund the coins (using a refund transaction). Compared to the smart contract approach, here the operational logic of attestation-based payment is encoded in the cryptographic operations of VweTS and *no* information is leaked or required to be stored on the blockchain. Using VweTS, Alice and Bob can start a ObC payment, such that the oracles can now either publish their attestations onto any public bulletin board (or the internet) or communicate the attestations privately to the users. The payment messages $(m_1, \ldots, m_M)$ are now cryptocurrency transactions, spending coins from the shared address between Alice and Bob, to an address of Bob. Importantly, all the communication and computation between Alice and Bob also happens off the chain. Given enough attestations on the outcome $\overline{m}_i$, Bob can obtain a signature σ_i on the transaction m_i, and publish the transaction and the signature on the blockchain. The blockchain is only ever involved in verifying the signature σ_i on the transaction m_i. Notice that we are able to bring real-world information onto the blockchain without actually recording or needing to interpret it on the blockchain.

Since no information about the ObC payment is recorded on the blockchain except the transaction m_i and the signature σ_i, our VweTS-based solution improves scalability. Moreover, given its simple signature verification on a regular looking transaction, we get better fungibility (compared to using smart contracts) and low on-chain cost. Finally, our VweTS constructions support the encryption of Schnorr, ECDSA or BLS signatures which are the payment signatures used in most major cryptocurrencies today.

D. *ObC Payments:* VweTS *vs DLC*

A somewhat different approach was initiated by the name of Discreet Log Contracts (DLC) [21], [29] and put forward by the Bitcoin community [28]. A DLC is a Bitcoin-compatible oracle contract enabling transactions from Alice to Bob to be contingent on signatures published by Olivia. DLC is promising because (1) it requires ECDSA or Schnorr signature verification and a timelock functionality from the underlying blockchain, which is available in many cryptocurrencies today; (2) it requires storing on the blockchain only a signed transaction from Alice to Bob (not even the signed message from Olivia), thereby minimizing the on-chain overhead (fee cost), and helping to preserve the fungibility of the cryptocurrency.

It is indeed the case that for a small number of event outcomes (see Figure 8), the DLC approach performs better in terms of running time than our VweTS. However, this approach comes with a number of shortcomings:

1) DLC does not support distributed trust among the oracles efficiently. In the multi-oracle setting, if we require t-of-N oracles to attest for Bob to claim the funds, neither Alice nor Bob know in advance which t-oracles are going to sign correctly. This implies that Alice has to send $\binom{N}{t}$ encrypted values, one for every combination of t out of the N oracles, leading to an exponential blowup.
2) Oracle attestation in DLC is strongly tied to the digital signature scheme used by Alice and Bob. Additionally, this digital signature scheme must be compatible with adaptor signatures. Hence, the protocol in [21], [29] cannot be used in cryptocurrencies such as the Chia network [2], where payments are authorized using a deterministic signature scheme (i.e., BLS) due to the impossibility result of [22].
3) DLC requires synchrony between the oracle and Alice, where the oracle has to announce some secret value periodically which Alice later uses in her promises to Bob.
4) There is no formal security analysis of DLC and in fact a number of attacks have been subsequently discovered [34]. These attacks can be roughly grouped into two families:
   - *Rogue key attacks*: The receiver of a conditional payment may choose their public key such that it cancels out the attestation of the oracle. As a result, the receiver could claim the payment without an attestation from the oracle. Similarly, in the multi-oracle setting, an oracle can choose its secret value for an event (as DLC requires synchrony) to cancel out attestations of another oracle. The malicious oracle in this case could produce an attestation that appear as if it had come from both oracles.
   - *Mix-and-match attack*: Here, instead of a designated combination of oracles and their attestations on different outcomes, the adversary can obtain the payment using

a different combination of outcome attestations. For example, Alice wants to pay Bob if and only if the first oracle attests $\overline{m}_1$ and the second oracle attests $\overline{m}_2$. The mix-and-match attack on DLC would allow Bob to obtain the payment even if the first oracle attests $\overline{m}_2$ and the second attests $\overline{m}_1$.

On the other hand, our VweTS-based solution is based on rigorous formalization of security and supports distributed trust in the threshold setting with many independent oracles efficiently. In our framework, there is no communication between the oracles and Alice prior to her promises to Bob, allowing Alice to make pre-scheduled payment promises to Bob (this is not possible in [21] due to the synchrony requirement). The oracle attestation in our VweTS is independent of the signature scheme used for the payment between Alice and Bob. This makes our solution more versatile to be used on different cryptocurrencies.

E. Other Related Work

Zhang et al. [38] proposed an approach for oracle contracts for data provenance where the functionality of the oracle (Olivia) is executed within a trusted execution environment (TEE). However, this approach again relies on smart contracts and requires trust assumption on the TEE which is unclear if it holds in practise [17], [14]. Zhang et al.[39] also present DECO, where they replace the TEE assumption with decentralized oracles while relying on smart contracts.

Eskandari et al. [23] present a systematization of knowledge of the oracle problem, where they present a modular workflow for the oracle system. They show the different phases of an oracle, from getting the ground truth to presenting the truth to a requester. The work does not propose a new concrete ObC payment problem and therefore is orthogonal to our work.

Witness Encryption Schemes. In terms of cryptographic work, concurrent to this work, Döttling et al. [20] proposed a witness encryption scheme for threshold signatures which is similar in functionality to VweTS, although in a completely different context. Their main application is to leverage the blockchain to do timed encryption, where if the blockchain reaches a certain height and a committee of validators attests a block, a ciphertext can be decrypted. In contrast to ours, their work is not concerned about the structure of the encrypted message. The technical crux of our paper is to efficiently prove the structure of the encrypted message (specifically, that it consists of a valid signature on a given message), for which we rely on new batching techniques for cut-and-choose.

The functionality of VweTS is also close to the functionality of *multi-authority attribute-based encryption (ABE)* [16]. Here a user can encrypt a message that can be decrypted only if the decryptor has attributes from enough number of authorities. We can think of the attributes as attestations from the oracles in our VweTS. Making use of multi-authority ABE potentially generalizes the attestation mechanism, which we leave as an interesting open problem. However, we wish to note that the verifiability aspect of VweTS is not covered even if we just use a multi-authority ABE. It is quite likely that to add verifiability to the above approach, we will have to make use of the techniques we introduce in this paper.

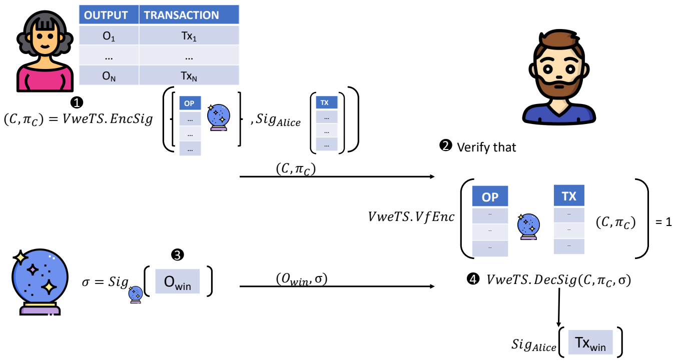


Fig. 1. Overview: ❶ Alice computes a signature on each transaction that corresponds to a different output. These transactions are then encrypted using a verifiable witness encryption, where the witness is a signature on the corresponding outcome. Alice sends these ciphertexts to Bob. ❷ Bob verifies that the encryption is computed correctly. ❸ An oracle provides a signature on the winning outcome to Bob ❹ Bob decrypts the corresponding ciphertext to get the signed transaction for the corresponding outcome.

II. Technical Overview

To establish some intuition for our solution, we describe our cryptographic techniques step by step with the goal of building oracle-based conditional (ObC) payments. For this, we consider the setting where Alice, with a key pair (sk, vk) of a digital signature DS, wants to transfer v coins to Bob in a payment m, if a certain outcome (represented by the message $\overline{m}$) of a real world event is attested by Olivia, with a key pair $(\overline{sk}, \overline{vk})$ of a digital signature $\overline{\mathsf{DS}}$ (possibly different to DS). To keep things simple, we assume that Olivia is honest (we will remove this assumption later).

A Strawman Solution. One trivial solution (depicted in Figure 1) is to resort to the notion of witness encryption [24]: Alice can create a ciphertext that includes $\sigma \leftarrow \mathsf{Sign}(sk, m)$ and that can only be decrypted if Bob has a witness (i.e, $\overline{\sigma}$) of the NP statement:

$$\{\exists\ \overline{\sigma}\ s.t.\ \overline{\mathsf{Vf}}(\overline{vk}, \overline{m}, \overline{\sigma}) = 1\}$$

i.e., Bob knows a valid signature on $\overline{m}$. While this solution would work theoretically (i.e., it would prevent Bob from getting the v coins if Olivia does not attest $\overline{m}$), there are two main issues. First, general purpose constructions of witness encryption are prohibitively expensive [24]. Second, Bob needs to trust Alice that the ciphertext contains a valid signature σ. The central challenge of our work is to build an *efficient* protocol that guarantees *verifiability* of Alice's ciphertexts.

Efficient Witness Encryption for Signatures. Our first observation is that the Boneh-Franklin (BF) identity-based encryption [12] can be thought of as a witness encryption scheme for a particular language. Recall that a key for an identity id in the BF scheme consists of a group element $H(\mathsf{id})^s$, where s is the master secret key. Furthermore, anyone can encrypt with respect to id, in such a way that the ciphertext can only be decrypted using $H(\mathsf{id})^s$ as the secret key. We observe that this is exactly the same structure that BLS [13] signatures have! Substituting identities id with messages $\overline{m}$, we can now compute ciphertexts that can only be decrypted knowing a signature on $\overline{m}$ (which is exactly $H(\overline{m})^s$). This yields a very efficient witness encryption scheme for the language of interest, provided that $\overline{\mathsf{DS}}$ is instantiated using the BLS signature scheme. Recall however that our goal is to let Alice encrypt a signature σ on m using DS, in a verifiable manner. We discuss how to address this challenge next.

Encrypting Adaptor Signatures. To understand our solution, it is useful to recall the notion of an adaptor signature (AS) [10]. In brief, AS allows Alice to generate a pre-signature $\hat{\sigma}$ on m, which is a verifiable encryption of a signature σ wrt. an NP statement $\{Y \mid Y := g^y\}$ where y is referred to as the witness and g is the generator of a cyclic group $\mathbb{G}$. With this tool at hand, Alice can: (1) create a pre-signature $\hat{\sigma}$ on m using a statement Y previously agreed with Bob; (2) use the BF-based witness encryption scheme mentioned above to encrypt y into ciphertext c for the identity $(\overline{vk}, \overline{m})$; (3) send $\hat{\sigma}$ and c to Bob. As soon as Olivia attests the event $\overline{m}$ by publishing a BLS signature with her key $\overline{sk}$, Bob can use the signature to extract y from c, and then use y to extract σ from $\hat{\sigma}$.

Verifiable Witness Encryption. To achieve verifiability efficiently, we adopt ideas from the *cut-and-choose* technique used in the verifiable encryption scheme of Camenisch et al. [15]. In a nutshell, Alice computes a pre-signature on the message as before and instead of generating a single BF ciphertext (*BF-cipher*), she generates λ (security parameter) tuples (*BF-cipher*, *sym-cipher*). Each *BF-cipher* contains a BF ciphertext that encrypts a random integer r_i for the identity $(\overline{vk}, \overline{m})$. In other words, Alice uses the same BF-based witness encryption as explained before to encrypt a random integer, instead of the adaptor witness y. Each *sym-cipher* is set to $(s_i = r_i + y)$, where y is the witness for the statement Y of AS and r_i is the random integer encrypted in *BF-cipher* at index i. Also, for all i, Alice computes $R_i = g^{r_i}$. At this point, Alice sends the λ-many *BF-cipher*$_i$, the λ-many R_i and the statement Y of AS to Bob. Intuitively, in this step, Alice commits to her setup of the cut-and-choose.

After receiving this information, Bob randomly samples[4] $\lambda/2$ pairs, for which Alice exposes the corresponding values r_i and the random coins used to encrypt r_i in *BF-cipher*$_i$ to Bob. For the other non-selected $\lambda/2$ pairs, Alice sends *sym-cipher* to Bob. The key question left, is to understand why this information would convince Bob of the fact that he will be able to get the signature σ after Olivia attests $\overline{m}$. To see that, Bob checks:

- For all $i \in [\lambda/2]$ not selected by Bob, $g^{s_i} \overset{?}{=} g^{r_i} \cdot Y$, intuitively checking that all *sym-cipher* are encrypting the value y using the randomness r_i as symmetric key of the one-time pad;
- For all $j \in [\lambda/2]$ chosen by Bob, recompute the BF ciphertext of r_j with random coins and check if $g^{r_j} \overset{?}{=} R_j$.

[4]This will be made non-interactive applying the Fiat-Shamir transformation.

If all these checks pass, Bob is guaranteed that there exists at least one well-formed BF ciphertext among those $\lambda/2$ not opened by Alice: meaning that it encrypts r_k such that $s_k = r_k + y$ for some k. Thus, when Olivia attests $\overline{m}$, Bob can decrypt the k-th BF ciphertext to compute r_k, extract $y = s_k - r_k$ from it and then use it to get σ from the pre-signature $\hat{\sigma}$ following the adaptor signature scheme.

Distributing the Trust. At the beginning of this overview, we have made the simplifying assumption that Olivia is honest. In order to relax this assumption, we show how to distribute the task of attesting the event $\overline{m}$ among a set of N oracles, each of them with a key pair $(\overline{sk}_i, \overline{vk}_i)$. Moreover, the event $\overline{m}$ is attested only when at least a threshold ρ number of oracles have signed it with their respective signing keys. Importantly, the N oracles are not required to coordinate, nor to talk to each other. A naive solution to this problem would be as follows: before proceeding with the cut-and-choose, Alice creates shares of the adaptor witness y into $(y_1, \dots, y_N)$ via (t-of-N)-Shamir secret sharing and additionally reveals the values $(Y_1, \dots, Y_N)$ where $Y_i := g^{y_i}$ so that one can verify the correctness of the secrete sharing via Lagrange interpolation. Finally, Alice executes N instances of the protocol described above. While this approach is correct, the verifiability proof would be very inefficient in terms of computation and communication cost. To this end, we develop a new batching technique (inspired by the work of Lindell and Riva [30] in the context of garbled circuit), for amortizing the costs of the cut-and-choose.

Batching Cut-and-Choose. We proceed by recalling the high-level idea of the Lindell-Riva cut-and-choose technique, adapted to our settings. As before, we let Alice generate *BF-cipher* encrypting random integers, but this time we generate 2*NB* of such *BF-cipher*, where B is a statistical security parameter. Bob then asks Alice to "open" NB number of *BF-ciphers* like in the previous case, while the rest of the "unopened" *BF-ciphers* are randomly mapped into N buckets, where each bucket consists of B *BF-ciphers*. By randomly sampling the bucket assignment (in the protocol it is specified by Bob) we are guaranteed that, with overwhelming probability, for *all* buckets there exists at least one "well-formed" ciphertext among the unopened ones. Each bucket is assigned to an oracle public key $\overline{vk}_i$, and Bob can then use the corresponding signature to recover the witness share y_i and ultimately reconstruct the adaptor witness y, if enough oracles signed the event.

However, a crucial step we overlooked in the outline above is that we cannot know ahead of time which bucket a *BF-cipher* will be mapped to later in the cut-and-choose step. In fact, it is *necessary* for the soundness of the cut-and-choose batching that we do not know the random mapping during the ciphertext generation. Therefore, it is unclear how we generate each of the *BF-cipher*, since we do not know the verification key $\overline{vk}$ that we want to encrypt against. To tackle this issue, during *BF-cipher* generation, we generate each of 2*NB* *BF-ciphers* (denoted by $(c'_1, \dots, c'_{2NB})$) w.r.t. to a BLS signature on a random (public) instance message $\overline{m}^*$ and a random instance verification key $\overline{vk}^*$. The instances $\overline{m}^*$ and $\overline{vk}^*$ can even be fixed ahead of time for the entire session. We proceed exactly as described above with these ciphertexts, until the random bucket mapping. Once we map an "unopened" *BF-cipher* $c'_{i,j}$ to the i-th bucket, we generate another *BF-cipher* $c_{i,j}$ w.r.t. a BLS signature on the correct instance message $\overline{m}$ and instance verification key $\overline{vk}_i$ (corresponding to the i-th bucket), which also encrypts the value r_j. We attach a Non-Interactive Zero-Knowledge (NIZK) proof to verify that the two *BF-ciphers* are well-formed and encrypt the same message. Crucially, such a NIZK is a simple proof for discrete logarithm equality, provided that we use the same random coins in both $c'_{i,j}$ and $c_{i,j}$ (which was shown to not compromise the security of the encryption scheme [11]).

The whole procedure is then made non-interactive by using the Fiat-Shamir heuristic, i.e., Alice generates the *NB* indices to open and the random bucket mapping using a cryptographic hash function applied to values generated prior. This concludes the construction of the cryptographic primitive *verifiable witness encryption based on threshold signatures (*VweTS*)*.

Extensions. The protocol described above forms the backbone of our construction, but a number of additional steps are needed to make the protocol practical. For starters, we need to consider the case where Alice has M different messages $(\overline{m}_1, \ldots, \overline{m}_M)$ instead of just one (Section IV-B). This allows Alice to condition a transaction m_i that pays to Bob if the outcome $\overline{m}_i$ is attested, which allows us to consider more realistic settings with multiple outcomes. We also consider the case where the signature scheme for authorizing a transaction is *not* an adaptor signature. In Appendix B, we show how to construct a protocol only based on BLS signatures (which do not imply adaptor signatures [22]).

A major bottleneck towards the practicality of the above scheme is the linear dependency on the number of payment messages M, which severely limits the applicability of our scheme. To overcome this, we study the notion of VweTS-*extension* where the expensive protocol (involving cut-and-choose verification) is run only for a handful of instances, where the number is independent of M. Nevertheless, security extends to *all* M instances, using only lightweight operations on the remaining instances. This notion is analogous to OT extension [25] and achieves similar guarantees, although our techniques are substantially different. We refer the curious reader to Section IV-C for more details.

III. Preliminaries

We denote by $\lambda \in \mathbb{N}$ the security parameter and by $x \leftarrow \mathcal{A}(\mathsf{in}; r)$ the output of the algorithm $\mathcal{A}$ on input in using $r \leftarrow \{0,1\}^*$ as its randomness. We often omit this randomness and only mention it explicitly when required. The notation $[n]$ denotes a set $\{1, \ldots, n\}$ and $[i, j]$ denotes the set $\{i, i+1, \ldots, j\}$. We consider *probabilistic polynomial time* (PPT) machines as efficient algorithms.

Digital Signatures. A digital signature scheme DS, formally, has a key generation algorithm $\mathsf{KGen}(1^\lambda)$ that takes the security parameter 1^λ and outputs the verification/signing key pair (vk, sk), a signing algorithm $\mathsf{Sign}(sk, m)$ inputs a signing key and a message $m \in \{0,1\}^*$ and outputs a signature σ, and a verification algorithm $\mathsf{Vf}(vk, m, \sigma)$ outputs 1 if σ is a valid signature on m under the verification key vk, and outputs 0 otherwise. We require unforgeability, which guarantees that a PPT adversary cannot forge a fresh signature on a message of its choice under a given verification key while having access to a signing oracle.

Non-Interactive Zero Knowledge Proofs. Let $R : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ be a n NP-witness-relation with corresponding NP-language $\mathcal{L} := \{x | \ \exists w \text{ s.t. } R(x, w) = 1\}$. A non-interactive zero-knowledge proof (NIZK) [19] system for the relation R is initialized with a setup algorithm $\mathsf{Setup}(1^\lambda)$ that, on input the security parameter, outputs a common reference string crs and a trapdoor td. A prover can show the validity of a statement x with a witness w by invoking $\mathsf{Prove}(crs, x, w)$, which outputs a proof π. The proof π can be efficiently checked by the verification algorithm $\mathsf{Vf}(crs, x, \pi)$. We require a NIZK system to be (1) *zero-knowledge*, where the verifier does not learn more than the validity of the statement x, and (2) *simulation sound*, where it is hard for any prover to convince a verifier of an invalid statement (chosen by the prover) even after having access to polynomially many simulated proofs for statements of his choosing.

Threshold Secret Sharing. Secret sharing is a method of creating shares of a given secret and later reconstructing the secret itself only if given a threshold number of shares. Shamir [36] proposed a threshold secret sharing scheme where the sharing algorithm takes a secret $s \in \mathbb{Z}_q$ and generates shares $(s_1, \ldots, s_n)$ each belonging to $\mathbb{Z}_q$. The reconstruction algorithm takes as input at least t shares and outputs the secret s via polynomial interpolation. The security of the secret sharing scheme demands that knowing only a set of $t-1$ shares does *not* leak any information about the choice of the secret s.

Hard Relations. We recall the notion of a hard relation R with statement/witness pairs (Y, y). We denote by $\mathcal{L}_R$ the associated language defined as $\mathcal{L}_R := \{Y | \ \exists y \ s.t. \ (Y, y) \in R\}$. The relation is called a hard relation if the following holds: (i) There exists a PPT sampling algorithm $\mathsf{GenR}(1^\lambda)$ that outputs a statement/witness pair $(Y, y) \in R$; (ii) The relation is poly-time decidable; (iii) For all PPT adversaries $\mathcal{A}$ the probability of $\mathcal{A}$ on input Y outputting a witness y is negligible.

Adaptor Signatures. Adaptor signatures [10] let users generate a pre-signature on a message m which by itself is not a valid signature, but can later be adapted into a valid signature using knowledge of some secret value. The formal definition of adaptor signatures is given below.

Definition 1 (Adaptor Signatures): An adaptor signature scheme AS w.r.t. a hard relation R and a signature scheme $\mathsf{DS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ consists of algorithms $(\mathsf{pSign}, \mathsf{Adapt}, \mathsf{pVf}, \mathsf{Ext})$ defined as:

- $\hat{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$: the pre-sign algorithm takes as input a signing key sk, message $m \in \{0,1\}^*$ and statement $Y \in L_R$, outputs a pre-signature $\hat{\sigma}$.
- $0/1 \leftarrow \mathsf{pVf}(vk, m, Y, \hat{\sigma})$: the pre-verify algorithm takes as input a verification key vk, message $m \in \{0,1\}^*$, statement

$Y \in L_R$ and pre-signature $\hat{\sigma}$, outputs either 1 (for valid) or 0 (for invalid).

- $\sigma \leftarrow \mathsf{Adapt}(\hat{\sigma}, y)$: the adapt algorithm takes as input a pre-signature $\hat{\sigma}$ and witness y, outputs a signature σ.
- $y \leftarrow \mathsf{Ext}(\sigma, \hat{\sigma}, Y)$: the extract algorithm takes as input a signature σ, pre-signature $\hat{\sigma}$ and statement $Y \in L_R$, outputs a witness y such that $(Y, y) \in R$, or $\perp$.

In addition to the standard signature correctness, an adaptor signature scheme has to satisfy *pre-signature correctness*. Informally, an honestly generated pre-signature w.r.t. a statement $Y \in L_R$ is a valid pre-signature and can be adapted into a valid signature from which a witness for Y can be extracted.

In terms of security, we want standard unforgeability even when the adversary is given access to pre-signatures with respect to the signing key sk. We also require that, given a pre-signature and a witness for the instance, one can always adapt the pre-signature into a valid signature (*pre-signature adaptability*). Finally, we require that, given a valid pre-signature and a signature with respect to the same instance, one can efficiently extract the corresponding witness (*witness extractability*). We refer the reader to Appendix A for the formal definitions of the properties of interest for adaptor signatures.

Witness Encryption Based on Signatures. Here we consider a special witness encryption scheme for a language $\mathcal{L} \in \mathsf{NP}$ defined with respect to a digital signature scheme $\mathsf{DS} := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$, where

$$\mathcal{L} := \{(vk, m) | \ \exists \sigma, \ s.t. \ , \mathsf{Vf}(vk, m, \sigma) = 1\}$$

where $(vk, sk) \in \mathsf{KGen}(1^\lambda)$. Here the verification key and the message (vk, m) is the instance and the signature σ is the witness. Informally, the relation states that there exist a signature σ such that σ is a signature on the message m and can be verified under the verification key vk.

We present below the formal definition of the witness encryption based on signatures scheme, its correctness, as well as its notion of security.

Definition 2 (Witness Encryption Based on Signatures): A *witness encryption scheme based on signatures (*WES*)* is a cryptographic primitive defined with respect to a digital signature scheme $\mathsf{DS} := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$, consisting of two PPT algorithms $(\mathsf{Enc}, \mathsf{Dec})$, defined below:

- $c \leftarrow \mathsf{Enc}((\tilde{vk}, \tilde{m}), m)$: the encryption algorithm takes as input a verification key $\tilde{vk}$ of the signature scheme, a message $\tilde{m}$ and the message to be encrypted m. It outputs a ciphertext c.
- $m \leftarrow \mathsf{Dec}(\tilde{\sigma}, c)$: the decryption algorithm takes as input a signature $\tilde{\sigma}$ and the ciphertext c. It outputs a message m.

Correctness is defined below.

Definition 3 (Correctness): A witness encryption scheme for signatures denoted by $\mathsf{WES} := (\mathsf{Enc}, \mathsf{Dec})$ defined with respect to a signature scheme $\mathsf{DS} := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ is said to be correct if for all $\lambda \in \mathbb{N}$, all $(\tilde{vk}, \tilde{sk}) \leftarrow \mathsf{KGen}(\lambda)$, all messages $\tilde{m}$ and m, all $c \leftarrow \mathsf{Enc}((\tilde{vk}, \tilde{m}), m)$, we have that $\Pr[\mathsf{Dec}(\tilde{\sigma}, c) = m] = 1$ where $\mathsf{Vf}(\tilde{vk}, \tilde{m}, \tilde{\sigma}) = 1$.

```
$\mathsf{IND\text{-}CPA}_{\mathsf{WES},\mathsf{DS},\mathcal{A}}(\lambda)$
    $Q := \emptyset$
    $(\tilde{vk}, \tilde{sk}) \leftarrow \mathsf{KGen}(\lambda)$
    $(\tilde{m}^*, m_0, m_1, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{Sign}\mathcal{O}}(\tilde{vk})$
    $b \leftarrow \{0, 1\}$
    $c_b \leftarrow \mathsf{Enc}((\tilde{vk}, \tilde{m}^*), m_b)$
    $b' \leftarrow \mathcal{A}^{\mathsf{Sign}\mathcal{O}}(\mathsf{st}_0, c_b)$
    $b_0 := (b = b')$
    $b_1 := (\tilde{m}^* \notin Q)$
    return $b_0 \wedge b_1$

$\mathsf{Sign}\mathcal{O}(\tilde{sk}, \tilde{m})$
    $\tilde{\sigma} \leftarrow \mathsf{Sign}(\tilde{sk}, \tilde{m})$
    $Q := Q \cup \{\tilde{m}\}$
    return $\tilde{\sigma}$
```

Fig. 2. Experiment for CPA security of a witness encryption scheme based on signatures.

The notion of security we want is similar to the chosen plaintext security of a standard public key encryption, except now the adversary has access to a signing oracle with key $\tilde{sk}$ while not being allowed to query the oracle on the message $\tilde{m}^*$, where the instance $(\tilde{vk}, \tilde{m}^*)$ is used to encrypt the challenge ciphertext. The reader familiar with the standard notion of security for witness encryption (which requires security only for false statements) will notice that our definition is stronger, although tailored for our specific language.

Definition 4 (Security): A witness encryption scheme for signatures denoted by $\mathsf{WES} := (\mathsf{Enc}, \mathsf{Dec})$ defined with respect to a signature scheme $\mathsf{DS} := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ is said to be *chosen plaintext attack* secure if for all $\lambda \in \mathbb{N}$, there exists a negligible function $\mathsf{negl}(\lambda)$, such that for all PPT adversaries $\mathcal{A}$, the following holds,

$$\Pr[\mathsf{IND\text{-}CPA}_{\mathsf{WES},\mathsf{DS},\mathcal{A}}(\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathsf{IND\text{-}CPA}$ is defined in Figure 2.

We give a construction for WES based on the BLS signature scheme. Our construction described in Figure 3 relies on efficiently computable bilinear pairings. We have the bilinear pairing operation e defined as $e : \mathbb{G}_0 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ where $\mathbb{G}_0, \mathbb{G}_1$ and $\mathbb{G}_T$ are groups of prime order q. We let g_0 and g_1 be the generators of $\mathbb{G}_0$ and $\mathbb{G}_1$ respectively and H_0, H_1 be a hash functions defined as $H_0 : \{0, 1\}^\lambda \rightarrow \mathbb{G}_1$ and $H_1 : \mathbb{G}_T \rightarrow \{0, 1\}^\lambda$.

The security of the construction follows similar to the IBE scheme from [12] based on Bilinear Diffie-Hellman assumption, when modelling the hash functions H_0, H_1 as random oracles.

IV. Verifiable Witness Encryption Based on Threshold Signatures

Consider the following language $\mathcal{L} \in \mathsf{NP}$ defined with respect to a signature scheme $\overline{\mathsf{DS}} := (\overline{\mathsf{KGen}}, \overline{\mathsf{Sign}}, \overline{\mathsf{Vf}})$, where $\mathcal{L}$ is defined as

$$\left\{ ((\overline{vk}_i)_{i \in [N]}, (\overline{m}_j)_{j \in [M]}, \rho) \middle| \begin{array}{c} \exists j \in [M], (\overline{\sigma}_i)_{i \in K \subset [N]}, \ s.t. \ , \\ |K| = \rho \ \wedge \\ \forall i \in K, \overline{\mathsf{Vf}}(\overline{vk}_i, \overline{m}_j, \overline{\sigma}_i) = 1 \end{array} \right\}$$

$\underline{\mathsf{Enc}((\tilde{vk},\tilde{m}),m)}$: The encryption algorithm proceeds as follows:

- Sample $r_1 \leftarrow \mathbb{Z}_q$ and $r_2 \leftarrow \mathbb{G}_T$.
- Set $c_1 := g_0^{r_1}$
- Compute $h := H_1(r_2)$.
- Compute $c_2 := (e(\tilde{vk}, H_0(\tilde{m}))^{r_1} \cdot r_2)$ and $c_3 := (h + m)$
- Return $c := (c_1, c_2, c_3)$.

$\underline{\mathsf{Dec}(\tilde{\sigma}, c)}$: The decryption algorithm proceeds as follows:

- Parse $c := (c_1, c_2, c_3)$.
- Compute $r := c_2 \cdot e(c_1, \tilde{\sigma})^{-1}$.
- Compute $h := H_1(r)$.
- Return $m := c_3 - h$.

Fig. 3. Witness encryption based on BLS signatures

where $(\overline{vk}_1, \ldots, \overline{vk}_N) \in \mathsf{SUPP}(\overline{\mathsf{KGen}}(1^\lambda))$. Here, the instance includes the verification keys of the oracles and the messages space. The witness is ρ-many signatures on a message $\overline{m}_j$ (where $\overline{m}_j$ is in the message space of the oracles). Informally, the relation states that there exist ρ number of signatures on a message $\overline{m}_j$ such that each of these signatures verify under the corresponding verification key specified in the instance.

We present a new primitive which is a witness encryption scheme for the above language, where we additionally consider another signature scheme DS. Moreover, the "secret" message(s) being encrypted by the witness encryption are themselves signatures $(\sigma_1, \ldots, \sigma_M)$ on messages $(m_1, \ldots, m_M)$ verifiable under a verification key vk with respect to DS. Intuitively, the primitive lets us encrypt signatures $(\sigma_1, \ldots, \sigma_M)$ such that the signature σ_j can be obtained after decryption, provided one holds a witness to the language $\mathcal{L}$.

A. Definitions

Definition 5 (Verifiable Witness Encryption Based on Threshold Signatures): A *verifiable witness encryption based on threshold signatures* is a cryptographic primitive parameterized by $\rho, N, M \in \mathbb{N}$, and is defined with respect to signature schemes $\mathsf{DS} := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ and $\overline{\mathsf{DS}} := (\overline{\mathsf{KGen}}, \overline{\mathsf{Sign}}, \overline{\mathsf{Vf}})$. It consists of three PPT algorithms $(\mathsf{EncSig}, \mathsf{VfEnc}, \mathsf{DecSig})$, that are defined below.

- $(c, \pi_c) \leftarrow \mathsf{EncSig}(((\overline{vk}_i)_{i\in[N]}, (\overline{m}_j)_{j\in[M]}), sk, (m_j)_{j\in[M]})$: the signature encryption algorithm takes as input tuples of instance verification keys $(\overline{vk}_i)_{i\in[N]}$, instance messages $(\overline{m}_j)_{j\in[M]}$, and messages $(m_j)_{j\in[M]}$ and a signing key sk. It outputs a ciphertext c and a proof π_c.
- $0/1 \leftarrow \mathsf{VfEnc}(c, \pi_c, ((\overline{vk}_i)_{i\in[N]}, (\overline{m}_j, m_j)_{j\in[M]}, vk))$: the encryption verification algorithm takes as input a ciphertext c, a proof π_c, tuples of instance verification keys $(\overline{vk}_i)_{i\in[N]}$, instance messages $(\overline{m}_j)_{j\in[M]}$, and messages $(m_j)_{j\in[M]}$, and a verification key vk. It outputs 1 (for valid) if its a valid ciphertext and 0 (for invalid) otherwise.
- $\sigma \leftarrow \mathsf{DecSig}(j, \{\overline{\sigma}_i\}_{i\in K}, c, \pi_c)$: the signature decryption algorithm takes as input an index $j \in [M]$, witness signatures $\{\overline{\sigma}_i\}_{i\in K}$ for $|K| = \rho$ and $K \subset [N]$, a ciphertext c, and proof π_c. It outputs a signature σ.

We define below the notion of correctness.

Definition 6 (Correctness): A (ρ, N, M)-VweTS is said to be *correct* if for all $\lambda \in \mathbb{N}$, all $(\overline{vk}_1, \ldots, \overline{vk}_N) \in \mathsf{SUPP}(\overline{\mathsf{KGen}}(\lambda))$, all $(vk, sk) \in \mathsf{KGen}(\lambda)$, all messages $(\overline{m}_j, m_j)_{j\in[M]}$, all (c, π_c) obtained by running EncSig algorithm on respective inputs, it holds that:

1) $\Pr\left[\mathsf{VfEnc}(c, \pi_c, ((\overline{vk}_i)_{i\in[N]}, (\overline{m}_j, m_j)_{j\in[M]}, vk)) = 1\right] = 1$.
2) For any $j \in [M], K \subset [N]$ and $|K| = \rho$, if for all $i \in K$ we have $\overline{\mathsf{Vf}}(\overline{vk}_i, \overline{m}_j, \overline{\sigma}_i) = 1$, then

$$\Pr[\mathsf{Vf}\,(vk, m_j, \mathsf{DecSig}(j, \{\overline{\sigma}_i\}_{i\in K}, c, \pi_c)) = 1] = 1.$$

One-Wayness. We require a notion called *one-wayness* for a VweTS scheme. Intuitively, the property guarantees that an adversary cannot output a valid signature σ^* for an index j^* encrypted in a VweTS ciphertext without access to ρ number of valid witness signatures on the corresponding instance message $\overline{m}_{j^*}$. The adversary is allowed to choose the signing keys of $\rho - 1$ number of instance verification keys of its choice, and is also given access to signing oracles conditioned on not allowing the adversary to trivially break the scheme. That is, the adversary cannot query the oracles for a signature on m_{j^*} wrt. the signing key sk and cannot query for a witness signature on the instance message $\overline{m}_{j^*}$. The intuition is captured formally in the following definition.

Definition 7 (One-Wayness): A (ρ, N, M)-VweTS is *one-way* if for all $\lambda \in \mathbb{N}$, there exists a negligible function $\mathsf{negl}(\lambda)$, such that for all PPT adversaries $\mathcal{A}$, the following holds:

$$\Pr\left[\mathsf{ExpOWay}^{\rho,N}_{\mathsf{VweTS},\mathsf{DS},\overline{\mathsf{DS}},\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

where ExpOWay is defined in Figure 4.

Verifiability. We require another notion of security called *verifiability* for a VweTS scheme. This property guarantees that it is infeasible for an adversary to output a ciphertext c along with a valid proof π_c, and valid witness signatures $(\overline{\sigma}_j)_{j\in K}$ on the instance message $\overline{m}_{j^*}$, such that the signature σ we get after decryption is in fact an invalid signature on the message m_{j^*} under the verification key vk. The intuition is formally captured in Definition 8.

Definition 8 (Verifiability): A (ρ, N, M)-VweTS is said to be *verifiable* if, for all $\lambda \in \mathbb{N}$, there exists a negligible function negl and no PPT adversary $\mathcal{A}$ that outputs $((m_j, \overline{m}_j)_{j\in[M]}, vk, (\overline{vk}_i)_{i\in[N]}, (\overline{\sigma}_j)_{j\in K}, j^*, c, \pi_c)$ *s.t.* all the following holds simultaneously with probability $\mathsf{negl}(\lambda)$:

- $K \subset [N]$ and $|K| = \rho$
- $(vk, \cdot) \in \mathsf{SUPP}(\mathsf{KGen})$ and for all $i \in [N]$ we have $(\overline{vk}_i, \cdot) \in \mathsf{SUPP}(\overline{\mathsf{KGen}})$ where SUPP denotes the support.
- $\forall j \in K, \overline{\mathsf{Vf}}(\overline{vk}_j, \overline{m}_{j^*}, \overline{\sigma}_j) = 1$
- $\mathsf{VfEnc}(c, \pi_c, ((\overline{vk}_i)_{i\in[N]}, (\overline{m}_j, m_j)_{j\in[M]}, vk)) = 1$
- $\mathsf{Vf}(vk, m_{j^*}, \sigma) = 0$, where $\sigma \leftarrow \mathsf{DecSig}(j^*, \{\overline{\sigma}_j\}_{j\in K}, c, \pi_c)$

```
$\mathsf{ExpOWay}^{\rho,N,M}_{\mathsf{VweTS},\mathsf{DS},\overline{\mathsf{DS}},\mathcal{A}}(\lambda)$
$Q_1 := Q_2 := \emptyset,\ Q_3 := []$
$(vk, sk) \leftarrow \mathsf{KGen}(1^\lambda)$
$(C, \mathsf{st}_0) \leftarrow \mathcal{A}(vk)$    $/\!\!/$ let $C \subset [N]$
$\forall i \in [N] \setminus C, (\overline{vk}_i, \overline{sk}_i) \leftarrow \overline{\mathsf{KGen}}(1^\lambda)$
$(q^*, \sigma^*, j^*) \leftarrow \mathcal{A}^{\overline{\mathsf{Sign}}\mathcal{O}, \mathsf{Sign}\mathcal{O}, \mathsf{EncSig}\mathcal{O}}(\mathsf{st}_0, \{\overline{vk}_i\}_{i \in [N]\setminus C})$
$(c, \pi_c, X) \leftarrow Q_3[q^*]$
$X := ((\overline{vk}_i)_{i\in[N]}, (\overline{m}_j)_{j\in[M]}), sk, (m_j)_{j\in[M]}$
$b_0 := ((m_{j^*}, \sigma^*) \notin Q_2)$
$b_1 := (\overline{m}_{j^*} \notin Q_1)$
$b_2 := (|C| \leq \rho - 1)$
$b_3 := (\mathsf{Vf}(vk, m_{j^*}, \sigma^*) = 1)$
return $b_0 \wedge b_1 \wedge b_2 \wedge b_3$

$\mathsf{EncSig}\mathcal{O}((\overline{m}_j, m_j)_{j\in[M]}, \{\overline{vk}_i\}_{i\in C})$
$X := ((\overline{vk}_i)_{i\in[N]}, (\overline{m}_j)_{j\in[M]}), sk, (m_j)_{j\in[M]}$
$(c, \pi_c) \leftarrow \mathsf{EncSig}(X)$
$Q_3 := Q_3 || (c, \pi_c, X)$
return $(c, \pi_c)$

$\overline{\mathsf{Sign}}\mathcal{O}(i, \overline{m})$
Ensure $i \in [N] \setminus C$
$\overline{\sigma} \leftarrow \overline{\mathsf{Sign}}(\overline{sk}_i, \overline{m})$
$Q_1 := Q_1 \cup \{\overline{m}\}$
return $\overline{\sigma}$

$\mathsf{Sign}\mathcal{O}(m)$
$\sigma \leftarrow \mathsf{Sign}(sk, m)$
$Q_2 := Q_2 \cup \{m, \sigma\}$
return $\sigma$
```

Fig. 4. Experiment for one-wayness.

B. Construction Based on Adaptor Signatures

Here we present a concrete construction of VweTS with parameters ρ, N and M relying on the following cryptographic building blocks:

- Signature scheme $\overline{\mathsf{DS}} := (\overline{\mathsf{KGen}}, \overline{\mathsf{Sign}}, \overline{\mathsf{Vf}})$ instantiated with BLS signature scheme (see Appendix A).
- Signature scheme $\mathsf{DS} := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ that is either Schnorr or ECDSA signature schemes (see Appendix A), based on a group $\mathbb{G}$ with generator g and order q.
- Witness encryption based on signatures $\mathsf{WES} := (\mathsf{Enc}, \mathsf{Dec})$ scheme (see Figure 3 for a concrete candidate).
- An adaptor signature scheme $\mathsf{AS} := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ for the signature scheme DS. The hard relation R for AS is that of the discrete log relation, where the language is defined as: $\mathcal{L}_R := \{Y |\ \exists y \in \mathbb{Z}_q^*,\ \ s.t.\ Y = g^y\}$.
- A NIZK proof $(\mathsf{Setup}_{\mathcal{L}_c}, \mathsf{Prove}_{\mathcal{L}_c}, \mathsf{Vf}_{\mathcal{L}_c})$ for the language $\mathcal{L}_c$ defined as

$$\left\{ (\overline{vk}_1, \overline{vk}_2, \overline{m}_1, \overline{m}_2, c_1, c_2) \middle| \begin{array}{c} \exists r \in \mathbb{Z}_q,\ \ s.t. \\ c_1 = \mathsf{WES.Enc}((\overline{vk}_1, \overline{m}_1), r) \wedge \\ c_2 = \mathsf{WES.Enc}((\overline{vk}_2, \overline{m}_2), r) \end{array} \right\}$$

where $(\overline{vk}_1, \cdot)$ and $(\overline{vk}_2, \cdot)$ are in the support of $\overline{\mathsf{KGen}}$. Informally, the relation here states that there exists an r such that c_1 encrypts r under $(\overline{vk}_1, \overline{m}_1)$ and c_2 encrypts r under $(\overline{vk}_2, \overline{m}_2)$, where $c_1, c_2, \overline{vk}_1, \overline{m}_1, \overline{vk}_2, \overline{m}_2$ constitutes the instance and r is the witness.

In Table I we provide an overview of the parameters used in our construction.

Overview. We present a high level overview of our construction, and the formal description is given in Figure 5. The signature encryption algorithm EncSig does the following:

1) Compute a random verification key $\overline{vk}^*$ and a random message $\overline{m}^*$.
2) Generate γ number of WES ciphertexts such that ciphertext c'_i encrypts a random integer r_i from $\mathbb{Z}_q$ wrt. the instance $(\overline{vk}^*, \overline{m}^*)$. It also encodes the integer r_i in the exponent by setting $R_i := g^{r_i}$.
3) The algorithm generates for each $i \in [M]$ an adaptor pre-signature on the message m_i wrt. an adaptor instance Y_i whose corresponding witness is y_i. Each of the adaptor witness y_i is further secret shared to generate shares $y_{i,j}$ for $j \in [N]$, such that the sharing can be verified with the aid of the group elements $Y_{i,j} := g^{y_{i,j}}$.
4) The algorithm then sets $\Sigma_1 := (\hat{\sigma}_i, Y_i, \{Y_{i,j}\}_{j\in[N]})_{i\in[M]}$. Looking ahead, the decryption algorithm will be able to compute the witness y_j corresponding to $\hat{\sigma}_j$ and adapt it to compute a full signature σ_j.
5) A bucket mapping Φ and γ bit values are generated by applying the Fiat-Shamir transform using the hash H_2 to the values computed thus far.
6) Now the algorithm performs the cut-and-choose, such that for all indices $i \in [\gamma]$ where the bit value from the Fiat-Shamir transform equals 1, the value r_i and the random coins used to generate the i-th WES ciphertext are added in plain to the set $\mathcal{S}_{\mathsf{op}}$. These values are considered to be opened by the cut-and-choose. On the other hand, for all indices i where the bit value equals 0, the index i is mapped to the bucket (α, β) using the map Φ. A value s_i is set to be the one-time pad of the adaptor witness share $y_{\alpha,\beta}$ and the value r_i. A new WES ciphertext c_i is generated encrypting the same value r_i as the WES ciphertext c'_i, but now wrt. the instance $(\overline{vk}_\beta, \overline{m}_\alpha)$, along with a NIZK proof that the two WES ciphertexts c_i and c'_i encrypt the same value r_i. The value s_i, the ciphertext c_i and the associated

TABLE I
NOTATIONS USED IN OUR CONSTRUCTION OF Π_{VweTS}, WHERE THE PUBLIC PARAMETERS ARE $(\mathbb{G}, \mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_2, q, \gamma, \Phi, H_2)$

M	Number of possible outcomes
N	Number of oracles
B	A statistical parameter
γ	Public parameter computed as $\gamma = 2NMB$
$\Phi : [\gamma] \to [M] \times [N]$	Mapping from an index $i \in [\gamma]$ to a bucket $(\alpha, \beta) \in [M] \times [N]$
$H_2 : \{0,1\}^* \to (\Phi, \mathbf{b})$	Hash function that outputs the mapping as well a bit-string of length γ
$\mathcal{S}_{\mathsf{op}}$	Set of opened values output by DecSig
$\mathcal{S}_{\mathsf{unop}}$	Set of unopened values output by EncSig

```
**Public parameters**: $(\mathbb{G}, g, q, \mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_T, \gamma, H_2, crs)$
$(c, \pi_c) \leftarrow \mathsf{EncSig}(((\overline{vk}_i)_{i\in[N]}, (\overline{m}_j)_{j\in[M]}, \rho), sk, (m_j)_{j\in[M]})$:
1) Sample random $\overline{vk}^* \in \mathbb{G}_0$ and $\overline{m}^* \in \{0,1\}^\lambda$, initialize $\mathcal{S}_{\mathsf{op}} = \mathcal{S}_{\mathsf{unop}} = \emptyset$.
2) For $i \in [\gamma]$:
   a) Sample $r_i \leftarrow \mathbb{Z}_q$ and compute $R_i := g^{r_i}$.
   b) Compute $c'_i := \mathsf{WES.Enc}((\overline{vk}^*, \overline{m}^*), r_i; r'_i)$ where $r'_i$ is the random coins used.
3) For $i \in [M]$:
   a) Sample $y_i \leftarrow \mathbb{Z}_q$ and compute $Y_i := g^{y_i}$.
   b) Compute $\hat{\sigma}_i \leftarrow \mathsf{AS.pSign}(sk, m_i, Y_i)$.
   c) For all $j \in [\rho-1]$ sample a uniform $y_{i,j} \leftarrow \mathbb{Z}_q$ and set $Y_{i,j} := g^{y_{i,j}}$.
   d) For all $j \in \{\rho, \ldots, N\}$ compute
      $y_{i,j} = \left(\left(y_i - \sum_{k\in[\rho-1]} y_{i,k} \cdot \ell_k(0)\right) \cdot \ell_j(0)^{-1}\right)$, $Y_{i,j} = \left(\frac{Y_i}{\prod_{k\in[\rho-1]} Y_{i,k}^{\ell_k(0)}}\right)^{\ell_j(0)^{-1}}$. Here $\ell_i$ is the $i$-th Lagrange
      polynomial.
4) Set $\Sigma_1 := (\hat{\sigma}_i, Y_i, \{Y_{i,j}\}_{j\in[N]})_{i\in[M]}$.
5) Compute $\{\Phi, (b_1, \ldots, b_\gamma)\} := H_2((c'_i, R_i)_{i\in[\gamma]}, \Sigma_1)$.
6) For $i \in [\gamma]$:
   a) If $b_i = 1$, then $\mathcal{S}_{\mathsf{op}} := \mathcal{S}_{\mathsf{op}} \cup \{(i, r_i, r'_i)\}$.
   b) If $b_i = 0$:
      i) Let $(\alpha, \beta) := \Phi(i)$.
      ii) Compute $s_i := r_i + y_{\alpha,\beta}$.
      iii) Compute $c_i := \mathsf{WES.Enc}((\overline{vk}_\beta, \overline{m}_\alpha), r_i; r''_i)$ with $r''_i$ as the random coins and set
           $\pi_i \leftarrow \mathsf{Prove}_{\mathcal{L}_c}(crs, (\overline{vk}_\beta, \overline{vk}^*, \overline{m}_\alpha, \overline{m}^*, c_i, c'_i), r_i)$.
      iv) Set $\mathcal{S}_{\mathsf{unop}} := \mathcal{S}_{\mathsf{unop}} \cup \{(i, s_i, c_i, \pi_i)\}$.
7) Return $c = \{c'_i\}_{i\in[\gamma]}, \pi_c = \{\mathcal{S}_{\mathsf{op}}, \mathcal{S}_{\mathsf{unop}}, \overline{vk}^*, \overline{m}^*, \{R_i,\}_{i\in[\gamma]}, \Sigma_1\}$.
$0/1 \leftarrow \mathsf{VfEnc}(c, \pi_c, ((\overline{vk}_i)_{i\in[N]}, (\overline{m}_j, m_j)_{j\in[M]}, vk))$:
1) Parse $c$ as $\{c'_i\}_{i\in[\gamma]}$ and $\pi_c$ as $\{\mathcal{S}_{\mathsf{op}}, \mathcal{S}_{\mathsf{unop}}, \overline{vk}^*, \overline{m}^*, \{R_i,\}_{i\in[\gamma]}, \Sigma_1\}$ where $\Sigma_1 := \{\hat{\sigma}_i, Y_i, \{Y_{i,j}\}_{j\in[N]}\}_{i\in[M]}$.
2) Compute $\{\Phi, (b_1, \ldots, b_\gamma)\} := H_2((c'_i, R_i)_{i\in[\gamma]}, \Sigma_1)$
3) For $i \in [\gamma]$:
   a) If $b_i = 1$, check that $(i, r_i, r'_i) \in \mathcal{S}_{\mathsf{op}}$ and that $c'_i := \mathsf{WES.Enc}((\overline{vk}^*, \overline{m}^*), r_i; r'_i)$
   b) If $b_i = 0$:
      i) $(\alpha, \beta) := \Phi(i)$
      ii) Check that $(i, s_i, c_i, \pi_i) \in \mathcal{S}_{\mathsf{unop}}$
      iii) Check that $g^{s_i} = R_i \cdot Y_{\alpha,\beta}$
      iv) Check $\mathsf{Vf}_{\mathcal{L}_c}(crs, (\overline{vk}_\beta, \overline{vk}^*, \overline{m}_\alpha, \overline{m}^*, c_i, c'_i), \pi) = 1$
      v) Check that $\mathsf{AS.pVf}(vk, m_\alpha, Y_\alpha, \hat{\sigma}_\alpha) = 1$
      vi) Let $T$ be a subset of $[N]$ of size $\rho - 1$, check that for every $k \in [N] \setminus T$: $\prod_{j\in T} Y_{\alpha,j}^{\ell_j(0)} \cdot Y_{\alpha,k}^{\ell_k(0)} = Y_\alpha$.
   c) If any of the checks fail output 0, else output 1.
$\sigma \leftarrow \mathsf{DecSig}(j, \{\overline{\sigma}_i\}_{i\in[K]}, c, \pi_c)$:
1) Parse $c$ as $\{c'_i\}_{i\in[\gamma]}$ and $\pi_c$ as $\{\mathcal{S}_{\mathsf{op}}, \mathcal{S}_{\mathsf{unop}}, \overline{vk}^*, \overline{m}^*, \{R_i,\}_{i\in[\gamma]}, \Sigma_1\}$ where $\Sigma_1 := \{\hat{\sigma}_i, Y_i, \{Y_{i,j}\}_{j\in[N]}\}_{i\in[M]}$.
2) For all $i \in [K]$, initialize $\mathsf{rShare}_i = \emptyset$ and compute $\{\Phi, (b_1, \ldots, b_\gamma)\} := H_2((c'_i, R_i)_{i\in[\gamma]}, \Sigma_1)$
3) For each $(i, s, c, \pi) \in \mathcal{S}_{\mathsf{unop}}$, compute $(\alpha, \beta) = \Phi(i)$. If $\alpha = j$ and if $\beta \in [K]$ s.t. $\mathsf{DS.Vf}(\overline{vk}_\beta, \overline{m}_\alpha, \overline{\sigma}_i) = 1)$
   a) Compute $r = \mathsf{WES.Dec}(\overline{\sigma}_i, c)$.
   b) Set $\mathsf{rShare}_\beta := \mathsf{rShare}_\beta \cup \{r\}$.
4) Denote each $r$ in $\mathsf{rShare}_i$ as $r_{i,a}$, where $(a, s_a, c_a, \pi_a) \in \mathcal{S}_{\mathsf{unop}}$. We are guaranteed that there exists at least one $r_{i,a}$
   such that $R_a = g^{r_{i,a}}$.
5) For $i \in [K]$, compute $y_{j,i} = s_a - r_{i,a}$.
6) Compute $y_j := \sum_{i\in[K]} y_{j,i} \cdot \ell_i(0)$.
7) Return $\sigma_j \leftarrow \mathsf{AS.Adapt}(\hat{\sigma}_j, y_j)$.
```

Fig. 5. VWeTS from adaptor signatures.

NIZK proof are added to the set $\mathcal{S}_{\mathsf{unop}}$. These values are considered to be unopened by the cut-and-choose.

7) The algorithm outputs all the WES ciphertexts, the two sets $\mathcal{S}_{\mathsf{op}}$ and $\mathcal{S}_{\mathsf{unop}}$, the instance $(\overline{vk}^*, \overline{m}^*)$, the group elements R_i and the adaptor instances along with the group elements for verifying the witness sharing.

To verify, the algorithm VfEnc does the following:

1) Retrieve the inputs of the Fiat-Shamir hash function from π_c.
2) Compute the hash function and check the correctness of the Fiat-Shamir transformation.
3) Check the well-formedness of the opened values in $\mathcal{S}_{\mathsf{op}}$ against the WES ciphertexts generated wrt. instance $(\overline{vk}^*, \overline{m}^*)$.
4) Check the unopened values in $\mathcal{S}_{\mathsf{unop}}$ by applying the mapping Φ for the corresponding index i and checking if the one-time pad of the value s_i is consistent by checking the relation in the exponent. It verifies the NIZK proofs and the pre-signatures against the corresponding adaptor instances. Finally, it checks if the adaptor witness sharing was performed correctly with Lagrange interpolation of the group elements $Y_{i,j}$ in the exponent.

The decryption algorithm DecSig does the following:

1) The algorithm parses as input the ciphertext c and the proof π_c as well as ρ valid witness signatures from the oracles on some instance message $\overline{m}_j$.
2) Corresponding to each signature $\overline{\sigma_i}$ the algorithm initializes a set rShare_i. Looking ahead, this set rShare_i will include the random r that were encrypted under $\overline{vk}_i$ and $\overline{m}_j$, where $\overline{\sigma_j}$ is a signature on $\overline{m}_j$. The algorithm also determines the mapping function Φ and the set of unopened and opened indices by evaluating the hash function H_2 on the corresponding indices.
3) For each index i in the unopened set $\mathcal{S}_{\mathsf{unop}}$, the decrypt algorithm DecSig first applies the bucket mapping Φ to obtain the bucket index (α, β). It proceeds to decrypt the ciphertext c_i using the i-th witness signature, provided the signature is valid on the instance message $\overline{m}_\alpha$ wrt. the instance verification key $\overline{vk}_\beta$ (where $\alpha = j$). The decrypted value r is added to a set rShare_β.
4) Notice that it is the case that for many $i' \neq i$ map to the same value β and therefore rShare_β will contain more than one element in it (more precisely, we will have $|\mathsf{rShare}_\beta| = B$). By the cut-and-choose, we are guaranteed that at least one of the values $r_{i,a} \in \mathsf{rShare}_i$ is consistent with the check $R_a = g^{r_{i,a}}$.
5) For each $i \in [K]$, where K stores the indices of the ρ valid witness signatures we have, we obtain the adaptor witness share $y_{j,i}$ using the consistent values $r_{i,a}$ from step 4.
6) We obtain ρ witness shares $y_{j,i}$ using which we can reconstruct the adaptor witness y_j.
7) The signature on the message m_j can now be easily output by adapting the j-th pre-signature using the witness y_j.

Analysis. In the full version [32], we formally show that our construction satisfies correctness according to Definition 6. Security of our construction is formally stated in the following theorem.

Theorem 1: Let DS and $\overline{\mathsf{DS}}$ be signature schemes that satisfy unforgeability, WES be a secure witness encryption based on signatures scheme, AS be a secure adaptor signature scheme for the signature scheme DS and $(\mathsf{Setup}_{\mathcal{L}_c}, \mathsf{Prove}_{\mathcal{L}_c}, \mathsf{Vf}_{\mathcal{L}_c})$ be NIZK proof system for the language $\mathcal{L}_c$ satisfying zero-knowledge and simulation soundness. Then the VweTS construction from Figure 5 is one-way and verifiable according to Definition 7 and Definition 8, respectively.

Proof 1: We first show that the protocol described in Figure 5 satisfies one-wayness as defined in Definition 7. To this end, we present a sequence of hybrids starting from the one-wayness experiment defined in Figure 4.

$\underline{\mathsf{Hyb}_0}$: This is the experiment defined in Figure 4.

$\underline{\mathsf{Hyb}_1}$: This hybrid is the same as Hyb_0 except that the challenger guesses q^* and j^* that are output by the adversary. For the oracle query $\mathsf{EncSig}\mathcal{O}$ corresponding to q^* the random oracle H_2 is simulated by lazy sampling. A random bit string $b_1, \ldots, b_\gamma$ and the mapping Φ is sampled and the output of the random oracle on the ciphertexts c'_i and R_i for $i \in [\gamma]$ and Σ_1 is set to $(\Phi, (b_1, \ldots, b_\gamma))$. The challenger guesses the query q^* correctly with probability $\frac{1}{|Q_3|}$.

$\underline{\mathsf{Hyb}_2}$: This hybrid is the same as Hyb_1 except that in the q^*-th query to the $\mathsf{EncSig}\mathcal{O}$ the zero knowledge proofs π_i are replaced by simulated zero knowledge proofs. By the zero knowledge property of the underlying NIZK scheme the two hybrids are indistinguishable.

$\underline{\mathsf{Hyb}_3}$: This hybrid is the same as Hyb_2, except that the ciphertexts c'_i for which $b_i = 1$ are replaced by ciphertexts of 0. By the IND-CPA security of the witness encryption scheme (Definition 2) the two hybrids are indistinguishable. Note that the adversary cannot know the witness σ which is a signature on a randomly sampled message $\overline{m}^*$ that can be verified by a randomly sampled key $\overline{vk}^*$. Since an adversary cannot efficiently compute sk^* from $\overline{vk}^*$ the adversary cannot compute a valid witness.

$\underline{\mathsf{Hyb}_4}$: This hybrid is the same as Hyb_3, except that the ciphertexts c_i which are encrypted under $\overline{vk}_\beta$ and $\overline{m}_\alpha$ such that $\beta \in [N] \setminus C$ and $\alpha = j^*$, are replaced by ciphertexts of 0. If $\overline{m}^*_j \in Q_1$, then abort. Note that since the experiment aborts if $\overline{m}^*_j \in Q_1$, the adversary cannot receive a valid witness (a signature on $\overline{m}^*_j$ under $\overline{vk}_\beta$) to decrypt the ciphertext c_i. By the IND-CPA security of the witness encryption scheme (Definition 2) the two hybrids are indistinguishable. Note that the challenger correctly guesses the message index j^* with probability $\frac{1}{|M|}$.

$\underline{\mathsf{Hyb}_5}$: This hybrid is the same as Hyb_4, except that $\hat{\sigma}^*_j$ is computed as $\hat{\sigma}^*_j = \mathsf{AS.pSign}(sk, m^*_j, Y^*_j)$ where $Y^*_j \leftarrow \mathbb{G}_0$. The shares of Y^*_j are computed by randomly sampling $Y_{j^*,k}$ for $k \in [1, \rho - 1]$. For $k \in [p, N]$, compute

$$Y_{j^*,k} = \left(\frac{Y^*_j}{\prod_{r \in [\rho-1]} Y_{j^*,r}^{\ell_r(0)}} \right)^{\ell_k(0)^{-1}}$$

where ℓ_i is the i-th Lagrange polynomial. The two hybrids are indistinguishable since the changes are syntactical and the distribution induced is identical in the two hybrids.

$\underline{\mathsf{Hyb}_6}$: This hybrid is the same as Hyb_5, except that for all i such that $\Phi(i) = (\alpha, \beta)$ where $\alpha = j^*$ and $\beta \in [N] \setminus C$ the variable s_i is randomly sampled as $s_i \leftarrow \mathbb{Z}_q$ and R_i is computed as $R_i = \frac{g^{s_i}}{Y_{\alpha,\beta}}$. The distribution of R_i and s_i are identical to the previous hybrid and therefore they are indistinguishable.

Now we show that one-wayness holds in Hyb_6. Consider an adversary $\mathcal{A}$ that wins the one-wayness experiment with non-negligible probability. We now describe another adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break the unforgeability of the adaptor signatures.

$\underline{\text{Adversary } \mathcal{B}}$:

1) Initialize $\mathcal{A}$ and simulate the experiment $\mathsf{ExpOWay}$.
2) While simulating $\mathsf{EncSig}\mathcal{O}$ for query q^* and message m^*, send m to the challenger.
3) Receive $\hat{\sigma}$ and Y from the challenger. Simulate the rest of the protocol as in Hyb_6 where Y is used instead of randomly sampling Y_j^* in computing $\hat{\sigma} = \mathsf{AS.pSign}(sk, m_j^*, Y)$.
4) Upon receiving any $\mathsf{Sign}\mathcal{O}$ calls forward the calls to the challenger and return the response to the adversary.
5) Upon receiving σ from $\mathcal{A}$, output σ to the challenger.

It is clear that

$$\begin{aligned}&\Pr\left[\mathsf{aSigForge}_{\mathcal{B},\mathsf{AS}}(\lambda)\right]\\ &= \frac{1}{|Q_3|}\frac{1}{|M|}\Pr\left[\mathsf{ExpOWay}^{\rho,N}_{\mathsf{VweTS},\mathsf{DS},\overline{\mathsf{DS}},\mathcal{A}}(\lambda) = 1\right].\end{aligned}$$

This implies that the success probability of the adversary is negligible, since we assume that the adaptor signature scheme is EUF-CMA secure and $|Q_3|$ and $|M|$ are polynomial in the security parameter λ. Thus, we prove one-wayness.

We now prove that the scheme is verifiable according to Definition 8. Assume that an adversary $\mathcal{A}$ breaks the verifiability of the protocol. This implies that the adversary outputs messages $(m_j, \overline{m}_j)_{j\in[M]}$ a verification key vk, oracle verification keys $(\overline{vk}_i)_{i\in[N]}$, oracle signatures on a message $\overline{m}_{j^*}$, $(\overline{\sigma}_j)_{j\in K}$ and outputs (c, π_c) of EncSig such that:

1) Each $\overline{\sigma}_j$ is a valid signature, i.e.,
$$\forall j \in K, \overline{\mathsf{Vf}}(\overline{vk}_j, \overline{m}_{j^*}, \overline{\sigma}_j) = 1.$$
2) The output of EncSig is valid, i.e.,
$$\mathsf{VfEnc}(c, \pi_c, ((\overline{vk}_i)_{i\in[N]}, (\overline{m}_j, m_j)_{j\in[M]}, vk)) = 1.$$
3) The final extracted signature does not verify, i.e., $\mathsf{Vf}(vk, m_{j^*}, \sigma) = 0$ where $\sigma \leftarrow \mathsf{DecSig}(j^*, \{\overline{\sigma}_j\}_{j\in K}, c, \pi_c)$.

Since we model the hash function as a random oracle, we can analyze the probability of this event happening in the interactive settings, by simulating the random oracle with lazy sampling.

We will now show that if the first and second conditions hold true, then DecSig will output a signature σ that verifies, except with negligible probability.

Recall that each $(\overline{m}_{j^*}, \overline{vk}_j)$ is assigned to a bucket $a = (j^*, j)$, and each bucket is associated with B-many ciphertexts $c_{a,1}, \ldots c_{a,B}$ that encrypt random values (denoted $r_{a,1}, \ldots, r_{a,B}$). Note that the DecSig algorithm decrypts these ciphertexts for each bucket a using $(\overline{\sigma}_j)$ to get the encrypted values $r_{a,1}, \ldots, r_{a,B}$.

Next, recall that since the VfEnc algorithm outputs 1, we are guaranteed that:

- $\mathsf{Vf}_{\mathcal{L}_c}(crs, (\overline{vk}_j, \overline{vk}^*, \overline{m}_{j^*}, \overline{m}^*, c_{a,k}, c'), \pi) = 1$. This implies that c_i and c_i' encrypt the same $r_{a,k}$ except with negligible probability. This is guaranteed by the soundness property of the underlying NIZK scheme.
- $g^{s_{a,k}} = R_{a,k}Y_a$ for $k \in [B]$ (where $R_{a,k} = g^{r_{a,k}}$). This implies that for a bucket a, each $r_{a,k}$ satisfies
$$g^{s_{a,k}} = R_{a,k}Y_a$$
which satisfies the following equation in the exponent:
$$s_{a,k} = r_{a,k} + y_a.$$
- $\mathsf{AS.pVf}(vk, m_{j^*}, Y_{j^*}, \hat{\sigma}_{j^*}) = 1$. This implies the pre-signature is valid, and if a valid witness y_{j^*} (such that $Y_{j^*} = g^{y_{j^*}}$) is used to adapt the signature, the decryption algorithm will output a valid signature σ.
- $\forall k \in [N] \setminus T$: $\prod_{j\in T} Y_{j^*,j}^{\ell_j(0)} \cdot Y_{j^*,k}^{\ell_k(0)} = Y_j^*$, where T is a subset of $[N]$ of size $\rho - 1$. This implies the secret sharing of y_{j^*} was done correctly, and that a valid share y_a was used in each bucket to compute the $s_{a,k}$ (for $k \in [B]$).

Setting the total number of ciphertext to $2MNB$, where $B = |\mathsf{bckt}|$ and $B \geq \frac{\lambda}{\log MN+1} + 1$ then the probability of all $r_{a,1}, \ldots, r_{a,B}$ are invalid is negligible, by Corollary 4.2 of [30]. More precisely, we are guaranteed that there exists at least one $r_{a,k}$, such that $c_{a,k} = \mathsf{WES.Enc}((\overline{vk}_j, \overline{m}_{j^*}), r_{a,k}; r'')$ (contingent on the soundness of the NIZK scheme) and $R_{a,k} = g^{r_{a,k}}$ (recall that $R_{a,k}$ was part of π_c). This implies a valid share of the witness y_{j^*} can be computed as
$$y_a = s_{a,k} - r_{a,k}.$$

Similarly, the DecSig algorithm computes $|K|$-many valid shares of y_{j^*} by repeating the above step for buckets (j^*, j) for all $j \in K$. These valid shares can be combined to compute a valid witness y_{j^*}. Finally, since the witness is valid and the presignature is valid, the presignature can be successfully adapted to compute a valid signature σ such that $\mathsf{Vf}(vk, m_{j^*}, \sigma) = 1$, hence, giving the property of verifiability.

Instantiating NIZK Proof for $\mathcal{L}_c$. The NIZK proof essentially proves that the two WES ciphertexts encrypt the same message. If we re-use encryption randomness in both WES ciphertexts [11], then the NIZK proof essentially reduces to proving a discrete logarithm relation over $\mathbb{G}_T$. This can be done efficiently using Schnorr sigma protocol [35].

C. VweTS *Extension*

In the construction described above, the communication and computation complexity of the protocol depends substantially on the number of messages signed in the EncSig procedure (i.e., the parameter M). Next, we show a modification to our protocol that allows us to substantially reduce this dependency. In particular, instead of executing the verifiable witness encryption for all the M instances $Y_i = g^{y_i}$, we will only execute this for $\log(M) = \mu$ values.

Construction. For all $j = \{1, \ldots, M\}$, Alice samples uniformly a $Y_j = g^{y_j}$ to be the instance of the hard relation for the adaptor signature (more details in Section IV-B). Instead of computing the witness encryption of y_j, Alice samples

$$\begin{bmatrix} Z_{0,1} & \ldots & Z_{0,\mu} \\ Z_{1,1} & \ldots & Z_{1,\mu} \end{bmatrix} = \begin{bmatrix} g^{z_{0,1}} & \ldots & g^{z_{0,\mu}} \\ g^{z_{1,1}} & \ldots & g^{z_{1,\mu}} \end{bmatrix}$$

where $z_{b,i} \leftarrow \mathbb{Z}_q$. Alice also computes

$$e_j = y_j + \sum_i z_{j[i],i}$$

for all $j = \{1, \ldots, M\}$, where $j[i]$ denotes the i-th bit of j. Alice proceeds as in Section IV-B, except that she computes the witness encryption of all $\{z_{b,i}\}_{b\in\{0,1\},i\in\{1,\ldots,\mu\}}$, each conditioned on knowing the signatures of a large enough fraction of oracles that attests that the i-th bit of the outcome equals b. The cut-and-choose proceeds in a similar fashion in proving that about the witness encryption ciphertexts and the associated group elements $\{Z_{b,i}\}_{b\in\{0,1\},i\in\{1,\ldots,\mu\}}$. Note that in Section IV-B we had the above cut-and-choose run for M such ciphertexts, but only have 2μ now. The bucket mapping Φ is similar to the previous protocol except that the bucket index now also includes a position $pos \in [\mu]$. The verification procedures is unchanged, except that Bob additionally checks

$$g^{e_j} \stackrel{?}{=} Y_j \cdot \prod_{i=1}^{\mu} Z_{j[i],i}$$

for all $j = \{1, \ldots, M\}$. The attestation is also unchanged, except that the oracles now provide one signature per bit of the outcome. I.e., each signature attests that the i-th bit of the outcome is equal to some bit b. Note that there are exactly μ signatures per oracle. For a precise definition of the algorithms, we refer the reader to the full version [32].

Correctness and Efficiency. In terms of correctness, obtaining enough attestations for an outcome $j = (j[1], \ldots, j[\mu])$ allows one to witness-decrypt the corresponding ciphertexts, thereby recovering the scalars $(z_{j[1],1}, \ldots, z_{j[\mu],\mu})$. Then, computing

$$y_j = e_j - \sum_{i=1}^{\mu} z_{j[i],i}$$

allows Bob to unmask y_j and consequently to recover the signature on the transaction corresponding to the outcome j. In terms of efficiency, the overall protocol complexity is still linear in the size of M (since Alice still needs to send M adaptor signatures to Bob), but now the "expensive" verifiable (threshold) witness encryption procedure is only run for 2μ values, resulting in substantial savings.

Proof Sketch. We now argue why the scheme is secure. Fix an outcome $\tilde{j}$. Observe that the security of the witness encryption scheme allows us to argue that the values $(z_{\tilde{j}[1]\oplus 1,1}, \ldots, z_{\tilde{j}[\mu]\oplus 1,\mu})$ are hidden, provided that the majority of the oracles is honest (this assumption is also necessary for the security of our main protocol in Section IV-B). Thus, all we need to argue is that, revealing the values $(z_{\tilde{j}[1],1}, \ldots, z_{\tilde{j}[\mu],\mu})$ does not allow the adversary to recover any signature beyond the one on $m_{\tilde{j}}$. We are going to show this via a reduction to the discrete logarithm problem.

Let Y^* be the challenge group element. The reduction guesses an index $i^* \in \{1, \ldots, \mu\}$ and a bit $b^* \in \{0, 1\}$ and sets $Z_{b^*,i^*} = Y^*$. All other values of $\{Z_{b,i}\}$ are sampled as in the original game (i.e., the reduction knows the corresponding discrete logarithm $z_{b,i}$). For all $j \in \{1 \ldots, M\}$, the reduction proceeds as follows. For all j such that $j[i^*] \neq b^*$, the reduction sets e_j and Y_j as in the original game (since it knows the discrete logarithm of the corresponding group elements). And, for all j such that $j[i^*] = b^*$, the reduction computes

$$e_j = r_j + \sum_{i \neq i^*} z_{j[i],i} \text{ and } Y_j = g^{r_j} \cdot (Y^*)^{-1}$$

where $r_j \leftarrow \mathbb{Z}_q$. Observe that all values up to this point are distributed identically as in the original game. The reduction proceeds as in the original game, except that it witness encrypts 0 instead of the discrete logarithm of Z_{b^*,i^*}. Let $\tilde{j}$ be the outcome of the event, and assume that the adversary is able to recover a signature on some message corresponding to the outcome $j^* \neq \tilde{j}$. If $j^*[i^*] \neq b^*$ and $\tilde{j}[i^*] \neq b^* \oplus 1$ the reduction aborts, otherwise it uses the signature on j^* to extract the discrete logarithm of Y_{j^*}. The reduction outputs $y^* = \mathsf{DLog}(Y^*)$ since

$$\mathsf{DLog}(Y_{j^*}) = \mathsf{DLog}(g^{r_{j^*}} \cdot (Y^*)^{-1}) = r_{j^*} - y^*.$$

It is clear that the reduction is efficient and, assuming that it completes the execution, it solves the discrete logarithm problem or breaks the witness extractability of the adaptor signature. We now argue that the view of the adversary induced by the reduction is computationally indistinguishable from the one of the original game. Note that the only difference is the computation of the witness encryption for the discrete logarithm of Z_{b^*,i^*} is substituted with a witness encryption of 0. Since $\tilde{j}[i^*] = b^* \oplus 1$, it follows by the security of witness encryption that the two views are computationally indistinguishable. Finally, note that the reduction does not abort if $j^*[i^*] = b^*$ and $\tilde{j}[i^*] = b^* \oplus 1$, which is an event that happens with non-negligible probability.

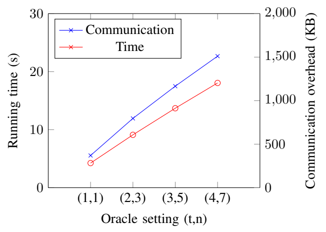
V. Performance Analysis

In this section, we describe the implementation and evaluate the practicality of our protocol for VweTS.

A. Implementation

We have developed a prototypical Rust implementation [8] to demonstrate the feasibility of our VweTS construction. The implementation encompasses the EncSig, VfEnc and DecSig algorithms, as described in Figure 5.

Implementation-level Optimizations. Alice can pre-compute several of the operations required in the VweTS.EncSig



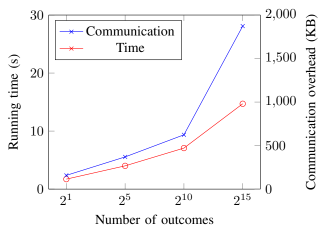
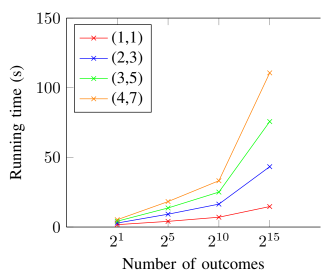


Fig. 6. Impact of the oracle setting (left) and number of outcomes (right) on running time (red) and communication overhead (blue). We set: Left: 128 outcomes; Right: 1 oracle. Security parameter is set to 2^5.



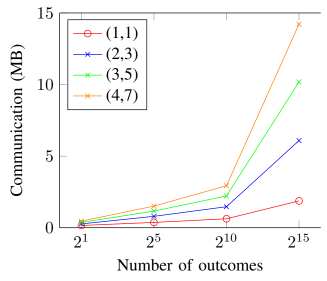


Fig. 7. Impact of the oracle setting and number of outcomes in the running time (left) and communication overhead (right) of VweTS. Security parameter is set to 128.

algorithm (see Figure 5), concretely bullet points 1, 2, 3, 4 (except for the sub-step b) and 5 (except for the complete sub-step b). The intuition behind this is that these steps use random values that are not linked to the inputs of the algorithm.

B. Performance

We conducted our experiments on a machine with a quad-core Intel Core i7 2,3 GHz and 16 GB of RAM. For our experiments, we run Alice and Bob's operations within the same machine, therefore they are communicated through localhost.

We evaluate the impact of two system parameters: (1) an increasing number of oracles and the different threshold settings; and (2) an increasing number of outcomes. The results are shown in Figure 6.

We make the following observations. First, augmenting the number of oracles, as well as the threshold for a fixed number of oracles, has a moderate impact in both running and communication time. They both seem to grow linearly on the number of oracles participating in the protocol. Second, the number of outcomes is also an impactful system parameter, both in terms of running time and communication overhead. Yet, even considering 2^{15} outcomes, the running time and communication overhead are well within reach of commodity hardware.

Impact of Increasing Oracles and Outcomes. So far, we have evaluated each system parameter in isolation. Now we evaluate the impact in time and communication overhead of increasing the threshold (i.e., the number of oracles) and the number of outcomes. The results are shown in Figure 7. As expected from previous experiments, both the running time and communication overhead are increasing faster the higher the number of outcomes we consider in the execution. Yet, even in the scenario of a threshold of 4 out of 7 oracles and a payment conditioned on up to 2^{15} different real-world event outcomes, the computation overhead is less than 150 seconds and the total communication overhead is below 15 MB.

Reducing the Impact of Number of Outcomes. So far, we have considered the setting where the oracles could attest to every single outcome set as parameter. In other words, we consider in our evaluation as many ciphertexts as outcomes. In practice, the number of ciphertexts can be considerably reduced if the messages to be attested represent values of a monotone function. For example, consider we have a scenario where Alice pays Bob if BTC/USD exchange rate is over 20K, or Alice can get refunded otherwise. In such scenario, it is not necessary to create ciphertext for every single value 20000, 20001, 20002, etc. Instead, we could envision an alternative approach where we encode attestations in the range of values (e.g., one attestation for $[0, 20000]$ and another one for $[20001, 40000]$, assuming that the maximum exchange rate is 40K.

Comparison with DLC. Discrete Log Contracts (DLC) is the proposal for oracle based conditional payments put forward by the community [18] and it is the closest in goals to our work here. Given that, we want to compare our approach with that of DLC when increasing the number of oracles and outcomes. We observe that while our approach requires a number of operations linear on the total number of oracles, the DLC approach requires a number of operations exponential in the (threshold) number of oracles since to construct a DLC for an outcome event using some threshold t-of-n oracles, they construct adaptor signatures for all outcomes for all possible combinations of t-of-t oracles [18]. Given that, for each threshold setting considered, there must exist a minimum number of outcomes after which VweTS is always more efficient than DLC. In order to test that, we have obtained a prototype implementation of the DLC design [7] where we have tested it for an increasing number of oracles and outcomes, and compared it with our implementation of VweTS. The results are shown in Figure 8.

We observe that for a small number of event outcomes, the DLC approach performs better in terms of running time than our VweTS. However, the scalability in DLC is worse than in our approach. When considering an oracle setting of 3 out of 5, our approach is faster than DLC when considering more than 2^{12} outcomes, while in a setting with 5 out of 9, our approach is faster considering more than 2^{10} outcomes. This trend continues as the number of oracle increases.

VI. Conclusions

In this work, we investigate the problem of oracle-based conditional payment that do not require Turing-complete language or are based on trusted execution environment. In particular, we present a new cryptographic primitive, verifiable witness encryption based on threshold signatures (VweTS). We give two practically efficient constructions: (i) the encrypted signatures are either Schnorr or ECDSA signatures; and (ii) the encrypted signatures are BLS signatures. In this manner, our

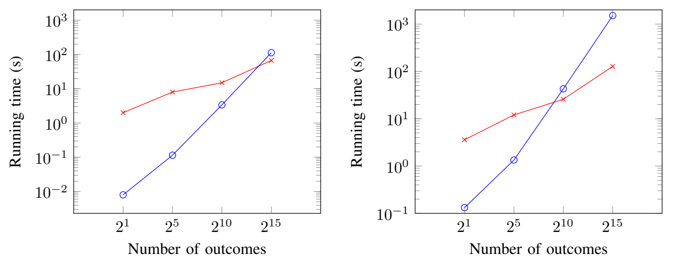


Fig. 8. Comparison running time DLC (blue) vs VweTS (red) for the oracle settings (3,5) in the left and (5,9) in the right. The security parameter is set to be 128.

constructions are compatible with many cryptocurrencies today, including Bitcoin. Moreover, we formally prove the security guarantees of our constructions. Finally, we have provided a prototype implementation and our benchmarks show that our construction is practical to be executed even in commodity hardware, and it scales better with the number of oracles compared to alternative solutions.

Acknowledgements. This work has been partially funded by Madrid regional government as part of the program S2018/TCS-4339 (BLOQUES-CM) co-funded by EIE Funds of the European Union; by the project HACRYPT (N00014-19-1-2292); by grant IJC2020-043391-I/MCIN/AEI/10.13039/501100011033 and European Union NextGenerationEU/PRTR; by PRODIGY Project (TED2021-132464B-I00) funded by MCIN/AEI/10.13039/501100011033/ and the European Union NextGenerationEU/PRTR; by SCUM Project (RTI2018-102043-B-I00) MCIN/AEI/10.13039/501100011033/ERDF A way of making Europe.

This work was partially funded by the German Federal Ministry of Education and Research (BMBF) in the course of the 6GEM research hub under grant number 16KISK038; by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany´s Excellence Strategy - EXC 2092 CASA – 390781972.

This works was also partially funded by a research grant from Protocol Labs.

References

[1] "Chainlink," https://chain.link/use-cases.

[2] "Chia network faq," https://www.chia.net/faq/.

[3] "Clear and unambiguous terms of merger agreement," https://corpgov.law.harvard.edu/2019/10/09/clear-and-unambiguous-terms-of-merger-agreement/.

[4] "Contractor payment schedules," https://www.levelset.com/blog/contractor-payment-schedule/.

[5] "Defi pulse website," https://www.defipulse.com/.

[6] "The oracle of truth: Where do blockchain betting projects get their event's results?" https://hackernoon.com/the-oracle-of-truth-where-do-blockchain-betting-projects-get-their-event-results-r44b2c99.

[7] "Source code for our test on the dlc project." https://github.com/LLFourn/rust-dlc/tree/lloyd-benchmark.

[8] "Source code for this project. implementation of bls-based attestations," https://github.com/LLFourn/dlc-verifiable-encryption-bls/tree/ndss-snapshot.

[9] "Timelock encryption/decryption made practical," https://github.com/drand/tlock.

[10] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostáková, M. Maffei, P. Moreno-Sanchez, and S. Riahi, "Generalized channels from limited blockchain scripts and adaptor signatures," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2021, pp. 635–664.

[11] M. Bellare, A. Boldyreva, and J. Staddon, "Randomness re-use in multi-recipient encryption schemeas," in *PKC 2003*, ser. LNCS, Y. Desmedt, Ed., vol. 2567. Miami, FL, USA: Springer, Heidelberg, Germany, Jan. 6–8, 2003, pp. 85–99.

[12] D. Boneh and M. K. Franklin, "Identity-based encryption from the Weil pairing," in *CRYPTO 2001*, ser. LNCS, J. Kilian, Ed., vol. 2139. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 19–23, 2001, pp. 213–229.

[13] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," in *ASIACRYPT 2001*, ser. LNCS, C. Boyd, Ed., vol. 2248. Gold Coast, Australia: Springer, Heidelberg, Germany, Dec. 9–13, 2001, pp. 514–532.

[14] J. V. Bulck, D. F. Oswald, E. Marin, A. Aldoseri, F. D. Garcia, and F. Piessens, "A tale of two worlds: Assessing the vulnerability of enclave shielding runtimes," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, L. Cavallaro, J. Kinder, X. Wang, and J. Katz, Eds. ACM, 2019, pp. 1741–1758. [Online]. Available: https://doi.org/10.1145/3319535.3363206

[15] J. Camenisch and I. Damgård, "Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes," in *ASIACRYPT 2000*, ser. LNCS, T. Okamoto, Ed., vol. 1976. Kyoto, Japan: Springer, Heidelberg, Germany, Dec. 3–7, 2000, pp. 331–345.

[16] M. Chase, "Multi-authority attribute based encryption," in *TCC 2007*, ser. LNCS, S. P. Vadhan, Ed., vol. 4392. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Feb. 21–24, 2007, pp. 515–534.

[17] G. Chen, S. Chen, Y. Xiao, Y. Zhang, Z. Lin, and T. Lai, "Sgxpectre: Stealing intel secrets from SGX enclaves via speculative execution," *IEEE Secur. Priv.*, vol. 18, no. 3, pp. 28–37, 2020. [Online]. Available: https://doi.org/10.1109/MSEC.2019.2963021

[18] D. community, "Specification for discreet log contracts," https://github.com/discreetlogcontracts/dlcspecs.

[19] A. De Santis, S. Micali, and G. Persiano, "Non-interactive zero-knowledge proof systems," in *Conference on the Theory and Application of Cryptographic Techniques*. Springer, 1987, pp. 52–72.

[20] N. Döttling, L. Hanzlik, B. Magri, and S. Wohnig, "Mcfly: Verifiable encryption to the future made practical," *Cryptology ePrint Archive*, 2022.

[21] T. Dryja, "Discreet log contracts," https://adiabat.github.io/dlc.pdf.

[22] A. Erwig, S. Faust, K. Hostáková, M. Maitra, and S. Riahi, "Two-party adaptor signatures from identification schemes," in *PKC 2021, Part I*, ser. LNCS, J. Garay, Ed., vol. 12710. Virtual Event: Springer, Heidelberg, Germany, May 10–13, 2021, pp. 451–480.

[23] S. Eskandari, M. Salehi, W. C. Gu, and J. Clark, "Sok: Oracles from the ground truth to market manipulation," in *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, 2021, pp. 127–141.

[24] S. Garg, C. Gentry, A. Sahai, and B. Waters, "Witness encryption and its applications," in *45th ACM STOC*, D. Boneh, T. Roughgarden, and J. Feigenbaum, Eds. Palo Alto, CA, USA: ACM Press, Jun. 1–4, 2013, pp. 467–476.

[25] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, "Extending oblivious transfers efficiently," in *CRYPTO 2003*, ser. LNCS, D. Boneh, Ed., vol. 2729. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 17–21, 2003, pp. 145–161.

[26] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, Aug 2001. [Online]. Available: https://doi.org/10.1007/s102070100002

[27] A. Juels, L. Breidenbach, A. Coventry, S. Nazarov, S. Ellis, and B. Magauran, "Mixicles: Simple private decentralized finance," 2019.

[28] N. Koheh, "Update on dlcs (new mailing list)," https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2021-January/018372.html.

[29] T. Le Guilly, N. Kohen, and I. Kuwahara, "Bitcoin oracle contracts: Discreet log contracts in practice," in *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2022, pp. 1–8.

[30] Y. Lindell and B. Riva, "Cut-and-choose Yao-based secure computation in the online/offline and batch settings," in *CRYPTO 2014, Part II*, ser.

LNCS, J. A. Garay and R. Gennaro, Eds., vol. 8617. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 17–21, 2014, pp. 476–494.
[31] B. Liu, P. Szalachowski, and J. Zhou, "A first look into defi oracles," in *IEEE International Conference on Decentralized Applications and Infrastructures, DAPPS 2021, Online Event, August 23-26, 2021*. IEEE, 2021, pp. 39–48. [Online]. Available: https://doi.org/10.1109/DAPPS52256.2021.00010
[32] V. Madathil, S. A. Thyagarajan, D. Vasilopoulos, L. Fournier, G. Malavolta, and P. Moreno-Sanchez, "Practical decentralized oracle contracts for cryptocurrencies," *Cryptology ePrint Archive*, 2022.
[33] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments."
[34] L. (pseudonym), "Secure dlcs," https://bitcoinproblems.org/problems/secure-dlcs.html.
[35] C.-P. Schnorr, "Efficient identification and signatures for smart cards," in *CRYPTO'89*, ser. LNCS, G. Brassard, Ed., vol. 435. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 20–24, 1990, pp. 239–252.
[36] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.
[37] S. M. Werner, D. Perez, L. Gudgeon, A. Klages-Mundt, D. Harz, and W. J. Knottenbelt, "Sok: Decentralized finance (defi)," *CoRR*, vol. abs/2101.08778, 2021. [Online]. Available: https://arxiv.org/abs/2101.08778
[38] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. ACM, 2016, pp. 270–282. [Online]. Available: https://doi.org/10.1145/2976749.2978326
[39] F. Zhang, D. Maram, H. Malvai, S. Goldfeder, and A. Juels, "Deco: Liberating web data using decentralized oracles for tls," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1919–1938.

Appendix

A. Extended Preliminaries

Adaptor Signatures. We recall the missing definitions for adaptor signatures.

Definition 9 (Pre-signature Correctness): An adaptor signature scheme AS satisfies pre-signature correctness if for every $\lambda \in \mathbb{N}$, every message $m \in \{0,1\}^*$ and every statement/witness pair $(Y, y) \in R$, the following holds:

$$\Pr\left[\begin{array}{c|l} \mathsf{pVf}(vk, m, Y, \hat{\sigma}) = 1 & (sk, vk) \leftarrow \mathsf{KGen}(1^\lambda) \\ \wedge & \hat{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y) \\ \mathsf{Vf}(vk, m, \sigma) = 1 & \sigma := \mathsf{Adapt}(\hat{\sigma}, y) \\ \wedge & y' := \mathsf{Ext}(\sigma, \hat{\sigma}, Y) \\ (Y, y') \in R & \end{array}\right] = 1.$$

Next, we formally define the security properties of an adaptor signature scheme.

Definition 10 (Unforgeability): An adaptor signature scheme AS is aEUF-CMA secure if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that:

$$\Pr\left[\mathsf{aSigForge}_{\mathcal{A},\mathsf{AS}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

where the experiment $\mathsf{aSigForge}_{\mathcal{A},\mathsf{AS}}$ is defined in Figure 9.

Definition 11 (Pre-signature Adaptability): An adaptor signature scheme AS satisfies pre-signature adaptability if for any $\lambda \in \mathbb{N}$, any message $m \in \{0,1\}^*$, any statement/witness pair $(Y, y) \in R$, any key pair $(sk, vk) \leftarrow \mathsf{KGen}(1^\lambda)$ and any pre-signature $\hat{\sigma} \leftarrow \{0,1\}^*$ with $\mathsf{pVf}(vk, m, Y, \hat{\sigma}) = 1$ we have:

$$\Pr[\mathsf{Vf}(vk, m, \mathsf{Adapt}(\hat{\sigma}, y)) = 1] = 1$$

Definition 12 (Witness Extractability): An adaptor signature scheme AS is *witness extractable* if for every PPT adversary $\mathcal{A}$, there exists a negligible function negl such that the following holds:

$$\Pr[\mathsf{aWitExt}_{\mathcal{A},\mathsf{AS}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

where the experiment $\mathsf{aWitExt}_{\mathcal{A},\mathsf{AS}}$ is defined in Figure 10.

BLS Signatures. We briefly recall here the BLS signature scheme [13]. Let $(\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_t)$ be a bilinear group of prime order q, where q is a λ bit prime. Let e be an efficiently computable bilinear pairing $e : \mathbb{G}_0 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$, where g_0 and g_1 are generators of $\mathbb{G}_0$ and $\mathbb{G}_1$ respectively. Let H be a hash function $H : \{0,1\}^* \rightarrow \mathbb{G}_1$.

- $(vk, sk) \leftarrow \mathsf{KGen}(1^\lambda)$: choose $\alpha \leftarrow \mathbb{Z}_q$, set $h \leftarrow g_0^\alpha \in \mathbb{G}_0$ and output $vk := h$ and $sk := \alpha$.
- $\sigma \leftarrow \mathsf{Sign}(sk, m)$: output $\sigma := H(m)^{sk} \in \mathbb{G}_1$.
- $0/1 \leftarrow \mathsf{Vf}(vk, m, \sigma)$: if $e(g_0, \sigma) = e(vk, H(m))$, then output 1 and otherwise output 0.

Schnorr Signatures. We briefly recall the Schnorr signature scheme [35], that is defined over a cyclic group $\mathbb{G}$ of prime order q with generator g, and use a hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$.

- $(vk, sk) \leftarrow \mathsf{KGen}(1^\lambda)$: choose $x \leftarrow \mathbb{Z}_q$ and set $sk := x$ and $vk := g^x$.
- $\sigma \leftarrow \mathsf{Sign}(sk, m; r)$: sample a randomness $r \leftarrow \mathbb{Z}_q$ to compute $R := g^r, c := H(g^x, R, m), s := r + cx$ and output $\sigma := (R, s)$.
- $0/1 \leftarrow \mathsf{Vf}(vk, m, \sigma)$: parse $\sigma := (R, s)$ and then compute $c := H(vk, R, m)$ and if $g^s = R \cdot vk^c$ output 1, otherwise output 0.

ECDSA Signatures. The ECDSA signature scheme [26] is defined over an elliptic curve group $\mathbb{G}$ of prime order q with base point (generator) g. The construction assumes the existence of a hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ and is given in the following.

- $(vk, sk) \leftarrow \mathsf{KGen}(1^\lambda)$: choose $x \leftarrow \mathbb{Z}_q$ and set $sk := x$ and $vk := g^x$.
- $\sigma \leftarrow \mathsf{Sign}(sk, m; r)$: sample an integer $k \leftarrow \mathbb{Z}_q$ and compute $c \leftarrow H(m)$. Let $(r_x, r_y) := R = g^k$, then set $r := r_x \mod q$ and $s := (c + rx)/k \mod q$. Output $\sigma := (r, s)$.

```
$\mathsf{aSigForge}_{\mathcal{A},\mathsf{AS}}(\lambda)$
$\mathcal{Q} := \emptyset$
$(sk, vk) \leftarrow \mathsf{KGen}(1^\lambda)$
$m \leftarrow \mathcal{A}^{\mathsf{Sign}\mathcal{O}(\cdot),\mathsf{pSign}\mathcal{O}(\cdot,\cdot)}(vk)$
$(Y, y) \leftarrow \mathsf{GenR}(1^\lambda)$
$\hat{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$
$\sigma \leftarrow \mathcal{A}^{\mathsf{Sign}\mathcal{O}(\cdot),\mathsf{pSign}\mathcal{O}(\cdot,\cdot)}(\hat{\sigma}, Y)$
return $(m \notin \mathcal{Q} \wedge \mathsf{Vf}(vk, m, \sigma))$

$\mathsf{Sign}\mathcal{O}(m)$
$\sigma \leftarrow \mathsf{Sign}(sk, m)$
$\mathcal{Q} := \mathcal{Q} \cup \{m\}$
return $\sigma$

$\mathsf{pSign}\mathcal{O}(m, Y)$
$\hat{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$
$\mathcal{Q} := \mathcal{Q} \cup \{m\}$
return $\hat{\sigma}$
```

Fig. 9. Unforgeabiltiy experiment of adaptor signatures

```
$\mathsf{aWitExt}_{\mathcal{A},\mathsf{AS}}(\lambda)$
    $\mathcal{Q} := \emptyset$
    $(sk, vk) \leftarrow \mathsf{KGen}(1^\lambda)$
    $(m, Y) \leftarrow \mathcal{A}^{\mathsf{Sign}\mathcal{O}(\cdot),\mathsf{pSign}\mathcal{O}(\cdot,\cdot)}(vk)$
    $\hat{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$
    $\sigma \leftarrow \mathcal{A}^{\mathsf{Sign}\mathcal{O}(\cdot),\mathsf{pSign}\mathcal{O}(\cdot,\cdot)}(\hat{\sigma})$
    $y' := \mathsf{Ext}(vk, \sigma, \hat{\sigma}, Y)$
    return $(m \notin \mathcal{Q} \wedge (Y, y') \notin R$
        $\wedge \mathsf{Vf}(vk, m, \sigma))$

$\mathsf{Sign}\mathcal{O}(m)$
    $\sigma \leftarrow \mathsf{Sign}(sk, m)$
    $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
    return $\sigma$

$\mathsf{pSign}\mathcal{O}(m, Y)$
    $\hat{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$
    $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
    return $\hat{\sigma}$
```

Fig. 10. Witness extractability experiment for adaptor signatures

- $0/1 \leftarrow \mathsf{Vf}(vk, m, \sigma)$: parse $\sigma := (r, s)$ and compute $c := H(m)$ and return 1 if and only if $(x, y) = (g^c \cdot h^r)^{s^{-1}}$ and $x = r \mod q$. Otherwise output 0.

B. Construction based on BLS signatures

In this section, we present another concrete construction of VweTS with parameters ρ, N and M relying on the same cryptographic building blocks as the previous construction, except that we replace DS with BLS signature scheme the same as $\overline{\mathsf{DS}}$.

High Level Overview. We present a high level overview of our construction, and the formal description is given in Figure 11. Similar to the adaptor signature based construction, we assume the availability of public parameters.

The signature generation algorithm proceeds similar to the previous construction, except that instead of generating adaptor pre-signatures on the message m_i, the algorithm generates BLS signatures on the message m_i wrt. secret key sk. It then secret shares each of the BLS signatures and for each of their verifiability, the algorithm also generates the shares of the verification key vk. The final point of difference is in the cut-and-choose where, for the unopened indices i such that $(\alpha, \beta) := \Phi(i)$, we set the value s_i to be the aggregate of the signature share $\sigma_{\alpha,\beta}$ and the value $g_1^{r_i}$. The rest of the algorithm proceeds as the adaptor signature based construction.

To verify, the algorithm proceeds as before except now instead of checking the correctness of adaptor witness sharing, it verifies the correctness of the signature sharing with a simple pairing check. The decryption algorithm also proceeds as before, except the difference is obtaining the signature share from s_i. Recall s_i is an aggregate of the signature share and a group element in this case. Therefore, to obtain the signature share, we divide away the masking group element and finally reconstruct the required signature via Lagrange interpolation.

Proof of Correctness. We prove the correctness of the scheme.

Theorem 2: Our VweTS construction from Figure 11 is correct according to Definition 6.

Proof 2: We let $(c, \pi_c) \leftarrow \mathsf{EncSig}(((\overline{vk}_i)_{i\in[N]}, (\overline{m}_j)_{j\in[M]}, \rho),\ sk,\ (m_j)_{j\in[M]})$. To prove correctness we first need to show that

$$\Pr\left[\mathsf{VfEnc}(c, \pi_c, ((\overline{vk}_i)_{i\in[N]}, (\overline{m}_j, m_j)_{j\in[M]}, vk)) = 1\right] = 1.$$

Note that VfEnc will output 0 if one of the following occurs:

1) If $b_i = 0$ and $c'_i \neq \mathsf{WES.Enc}((\overline{vk}^*, \overline{m}^*), r_i; r'_i)$. Provided the encryption is done correctly, this occurs with zero probability.
2) If $b_i = 1$ and $e(g_0, s_i) \neq e(R_i, g_1) \cdot e(h_{\alpha,\beta}, H(m_\alpha))$. Note that by construction we have $s_i = \sigma_{\alpha,\beta} \cdot g_1^{r_i}$. This implies
$$\begin{aligned} e(g_0, s_i) &= e(g_0, \sigma_{\alpha,\beta} \cdot g_1^{r_i}) \\ &= e(g_0, H_0(m_\alpha)^{x_{\alpha,\beta}} \cdot g_1^{r_i}) \\ &= e(g_0, H_0(m_\alpha)^{x_{\alpha,\beta}}) \cdot e(g_0, g_1^{r_i}) \\ &= e(g_0^{x_{\alpha,\beta}}, H_0(m_\alpha)) \cdot e(g_0^{r_i}, g_1) \\ &= e(h_{\alpha,\beta}, H_0(m_\alpha)) \cdot e(R_i, g_1) \end{aligned}$$
and therefore this case never occurs.
3) If $b_i = 1$ and $\Pi_{\mathcal{L}_c}.\mathsf{Vf}(\overline{vk}_\beta, \overline{vk}^*, \overline{m}_\alpha, \overline{m}^*, c, c'_i, \pi) = 0$. By the completeness of the zero-knowledge protocol this occurs with zero probability.
4) If $b_i = 1$ and $\prod_{j\in T} h_{\alpha,j}^{\ell_j(0)} \cdot h_{\alpha,k}^{\ell_k(0)} = vk$. This case is impossible by construction of the shares of vk for $\alpha \in [M]$ and $k \in [N]$.

Thus we have shown that if EncSig is computed correctly, VfEnc outputs 1 with probability 1.

Next we need to show that for any $j \in [M], K \subset [N]$ and $|K| = \rho$, if for all $i \in K$ we have $\overline{\mathsf{Vf}}(\overline{vk}_i, \overline{m}_j, \overline{\sigma}_i) = 1$, then

$$\Pr[\mathsf{Vf}\left(vk, m_j, \mathsf{DecSig}(j, \{\overline{\sigma}_i\}_{i\in K}, c, \pi_c)\right) = 1] = 1.$$

We are given that for all $i \in K$, $\overline{\mathsf{Vf}}(\overline{vk}_i, \overline{m}_j, \overline{\sigma}_i) = 1$. By construction, we have N buckets of size B that correspond to the message m_j. Let these buckets be denoted as $\mathsf{bckt}_{j,1}, \ldots, \mathsf{bckt}_{j,N}$. W.l.o.g. let K correspond to the first $|K|$ of these N buckets. And let each bucket $\mathsf{bckt}_{j,i}$ contain ciphertexts $c_1, \ldots c_B$ For $i \in K$:

1) Let rShare_i denote the set of values that are decrypted from $\mathsf{bckt}_{j,i}$
2) For each $c_k \in \mathsf{bckt}_{j,i}$
   a) Compute $r = \mathsf{WES.Dec}(\overline{\sigma}_i, c_k)$
   b) Update $\mathsf{rShare}_i = \mathsf{rShare}_i \cup r$. By the correctness property of WES we can correctly compute a r.

Let each r in rShare_i be denoted as $r_{i,a}$ for each $\mathsf{bckt}_{j,i}$. To each $r_{i,a}$ is associated an (a, s_a, c_a, π_a). By construction it is guaranteed that $R_a = g_0^{r_{i,a}}$. Pick any $r_{i,a}$ from the rShare_i. Since by construction, $s_a = \sigma_{j,\beta} \cdot g_1^{r_{i,a}}$ (j is the message number and β is the server number), one can compute $\sigma_{j,\beta} = s_a / g_1^{r_{i,a}}$.

Since $\sigma_{j,\beta} = \left(\frac{\sigma_j}{\prod_{i\in[t-1]} \sigma_{j,i}^{\ell_j(0)}}\right)^{\ell_i(0)^{-1}}$ by construction, one can compute $\sigma_j = \prod_{i\in K} \sigma_{j,i} \cdot \ell_i(0)$.

```
**Public parameters**: $(\mathbb{G}_0, g_0, \mathbb{G}_1, g_1, q, \mathbb{G}_T, \gamma, H_2, crs)$
$(c, \pi_c) \leftarrow \mathsf{EncSig}(((\overline{vk}_i)_{i\in[N]}, (\overline{m}_j)_{j\in[M]}, \rho), sk, (m_j)_{j\in[M]})$:
1) Sample random $\overline{vk}^* \in \mathbb{G}_0$ and $\overline{m}^* \in \{0,1\}^\lambda$, initialize $\mathcal{S}_{\mathsf{op}} = \mathcal{S}_{\mathsf{unop}} = \emptyset$.
2) For $i \in [\gamma]$:
    a) Sample $r_i \leftarrow \mathbb{Z}_q$ and compute $R_i := g_0^{r_i}$.
    b) Compute $c'_i := \mathsf{WES.Enc}((\overline{vk}^*, \overline{m}^*), r_i; r'_i)$ where $r'_i$ is the random coins used.
3) For $i \in [1, M]$:
    a) Compute $\sigma_i = \mathsf{DS.Sign}(sk, m_i)$.
    b) For $j \in [\rho - 1]$, sample a uniform $x_{i,j} \leftarrow \mathbb{Z}_q$ and set $\sigma_{i,j} = H_0(m_i)^{x_{i,j}}$ and set $h_{i,j} = g_0^{x_{i,j}}$.
    c) For all $j \in \{t, \dots, N\}$ compute $\sigma_{i,j} = \left(\frac{\sigma_i}{\prod_{j\in[t-1]} \sigma_{i,j}^{\ell_j(0)}}\right)^{\ell_i(0)^{-1}}$, $h_{i,j} = \left(\frac{vk}{\prod_{j\in[t-1]} h_{i,j}^{\ell_j(0)}}\right)^{\ell_i(0)^{-1}}$.
4) Set $\Sigma_1 = \{h_{i,j}\}_{i\in[M], j\in[N]}$.
5) Compute $\{\Phi, (b_1, \dots, b_\gamma)\} := H_2((c'_i, R_i)_{i\in[\gamma]}, \Sigma_1)$.
6) For $i \in [\gamma]$:
    a) If $b_i = 1$, do $\mathcal{S}_{\mathsf{op}} = \mathcal{S}_{\mathsf{op}} \cup (i, r_i, r'_i)$.
    b) If $b_i = 0$:
        i) Let $(\alpha, \beta) := \Phi(i)$.
        ii) Compute $s_i = \sigma_{\alpha,\beta} \cdot g_1^{r_i}$.
        iii) Compute $c_i := \mathsf{WES.Enc}((\overline{vk}_\beta, \overline{m}_\alpha), r_i; r'_i)$ and $\pi_i \leftarrow \Pi_{\mathcal{L}_c}.\mathsf{Prove}(\overline{vk}_\beta, \overline{vk}^*, \overline{m}_\alpha, \overline{m}^*, c_i, c'_i)$.
        iv) Set $\mathcal{S}_{\mathsf{unop}} = \mathcal{S}_{\mathsf{unop}} \cup (i, c_i, \pi_i, s_i)$.
7) Return $c = \{c'_i\}_{i\in[\gamma]}, \pi_c = \{\mathcal{S}_{\mathsf{op}}, \mathcal{S}_{\mathsf{unop}}, \overline{vk}^*, \overline{m}^*, \{R_i\}_{i\in[\gamma]}, \Sigma_1\}$.
$0/1 \leftarrow \mathsf{VfEnc}(c, \pi_c, ((\overline{vk}_i)_{i\in[N]}, (\overline{m}_j, m_j)_{j\in[M]}, vk))$:
1) Parse $c$ as $\{c'_i\}_{i\in[\gamma]}$ and $\pi_c$ as $\{\mathcal{S}_{\mathsf{op}}, \mathcal{S}_{\mathsf{unop}}, \overline{vk}^*, \overline{m}^*, \{R_i\}_{i\in[\gamma]}, \Sigma_1$ and $\Sigma_1 = \{h_{i,j}\}_{i\in[M], j\in[N]}\}$.
2) Compute $\{\Phi, (b_1, \dots, b_\gamma)\} := H_2((c'_i, R_i)_{i\in[\gamma]}, \Sigma_1)$.
3) For $i \in [\gamma]$:
    a) If $b_i = 0$, check that $(i, r_i, r'_i) \in \mathcal{S}_{\mathsf{op}}$ and that $c'_i := \mathsf{WES.Enc}((\overline{vk}^*, \overline{m}^*), r_i; r'_i)$.
    b) If $b_i = 1$:
        i) $(\alpha, \beta) := \Phi(i)$.
        ii) Check that $(i, c_i, \pi_i, s_i) \in \mathcal{S}_{\mathsf{unop}}$.
        iii) Check that $e(g_0, s_i) = e(R_i, g_1) \cdot e(h_{\alpha,\beta}, H_0(m_\alpha))$.
        iv) Check $\Pi_{\mathcal{L}_c}.\mathsf{Vf}(\overline{vk}_\beta, \overline{vk}^*, \overline{m}_\alpha, \overline{m}^*, c, c'_i, \pi_i) = 1$.
        v) Let $T$ be a subset of $[N]$ of size $\rho - 1$, check that for every $k \in [N] \setminus T$: $\prod_{j\in T} h_{\alpha,j}^{\ell_j(0)} \cdot h_{\alpha,k}^{\ell_k(0)} = vk$.
    c) If any of the checks fail output 0, else output 1.
$\sigma \leftarrow \mathsf{DecSig}(j, \{\overline{\sigma}_i\}_{i\in[K]}, c, \pi_c)$:
1) Parse $c$ as $\{c'_i\}_{i\in[\gamma]}$ and and $\pi_c$ as $\{\mathcal{S}_{\mathsf{op}}, \mathcal{S}_{\mathsf{unop}}, \overline{vk}^*, \overline{m}^*, \{R_i\}_{i\in[\gamma]}, \Sigma_1$ and $\Sigma_1 = \{h_{i,j}\}_{i\in[M], j\in[N]}\}$.
2) Initialize $\mathsf{rShare}_i = \emptyset$ for $i \in [K]$.
3) For each $(i, c_i, \pi_i, s_i) \in \mathcal{S}_{\mathsf{unop}}$, compute $(\alpha, \beta) = \Phi(i)$. If $\alpha = j$ and $\beta \in [K]$ s.t. $\mathsf{DS.Vf}(\overline{vk}_\beta, (\overline{m}_\alpha, \overline{\sigma}_i) = 1)$.
    a) Compute $r = \mathsf{WES.Dec}(\overline{\sigma_i}, c_i)$.
    b) $\mathsf{rShare}_\beta := \mathsf{rShare}_\beta \cup \{r\}$.
4) It is guaranteed that at least one $r$ in each $\mathsf{rShare}_i$ is valid. Denote this as $r_{i,a}$, where $(a, c_a, \pi_i, s_a) \in \mathcal{S}_{\mathsf{unop}}$.
5) For $i \in [K]$, compute $\sigma_{j,i} = s_a / g_1^{r_{i,a}}$.
6) Return $\sigma_j = \prod_{i\in[K]} \sigma_{j,i}^{\ell_i(0)}$.
```

Fig. 11. VWeTS from BLS signatures.