Practical Decentralized Oracle Contracts for Cryptocurrencies

Varun Madathil
North Carolina State University
vrmadath@ncsu.edu

Sri AravindaKrishnan Thyagarajan
Carnegie Mellon University
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

The lack of data feeds about real-world events happening "outside" of the blockchain environment is a critical obstacle to the development of smart contracts. This has motivated the introduction of trusted identities, the so-called "Oracles", that attest the information about real-world events into the blockchain. This enables mutually distrustful parties to establish contracts based on said events.

Previous proposals for oracle-based contracts rely either on Turing-complete smart contracts or on trusted hardware. While the latter imposes an additional trust assumption, the former relies on a Turing-complete language to write the complete data feed on-chain, imposing thus an undesirable on-chain storage overhead and being incompatible with many popular cryptocurrencies that do not support Turing-complete language such as Bitcoin. Moreover, no proposal so far comes with provable cryptographic guarantees.

In this work, we lay the foundations of oracle contracts for cryptocurrencies. We present game-based definitions that model the security properties of oracle contracts, and we propose the first construction with provable security guarantees. Moreover, our construction does not incur any additional on-chain overhead and is compatible with all cryptocurrencies. Finally, our evaluation shows that our construction is practical even in commodity hardware.

As a contribution of independent interest, we show an efficient construction of *witness encryption* for the class of languages: $\{(vk, m) \in \mathcal{L} : \exists\, \sigma \text{ s.t. } \mathsf{Verify}(vk, \sigma, m) = 1\}$ where σ is a BLS signature on m. We show how this can be efficiently extended to the threshold settings (allowing the distribution of trust among several "Oracles") and how to prove that the encrypted message has a certain structure (e.g., it is itself a valid signature on some message). To guarantee the latter in a practically efficient manner, we develop a new batching technique for cut-and-choose, inspired by the work of Lindell-Riva on garbled circuits.

1 INTRODUCTION

From their inception, blockchain-based cryptocurrencies have provided a mean for payments governed by a consensus protocol executed by mutually distrusting parties located worldwide. Less than 15 years later, they have evolved and offer a complex financial architecture, the so-called Decentralized Finance (DeFi), that includes components to support lending, decentralized exchange of assets, markets of derivatives, among others [28].

In principle, the most compelling applications of smart contracts are inherently limited, since they require access to data about state of the real-world state and events that are external to the blockchain [20]. For instance, it might be necessary for a smart contract implementing a decentralized exchange across different currencies to have access to information about up-to-date exchange rates to carry out the exchanges weighted accordingly. Oracles (also known as data feeds) aim to meet this need. In fact, many of Ethereum-based DeFi applications rely at its core on oracle based contracts [1]. For example, active outstanding loans from only four open lending contracts (MakerDAO, Fulcrum, dYdX, and Compound) that use oracle services, are worth above \$200 million [24]. Moreover, there exist companies such as Chainlink whose business model consists on offering the oracle service to current and future smart contracts.

In a nutshell, the functionality of an oracle is that it attests the information about real-world events into the blockchain so that other smart contracts can perform operations accordingly. In the simplest form, say there are three mutually distrusting parties in the oracle-based contract process: Alice, and Bob, the contract counterparties and Olivia, the oracle. Alice and Bob make and execute today a smart contract whose payout is defined by the outcome of a real world event in the future. After the event happens, Olivia attests the outcome of the event to the smart contract and the corresponding user (either Alice or Bob) gets paid. The realization of this vision, however, poses a number of technical challenges, most notably, *unforgeability* and *verifiability*.

An oracle contract that provides unforgeability must ensure that Bob does not get the payout of the contract before Olivia provides the corresponding attestation. Additionally, an oracle contract provides verifiability if after Alice sets up the contract with Bob, the latter is guaranteed that he will get a payout from it if Olivia attests the corresponding event correctly.

Related Work. Existing approaches can roughly be grouped in three trends. The first one consists in including the operation logic of Alice, Bob, and Olivia in a smart contract that controls the complete lifecycle. While this approach is already used in practice [24, 28], it suffers from several drawbacks: (i) it is tailored to the characteristics offered by a *restricted* set of currencies (e.g., those supporting Turing-complete scripting languages); (ii) it hinders *scalability* since the complete operation logic as well as attestation data is stored on the blockchain; (iii) it hampers *fungibility* since an oracle contract is trivially distinguishable from other contracts by a blockchain observer.

A second approach was proposed by Zhang et al. [29] where the functionality of Olivia within the oracle contract is executed within a trusted execution environment (TEE). This approach provides the correctness guarantee of the data attested by Olivia. However, this approach suffers from the same drawbacks as mentioned above, as the rest of the functionality (including the verification of the attested data provided by the TEE) is executed within a smart contract as before. Moreover, this approach adds a trust assumption on the TEE which it is unclear to hold in practice [10, 13] and it is against the decentralization philosophy of blockchains to start with.

A somewhat different approach was initiated by the name of Discreet Log Contracts [17] and put forward by the Bitcoin community [22]. A Discreet Log Contract (DLC) is a Bitcoin-compatible oracle contract enabling transactions from Alice to Bob to be contingent on signatures published by Olivia. This approach is promising because (i) it requires only an adaptor-compatible signature scheme [4] such as ECDSA or Schnorr and a timelock functionality from the underlying blockchain, which is available in many cryptocurrencies today; (ii) 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, associated fee cost and helping to preserve the fungibility of the cryptocurrency.

However, none of the previous approaches provide a formal description of the oracle contract problem along with the security notions of interest. This is the gap that we aim to bridge in this work. We also provide practically efficient constructions that can be used across any currency.

1.1 Our Contributions

Our contributions can be summarized as follows:

- We formally define the notion of oracle contracts for cryptocurrencies. We provide a formal model with game-based security definitions that model the properties of interest for this new primitive. We also propose an efficient construction and formally prove its security. Our protocol is the first one that comes with provable guarantees, while being more versatile and capturing more application scenarios, whilst simultaneously overcoming the compatibility and scalability issues with state-of-the-art approaches (more details in Section 1.2)
- As our main building block, we present a new cryptographic primitive of *verifiable witness encryption based on threshold signatures (*VweTS*)*: VweTS allow one to (verifiably) encrypt a message that can be decrypted using (any set of a minimum number of) *witness* signatures on an instance message. Moreover, the encrypted message is itself a valid signature, which in the case of oracle contracts is a valid signature on a transaction. We provide formal definitions for VweTS, along with 2 *efficient* constructions: (1) where the signature scheme used to sign transactions is an adaptor-based signature scheme like Schnorr or ECDSA and (2) where the signature scheme is the BLS signature scheme. In both of our constructions, the witness signatures are BLS signatures. For efficient verifiability, we combine techniques from the Camenish-Damgård *cut-and-choose* approach [11] along with the batching technique of Lindell and Riva [23], originally developed to optimize garbled circuit computations over many executions.
- We provide a prototype implementation of VweTS and evaluate how the running time and the communication overhead is affected by the system parameters such as the security parameter, the oracle threshold setting as well as the number of possible outcomes of an event. Our evaluation shows that our construction is practical to be executed even in commodity hardware. Moreover, our evaluation shows that our approach is faster than current DLC when considering an increasing number of oracles.

1.2 Concurrent Work and Comparison

Concurrent to this work, Döttling et al. [16] proposed a witness encryption similar to ours for the same class of languages, 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.

A solution for oracle contracts was proposed by Dryja [17], however it comes with a number of shortcomings: (i) It only supports a single oracle and only a constant amount of oracle outcomes. This leads to single point of failure, and it is unclear how one can extend their protocol to a setting with multiple oracles and a distributed trust, without assuming some sort of coordination among the oracles. Also, (ii) the protocol in [17] requires a synchronous communication between the oracle and Alice, where the oracle has to announce some secret value periodically which Alice uses in her promises to Bob. Furthermore, (iii) the oracle attestation is strongly tied to the signature scheme of the transaction scheme used by Alice and Bob. Finally, (iv) their solution is not formally analyzed and in fact a number of attacks have been subsequently discovered [25].

On the other hand, our VweTS-based solution supports distributed trust using the threshold setting with many independent oracles who need not interact with each other at any point in time. In our framework, there is no communication between the oracles and Alice prior to her promises to Bob. This, for example, allows for Alice to make several promises of payments that can be scheduled for the future, like a monthly Netflix subscription, monthly utility payments or salaries to employees. With the solution from [17], Alice had to wait for the communication from the oracles every time before making a promise and therefore cannot schedule payments for the future. The oracle attestation in our oracle contracts is independent of the authentication mechanism, i.e., the signature scheme of the transaction scheme. This makes our solution more versatile to different currencies.

2 TECHNICAL OVERVIEW

To establish some intuition for our problem, we consider the setting where Alice, with a key pair (sk_A, vk_A) of a digital signature DS, wants to transfer v coins to Bob in a transaction tx, if a certain real world event (represented by the message $\overline{m}$) is attested by Olivia, with a key pair $(\overline{sk_O}, \overline{vk_O})$ of a digital signature $\overline{\mathsf{DS}}$ (possibly

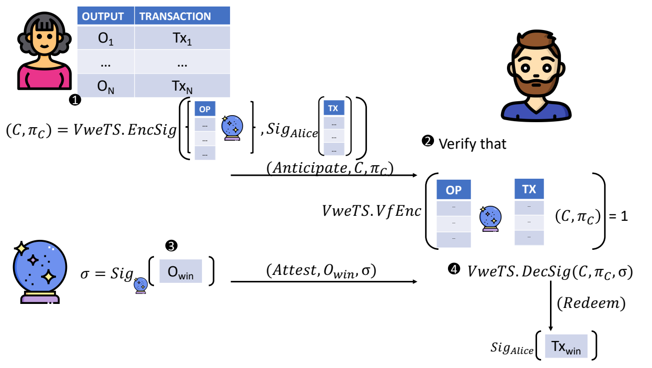

Figure 1: Overview: ❶ Alice computes a signature on each transaction that corresponds to a different output. These transactions are then encrypted using a verifiable witness encryption, where the witness is a signature on the corresponding output. Alice sends these ciphertexts to Bob. ❷ Bob verifies 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.

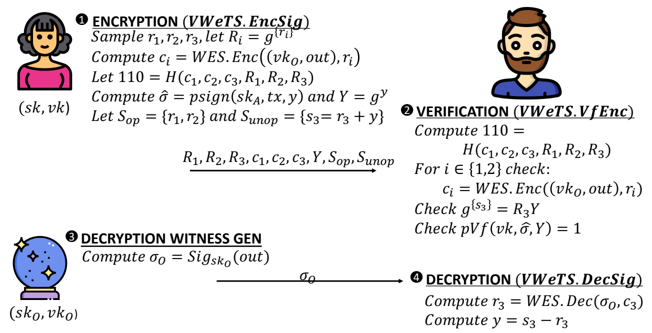
❶ ENCRYPTION ($VWeTS.EncSig$)
Sample r_1, r_2, r_3, let $R_i = g^{\{r_i\}}$
Compute $c_i = WES.Enc((vk_O, out), r_i)$
Let $110 = H(c_1, c_2, c_3, R_1, R_2, R_3)$
Compute $\hat{\sigma} = psign(sk_A, tx, y)$ and $Y = g^y$
Let $S_{op} = \{r_1, r_2\}$ and $S_{unop} = \{s_3 = r_3 + y\}$
(sk, vk)
$R_1, R_2, R_3, c_1, c_2, c_3, Y, S_{op}, S_{unop}$
❷ VERIFICATION ($VWeTS.VfEnc$)
Compute $110 = H(c_1, c_2, c_3, R_1, R_2, R_3)$
For $i \in \{1,2\}$ check:
$c_i = WES.Enc((vk_O, out), r_i)$
Check $g^{\{s_3\}} = R_3 Y$
Check $pVf(vk, \hat{\sigma}, Y) = 1$
❸ DECRYPTION WITNESS GEN
Compute $\sigma_O = Sig_{sk_O}(out)$
(sk_O, vk_O)
σ_O
❹ DECRYPTION ($VWeTS.DecSig$)
Compute $r_3 = WES.Dec(\sigma_O, c_3)$
Compute $y = s_3 - r_3$
Compute $\sigma = Adapt(\hat{\sigma}, y)$

Figure 2: Example of VweS: ❶ Alice encrypts three random values, and the (non-interactive) cut-and-choose directs Alice to open indices 1 and 2. Alice computes $s_3 = r_3 + y$, where y is the witness required to adapt $\hat{\sigma}$ to σ. ❷ Bob is able to verify that the encryption in indices 1 and 2 were computed correctly. By the cut and choose guarantee, the *BF-cipher* c_3 is also computed correctly. Moreover, the *sym-sipher* is also verified by checking $g^{s_3} = R_3 Y$ ❸ The oracle provides a witness - σ_O. ❹ Bob uses this signature and decrypts c_3 to get r_3. He can then compute $y = s_3 - r_3$ and then uses y to adapt $\hat{\sigma}$ to get σ

different to DS). To keep things simple, we assume that Olivia is honest (we will remove this assumption later). One trivial solution is to resort to the notion of witness encryption [19]: Alice can create a ciphertext that includes $\sigma \leftarrow \mathsf{Sign}(sk_A, tx)$ and that can only be decrypted if Bob has a witness (i.e, $\overline{\sigma}$) of the NP statement:

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

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: (i) For starters, general purpose constructions of witness encryption are prohibitively expensive. Second, (ii) 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. See Figure 1 for an overview of our scheme.

Efficient Witness Encryption for Signatures. Our first observation is that the Boneh-Franklin (BF) identity-based encryption [7] 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 $H(\mathrm{id})^s$, 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 $H(\mathrm{id})^s$ as the secret key. We observe that this is exactly the same structure that BLS [9] 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 $\overline{DS}$ is instantiated using the BLS signature scheme. Recall however that our goal is to let Alice encrypt a signature σ on tx 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) [4]. In brief, AS allows Alice to generate a pre-signature $\hat{\sigma}$ on tx, which is a verifiable encryption of a signature σ wrt. an NP statement $\{Y |\ Y := g^y\}$ 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: (i) create a pre-signature $\hat{\sigma}$ on tx using a statement Y previously agreed with Bob; (ii) use the BF-based witness encryption scheme mentioned above to encrypt y into ciphertext c for the identity $(\overline{vk}_O, \overline{m})$; (iii) send $\hat{\sigma}$ and c to Bob. As soon as Olivia attests the event $\overline{m}$ by publishing a BLS signature with her key $\overline{sk}_O$, 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. [11]. In a nutshell, Alice computes a pre-signature on the message as before and instead of generating a single BF ciphertext (*BF-cipher*), Alice 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 $(\overline{vk}_O, \overline{m})$. 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[1] $\lambda/2$ 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 correctly encrypting the value y using the randomness r_i as symmetric key of the one-time pad;

[1]This can be made non-interactive applying the Fiat-Shamir transformation.

- For all $j \in [\lambda/2]$ chosen by Bob, recompute the BF ciphertext of r_j with random coins and check if it is the same as sent by Alice.

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. We present an illustrative example in Fig. 2.

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-off-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 [23] 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 $2NB$ number 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*. The random mapping is also specified by Bob. As before, each of the j-th "unopened" *BF-cipher* in the i-th bucket denoted by $c_{i,j}$, is also associated with the *sym-cipher* value $s_{i,j} := r_j + y_i$, where r_j is the value encrypted in the *BF-cipher* $c_{i,j}$, and recall that y_i is the i-th share of the adaptor witness y. The high level idea is that the i-th bucket is now associated with the instance verification key $\overline{vk}_i$. The soundness guarantee of this batching technique is that, with overwhelming probability, there exists a $j' \in [B]$ in each bucket $i \in [N]$, such that the *BF-cipher* $c_{i,j'}$ is a well-formed BF ciphertext and the underlying message $r_{j'}$ satisfies the check

$$g^{s_{i,j'}} \stackrel{?}{=} g^{r'_j} \cdot Y_i.$$

The hope now is that if we have a witness BLS signature $\overline{\sigma}_i$ on the message $\overline{m}$ that is valid w.r.t. a key $\overline{vk}_i$, then Bob is able to decrypt $c_{i,j'}$ to obtain $r_{j'}$ and consequently the witness share y_i. If we have ρ number of witness BLS signatures $(\overline{\sigma}_i)_{i \in K}$, where $K \subset [N]$ and $|K| = \rho$, then we are guaranteed to obtain ρ number of valid witness shares $(y_i)_{i \in K}$, and we can reconstruct the adaptor witness y, and adapt the pre-signature $\hat{\sigma}$ into a valid signature σ.

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. Therefore, it is unclear how we generate each of the *BF-cipher*, meaning, it is unknown at the stage of the *BF-cipher* generation w.r.t. which instance verification key do we set it to. In fact, it is necessary for the soundness cut-and-choose batching that we do not know the random mapping during the ciphertext generation. To tackle this issue, during *BF-cipher* generation, we generate each of $2NB$ number of *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. It turns out that such NIZK corresponds to 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 [5]). The rest of the cut-and-choose proceeds as before.

To sum up, Alice returns the $2NB$ *BF-ciphers* generated wrt. $\overline{m}^*$ and $\overline{vk}^*$, the pre-signature $\hat{\sigma}$, values g^{r_j} for all $j \in [2NB]$, adaptor statement shares $(Y_1, \dots, Y_N)$, the "opened" values and the "unopened" values similar to the simplified case above, but now additionally consists of the new *BF-ciphers* generated wrt. the correct instance message and the instance verification corresponding to the bucket, and the associated NIZK proofs as described above. The verification by Bob is canonical, with only additional checks needed for the correctness of the bucket mapping and the validity of the NIZK proofs for the "unopened" *BF-ciphers*. Given witness BLS signature $\overline{\sigma}_i$ on the message $\overline{m}^*$ w.r.t. key $\overline{vk}_i$, Bob decrypts all the *BF-cipher* $c_{i,j}$ in the i-th bucket. He then obtains shares of y as described before, and provided that Bob has ρ of them, he can reconstruct y and later adapt $\hat{\sigma}$ into the valid signature σ.

Extensions. We can extend the above techniques to the case where we have M different messages $(\overline{m}_1, \dots, \overline{m}_M)$ instead of just one. In this case, Alice has transaction tx_i paying to Bob if the message $\overline{m}_i$ is attested. For more details on this general case, we refer the reader to Section 4.2. We extend our techniques even to the case where the signature scheme for authorizing a transaction, i.e., DS is the BLS signature scheme. Note that it was shown in [18] that it is impossible to construct an AS scheme for BLS signatures. Thus, we resort to different techniques to achieve our goal of constructing verifiable witness encryption based on threshold signatures (VweTS). We include more details on our BLS based construction in Appendix E.

Furthermore, in the protocols overviewed so far, the communication complexity grows linearly in the number of messages $\bar{m}$. In particular, this implies that the number of messages must be bounded by a given polynomial (in the security parameter). In Appendix G we outline how to modify our protocol to remove this bound and support an event with an *exponential* number of outcomes, without increasing the communication complexity of the protocol proportionately.

3 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 $R : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}$ 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) [15] 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 soundness*, 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 [27] 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 a secret s via polynomial interpolation. The security of the secret sharing scheme demands that knowing only a set of shares smaller than the threshold size does *not* help in learning 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 [4] 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* DS $= (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ *consists of algorithms* $(\mathsf{pSign}, \mathsf{Adapt}, \mathsf{pVf}, \mathsf{Ext})$ *defined as:*

$\underline{\hat{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)}$*: 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 $\hat{\sigma}$.*

$\underline{0/1 \leftarrow \mathsf{pVf}(vk, m, Y, \hat{\sigma})}$*: 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).*

$\underline{\sigma \leftarrow \mathsf{Adapt}(\hat{\sigma}, y)}$*: The adapt algorithm takes as input a pre-signature $\hat{\sigma}$ and witness y, outputs a signature σ.*

$\underline{y \leftarrow \mathsf{Ext}(\sigma, \hat{\sigma}, Y)}$*: 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. 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:*

$\underline{c \leftarrow \mathsf{Enc}((\tilde{vk}, \tilde{m}), m)}$*: 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 returns as output a ciphertext c.*

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

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

Figure 3: Experiment for CPA security of a witness encryption scheme based on signatures.

$m \leftarrow \mathsf{Dec}(\tilde{\sigma}, c)$*: the decryption algorithm takes as input a signature* $\tilde{\sigma}$ *and the ciphertext* c. *It returns as output a message* m.

The correctness of a witness encryption based on signatures is defined below.

Definition 3 (Correctness of Witness Encryption for Signatures). *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* $\mathsf{Vf}(\tilde{vk}, \tilde{m}, \tilde{\sigma}) = 1$.

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,*

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

where IND-CPA *is defined in Fig. 3.*

We give a construction for WES based on the BLS signature scheme. Our construction described in Fig. 4 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 $H_1 : \mathbb{Z}_q \rightarrow \{0,1\}^\lambda$.

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

- Sample $r_1, r_2 \leftarrow \mathbb{Z}_q$.
- 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 $c := (c_1, c_2, c_3)$.

$\mathsf{Dec}(\tilde{\sigma}, c)$: 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 $m := c_3 - h$.

Figure 4: Witness encryption based on BLS signatures

The security of the construction follows similar to the IBE scheme from [7] based on Bilinear Diffie-Hellman assumption, when modelling the hash functions H_0 and H_1 as random oracles.

4 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

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

where $(\overline{vk}_1, \ldots, \overline{vk}_N) \in \mathsf{SUPP}(\overline{\mathsf{KGen}}(1^\lambda))$.

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 $\mathcal{L}$ as defined above.

4.1 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* verifiable witness encryption based on threshold signatures *scheme denoted by* (ρ, N, M)-$\mathsf{VweTS} :=$ $(\mathsf{EncSig}, \mathsf{VfEnc}, \mathsf{DecSig})$ *is parameterized by* $\rho, N, M \in \mathbb{N}$ *and defined with respect to signature schemes* $\mathsf{DS} := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ *and* $\overline{\mathsf{DS}} := (\overline{\mathsf{KGen}}, \overline{\mathsf{Sign}}, \overline{\mathsf{Vf}})$ *is said to be* correct *if the following holds. 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, we have the following that hold simultaneously:*

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

(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*

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

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* verifiable witness encryption based on threshold signatures *scheme denoted by* (ρ, N, M)-$\mathsf{VweTS} :=$ $(\mathsf{EncSig}, \mathsf{VfEnc}, \mathsf{DecSig})$ *is parameterized by* $\rho, N, M \in \mathbb{N}$ *and defined with respect to signature schemes* $\mathsf{DS} := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ *and* $\overline{\mathsf{DS}} := (\overline{\mathsf{KGen}}, \overline{\mathsf{Sign}}, \overline{\mathsf{Vf}})$ *is said to be* 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,*

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

where ExpOWay *is defined in Fig. 5.*

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 the definition below.

Definition 8 (Verifiability). *A* verifiable witness encryption for threshold signatures *scheme denoted by* (ρ, N, M)-$\mathsf{VweTS} :=$

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

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

$\overline{\mathsf{Sign}}O(i, \overline{m})$
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 $\overline{\sigma}$

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

Figure 5: Experiment for one-wayness.

$(\mathsf{EncSig}, \mathsf{VfEnc}, \mathsf{DecSig})$ *parameterized by* $\rho, N, M \in \mathbb{N}$ *and defined with respect to signature schemes* $\mathsf{DS} := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ *and* $\overline{\mathsf{DS}} :=$ $(\overline{\mathsf{KGen}}, \overline{\mathsf{Sign}}, \overline{\mathsf{Vf}})$ *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* $((m_j, \overline{m}_j)_{j\in[M]}, vk, (\overline{vk}_i)_{i\in[N]}, (\overline{\sigma}_j)_{j\in K}, j^*, c, \pi_c)$ *such that all the following holds simultaneously except with probability* $\mathsf{negl}(\lambda)$*:*

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

4.2 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:

(1) Signature scheme $\overline{\mathsf{DS}} := (\overline{\mathsf{KGen}}, \overline{\mathsf{Sign}}, \overline{\mathsf{Vf}})$ instantiated with BLS signature scheme (see Appendix B).
(2) Signature scheme $\mathsf{DS} := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ that is either Schnorr or ECDSA signature schemes (see Appendix B), based on a group $\mathbb{G}$ with generator g and order q.

(3) Witness encryption based on signatures $\mathsf{WES} := (\mathsf{Enc}, \mathsf{Dec})$ scheme (see Fig. 4 for a concrete candidate).
(4) 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: $\mathcal{L}_R := \{Y : \exists y \in \mathbb{Z}_q^*,\ s.t.\ Y = g^y\}$.
(5) A NIZK proof $(\mathsf{Setup}_{\mathcal{L}_c}, \mathsf{Prove}_{\mathcal{L}_c}, \mathsf{Vf}_{\mathcal{L}_c})$ for the language

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

where $(\overline{vk}_1, \cdot)$ and $(\overline{vk}_2, \cdot)$ are in the support of $\overline{\mathsf{KGen}}$.

Our construction of VweTS based on BLS signatures follows a similar outline and is therefore deferred to Appendix E due to space constrains.

```
**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) Compute $\{\Phi, (b_1, \dots, b_\gamma)\} := H_2((c'_i, R_i)_{i\in[\gamma]})$.
(4) 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 $j \in \{\rho, \dots, N\}$ 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.
(5) Set $\Sigma_1 := (\hat{\sigma}_i, Y_i, \{Y_{i,j}\}_{j\in[N]})_{i\in[M]}$.
(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 $\{\Phi, (b_1, \dots, b_\gamma)\} := H_2((c'_i, R_i)_{i\in[\gamma]})$
(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 $\mathsf{rShare}_i = \emptyset$.
(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 $\sigma_j \leftarrow \mathsf{AS.Adapt}(\hat{\sigma}_j, y_j)$.
```

Figure 6: Verifiable witness encryption based on threshold signatures from adaptor signatures.

Parameters. We assume the setup algorithm $\mathsf{Setup}_{\mathcal{L}_c}$ has been executed and the resulting *crs* is part of public parameters which also include the group descriptions of groups $\mathbb{G}, \mathbb{G}_0, \mathbb{G}_1$ and $\mathbb{G}_2$, the value q which is the order of the group $\mathbb{G}$, a value $\gamma := 2NMB$ where B is a statistical parameter, a mapping function $\Phi : [\gamma] \to [M]\times[N]$ with $\gamma := 2NMB$, and the description of the hash function $H_2 : \{0,1\}^* \to I$ such that $I \in ([\gamma] \to [M]\times[N]) \cup \{0,1\}^\gamma$, modeled as a random oracle.

Overview. We present a high level overview of our construction, and the formal description is given in Fig. 6. The signature encryption algorithm first generates γ number of WES ciphertexts such that ciphertext c'_i encrypts a random integer r_i from $\mathbb{Z}_q$ wrt. the instance $(\overline{vk}^*, \overline{m}^*)$. Here $\overline{vk}^*$ and $\overline{m}^*$ are random verification key and message, respectively. It also encodes the integer r_i in the exponent by setting $R_i := g^{r_i}$. A bucket mapping Φ and γ bit values are generated by applying the Fiat-Shamir transform using the hash H_2. The algorithm generates for each $i \in [M]$ a 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 $Y_{i,j} := g^{y_{i,j}}$.

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 NIZK proof are added to the set $\mathcal{S}_{\mathsf{unop}}$. These values are considered to be unopened by the cut-and-choose. 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 first checks the correctness of the Fiat-Shamir transform, and checks the well-formedness of the opened values in $\mathcal{S}_{\mathsf{op}}$ against the WES ciphertexts generated wrt. instance $(\overline{vk}^*, \overline{m}^*)$. It then checks 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.

To decrypt the j-th signature, we require at least ρ valid witness signatures on the instance message $\overline{m}_j$ wrt. any ρ verification keys in $(\overline{vk}_i)_{i\in[N]}$. 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. 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 $R_a = g^{r_{i,a}}$. 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 the previous step. We obtain ρ witness shares $y_{j,i}$ using which we can reconstruct the adaptor witness y_j. 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 Appendix C, we formally show that our construction satisfies correctness according to Definition 6. Security of our construction is formally stated in the following theorem, and the proof is deferred to Appendix D.

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 Fig. 6 is one-way and verifiable according to Definition 7 and Definition 8, respectively.*

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 [5], 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 [26].

4.3 Large Universe of Outcomes

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 outline 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. In Appendix G we present a more detailed description of the construction along with the security analysis. We also show a different scheme, which is asymptotically optimal, but concretely less efficient.

In the modified EncSig algorithm, we additionally compute

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

where $z_{b,i} \leftarrow \mathbb{Z}_q$, along with

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

for all $j = \{1, \dots, M\}$, where $j[i]$ denotes the i-*th* bit of j. Instead of witness encrypting $(y_1, \dots, y_M)$, we witness encrypt $\{z_{b,i}\}$, each conditioned on knowing the signatures of a large enough fraction of oracles that attest that the i-th bit of the message equals b. The verification procedure is unchanged, except that we add the following check

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

for all $j = \{1, \ldots, M\}$. Assuming that the signatures of $\overline{\mathsf{DS}}$ are on a suitable encoding of the bits of the messages[2], then it is not hard to see that obtaining a large enough fraction of signatures, allows one to successfully decrypt. In particular, obtaining enough signatures on $j = (j[1], \ldots, j[\mu])$, 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 one to unmask y_j and consequently to recover σ_j calling the AS.Adapt algorithm.

5 ORACLE CONTRACTS

We present the interfaces for oracle contracts and we formalize their security properties, namely unforgeability, verifiability and attestation unforgeability. Then we present a construction based on VweTS.

Definition 9 (Oracle Contracts). Oracle Contracts *is a protocol parameterized by* $\rho, N, M \in \mathbb{N}$ *(s.t.* $\lceil \frac{N}{2} \rceil \leq \rho \leq N$*) and run among a set of entities: N oracles* $\{O_1, \ldots, O_N\}$*, and two users, Alice A (signing party) and Bob B (verifying party). The oracle contracts protocol is defined with respect to a digital signature scheme* $\Pi_{\mathsf{BDS}} := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ *and consists of five PPT algorithms* (OKGen, Attest, AttestVf, Anticipate, AnticipateVf, Redeem)*, that are defined below.*

- $\underline{(pk^O, sk^O) \leftarrow \mathsf{OKGen}(1^\lambda)}$*: the oracle key generation algorithm takes as input the security parameter* λ *and outputs the oracle public key* pk^O *and the corresponding oracle secret key* sk^O*.*
- $\underline{att \leftarrow \mathsf{Attest}(sk^O, o)}$*: the event attestation algorithm takes as input oracle's secret key* sk^O*, and the event outcome o, and outputs the outcome attestation att.*
- $\underline{\{0,1\} \leftarrow \mathsf{AttestVf}(pk^O, att, o)}$*: the attestation verification algorithm takes as input oracle's public key* pk^O*, the outcome attestation att and the outcome o, and returns* 1 *if att attests to o being the outcome of the event and* 0 *otherwise.*
- $\underline{ant \leftarrow \mathsf{Anticipate}(sk_A, (pk_i^O)_{i\in[N]}, (o_j, \mathsf{Tx}_j)_{j\in[M]})}$*: the attestation anticipation algorithm takes as input the signing party's secret key* sk_A*, oracles' public keys* $(pk_i^O)_{i\in[N]}$*, and tuples of outcomes and transactions* $(o_j, \mathsf{Tx}_j)_{j\in[M]}$*, and outputs the anticipation ant.*
- $\underline{\{0,1\} \leftarrow \mathsf{AnticipateVf}(pk_A, ant, (pk_i^O)_{i\in[N]}, (o_j, \mathsf{Tx}_j)_{j\in[M]})}$*: the anticipation verification algorithm takes as inputs the signing party's public key* pk_A*, the anticipation ant, oracles' public keys* $(pk_i^O)_{i\in[N]}$*, and tuples of outcomes and transactions* $(o_j, \mathsf{Tx}_j)_{j\in[M]}$*, and outputs* 1 *if ant is well-formed and* 0 *otherwise.*
- $\underline{\sigma \leftarrow \mathsf{Redeem}(j, (att_i)_{i\in[K]}, ant)}$*: the redeem algorithm takes as input an index* $j \in [M]$*, attestations* $(att_i)_{i\in[K]}$ *for* $|K| = \rho$ *and* $K \subset [N]$*, and the anticipation ant. It returns as output a signature* σ *on the transaction* Tx_j*.*

[2]E.g., each bit should also be signed together with its position, to avoid mix-and-match attacks

An oracle contract scheme is correct if (i) honestly created attestations verify correctly; (ii) honestly generated attestation anticipations verify correctly; and (iii) honestly generated anticipations and attestations are redeemable. We defer a formal definition to Appendix H. We additionally introduce the security notions of interest for oracle contracts. We give here their intuition and refer the reader to Appendix H for the formal details.

We first introduce the notion of unforgeability. Unforgeability means that an adversary cannot redeem a contract on an outcome that is different from the winning outcome announced by the oracles. A second notion of interest in oracle contracts is verifiability. With verifiability, we aim to capture the property that if an anticipation is correctly computed and verified, a conditional payment on this anticipation is redeemable by the counter-party except with negligible probability.

Another notion of interest in oracle contracts is attestation unforgeability. Attestation unforgeability means that an adversary cannot counterfeit an attestation from an oracle. We note that the notion of attestation unforgeability is important to ensure that the scheme achieves *accountability*. With accountability, we aim to capture the property that, if an oracle attests to more than one outcome for an event, it can be detected by Alice and Bob. In case of a dispute between Alice and Bob regarding the correct outcome (where Alice claims outcome j and Bob claims outcome j'), they are both asked to present ρ valid signatures on j and j'. We then distinguish three cases:

(1) Alice fails to present valid signatures on j: In this case, Alice is blamed, since she cannot substantiate the outcome with signatures on behalf of the oracles.

(2) Bob fails to present valid signatures on j': Analogously, in this case, Bob is blamed.

(3) Both Alice and Bob present enough signatures on both j and j'. Then, there must exist an oracle that signed two different outcomes for a given event (since $\rho > N/2$), which is blamed. Note that Alice and Bob cannot frame the oracles without breaking the attestation unforgeability of the signature scheme of the oracles.

5.1 Our Protocol

In this section, we present a concrete construction of oracle contracts with parameters ρ, N and M relying on the VweTS cryptographic building block. We set $\rho > N/2$. More precisely, algorithms OKGen, Attest, and AttestVf are instantiated using the signature scheme $\overline{\mathsf{DS}} := (\overline{\mathsf{KGen}}, \overline{\mathsf{Sign}}, \overline{\mathsf{Vf}})$, algorithms Anticipate, AnticipateVf and Redeem are instantiated using the verifiable witness encryption based on threshold signatures scheme VweTS := (EncSig, VfEnc, DecSig) and the signature scheme $\Pi_{\mathsf{BDS}} := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ is mapped to $\mathsf{DS} := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$. The formal description of our construction is given in Fig. 7. Moreover, we state below our formal security claim and defer the formal proofs to the Appendix J.

Theorem 2 (Oracle contract security). *Let* (ρ, N, M)-VweTS *be a* one-way *verifiable witness encryption for threshold signatures scheme defined with respect to* $\mathsf{DS} := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ *and* $\overline{\mathsf{DS}} := (\overline{\mathsf{KGen}}, \overline{\mathsf{Sign}}, \overline{\mathsf{Vf}})$*. Let* $\overline{\mathsf{DS}} := (\overline{\mathsf{KGen}}, \overline{\mathsf{Sign}}, \overline{\mathsf{Vf}})$ *be an EUF-CMA secure digital signature scheme. Then, our protocol is an* unforgeable, verifiable *and* attestation unforgeable (ρ, N, M)*-oracle contract protocol defined with respect to the signature scheme* $\Pi_{\mathsf{BDS}} := \mathsf{DS}$*.*

Oracle Key Generation: Algorithm $\underline{\mathsf{OKGen}(1^\lambda)}$ is run by oracles O_i for $i \in [N]$, which does the following:

- Sample keys $(\overline{vk}_i, \overline{sk}_i) \leftarrow \overline{\mathsf{DS}}.\overline{\mathsf{KGen}}(1^\lambda)$
- Return $(pk_i^O, sk_i^O) := (\overline{vk}_i, \overline{sk}_i)$.

Event Attestation: Algorithm $\underline{\mathsf{Attest}(sk_i^O, o)}$ is run by the oracles O_i for $i \in [N]$, which does the following:

- Parse $sk_i^O := \overline{sk}_i$
- Generate $\overline{\sigma}_i \leftarrow \overline{\mathsf{DS}}.\overline{\mathsf{Sign}}(\overline{sk}_i, o)$.
- Return $att_i := \overline{\sigma}_i$.

Attestation Verification: Algorithm $\underline{\mathsf{AttestVf}(pk^O, att, o)}$ does the following:

- Parse $pk_i^O := \overline{vk}_i$ and $att := \overline{\sigma}_i$
- Check if $\overline{\mathsf{DS}}.\overline{\mathsf{Vf}}(\overline{vk}_i, o, \overline{\sigma}_i) = 1$
- Return 1 if the above check is successful, and 0 otherwise.

Event Anticipation: Algorithm $\underline{\mathsf{Anticipate}(sk_A, (pk_i^O)_{i\in[N]}, (o_j, \mathsf{Tx}_j)_{j\in[M]})}$ does the following:

- Parse $sk_A := sk$ and $(pk_i^O)_{i\in[N]} := (\overline{vk}_i)_{i\in[N]}$
- Set $(c, \pi_c) \leftarrow$ $\mathsf{VweTS.EncSig}(((\overline{vk}_i)_{i\in[N]}, (o_j)_{j\in[M]}), sk, (\mathsf{Tx}_j)_{j\in[M]})$
- Return $ant := (c, \pi_c)$.

Anticipation Verification: Algorithm $\underline{\mathsf{AnticipateVf}(pk_A, ant, (pk_i^O)_{i\in[N]}, (o_j, \mathsf{Tx}_j)_{j\in[M]})}$ does the following:

- Parse $ant := (c, \pi_c)$, $pk_A := vk$ and $(pk_i^O)_{i\in[N]} := (\overline{vk}_i)_{i\in[N]}$
- Check if $\mathsf{VweTS.VfEnc}(c, \pi_c, ((\overline{vk}_i)_{i\in[N]}, (o_j, \mathsf{Tx}_j)_{j\in[M]}, vk)) = 1$
- Return 1 if the above check is successful, and 0 otherwise.

Contract Redeem: Algorithm $\underline{\mathsf{Redeem}(j, (att_i)_{i\in[K]}, ant)}$ does the following:

- Parse $ant := (c, \pi_c)$ and $(att_i)_{i\in[K]} := (\overline{\sigma}_i)_{i\in[K]}$
- Set $\sigma \leftarrow \mathsf{VweTS.DecSig}(j, \{\overline{\sigma}_i\}_{i\in[K]}, c, \pi_c)$
- Return σ

Figure 7: Oracle Contracts construction based on VweTS.

6 PERFORMANCE ANALYSIS

In this section, we describe the implementation and evaluate the practicality of our protocol for oracle contracts. We also suggest some implementation-level optimizations for VweTS.

6.1 Implementation

We have developed a prototypical Rust implementation [2] to demonstrate the feasibility of our construction. The implementation encompasses the *event anticipation* (i.e., Anticipate algorithm), the *anticipation verification* (i.e., AnticipateVF algorithm) and the *contract redeem* (i.e., Redeem algorithm), thus simulating the functionality that would be performed by Alice and Bob in the protocol. We omit the operations regarding attestations since they are simple signature creation and verification of a digital signature scheme.

Implementation-level Optimizations. Alice can pre-compute several of the operations required in the VweTS.EncSig algorithm (see Fig. 6), 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.

6.2 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 three system parameters: (i) the security parameter of the cut-and-choose; (ii) the number of oracles; and (iii) the number of outcomes. For each parameter we study, we vary this parameter while we fix a value for the other two. We thereby compute the impact of such parameter in the overall process, that is, the execution of Anticipate, AnticipateVF and Redeem. The results are shown in Figure 8.

In the view of these results, we make the following observations. First, augmenting the security parameter of the cut-and-choose used within VweTS has the least impact on both running time and communication time. Second, augmenting the number of oracles, as well as the threshold, 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. Third, the number of outcomes seems to be the most impactful system parameter, since both running time and communication overhead seem to grow worse than linearly on the number of outcomes. In any case, even in the possibly unrealistic setting of considering 1000 outcomes, the running time and communication overhead are well within reach of commodity hardware.

6.3 Further Optimizations and Comparison

Runtime and Communication Overhead. After further inspection of our implementation, we have observed that one of the main bottlenecks is the handle of the pairing operations appearing during the VweTS. Intuitively, they appear since we require a witness encryption where the witness is a BLS signature from the oracles. We have thus observed that we can improve the performance if we change the digital signature of the oracle attestations to be one in the DLog setting (e.g., Schnorr) and adapt VweTS accordingly. In this modified implementation, available at [2], we have used ristretto to implement the elliptic curve. To illustrate the performance gain, we have tested the same parameters set as described in the previous section with this modified implementation, with the results shown in Figure 9. We make two main observations. First, in all the tested setting, the optimization shows a tremendous gain in both running time and communication overhead. Second, the impact of the different settings is maintained, that is, the number of outcomes keeps having the highest impact in both running time and communication overhead, while the security parameter has the lowest.

Increasing the Number of Oracles. In this experiment, we want to compare the performance of our design with that of DLC when increasing the number of oracles. For that, we have obtained a

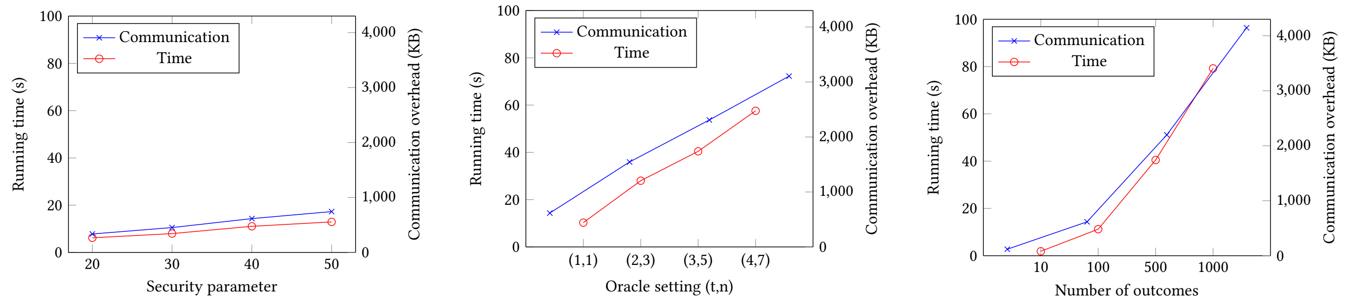

Figure 8: Impact of the security parameter (left), oracle setting (middle) and number of outcomes (right) on running time (red) and communication overhead (blue). For this evaluation, we have set up the parameters as follows: Left: 1 oracle, 100 outcomes; Middle: security parameter 40, 100 outcomes; Right: 1 oracle, security parameter 40.

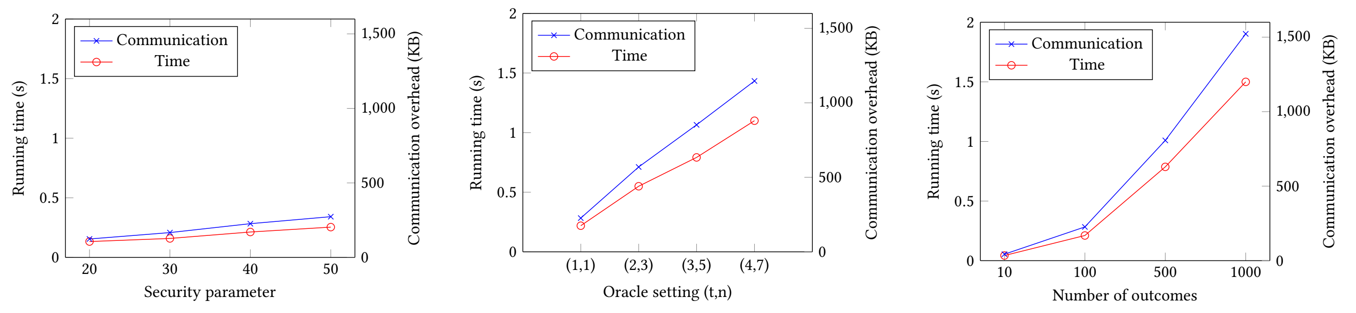

Figure 9: Impact of the security parameter (left), oracle setting (middle) and number of outcomes (right) on running time (red) and communication overhead (blue). For this evaluation, we have set up the parameters as follows: Left: 1 oracle, 100 outcomes; Middle: security parameter 40, 100 outcomes; Right: 1 oracle, security parameter 40.

prototype implementation of the DLC design [2] where we have tested it for an increasing number of oracles. The results are shown in Figure 10. We observe that running time of the DLC approach is the one scaling the worst when we increase the number of oracles. 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 [14]. Therefore, starting from a small setting of 5 oracles and a threshold of $t = 3$, their approach is less efficient and the different keeps growing as we increment the values of n and t.

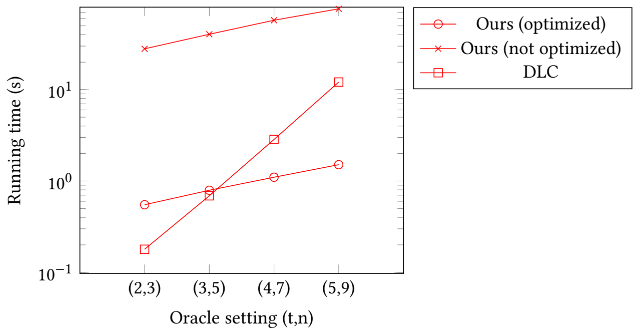

Figure 10: Running time of our approach and DLC with an increasing number of oracles. We fix security parameter to 40 and number of outcomes to 100. The y-axis is in log scale.

7 CONCLUSIONS

In this work, we investigate the problem of oracle contracts that do not require Turing-complete language or are based on the trusted execution environment. In particular, we design game-based definitions that model the security properties of oracle contracts, and we propose the first construction with provable security guarantees that is compatible with many cryptocurrencies today, including Bitcoin. As a contribution of independent interest, we design an efficient protocol for witness encryption for the general class of languages $\{(vk, m) \in \mathcal{L} : \exists\, \sigma \text{ s.t. } \mathsf{Verify}(vk, \sigma, m) = 1\}$, where σ is a BLS digital signature on m. Moreover, we show extensions to (i) the threshold setting; (ii) how to efficiently prove that the encrypted message has a certain structure; and (iii) how to support an event with an exponential number of outcomes without increasing the communication complexity of the protocol proportionally. Finally, we provide a prototypical implementation and evaluated, showing that not only our construction is practical even in commodity hardware but also copes better with a growing number of oracles than the currently proposed Discreet Log Contracts approach.

REFERENCES

[1] [n.d.]. DeFi Pulse Website. https://www.defipulse.com/.

[2] [n.d.]. Source code for this project. https://sites.google.com/view/ccs343implementation/home.

[3] Arash Afshar, Payman Mohassel, Benny Pinkas, and Ben Riva. 2014. Non-Interactive Secure Computation Based on Cut-and-Choose. In *EUROCRYPT 2014 (LNCS, Vol. 8441)*, Phong Q. Nguyen and Elisabeth Oswald (Eds.). Springer, Heidelberg, Germany, Copenhagen, Denmark, 387–404. https://doi.org/10.1007/978-3-642-55220-5_22

[4] Lukas Aumayr, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostáková, Matteo Maffei, Pedro Moreno-Sanchez, and Siavash Riahi. 2021. Generalized channels from limited blockchain scripts and adaptor signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 635–664.

[5] Mihir Bellare, Alexandra Boldyreva, and Jessica Staddon. 2003. Randomness Re-use in Multi-recipient Encryption Schemeas. In *PKC 2003 (LNCS, Vol. 2567)*, Yvo Desmedt (Ed.). Springer, Heidelberg, Germany, Miami, FL, USA, 85–99. https://doi.org/10.1007/3-540-36288-6_7

[6] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. 2012. Foundations of garbled circuits. In *ACM CCS 2012*, Ting Yu, George Danezis, and Virgil D. Gligor (Eds.). ACM Press, Raleigh, NC, USA, 784–796. https://doi.org/10.1145/2382196.2382279

[7] Dan Boneh and Matthew K. Franklin. 2001. Identity-Based Encryption from the Weil Pairing. In *CRYPTO 2001 (LNCS, Vol. 2139)*, Joe Kilian (Ed.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 213–229. https://doi.org/10.1007/3-540-44647-8_13

[8] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. 2003. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *EUROCRYPT 2003 (LNCS, Vol. 2656)*, Eli Biham (Ed.). Springer, Heidelberg, Germany, Warsaw, Poland, 416–432. https://doi.org/10.1007/3-540-39200-9_26

[9] Dan Boneh, Ben Lynn, and Hovav Shacham. 2001. Short Signatures from the Weil Pairing. In *ASIACRYPT 2001 (LNCS, Vol. 2248)*, Colin Boyd (Ed.). Springer, Heidelberg, Germany, Gold Coast, Australia, 514–532. https://doi.org/10.1007/3-540-45682-1_30

[10] Jo Van Bulck, David F. Oswald, Eduard Marin, Abdulla Aldoseri, Flavio D. Garcia, and Frank Piessens. 2019. 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*, Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz (Eds.). ACM, 1741–1758. https://doi.org/10.1145/3319535.3363206

[11] Jan Camenisch and Ivan Damgård. 2000. Verifiable Encryption, Group Encryption, and Their Applications to Separable Group Signatures and Signature Sharing Schemes. In *ASIACRYPT 2000 (LNCS, Vol. 1976)*, Tatsuaki Okamoto (Ed.). Springer, Heidelberg, Germany, Kyoto, Japan, 331–345. https://doi.org/10.1007/3-540-44448-3_25

[12] Ran Canetti, Abhishek Jain, and Alessandra Scafuro. 2014. Practical UC security with a Global Random Oracle. In *ACM CCS 2014*, Gail-Joon Ahn, Moti Yung, and Ninghui Li (Eds.). ACM Press, Scottsdale, AZ, USA, 597–608. https://doi.org/10.1145/2660267.2660374

[13] Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, and Ten-Hwang Lai. 2020. SgxPectre: Stealing Intel Secrets From SGX Enclaves via Speculative Execution. *IEEE Secur. Priv.* 18, 3 (2020), 28–37. https://doi.org/10.1109/MSEC.2019.2963021

[14] DLC community. [n.d.]. Specification for Discreet Log Contracts. https://github.com/discreetlogcontracts/dlcspecs.

[15] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. 1987. Non-interactive zero-knowledge proof systems. In *Conference on the Theory and Application of Cryptographic Techniques*. Springer, 52–72.

[16] Nico Döttling, Lucjan Hanzlik, Bernardo Magri, and Stella Wohnig. 2022. McFly: Verifiable Encryption to the Future Made Practical. *Cryptology ePrint Archive* (2022).

[17] Thaddeus Dryja. [n.d.]. Discreet Log Contracts. https://adiabat.github.io/dlc.pdf.

[18] Andreas Erwig, Sebastian Faust, Kristina Hostáková, Monosij Maitra, and Siavash Riahi. 2021. Two-Party Adaptor Signatures from Identification Schemes. In *PKC 2021, Part I (LNCS, Vol. 12710)*, Juan Garay (Ed.). Springer, Heidelberg, Germany, Virtual Event, 451–480. https://doi.org/10.1007/978-3-030-75245-3_17

[19] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. 2013. Witness encryption and its applications. In *45th ACM STOC*, Dan Boneh, Tim Roughgarden, and Joan Feigenbaum (Eds.). ACM Press, Palo Alto, CA, USA, 467–476. https://doi.org/10.1145/2488608.2488667

[20] Gideon Greenspan. [n.d.]. Why Many Smart Contract Use Cases Are Simply Impossible. https://www.coindesk.com/markets/2016/04/17/why-many-smart-contract-use-cases-are-simply-impossible/.

[21] Don Johnson, Alfred Menezes, and Scott Vanstone. 2001. The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security* 1, 1 (01 Aug 2001), 36–63. https://doi.org/10.1007/s102070100002

[22] Nadav Koheh. [n.d.]. Update on DLCs (new mailing list). https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2021-January/018372.html.

[23] Yehuda Lindell and Ben Riva. 2014. Cut-and-Choose Yao-Based Secure Computation in the Online/Offline and Batch Settings. In *CRYPTO 2014, Part II (LNCS, Vol. 8617)*, Juan A. Garay and Rosario Gennaro (Eds.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 476–494. https://doi.org/10.1007/978-3-662-44381-1_27

[24] Bowen Liu, Pawel Szalachowski, and Jianying Zhou. 2021. A First Look into DeFi Oracles. In *IEEE International Conference on Decentralized Applications and Infrastructures, DAPPS 2021, Online Event, August 23-26, 2021*. IEEE, 39–48. https://doi.org/10.1109/DAPPS52256.2021.00010

[25] LLFourn (pseudonym). [n.d.]. Secure DLCs. https://bitcoinproblems.org/problems/secure-dlcs.html.

[26] Claus-Peter Schnorr. 1990. Efficient Identification and Signatures for Smart Cards. In *CRYPTO'89 (LNCS, Vol. 435)*, Gilles Brassard (Ed.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 239–252. https://doi.org/10.1007/0-387-34805-0_22

[27] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.

[28] Sam M. Werner, Daniel Perez, Lewis Gudgeon, Ariah Klages-Mundt, Dominik Harz, and William J. Knottenbelt. 2021. SoK: Decentralized Finance (DeFi). *CoRR* abs/2101.08778 (2021). arXiv:2101.08778 https://arxiv.org/abs/2101.08778

[29] Fan Zhang, Ethan Cecchetti, Kyle Croman, Ari Juels, and Elaine Shi. 2016. 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*, Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi (Eds.). ACM, 270–282. https://doi.org/10.1145/2976749.2978326

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

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

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

Figure 11: Unforgeabiltiy experiment of adaptor signatures

A MORE PRELIMINARIES

A.1 Adaptor Signatures

Definition 10 (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 11 (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 as follows:*

Definition 12 (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 13 (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:*

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

where the experiment $\mathsf{aWitExt}_{\mathcal{A},\mathsf{AS}}$ *is defined as follows*

B SIGNATURE SCHEMES

BLS Signatures. We briefly recall here the BLS signature scheme [9]. 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 $H : \{0,1\}^* \rightarrow \mathbb{G}_1$.

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

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

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

Figure 12: Witness extractability experiment for adaptor signatures

- $(vk, sk) \leftarrow \mathsf{KGen}(1^\lambda)$: 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 [26], 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 [21] 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 $\sigma := (r, s)$.
- $0/1 \leftarrow \mathsf{Vf}(vk, m, \sigma)$: 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.

C PROOFS OF CORRECTNESS OF ADAPTOR BASED VweTS

Theorem 3. *Our* VweTS *construction from Fig. 6 is correct according to Definition 6.*

Proof. Let $(c, \pi_c) \leftarrow \mathsf{EncSig}(((\overline{vk}_i)_{i \in [N]}, (\overline{m}_j)_{j \in [M]}), sk, (m_j)_{j \in [M]})$. 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 $g^{s_i} \neq R_i \cdot Y_{\alpha,\beta}$. Note that by construction we have $s_i = r_i + y_{\alpha,\beta}$. This implies $g^{s_i} = g^{r_i} \cdot g^{y_{\alpha,\beta}} = R_i \cdot Y_{\alpha,\beta}$ and therefore this case never occurs.
(3) If $b_i = 1$ and $\mathsf{Vf}_{\mathcal{L}_c}(\overline{vk}_\beta, \overline{vk}^*, \overline{m}_\alpha, \overline{m}^*, c, c'_i, \pi) = 0$. By the completeness of the zero-knowledge protocol this occurs with zero probability.
(4) If $b_i = 1$ and $\mathsf{AS.pVf}(vk, m_\alpha, Y_\alpha, \hat{\sigma}_\alpha) \neq 1$. Since $\hat{\sigma}_i$ is computed using m_i and Y_i, by the correctness property of pSign, it is guaranteed pVf outputs 0 with zero probability.
(5) If $b_i = 1$ and $\prod_{j \in T} Y_{\alpha,j}^{\ell_j(0)} \cdot Y_{\alpha,k}^{\ell_k(0)} \neq Y_\alpha$ for some $k \in [N] \setminus T$. This case is impossible by construction of the shares $Y_{\alpha,k}$ 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

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

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. Denote these buckets 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 $\mathsf{bckt}_{j,i}$ 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 $R_a = g^{r_{i,a}}$. Pick any $r_{i,a}$ from the rShare_i. Since by construction, $s_a = r_{i,a} + y_{j,i}$ (j is the message number and i is the server number), one can compute $y_{j,i} = s_a - r_{i,a}$. Since $y_{j,i} = \left(\left(y_j - \sum_{k \in [\rho-1]} y_{j,k} \cdot \ell_k(0)\right) \cdot \ell_i(0)^{-1}\right)$ by construction, we can compute $y_j = \sum_{i \in K} y_{j,i} \cdot \ell_i(0)$. Finally, we can adapt the signature $\hat{\sigma}_j$ using y_j to get σ_j, and by the correctness of the adaptor signature AS, the validity of the signature σ_j is guaranteed. □

D SECURITY ANALYSIS OF VweTS CONSTRUCTION FROM ADAPTOR SIGNATURES

Proof of Theorem 1. We first show that the protocol described in Figure 6 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 5.
$\underline{\mathsf{Hyb}_0}$: This is the experiment defined in Figure 5.

$\underline{\mathsf{Hyb}_1}$: 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 $i \in [\gamma]$ is set to $(\Phi, (b_1, \ldots, b_\gamma))$. The challenger guesses that 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 encryptions c'_i for which $b_i = 1$ are replaced by encryptions of 0. By the IND-CPA security of the witness encryption scheme (Definition 3) 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 encryptions 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 encryptions 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 3) 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. In particular we show that an adversary that wins the one-wayness experiment can be used to break the unforgeability property (Definition 11) of the underlying adaptor signature.

Consider an adversary $\mathcal{A}$ that wins the one-wayness experiment with non-negligble probability. We now describe another adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win the unforgeability game of the adaptor signatures.

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

(1) Initialize $\mathcal{A}$ and simulate the experiment ExpOWay towards $\mathcal{A}$.
(2) While simulating 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 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 the

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

This implies that $\Pr\left[\mathsf{ExpOWay}^{\rho,N}_{\mathsf{VweTS},\mathsf{DS},\overline{\mathsf{DS}},\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$ since we assume that the adaptor signature scheme is EUF-CMA secure and $|Q_3|$ and $|M|$ are polynomial in the security parameter λ. This concludes our proof of security of one-wayness.

We now prove that the scheme is verifiable according to Definition 8. We analyze the protocol in the interactive version and the verifiability must follow from the Fiat-Shamir transformation. Assume that an adversary $\mathcal{A}$ breaks the verifiability of the protocol. This implies that the adversary outputs $((m_j, \overline{m}_j)_{j\in[M]}, vk, (\overline{vk}_i)_{i\in[N]}, (\overline{\sigma}_j)_{j\in K}, j^*, c, \pi_c)$ such that

(1) $\forall j \in K, \overline{\mathsf{Vf}}(\overline{vk}_j, \overline{m}_{j^*}, \overline{\sigma}_j) = 1$
(2) $\mathsf{VfEnc}(c, \pi_c, ((\overline{vk}_i)_{i\in[N]}, (\overline{m}_j, m_j)_{j\in[M]}, vk)) = 1$
(3) $\mathsf{Vf}(vk, m_{j^*}, \sigma) = 0$, where $\sigma \leftarrow \mathsf{DecSig}(j^*, \{\overline{\sigma}_j\}_{j\in K}, c, \pi_c)$

Now since $\forall j \in K, \overline{\mathsf{Vf}}(\overline{vk}_j, \overline{m}_{j^*}, \overline{\sigma}_j) = 1$, the adversary is able to compute some $r = \mathsf{WES.Dec}(\overline{\sigma_j}, c)$ for every $(i, s, c, \pi) \in \mathcal{S}_{\mathsf{unop}}$ such that $\Phi(i) = (j^*, j)$. This r is then added to rShare_j.

Now following Corollary 4.2 of [23] we pick parameters such that the probability of all r in any rShare_j to be invalid is negligible. More specifically, if the total number of ciphertexts is set to $2MNB$, where $B = |\mathsf{bckt}|$ and $B \geq \frac{\lambda}{\log MN+1} + 1$ then the probability of all r in any rShare being invalid is negligible.

Since VfEnc outputs 1, this implies that $g^{s_i} = R_i \cdot Y_{\alpha,\beta}$. Moreover, the ciphertexts are well formed except with negligible probability by the soundness of the NIZK scheme. This implies that the secret shares $y_{j,i}$ can be computed as $s_a - r_{i,a}$. Given K shares the party is able to reconstruct to compute y_j. Finally since $\mathsf{AS.pVf}(vk, m_\alpha, Y_\alpha, \hat{\sigma}_\alpha) = 1$ by the pre-signature adaptibility property of AS the party is able to compute the signature σ with high probability. □

E 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 $\overline{\mathsf{DS}}$.

E.0.1 High Level Overview. We present a high level overview of our construction, and the formal description is given in Fig. 13. 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. In Appendix C, we formally show that our construction satisfies correctness according to Definition 6. Security of our construction is formally stated in the following theorem, and the proof is deferred to Appendix F.

THEOREM 4. *Let BLS signature scheme be unforgeable (*DS *and* $\overline{\mathsf{DS}}$*),* WES *be a secure witness encryption based on signatures scheme, 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 Fig. 6 is one-way and verifiable according to Definition 7 and Definition 8, respectively.*

THEOREM 5. *Our* VweTS *construction from Fig. 13 is correct according to Definition 6.*

PROOF. 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.

```
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) Compute $\{\Phi, (b_1, \dots, b_\gamma)\} := H_2((c_i', R_i)_{i\in[\gamma]})$.
(4) 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 $\sigma_{i,j} = \left(\frac{\sigma_i}{\prod_{j\in[t-1]} \sigma_{i,j}^{\ell_j(0)}}\right)^{\ell_i(0)^{-1}}$, $h_{i,j} = \left(\frac{vk}{\prod_{j\in[t-1]} h_{i,j}^{\ell_j(0)}}\right)^{\ell_i(0)^{-1}}$.
(5) Set $\Sigma_1 = \{h_{i,j}\}_{i\in[M], j\in[N]}$.
(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 $\{\Phi, (b_1, \dots, b_\gamma)\} := H_2((c_i', R_i)_{i\in[\gamma]})$.
(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 $\sigma_j = \prod_{i\in[K]} \sigma_{j,i}^{\ell_i(0)}$.
```

Figure 13: Verifiable witness encryption based on threshold signatures from BLS signatures.

(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

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

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 $\sigma_j = \prod_{i \in K} \sigma_{j,i} \cdot \ell_i(0)$. □

F SECURITY ANALYSIS OF VweTS CONSTRUCTION FROM BLS SIGNATURES

Before proceeding with the proof of the theorem we recall the aggregate extraction problem, as defined in [8]. For a uniformly sampled bilinear group $(\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_T)$ with uniformly sampled generators (g_0, g_1), the aggregate extraction problem gives the attacker the following information

$$(g_0, g_1, g_0^r, g_0^s, g_1^{r+s})$$

where $r, s \leftarrow_\$ \mathbb{Z}_q$. The adversary wins if it outputs g_1^s. It is not hard to see that this variant of the problem is as hard as the computational Diffie-Hellman (CDH) problem. On input $(g_0, g_1, X = g_0^x)$, the reduction samples y and set $Y = g_1^y$. Then it feeds the adversary with $(g_0, g_1, X, g_0^y / X, Y)$ and returns whatever the adversary returns. It can be verified that the tuple is identically distributed as the challenge for the aggregate extraction problem and a solution immediately yields a solution for the CDH problem.

Proof of Theorem 4. We first show that the protocol described in Figure 13 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 5.

$\mathsf{Hyb}_0 - \mathsf{Hyb}_4$: Defined as in the proof of Theorem 1.

Hyb_5: This hybrid is the same as Hyb_4 except that for j^*

(1) For $i \in C$:
  (a) Sample a uniform $x_{i,j^*} \leftarrow \mathbb{Z}_q$
  (b) Set $\sigma_{i,j^*} = H_0(m_{j^*})^{x_{i,j^*}}$
  (c) Set $h_{i,j^*} = g_0^{x_{i,j^*}}$
(2) For $i \in [N] \setminus C$:
  (a) Compute $h_{i,j^*} = \left(\frac{vk}{\prod_{k \in C} h_{i,k}^{\ell_k(0)}} \right)^{\ell_i(0)^{-1}}$
  (b) Sample $r \leftarrow_\$ \mathbb{Z}_q$
  (c) Let a be s.t. $\Phi(a) = (i, j^*)$ compute $s_a = g_1^r \cdot \left(\frac{\sigma_i}{\prod_{k \in C} \sigma_{i,k}^{\ell_k(0)}} \right)^{\ell_i(0)^{-1}}$
  (d) Set $R_a = g_0^r$.

For the malicious parties ($i \in C$) the variables σ_{i,j^*}, h_{i,j^*} and s_{i,j^*} are computed exactly as in Hyb_4.

For the honest parties ($i \in [N] \setminus C$), the variables are computed such that the distribution of R_i, s_i are indistinguishable from the previous hybrid and h_{i,j^*} is computed as in the previous hybrid. Therefore the two hybrids are indistinguishable.

Now we show that one-wayness holds in Hyb_5. In particular we show that an adversary that wins the one-wayness experiment can be used to solve the aggregate extraction problem. 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 win the aggregate extraction problem.

Adversary $\mathcal{B}$:

(1) Initialize $\mathcal{A}$ and simulate the experiment $\mathsf{ExpOWay}$ towards the adversary as in Hyb_5.
(2) Upon receiving a challenge (G, H, σ, g_0, g_1) do the following. For $i \in C$, do as in Hyb_5. For $i \in [N] \setminus C$:
  (a) Sample $\alpha \leftarrow \mathbb{Z}_q$
  (b) replace s_a with $\sigma \cdot g_1^\alpha$
  (c) replace h_{i,j^*} with H
  (d) replace R_a with $G \cdot g_0^\alpha$.
(3) Upon receiving $\mathsf{Sign}O$ calls simulate the signature by programming the random oracle appropriately.
(4) Upon receiving σ^* from $\mathcal{A}$, compute $\sigma' = \left(\frac{\sigma^*}{\prod_{i \in C} \sigma_{i,j^*}^{l_{j^*}(0)}} \right)^{(l_{j^*}(0))^{-1}}$ and output σ'.

Observe that if σ^* is a valid signature then $\sigma^* = \prod_{i \in [K]} \sigma_{j,i}^{\ell_i(0)}$. This implies atleast one of the $\sigma_{j,i}$ corresponds to an $i \in [N] \setminus C$.

Now, $\sigma' = \left(\frac{\sigma^*}{\prod_{i \in C} \sigma_{i,j^*}^{l_{j^*}(0)}} \right)^{(l_{j^*}(0))^{-1}}$ returns $\sigma' = \sigma_{j,i}$ that corresponds to an $i \in [N] \setminus C$

This implies $\sigma' = s_a / g_1^{r_a}$ for some a. The reduction playing the AggExt experiment sets $s_a = g_1^{r+s} \cdot g_1^\alpha$ and $R_a = g_0^r \cdot g_0^\alpha$. The latter implies $r_a = r + \alpha$ and therefore $\sigma' = s_a / g_1^{r_a} = \frac{g_1^{r+s} \cdot g_1^\alpha}{g_1^{r+\alpha}} = g_1^s$

Thus, $\Pr\left[\mathsf{AggExt}_{\mathcal{A},\mathbb{G}_0,\mathbb{G}_1,\mathbb{G}_T}(\lambda) = 1\right] = \Pr\left[\mathsf{Hyb}_5{}^{\rho,N}_{\mathsf{VweTS},\mathsf{DS},\overline{\mathsf{DS}},\mathcal{A}}(\lambda) = 1\right]$
$= \frac{1}{|Q_3|} \frac{1}{M} \Pr\left[\mathsf{ExpOWay}^{\rho,N}_{\mathsf{VweTS},\mathsf{DS},\overline{\mathsf{DS}},\mathcal{A}}(\lambda) = 1\right]$

We now prove that the scheme is verifiable according to Definition 8. We analyze the protocol in the interactive version and the verifiability must follow from the Fiat-Shamir transformation. Assume that an adversary $\mathcal{A}$ breaks the verifiability of the protocol. This implies that the adversary outputs $((m_j, \overline{m}_j)_{j \in [M]}, vk, (\overline{vk}_i)_{i \in [N]}, (\overline{\sigma}_j)_{j \in K}, j^*, c, \pi_c)$ such that

(1) $\forall j \in K, \overline{\mathsf{Vf}}(\overline{vk}_j, \overline{m}_{j^*}, \overline{\sigma}_j) = 1$
(2) $\mathsf{VfEnc}(c, \pi_c, ((\overline{vk}_i)_{i \in [N]}, (\overline{m}_j, m_j)_{j \in [M]}, vk)) = 1$
(3) $\mathsf{Vf}(vk, m_{j^*}, \sigma) = 0$, where $\sigma \leftarrow \mathsf{DecSig}(j^*, \{\overline{\sigma}_j\}_{j \in K}, c, \pi_c)$

Now since $\forall j \in K, \overline{\mathsf{Vf}}(\overline{vk}_j, \overline{m}_{j^*}, \overline{\sigma}_j) = 1$, the adversary is able to compute some $r = \mathsf{WES.Dec}(\overline{\sigma_j}, c)$ for every $(i, s, c, \pi) \in \mathcal{S}_{\mathsf{unop}}$ such that $\Phi(i) = (j^*, j)$. This r is then added to rShare_j.

Now following Corollary 4.2 of [23] we pick parameters such that the probability of all r in any rShare$_j$ to be invalid is negligible. More specifically, if the total number of ciphertexts is set to $2MNB$, where $B = |\text{bckt}|$ and $B \geq \frac{\lambda}{\log MN+1} + 1$ then the probability of all r in any rShare being invalid is negligible.

Since VfEnc outputs 1, this implies that $e(g_0, s_i) = e(R_i, g_1) \cdot e(h_{\alpha,\beta}, H_0(m_\alpha))$. Moreover, the ciphertexts are well formed except with negligible probability by the soundness of the NIZK scheme. This implies that the secret shares σ_{i,j^*} can be computed as $s_a/g_1^{r_{i,a}}$. Given K shares the party is able to reconstruct to compute σ_{j^*}. □

G CONSTRUCTIONS FOR LARGE UNIVERSE OF OUTCOMES

In the protocols described previously, the communication complexity grows linearly in the number of outcomes. In particular, this implies that the number of outcomes of a given event must be bounded by a given polynomial (in the security parameter). In what follows we outline two modifications to decrease the dependency of the computation and communication complexity in the number of outputs. The first is practically more efficient, but asymptotically still linear in the number of outcomes. The second is concretely less efficient, by asymptotically logarithmic in the size of the output space. In particular, this means that we can even support an exponential-size universe of outcomes using the latter construction.

G.1 Concretely Efficient Construction

Let M be the size of the universe of outcomes, which we assume to be bounded by some polynomial in the security parameter. For convenience, we assume that M is a power of 2 and we denote $M = 2^\mu$, where $\mu = O(\log(\lambda))$. We present the formal construction in Fig. 14.

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 (refer to Section 4.2 for more details). Instead of computing the witness encryption of the y_j, Alice additionally 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 4.2, 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 4.2 we had the above cut-and-choose run for M such ciphertexts, but only have 2μ now. The verification procedures is unchanged, except that Bob additionally checks whether

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

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.

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 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.

Security Analysis. We now argue that 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 4.2). 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). On the other hand, 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^*}. Since

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

the reduction can output $y^* = \mathsf{DLog}(Y^*)$.

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. This concludes our proof.

G.2 Asymptotically Optimal Construction

The main ingredient used in this protocol are garbled circuits (see [6] for a formal treatment of garbled circuits). Instead of computing signatures for each outcome and encrypting separately, Alice now garbles a circuit that does the following: On input an outcome j, it outputs a signature (using Alice's secret key) of the corresponding message m_j. Let $\{\ell_{i,0}, \ell_{i,1}\}_{i \in \log(M)}$ be the labels of the garbled circuits, where M is the size of the universe of outcomes.[3] Alice then uses the scheme described in the previous section to encrypt each label $\ell_{i,b}$, conditioned on the oracle signing a message encoding the position i and the bit b. The output of this algorithm consists of the encryptions of the labels, and the garbled circuit.

For the oracles, the scheme is defined identically, except that, on input an event $j \in M$, each oracle signs separately each bit of $j = (j_1, \dots, j_{\log(M)})$ along with an identifier for the position, e.g., it signs the messages $(j_1, 1), \dots, (j_{\log(M)}, \log(M))$. To decrypt, Bob can then use the signatures of the oracles to recover the set of labels $\{\ell_{i,j_i}\}_{i \in \log(M)}$ and use such labels to evaluate the garbled circuit, which returns a signature on m_j under Alice's key.

Note that in the description above we did not consider the verifiability of the encryptions. We require two guarantees of verifiability: (i) The encryptions are computed correctly and (ii) the garbled circuits are computed correctly. The first guarantee comes for free using the scheme described in our previous section. To achieve the latter, one can resort to known techniques in the literature, such as cut-and-choose protocols presented in [3, 12]. We leave this extension as ground for future work.

H FORMAL DEFINITIONS FOR ORACLE CONTRACTS

Definition 14 (Correctness). *An* Oracle Contract *scheme is correct if the following holds simultaneously:*

- Honest attestations must verify correctly. *For all $\lambda \in \mathbb{N}$, all $(pk^O, sk^O) \in$ SUPP(OKGen(λ)), all outcomes o, it must hold that:*

$$Pr[\mathsf{AttestVf}(pk^O, \mathsf{Attest}(sk^O, o), o) = 1] = 1$$

- Honestly generated attestation anticipations must verify correctly. *For all $\lambda \in \mathbb{N}$, all $(pk^O_1, \dots, pk^O_N) \in \mathsf{SUPP}(\mathsf{OKGen}(\lambda))$, all $(pk_A, sk_A) \in \mathsf{SUPP}(\Pi_{\mathsf{BDS}}.\mathsf{KGen}(\lambda))$ all pairs of the form $(o_j, \mathsf{Tx}_j)_{j \in [M]}$, it must hold that:*

$$Pr[\mathsf{AnticipateVf}(pk_A, ant, (pk^O_i)_{i \in [N]}, (o_j, \mathsf{Tx}_j)_{j \in [M]}) = 1] = 1$$

where $ant \leftarrow \mathsf{Anticipate}(sk_A, (pk^O_i)_{i \in [N]}, (o_j, \mathsf{Tx}_j)_{j \in [M]})$.

- Honest generated anticipations and attestations must be redeemable by the counter-party. *For all $\lambda \in \mathbb{N}$, all set of public keys $(pk^O_1, \dots, pk^O_N) \in \mathsf{SUPP}(\mathsf{OKGen}(\lambda))$, all $(pk_A, sk_A) \in \mathsf{SUPP}(\Pi_{\mathsf{BDS}}.\mathsf{KGen}(\lambda))$, all pairs $(o_j, \mathsf{Tx}_j)_{j \in [M]}$, any $j \in [M]$ and any $K \subset [N]$, where $|K| = \rho$, it must hold that:*

$$Pr[\Pi_{\mathsf{BDS}}.\mathsf{Vf}(pk_A, \mathsf{Tx}_j, \mathsf{Redeem}(j, (att_i)_{i \in [K]}, ant)) = 1] = 1$$

where $ant \leftarrow \mathsf{Anticipate}(sk_A, (pk^O_i)_{i \in [N]}, (o_j, \mathsf{Tx}_j)_{j \in [M]})$ and $\forall i \in [K] : att_i \leftarrow \mathsf{Attest}(sk^O_i, o_j)$.

Definition 15 (Unforgeability). *A oracle contract scheme $(\rho - N - M) - OC :=$ (OKGen, Attest, AttestVf, Anticipate, AnticipateVf, Redeem) parameterized by $\rho, N, M \in \mathbb{N}$ and defined with respect to a signature scheme $\Pi_{\mathsf{BDS}} :=$ (KGen, Sign, Vf) is said to be* unforgeable *if for all $\lambda \in \mathbb{N}$, there exists a negligible function* negl(λ)*, such that for all PPT adversaries $\mathcal{A}$, the following holds,*

$$\Pr\Big[\mathsf{ExpForge}^{\rho,N,M}_{OC,\Pi_{\mathsf{BDS}},\mathcal{A}}(\lambda) = 1\Big] \leq \mathsf{negl}(\lambda)$$

where ExpForge *is defined in Fig. 15.*

Definition 16 (Verifiability). *A oracle contract scheme $(\rho - N - M) - OC :=$ (OKGen, Attest, AttestVf, Anticipate, AnticipateVf, Redeem) parameterized by $\rho, N, M \in \mathbb{N}$ and defined with respect to a signature scheme $\Pi_{\mathsf{BDS}} :=$ (KGen, Sign, Vf) is said to be* verifiable *if for all $\lambda \in \mathbb{N}$, there exists a negligible function* negl(λ)*, and no PPT adversary $\mathcal{A}$ that outputs $((o_j, \mathsf{Tx}_j)_{j \in [M]}, pk_A, \{pk^O_i\}_{i \in N}, \{att_i\}_{i \in K}, j^*, ant)$ such that all the following holds simultaneously except with probability* negl(λ):

(1) $K \subset [N]$ and $|K| = \rho$
(2) $(pk_A, \cdot) \in SUPP(\Pi_{\mathsf{BDS}}.\mathsf{KGen})$ and for all $i \in [N]$ we have $(pk^O_i, \cdot) \in SUPP(\mathsf{OKGen})$ where SUPP denotes to the support.
(3) $\forall i \in K, \mathsf{AttestVf}(pk^O_i, o_{j^*}, att_i) = 1$
(4) $\mathsf{AnticipateVf}(pk_A, ant, (pk^O_i)_{i \in [N]}), (o_j, \mathsf{Tx}_j)_{j \in [M]}) = 1$
(5) $\Pi_{\mathsf{BDS}}.\mathsf{Vf}(pk_A, \mathsf{Tx}_{j^*}, \sigma) = 0$, where $\sigma \leftarrow \mathsf{Redeem}(j^*, \{att_i\}_{i \in K}, ant)$

Definition 17 (Attestation Unforgeability). *A oracle contract scheme $(\rho - N - M) - OC :=$ (OKGen, Attest, AttestVf, Anticipate, AnticipateVf, Redeem) parameterized by $\rho, N, M \in \mathbb{N}$ is said to be* attestation unforgeable *if for all $\lambda \in \mathbb{N}$, there exists a negligible function* negl(λ)*, such that for all PPT adversaries $\mathcal{A}$, the following holds,*

$$\Pr\Big[\mathsf{ExpAttestForge}^{\rho,N,M}_{OC,\Pi_{\mathsf{BDS}},\mathcal{A}}(\lambda) = 1\Big] \leq \mathsf{negl}(\lambda)$$

where ExpAttestForge *is defined in Fig. 16.*

I PROOFS OF CORRECTNESS OF ORACLE CONTRACTS

Theorem 6. *Our oracle contracts contraction from Fig. 7 is correct according to Definition 14.*

Proof. To prove correctness we first need to show that

$$Pr[\mathsf{AttestVf}(pk^O, \mathsf{Attest}(sk^O, o), o) = 1] = 1.$$

Note that AttestVf will output 0 if $\overline{\mathsf{DS}}.\overline{\mathsf{Vf}}(\overline{vk}, o, \overline{\sigma}) \neq 1$. Since $\overline{\sigma}$ is computed using o, by the correctness property of $\overline{\mathsf{DS}}.\overline{\mathsf{Sign}}$, it is guaranteed that $\overline{\mathsf{DS}}.\overline{\mathsf{Vf}}$ outputs 0 with zero probability. Thus, if Attest is computed correctly, AttestVf outputs 1 with probability 1.

Next we need to show that

$$Pr[\mathsf{AnticipateVf}(pk_A, ant, (pk^O_i)_{i \in [N]}, (o_j, \mathsf{Tx}_j)_{j \in [M]}) = 1] = 1.$$

[3] E.g., setting $M = 2^\lambda$ gives us an exponential size universe of outcomes.

Note that AnticipateVf will output 0 if $\mathsf{VweTS.VfEnc}(c, \pi_c, ((\overline{vk}_i)_{i\in[N]}, (o_j, \mathsf{Tx}_j)_{j\in[M]}, vk)) \neq 1$. Since $att := (c, \pi_c)$ is computed using $(o_j, \mathsf{Tx}_j)_{j\in[M]}$, by the correctness property of VweTS.EncSig, it is guaranteed that VweTS.VfEnc outputs o with zero probability. Thus, if Anticipate is computed correctly, AnticipateVf outputs 1 with probability 1.

Finally, we need to show that for any $j \in [M], K \subset [N]$ and $|K| = \rho$, if for all $i \in [K]$ we have $\mathsf{AttestVf}(pk_i^O, att_i, o_j) = 1$ then

$$Pr[\Pi_{\mathsf{BDS}}.\mathsf{Vf}(pk_A, \mathsf{Tx}_j, \mathsf{Redeem}(j, (att_i)_{i\in[K]}, ant)) = 1] = 1.$$

We are given that for all $i \in K$, $\overline{\mathsf{DS}}.\overline{\mathsf{Vf}}(\overline{vk}, o, \overline{\sigma}) = 1$. By construction, we have $\sigma \leftarrow \mathsf{VweTS.DecSig}(j, \{\overline{\sigma}_i\}_{i\in[K]}, c, \pi_c)$, thus by the correctness of the verifiable witness encryption based on threshold signatures scheme VweTS, the validity of the signature σ is guaranteed. □

J SECURITY ANALYSIS OF ORACLE CONTRACTS

Theorem 7 (Oracle contract security). *Let (ρ, N, M)-VweTS be a* one-way *verifiable witness encryption for threshold signatures scheme defined with respect to signature schemes* $\mathsf{DS} := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ *and* $\overline{\mathsf{DS}} := (\overline{\mathsf{KGen}}, \overline{\mathsf{Sign}}, \overline{\mathsf{Vf}})$. *Let* $\overline{\mathsf{DS}} := (\overline{\mathsf{KGen}}, \overline{\mathsf{Sign}}, \overline{\mathsf{Vf}})$ *be an an EUF-CMA secure digital signature scheme. Then, our protocol is an* unforgeable, verifiable *and* attestation unforgeable *(ρ, N, M)-oracle contract protocol defined with respect to the signature scheme* $\Pi_{\mathsf{BDS}} := \mathsf{DS}$.

Proof. We give a proof by reduction for three adversaries playing the games of unforgeability, verifiablity and attestation unforgeability, respectively.

Unforgeability. Let $\mathcal{A}$ be a PPT adversary with non-negligible advantage in the $\mathsf{ExpForge}^{\rho,N,M}_{OC,\Pi_{\mathsf{BDS}},\mathcal{A}}(\lambda)$ game. We now construct and adversary $\mathcal{R}$ which uses $\mathcal{A}$ to win the $\mathsf{ExpOWay}^{\rho,N,M}_{\mathsf{VweTS},\mathsf{DS},\overline{\mathsf{DS}},\mathcal{A}}(\lambda)$ game.

$\mathcal{R}$ is given a verification key vk by the $\mathsf{ExpOWay}^{\rho,N,M}_{\mathsf{VweTS},\mathsf{DS},\overline{\mathsf{DS}},\mathcal{A}}(\lambda)$ game. It then runs $\mathcal{A}$ on input $pk_A := vk$ to get as output a pair (C, st_0). $\mathcal{R}$ forwards the same pair to the challenger.

On input st_0, $\{\overline{vk}_i\}_{i\in[N]\setminus C}$, $\mathcal{R}$ sets $\{pk_i^O\}_{i\in[N]\setminus C} := \{\overline{vk}_i\}_{i\in[N]\setminus C}$ and invokes $\mathcal{A}$ get the tuple (q^*, j^*, σ^*). The reduction $\mathcal{R}$ simply forwards this tuple to the challenger as the output of the game.

Additionally, $\mathcal{R}$ must simulate $\mathcal{A}$'s oracle access to Anticipate$O$, Attest$O$ and Sign$O$. This can be trivially done as follows. Every time that $\mathcal{A}$ queries Anticipate$O$ on input $(o_j, \mathsf{Tx}_j)_j \in [M], \{pk_i^O\}_{i\in C}$, $\mathcal{R}$ queries its own oracle EncSig$O$ on the same input and forwards the output. Every time that $\mathcal{A}$ queries Attest$O$ on input i, o, $\mathcal{R}$ queries $\overline{\mathsf{Sign}O}$ on input the same input i, o and return the attestation att_i to $\mathcal{A}$. Finally, every time that $\mathcal{A}$ queries Sign$O$, $\mathcal{R}$ forwards the query to its own Sign$O$ and returns the output signature σ to $\mathcal{A}$.

After $\mathcal{A}$ returns the tuple (q^*, j^*, σ^*) as the forgery for the unforgeability game of oracle contracts, $\mathcal{R}$ outputs (q^*, j^*, σ^*) as the output of its own game. It is easy to see that $\mathcal{R}$ is an efficient algorithm and that faithfully simulates the view of $\mathcal{A}$. It is left to show that $\mathcal{R}$ wins its game with the same probability as $\mathcal{A}$ wins its corresponding game. For that, we observe the following:

- b_0: Q_2 is updated in the same way in both games. Moreover $\mathcal{R}$ simply forwards calls from $\mathcal{A}$ to its own oracle, therefore if b_0 holds for $\mathcal{A}$, it holds in $\mathcal{R}$
- b_1: It holds in $\mathcal{R}$ by the same argument as before but applied to the oracle $\overline{\mathsf{Sign}O}$.
- b_2: This is exactly the same condition in both games. Moreover, C is a value received from $\mathcal{A}$ and unmodified by $\mathcal{R}$. Therefore, it must hold for $\mathcal{R}$ if it holds for $\mathcal{A}$.
- b_3: Our $\mathcal{R}$ maps pk_A to vk and Tx_{j^*} to m_{j^*} during the reduction. Therefore, the condition is the same in both games and must hold in both.

Therefore, by assumption, $\mathcal{A}$ succeeds with non-negligible probability, and thus $\mathcal{R}$ also wins with non-negligible probability. This violates the assumption that (ρ, N, M)-VweTS be a *one-way* verifiable witness encryption for threshold signatures scheme, implying that no such adversary $\mathcal{A}$ can exist.

Verifiability. Let $\mathcal{A}$ be a PPT adversary that can break the verifiablity of our (ρ, N, M)-oracle contract protocol non-negligible probility. We now construct and adversary $\mathcal{R}$ which uses $\mathcal{A}$ to break the verifiablity of (ρ, N, M)-VweTS.

Our reduction $\mathcal{R}$ maps $(m_j, \overline{m}_j)_{j\in[M]}$ to $(o_j, \mathsf{Tx}_j)_{j\in[M]}$, vk to pk_A, $(\overline{vk}_i)_{i\in[N]}$ to $\{pk_i^O\}_{i\in[N]}$, $(\overline{\sigma}_j)_{j\in K}$ to $\{att_i\}_{i\in K}$, j^* to j^* and (c, π_c) to ant.

After $\mathcal{A}$ returns the tuple $((o_j, \mathsf{Tx}_j)_{j\in[M]}, pk_A, \{pk_i^O\}_{i\in N}, \{att_i\}_{i\in K}, j^*, ant)$ that breaks the verifiability of oracle contracts, $\mathcal{R}$ outputs $((m_j, \overline{m}_j)_{j\in[M]}, vk, (\overline{vk}_i)_{i\in[N]}, (\overline{\sigma}_j)_{j\in K}, j^*, c, \pi_c)$ as the output of its own game. It is easy to see that $\mathcal{R}$ is an efficient algorithm and that faithfully simulates the view of $\mathcal{A}$. Finally, we see that the conditions in both definitions are exactly the same and as a consequence they all must hold for $\mathcal{R}$ if they hold for $\mathcal{A}$. Hence, $\mathcal{R}$ wins with the same probability as $\mathcal{A}$.

Therefore, by assumption, $\mathcal{A}$ succeeds with non-negligible probability, and thus $\mathcal{R}$ also wins with non-negligible probability. This violates the assumption that (ρ, N, M)-VweTS be a *verifiable* witness encryption for threshold signatures scheme, implying that no such adversary $\mathcal{A}$ can exist.

Attestation unforgeability. Let $\mathcal{A}$ be a PPT adversary with non-negligible advantage in the $\mathsf{ExpAttestForge}^{\rho,N,M}_{OC,\mathcal{A}}(\lambda)$ game. We now construct and adversary $\mathcal{R}$ which uses $\mathcal{A}$ to win the $\mathsf{SigForge}_{\mathcal{A},\mathsf{DS}}(\lambda)$ game.

$\mathcal{R}$ is given a verification key vk by the $\mathsf{SigForge}_{\mathcal{A},\mathsf{DS}}(\lambda)$ game. It then runs $\mathcal{A}$ on input $pk^O := vk$ to get as output a pair (o^*, att^*). $\mathcal{R}$ forwards the pair $(m^* := o^*, \sigma := att^*)$ to the challenger as the output of the game.

Additionally, $\mathcal{R}$ must simulate $\mathcal{A}$'s oracle access to Attest$O$. This can be trivially done as follows. Every time that $\mathcal{A}$ queries Attest$O$ on input o, $\mathcal{R}$ queries $\overline{\mathsf{Sign}O}$ on input the same input $m := o$ and return the attestation $\sigma := att$ to $\mathcal{A}$.

After $\mathcal{A}$ returns the pair (o^*, att^*) as the forgery for the attestation unforgeability game of oracle contracts, $\mathcal{R}$ outputs $(m^* := o^*, \sigma := att^*)$ as the output of its own game. It is easy to see that $\mathcal{R}$ is an efficient algorithm and that faithfully simulates the view of $\mathcal{A}$. It is left to show that $\mathcal{R}$ wins its game with the same probability as $\mathcal{A}$ wins its corresponding game. For that, we observe the following:

- b_0: Q is updated in the same way in both games. Moreover $\mathcal{R}$ simply forwards calls from $\mathcal{A}$ to its own oracle, therefore if b_0 holds for $\mathcal{A}$, it holds in $\mathcal{R}$
- b_1: Our $\mathcal{R}$ maps pk^O to vk and o^* to m^* during the reduction. Therefore, the condition is the same in both games and must hold in both.

Therefore, by assumption, $\mathcal{A}$ succeeds with non-negligible probability, and thus $\mathcal{R}$ also wins with non-negligible probability. This violates the assumption that $\overline{\mathsf{DS}} := (\overline{\mathsf{KGen}}, \overline{\mathsf{Sign}}, \overline{\mathsf{Vf}})$ be an EUF-CMA secure digital signatures scheme, implying that no such adversary $\mathcal{A}$ can exist. □

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 $i \in [\gamma]$: where $\gamma$ is $2NB(2\log M)$.
    (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) Compute $\{\Phi, (b_1, \ldots, b_\gamma)\} := H_2((c'_i, R_i)_{i\in[\gamma]})$.
(4) For $i \in [\log M]$ and $b \in \{0,1\}$
    (a) Compute $z_{i,b} \leftarrow \mathbb{Z}_q$ and $Z_{i,b} = g^{z_{i,b}}$
    (b) For all $j \in [\rho - 1]$ sample a uniform $z_{i,b,j} \leftarrow \mathbb{Z}_q$ and set $Z_{i,b,j} := g^{z_{i,b,j}}$.
    (c) For all $j \in \{\rho, \ldots, N\}$ compute $z_{i,b,j} = \left(\left(z_{i,b} - \sum_{k\in[\rho-1]} z_{i,b,k} \cdot \ell_k(0)\right) \cdot \ell_j(0)^{-1}\right)$, $Z_{i,b,j} = \left(\frac{Z_{i,b}}{\prod_{k\in[\rho-1]} Z_{i,b,k}^{\ell_k(0)}}\right)^{\ell_j(0)^{-1}}$. Here $\ell_i$ is
        the $i$-th Lagrange polynomial.
(5) 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) Compute $e_i = y_i + \sum_j z_{j,i[j]}$
(6) Set $\Sigma_1 = \{(\hat{\sigma}_i, e_i)_{i\in[M]}, \{Z_{i,b}\}_{i\in[\log M], b\in\{0,1\}}, \{Z_{i,b,j}\}_{i\in[\log M], j\in[N], b\in\{0,1\}}\}$
(7) 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 $(\alpha, \beta, pos) := \Phi(i)$.
        (ii) 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
             $\pi_i \leftarrow \mathsf{Prove}_{\mathcal{L}_c}(crs, (\overline{vk}_\beta, \overline{vk}^*, \overline{m}_\alpha, \overline{m}^*, c_i, c'_i), r_i)$.
        (iii) Compute $s_i = z_{pos,\alpha[pos],\beta} + r_i$
        (iv) Set $\mathcal{S}_{\mathsf{unop}} := \mathcal{S}_{\mathsf{unop}} \cup \{(i, c_i, s_i, \pi_i)\}$.
(8) 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
    $\Sigma_1 := \{(\hat{\sigma}_i, e_i)_{i\in[M]}, \{Z_{i,b}\}_{i\in[\log M], b\in\{0,1\}}, \{Z_{i,b,j}\}_{i\in[\log M], j\in[N], b\in\{0,1\}}\}$.
(2) Compute $\{\Phi, (b_1, \ldots, b_\gamma)\} := H_2((c'_i, R_i)_{i\in[\gamma]})$
(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) $(\alpha, \beta, pos) := \Phi(i)$
        (ii) Check that $(i, c_i, s_i, \pi_i) \in \mathcal{S}_{\mathsf{unop}}$
        (iii) Check that $g^{s_i} = R_i \cdot Z_{pos,\alpha[pos],\beta}$
        (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 $k \in [N] \setminus T$: $\prod_{j\in T} Z_{pos,\alpha[pos],j}^{\ell_j(0)} \cdot Z_{pos,\alpha[pos],k}^{\ell_k(0)} = Z_{pos,\alpha[pos]}$.
    (c) For $i \in [M]$ Check that $g^{e_i} = Y_i \cdot \prod_i Z_{j,i[j]}$
    (d) 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
    $\Sigma_1 := \{(\hat{\sigma}_i, e_i)_{i\in[M]}, \{Z_{i,b}\}_{i\in[\log M], b\in\{0,1\}}, \{Z_{i,b,j}\}_{i\in[\log M], j\in[N], b\in\{0,1\}}\}$.
(2) For all $(i, j) \in [K] \times [\log M]$, initialize $\mathsf{rShare}_{i,j} = \emptyset$.
(3) For each $(i, c, s, \pi) \in \mathcal{S}_{\mathsf{unop}}$, compute $(\alpha, \beta, pos) = \Phi(i)$. If $\alpha = j$ and if $\beta \in [K]$ s.t. $\mathsf{DS.Vf}(\overline{vk}_\beta, \alpha[pos], \overline{\sigma}_i) = 1)$
    (a) Compute $r = \mathsf{WES.Dec}(\overline{\sigma_i}, c)$.
    (b) Set $\mathsf{rShare}_{\beta,pos} := \mathsf{rShare}_{\beta,pos} \cup \{r\}$.
(4) Denote each $r$ in $\mathsf{rShare}_{i,j}$ 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 $k \in [K]$ and $i \in [\log M]$, compute $z_{i,j[i],k} = s_a - r_{i,a}$
(6) Compute $z_{i,j[i]} = \sum_{k\in[K]} z_{i,j[i],k} \cdot \ell_k(0)$
(7) Compute $y_j = e_j - \sum_i z_{i,j[i]}$
(8) Return $\sigma_j \leftarrow \mathsf{AS.Adapt}(\hat{\sigma}_j, y_j)$.
```

Figure 14: Verifiable witness encryption based on threshold signatures from adaptor signatures for exponential outcomes

```
$\mathsf{ExpForge}^{\rho,N,M}_{OC,\Pi_{\mathsf{BDS}},\mathcal{A}}(\lambda)$
$Q_1 := Q_2 := \emptyset, Q_3 := []$
$(pk_A, sk_A) \leftarrow \Pi_{\mathsf{BDS}}.\mathsf{KGen}(1^\lambda)$
$(C, \mathsf{st}_0) \leftarrow \mathcal{A}(pk_A)$   / let $C \subset [N]$
$\forall i \in [N] \setminus C, (pk_i^O, sk_i^O) \leftarrow \mathsf{OKGen}(1^\lambda)$
$(q^*, \sigma^*, j^*) \leftarrow \mathcal{A}^{\mathsf{Anticipate}\mathcal{O},\mathsf{Attest}\mathcal{O},\mathsf{Sign}\mathcal{O}}(\mathsf{st}_0, \{pk_i^O\}_{i\in[N]\setminus C})$
$(ant, X) \leftarrow Q_3[q^*]$
$X := (sk_A, (pk_i^O)_{i\in[N]}, (o_j, \mathsf{Tx}_j)_{j\in[M]})$
$b_0 := ((\mathsf{Tx}_j{}^*, \sigma^*) \notin Q_2)$
$b_1 := ((o_j{}^*) \notin Q_1)$
$b_2 := (|C| \leq \rho - 1)$
$b_3 := (\Pi_{\mathsf{BDS}}.\mathsf{Vf}(pk_A, \mathsf{Tx}_j{}^*, \sigma^*) = 1)$
return $b_0 \wedge b_1 \wedge b_2 \wedge b_3$

$\mathsf{Anticipate}\mathcal{O}((o_j, \mathsf{Tx}_j)_{j\in[M]}, \{pk_i^O\}_{i\in C})$
$X := (sk_A, (pk_i^O)_{i\in[N]}, (o_j, \mathsf{Tx}_j)_{j\in[M]})$
$ant \leftarrow \mathsf{Anticipate}(X)$
$Q_3 := Q_3 || (ant, X)$
return $ant$

$\mathsf{Attest}\mathcal{O}(i, o)$
Ensure $i \in [N] \setminus C$,
$att_i \leftarrow \mathsf{Attest}(sk_i, o)$
$Q_1 := Q_1 \cup \{o\}$
return $att_i$

$\mathsf{Sign}\mathcal{O}(\mathsf{Tx})$
$\sigma \leftarrow \Pi_{\mathsf{BDS}}.\mathsf{Sign}(sk_A, \mathsf{Tx})$
$Q_2 := Q_2 \cup \{\mathsf{Tx}, \sigma\}$
return $\sigma$
```

Figure 15: Experiment for Unforgeability of Oracle Contracts.

```
$\mathsf{ExpAttestForge}^{\rho,N,M}_{OC,\mathcal{A}}(\lambda)$
$Q := \emptyset$
$(pk^O, sk^O) \leftarrow \mathsf{OKGen}(1^\lambda)$
$(o^*, att^*) \leftarrow \mathcal{A}^{\mathsf{Attest}\mathcal{O}(\cdot)}(pk^O)$
$b_0 := (o \notin Q)$
$b_1 := (\mathsf{AttestVf}(pk^O, o^*, att^*) = 1)$
return $b_0 \wedge b_1$

$\mathsf{Attest}\mathcal{O}(o)$
$att \leftarrow \mathsf{Attest}(sk^O, o)$
$Q := Q \cup \{o\}$
return $att$
```

Figure 16: Experiment for Attestation Unforgeability of Oracle Contracts.

```
$\underline{\mathsf{SigForge}_{\mathcal{A},\mathsf{DS}}(\lambda)}$
$Q := \emptyset$
$(vk, sk) \leftarrow \mathsf{KGen}(1^\lambda)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}O(\cdot)}(vk)$
$b_0 := (m \notin Q)$
$b_1 := (\mathsf{Vf}(vk, m^*, \sigma^*) = 1)$
return $b_0 \wedge b_1$

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

Figure 17: Experiment for EUF-CMA of Digital Signatures.