MixBuy: Contingent Payment in the Presence of Coin Mixers

Diego Castejon-Molina
IMDEA Software Institute
Universidad Politécnica de Madrid

Dimitrios Vasilopoulos
IMDEA Software Institute

Pedro Moreno-Sanchez
IMDEA Software Institute
VISA Research

ABSTRACT

A contingent payment protocol involves two mutually distrustful parties, a buyer and a seller, operating on the same blockchain, and a digital product, whose ownership is not tracked on a blockchain (e.g. a digital book, but not a NFT). The buyer holds coins on the blockchain and transfers them to the seller in exchange for the product. However, if the blockchain does not hide transaction details, any observer can learn that a buyer purchased some product from a seller. In this work, we take contingent payment a step further: we consider a buyer who wishes to buy a digital product from a seller routing the payment via an untrusted mixer. Crucially, we require that said payment is unlinkable, meaning that the mixer (or any other observer) does not learn which buyer is paying which seller. We refer to such setting as *unlinkable contingent payment* (UCP).

We present MixBuy, a system that realizes UCP. Mixbuy relies on *oracle-based unlinkable contingent payment* (O-UCP), a novel four-party cryptographic protocol where the mixer pays the seller and the seller provides the buyer with the product only if a semi-trusted notary attests that the buyer has paid the mixer. More specifically, we require four security notions: (i) *mixer security* that guarantees that if the mixer pays the seller, the intermediary must get paid from the buyer; (ii) *seller security* that guarantees that if the seller delivers the product to the buyer, the seller must get paid from the intermediary; (iii) *buyer security* that guarantees that if the buyer pays the intermediary, the buyer must obtain the product; and (iv) *unlinkability* that guarantees that given a set of buyers and sellers, the intermediary should not learn which buyer paid which seller.

We present a provably secure and efficient cryptographic construction for O-UCP. Our construction can be readily used to realize UCP on most blockchains, as it has minimal functionality requirements (i.e., digital signatures and timelocks). To demonstrate the practicality of our construction, we provide a proof of concept for O-UCP and our benchmarks in commodity hardware show that the communication overhead is small (a few kB per message) and the running time is below one second.

1 INTRODUCTION

Given the increasing deployment of cryptocurrencies, they are now accepted for purchases of digital products such as music, software, e-books, authentication token for a website or mobile phone plan (e.g. [1, 3, 11, 22, 44, 56, 58]). A *contingent payment* involves a buyer and a seller, a blockchain $\mathcal{B}$ and a digital product p whose ownership is not tracked on a blockchain (e.g. a digital book, but not a NFT). Buyer holds α coins on $\mathcal{B}$ and wants to transfer them to the seller in exchange for the product p. In the contingent payment setting, buyer and seller have addresses (or accounts) in the same blockchain $\mathcal{B}$. Hence, with the exception of blockchains like Monero [53] or ZeroCash [8] which support anonymous transactions, an observer who identifies seller's accounts can find out which accounts have been used to purchase goods from a seller and for which amounts.

In this work, we strive to take the contingent payment a step further adding the property of unlinkability between buyer and seller. We call this extension *unlinkable contingent payment.* Here, a group of buyers and a group of sellers route their payments through a mixer such that neither the mixer, nor any other observer to the blockchain knows which buyer is paying which seller.

Problem Description. An unlinkable contingent payment (UCP) involves a blockchain $\mathcal{B}$, a product p, and three participants: buyer, seller, and mixer. Initially, the buyer holds α coins, the mixer holds β coins, and the seller holds product p. At the end of a successful UCP, the buyer should have transferred α coins to the mixer, the mixer should have transferred β coins to the seller (we assume $\alpha - \beta \geq 0$ is mixer's fee), and the buyer should have received p from the seller. A protocol for UCP should enforce the following security and privacy properties: (a) if the mixer transfers β coins to the seller, the mixer obtains α coins from the buyer (mixer security); (b) if the seller delivers p to the buyer, the seller receives β coins from the mixer (seller security); (c) if the buyer transfers α coins to the mixer, the buyer obtains the product p (buyer security); (d) for a set of buyers and sellers, the mixer should not learn which buyer paid which seller (unlinkability).

Designing a protocol for the problem described above turns out to be a non-trivial task. To illustrate the obstacles, consider a setting where a buyer locks some funds into a shared address with the mixer for a pre-determined amount of time T. Similarly, assume that the mixer locks some funds into a shared address with a seller, also for time T. In blockchains this is a standard, well-established procedure realizable, e.g., with 2-out-of-2 multisig addresses [64]. This is needed to ensure that the buyer and the mixer do not quit the protocol prematurely. The funds are unlocked after T, which determines the maximum duration of the protocol. To complete the UCP protocol, (i) the buyer cannot send to the mixer the signed transaction before receiving the product p from the seller; (ii) the mixer cannot send to the seller the signed transaction before receiving a signed transaction from the buyer; (iii) the seller cannot deliver product p to the buyer before receiving a signed transaction from the mixer.

Hence, we end up on a fair exchange of three items of interest (i.e., coins or product) between three mutually untrusted parties. It is established that such fair exchange cannot be achieved in the standard model [10]. However, it has been shown that the (allegedly weak) synchronicity guarantees provided by blockchains (often called claim-or-refund [10]) suffice to solve a weaker version of fair exchange: either each party receives the expected item of interest before a pre-defined time T, or they get refunded their initial item of interest. In fact, several blockchain applications have been proposed in the literature that leverage this claim-or-refund model to provide a trade-off between functionality, security and unlinkability.

Table 1: Related Work. Intermediated unlinkable contingent payment has not been explored yet.

	Two-Party	Intermediated	
		Linkable	Unlinkable
Payment Coordination	[9, 38, 40, 50, 62]	[4, 25, 39, 41, 51, 55]	[28, 29, 33, 35, 37, 57, 59]
Contingent Payment	[12, 14, 21, 27, 52]	[45, 54]	This Work

1.1 Related Work

We classify existing works with respect to the type of assets exchanged. We categorize as *payment coordination* the protocols where the ownership of all of exchanged assets is tracked by a blockchain. We categorize as *contingent payment* the protocols where the ownership of all assets except for one (i.e., the product) is tracked by the blockchain. We categorize as *intermediated* the protocols in which sender/receiver or buyer/seller rely on an untrusted intermediary to route the payment between them.

Two-party Payment Coordination. Also known as *atomic swaps*, involves two parties: Alice and Bob. Alice has α coins in $\mathcal{B}_1$, while Bob has β coins in $\mathcal{B}_2$. Their objective is to have Bob own α coins in $\mathcal{B}_1$, while Alice owns β coins in $\mathcal{B}_2$. This problem has been explored thoroughly by the research community and several solutions are proposed based on cryptographic protocols (e.g., [62]), smart contracts (e.g., [38, 40, 50]), and trusted hardware (e.g., [9]). However, these protocols are restricted to the coordinated exchange of assets whose ownership is tracked on the blockchain.

Intermediated Payment Coordination. Intermediated payment coordination involves at least three parties: Alice, Bob and an intermediary. We discuss three common approaches, *multi-hop payments*, *centralized coin mixers* and *cyclic swaps*. In multi-hop payments, Alice has α coins in $\mathcal{B}_1$, the intermediary has β coins in $\mathcal{B}_2$ and Bob operates in $\mathcal{B}_2$. Their objective is to have the intermediary own α coins in $\mathcal{B}_1$ while Bob owns β coins in $\mathcal{B}_2$. In this sense, Alice paid Bob using the intermediary as an exchange between $\mathcal{B}_1$ and $\mathcal{B}_2$. In practice, multi-hop payments have been proposed for scalability/layer 2 networks, such as the Lightning Network [4, 51, 55], or cross currency payments [25]. Multi-hop payment protocols coordinate the transfer of assets whose ownership is tracked by the blockchain. Moreover, the intermediary is able to link the incoming payment received from Alice with the outgoing payment to Bob. In order to prevent leaking such information to the intermediary, centralized coin mixers [28, 29, 33, 35, 37, 57, 59] have been proposed. Centralized coin mixers involve three type of parties: senders (Alice), receivers (Bob) and mixer (also called hub or tumbler). In this setting, the mixer collects α coins from each sender. Each receiver collects β coins from the mixer in a randomized order, which prevents the mixer from learning which sender paid to which receiver. Although centralized coin mixers provide unlinkability towards the mixer, they only model the coordinated transfer of assets whose ownership is tracked on the blockchain. In cyclic swaps [39, 41], Alice has α coins in $\mathcal{B}_1$, the intermediary has β coins in $\mathcal{B}_2$ and Bob has an asset, for example a product p whose ownership is tracked in $\mathcal{B}_3$. The objective is to have the intermediary own α coins in $\mathcal{B}_1$, Bob own β coins in $\mathcal{B}_2$ and Alice own p in $\mathcal{B}_3$. Although cyclic swaps can be used to model the intermediated purchase of a product p, note that the ownership of p is tracked by $\mathcal{B}_3$.

Two-party Contingent Payment. Two-party contingent payment, called zero-knowledge contingent payment (zkCP) (e.g. [12, 14, 21, 27, 52]), is an operation between a buyer and a seller. The buyer owns α coins in a blockchain $\mathcal{B}$ and the seller holds the product p. Crucially, in zkCP the ownership of p is not tracked in a blockchain. The goal of a zkCP is to have the buyer own the product p and the seller own the corresponding α coins in $\mathcal{B}$. Existing works in zkCP do not model the inclusion of a mixer.

Intermediated Contingent Payment. In practice, they derive from multi-hop payments. For instance, Alice owns α coins in $\mathcal{B}_1$, the mixer owns β coins in $\mathcal{B}_2$ and Bob, who also operates in $\mathcal{B}_2$, owns product p. The objective is to have Alice own product p, mixer own α coins in $\mathcal{B}_1$ and Bob own β coins in $\mathcal{B}_2$. This problem has been explored in practice with the Lightning Service Authentication Token (LSAT) [45, 54], but the security of the protocol has not been formally proven. Moreover, the mixer knows that Alice paid Bob and thus unlinkability is not achieved.

In summary, none of the existing related works give a satisfactory solution to the functionality of UCP.

1.2 Our Goal and Contributions.

As summarized in Table 1, none of the existing works simultaneously provide the functionality, security and privacy properties required by UCP. Hence, the following question naturally raises: *Can we provide a secure protocol for unlinkable contingent payment?*

We answer this question in the affirmative. For that, in this work we present MixBuy, the first protocol for unlinkable contingent payment. In particular:

- We describe MixBuy, which comprises two phases: the setup phase in which the shared addresses are prepared and funded whereas the product is prepared to be delivered; and the execution phase, in which the payment and product delivery takes place. We base our setup phase on prior work on zkCP, while the execution phase is a novel contribution of this work.
- We formalize the execution phase with the notion of *oracle-based unlinkable contingent payment* (O-UCP), a novel four-party cryptographic protocol where the mixer pays the seller and the seller delivers the product only if a semi-trusted notary attests that the buyer has paid the mixer. We present a provably secure and efficient cryptographic construction for O-UCP.
- We provide a proof of concept for O-UCP. Our performance evaluation in commodity hardware shows small communication overhead (few kB per message) and running times below one second, thereby demonstrating the practicality of our approach.

2 TECHNICAL OVERVIEW

2.1 Unlinkable Contingent Payment Overview

An unlinkable contingent payment (UCP) involves a product p, and three parties: buyer B, mixer M and seller S. As shown in Fig. 1, at the beginning of the UCP the buyer owns a key pair (vk_B, sk_B) that controls α coins. The mixer owns a key pair (vk_M, sk_M) that controls β coins. Finally, the seller owns a key pair (vk_S, sk_S) that represents seller's address. UCP is divided in two phases: *setup phase* and *execution phase*. We next overview the setup phase.

UCP: Setup Phase. During the setup phase, parties proceed as follows. In this description, we assume that there is a predefined timeout T known by every party that denotes the upper bound on the protocol completion time. First, the buyer and the mixer create a shared address (vk_B, vk_M) (e.g., in the form of 2-of-2 multisig), and the buyer transfers α coins to that shared address. Analogously, mixer and seller create a shared address (vk_M, vk_S) to which the mixer transfers β coins. Both shared addresses are set with a timeout T after which the coins can be refunded to their original owners.

Second, the seller prepares the delivery of digital product p to the buyer. As in zkCP protocols [12, 14, 27, 52], the seller samples an encryption/decryption key pair (pek, pdk) and encrypts the digital product p with the encryption key pek. Then, the seller generates a zero-knowledge proof π certifying that (i) the ciphertext is the encryption of p under pek; and (ii) p satisfies some predicate ϕ. For instance, the product p may be a file (e.g. digital book) and ϕ outputs 1 if hashing p results into some fixed value h (i.e., $h = \mathsf{H}(p)$).[1] The setup phase ends with the buyer checking the proof π.

The described setup phase is defined and analyzed in previous zkCP protocols. MixBuy also borrows this setup phase. The open technical challenge that we tackle in this work is thus the design of the execution phase. Next, we overview the expected functionality of the execution phase.

UCP: Execution Phase. The execution phase starts in a setting where α coins are locked in the shared address (vk_B, vk_M), β coins are locked in the shared address (vk_M, vk_S), and the buyer holds a pair (c, π), where c is the encryption of the product p under public key pek and π is a zero-knowledge proof. The execution phase must be designed to achieve the following outcomes: (1) mixer gets $\sigma_B \leftarrow \mathsf{Sig}(sk_B, m_B)$ from the buyer, where m_B is a transaction that transfers α coins from (vk_B, vk_M) to vk_M; (2) seller gets $\sigma_M \leftarrow \mathsf{Sig}(sk_M, m_M)$ from the mixer, where m_M is a transaction that transfers β coins from (vk_M, vk_S) to vk_S; (3) buyer gets pdk and thus can get p decrypting ciphertext c. Hence, it must ensure buyer security, mixer security, seller security and unlinkability.

Designing such a protocol is technically challenging. Among the properties that such protocol needs to provide, we find unlinkabilty to be the most challenging one, motivating us to inspire our approach from centralized coin mixers [28, 29, 33, 35, 37, 57, 59]. In a nutshell, a centralized coin mixing protocol provides the same outcomes (1) and (2) as required by the execution phase of UCP. However, a direct application of a centralized coin mixing protocol would fail to provide outcome (3). Moreover, in the coin mixing setting, buyer and seller must collaborate with each other to arrive to the desired outcomes (1) and (2), an assumption that cannot be made in UCP, where buyer and seller are mutually distrustful.

[1] The reader might wonder how buyer knows if h corresponds to $H(p)$ (i.e., a malicious seller has not used $h' = H(p')$). This is an orthogonal problem for which solutions exist (e.g., a penalization mechanism is proposed in [21]).

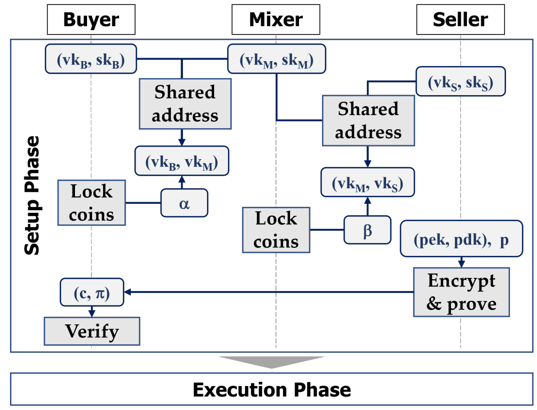

Figure 1: Buyer and mixer create the shared address (vk_B, vk_M) which the buyer funds with α coins. Mixer and seller create the shared address (vk_M, vk_S) which the mixer funds with β coins. Finally, seller encrypts the product (c) and proves in zero knowledge (π) that c contains the product.

2.2 Towards our Solution

For context, we first overview how a centralized coin mixing protocol works. In particular, we review the puzzle-promise and puzzle-solve paradigm, first introduced in [37], and later followed by other designs of centralized coin mixing protocols.

The Puzzle-promise and Puzzle-solve Paradigm. A centralized coin mixing protocol assumes that the same setup as described for UCP has been successfully executed, except for the preparation for the delivery of the product that is naturally not considered. Concretely, there are also three parties: Alice, mixer, and Bob. α coins are locked in shared address (vk_{Alice}, vk_M), and β coins are locked in shared address (vk_M, vk_{Bob}).

The protocol is run in epochs and consists of two steps, namely, puzzle-promise and puzzle-solve (cf. Fig. 2 (i)).
Puzzle-promise. During epoch $\mathcal{E}_i$, the mixer hides signature σ_M in a *randomizable puzzle* rP_1 and sends it to Bob. A randomizable puzzle ensures that one cannot learn σ_M from rP_1. Bob verifies that learning the solution s_1 corresponding to rP_1 would allow to extract σ_M. In the affirmative case, Bob chooses a random value r and uses it to randomize rP_1 into rP_2 so that they cannot be linked together. Moreover, the solution s_2 to rP_2 is a randomization of s_1 with r. Bob sends rP_2 to Alice, who holds it until the end of epoch $\mathcal{E}_i$.
Puzzle-solve. At the beginning of epoch $\mathcal{E}_{i+1}$, Alice engages with the mixer in a protocol where the mixer gets σ_A only if Alice learns s_2. Alice forwards s_2 to Bob, who in turn can derandomize it to obtain s_1 and then σ_M from rP_1.

The key observation regarding unlinkability is that the randomization factor r is unknown to the mixer, hence the mixer cannot link rP_1 to rP_2. Assume n honest Bobs that interact with the mixer

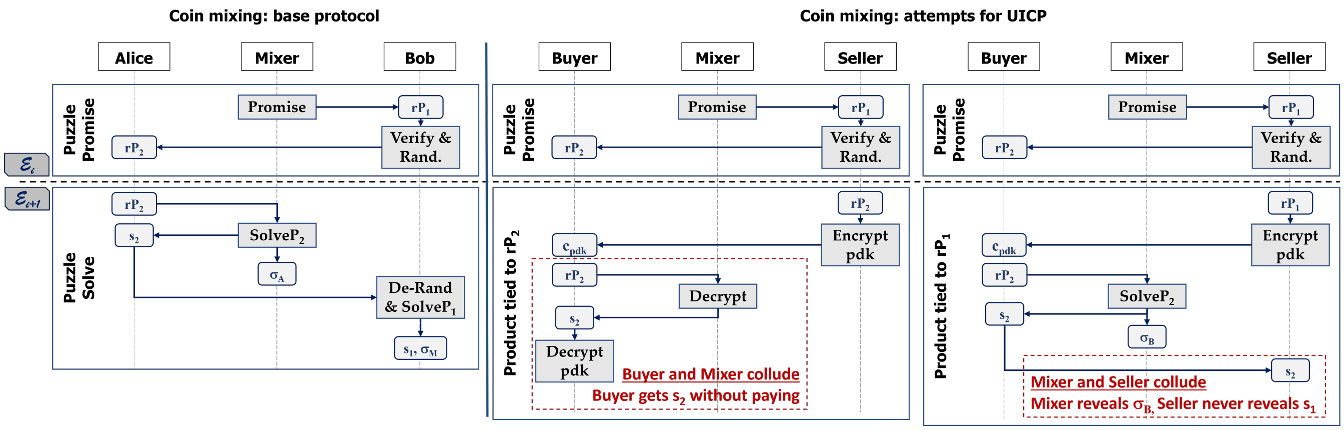

Figure 2: From left to right: (i) *Coin mixing protocol.* During puzzle-promise, mixer sends rP_1 to Bob, who randomizes into rP_2 and sends it to Alice. During puzzle-solve, Alice provides mixer with rP_2, who solves it, allowing Alice to learn s_2 and mixer to publish σ_A. Then, Alice forwards s_2 to Bob, who derandomizes it to get s_1 and solve rP_1, obtaining σ_M. (ii) *Attempt to build UCP linking the reveal of product decryption key pdk to* s_2. After puzzle-promise, the seller encrypts *pdk* using rP_2 and forwards c_{pdk} to the buyer. *Attack:* buyer and mixer collude such that mixer reveals s_2 without the buyer publishing σ_B. Hence, the buyer gets *pdk* without paying. (iii) *Attempt to build UCP linking the reveal of pdk to* s_1. After puzzle-promise, the seller encrypts *pdk* using rP_1 and forwards c_{pdk} to the buyer. Buyer and mixer begin puzzle-solve, which reveals s_2 and publish σ_B. *Attack:* seller and mixer collude such that seller never de-randomizes s_2 to reveal s_1 or σ_M. Hence, the buyer paid but did not receive *pdk*.

during epoch $\mathcal{E}_i$ (i.e., puzzle-promise step). Thenceforth, n corresponding honest Alices interact with the mixer in any order during $\mathcal{E}_{i+1}$ (i.e., puzzle-solve step). Following the aforementioned observation, the mixer cannot link who paid to whom, up to what is leaked by the content of the transactions themselves (e.g., payment amounts). We discuss these system aspects in Section 6. The unlinkability of the puzzle-promise, puzzle-solve paradigm in coin mixing protocols has been formally analyzed in [29].

Limitations of Puzzle-promise, Puzzle-solve Paradigm in UCP. Recall that a two-party contingent payment ties the published transaction paying the seller to the disclosure of the product decryption key *pdk* (hence, the delivery of product *p*) to the buyer. In other words, the buyer engages with the seller in a protocol where the seller gets σ_B only if buyer learns *pdk*. Likewise, in UCP we want to tie the published transaction on the blockchain that sends β coins from the mixer to the seller (i.e., m_M), to the disclosure of *pdk*. Note that if we use the puzzle-promise, puzzle-solve paradigm off-the-shelf as implementation of the execution phase in UCP, we are missing the guarantee that the buyer learns *pdk*.

Designing such a protocol is technically challenging. In the next, we describe how any attempt to leverage the blockchain in such a manner that one of the solutions s_1, s_2 to puzzles rP_1, rP_2, respectively, leads to the reveal of *pdk* is futile. The root of the problem lies in the fact that contrary to the puzzle-promise puzzle-solve paradigm, where the buyer and the seller cooperate in order to route an unlikable payment via the mixer, in the UCP setting the three parties are mutually distrustful.

More specifically, assume that we tie the disclosure of *pdk* to puzzle solution s_2, e.g., by encrypting *pdk* into ciphertext c_{pdk} such that it can only be decrypted with s_2. We deploy a smart contract that reveals s_2 if transaction m_B, that sends α coins from the buyer to the mixer, is published (cf. Fig. 2 (ii)). Nevertheless, the following attack on seller security is possible: at the end of the puzzle-promise step, when the honest seller forwards the puzzle rP_2 to the malicious buyer, the latter can collude with the malicious mixer such that the buyer learns the puzzle solution s_2 without publishing the transaction m_B. As a result, the buyer can use s_2 to get *pdk*, while the seller does not get paid because s_2 cannot be obtained from the blockchain.

Conversely, assume that we tie the disclosure of *pdk* to puzzle solution s_1 and we deploy a smart contract that reveals s_1 if transaction m_M, sending β coins from the mixer to the seller, is published. (cf. Fig. 2 (iii)). Nevertheless, the following attack on buyer security is possible: during the puzzle-solve step, the malicious seller colludes with the malicious mixer such that the latter does not publish transaction m_M.[2] As a result, the honest buyer, who according to the puzzle-promise, puzzle-solve paradigm has already published transaction m_B, does not get *pdk* because s_1 cannot be obtained.

Solving the Fair Exchange Problem. In order to cope with the above deadlock, we introduce a fourth party, called *notary*, that is trusted to carry out a simple task, namely, to attest all transactions published on the blockchain. A transaction's attestation is a signature on such transaction verifiable under the notary's verification key $\widehat{vk}$, that is disseminated through a public channel (e.g., a bulletin board or a blockchain). The notary's functionality is thus similar to that of oracle and data feeds that have been largely studied in the literature [20, 43, 47, 49, 63, 65] and deployed solutions exist.[3] The advantages of such limited trust on the notary are twofold: (a) the notary is oblivious about what attested transactions are used for, i.e., no communication between the notary and the other three parties is required in order to carry out an UCP; and

[2] Colluding parties can split buyer's coins with a transaction different to m_M.
[3] ChainLink: https://chain.link; SupraOracles: https://supra.com

(b) the limited requirements on notary's functionality reduces the burden on deploying it in practice.

A key technical contribution of our work is a novel cryptographic construction that leverages notary's attestation to tie the inclusion of m_B in the blockchain (i.e., buyer's payment to the mixer) to both: (i) the disclosure of decryption key *pdk* to the buyer; and (ii) the disclosure of σ_M to the seller. In this construction, notary's attestation is independent of the authorization scheme of the blockchain and is only required for security, but not for unlinkability. Next, we overview this construction (and the rest of MixBuy).

2.3 Overview of MixBuy

MixBuy provides the functionality of UCP, ensuring buyer security, mixer security, seller security and unlinkability.

Setup Phase. The setup phase in MixBuy is identical to the one described in Section 2.1. Additionally, the notary's verification key $\widehat{vk}$ is disseminated through a public channel (e.g., a bulletin board or a blockchain).

Execution Phase. The execution phase in MixBuy is run in epochs and consists of three steps, namely, *puzzle-promise*, *puzzle-link*, and *attest-and-solve*, as shown in Fig. 3. The puzzle-promise step comprises the same operations as the puzzle-promise step of coin mixing. On the contrary, steps puzzle-link and attest-and-solve fully differ from the puzzle-solve in coin mixing and instead are based on a novel cryptographic construction described hereafter.

Puzzle-promise. During epoch $\mathcal{E}_i$, the mixer creates a re-randomizable puzzle rP_1 containing σ_M and sends it to the seller, who randomizes it into rP_2. Buyer receives rP_2 and holds it until epoch $\mathcal{E}_i$ ends.

Puzzle-link. At the beginning of epoch $\mathcal{E}_{i+1}$, the buyer forwards rP_2 to the mixer. Note that at this point, similarly to the puzzle-promise, puzzle-solve paradigm, the randomization factor r used by the seller to randomize s_1 is unknown to the mixer, hence the mixer cannot link s_1 to s_2. In this way, MixBuy achieves unlinkability.

The mixer then opens rP_2 and includes the solution s_2 into an *attestation puzzle* aP_3. It is crucial to see here that although rP_2 and aP_3 hide the same value, we have designed attestation puzzle aP_3 in such a way that it can be opened only if the notary attests m_B (i.e., a payment from the buyer to the mixer). At this point, the mixer is ensured that in order to obtain the solution to rP_2, the buyer must have included m_B in the blockchain, meaning that the mixer has got α coins from the buyer if rP_2 (hence, rP_1) is solved. In this way, MixBuy achieves mixer security.

Thereafter, the mixer sends aP_3 to the buyer who forwards it to the seller. The possession of aP_3 ensures the seller that if the buyer pays the mixer using m_B, then the notary will provide an attestation for such transaction (i.e., the notary is trusted for this task), thus the seller opens aP_3, learns the solution s_2, de-randomizes it to learn s_1 and finally obtains σ_M from rP_1. Henceforth, the seller is safe to provide the buyer with an attestation puzzle aP_4 containing the product decryption key *pdk* that the buyer needs to obtain the product. As with aP_3, the solution to aP_4 can only be obtained if the notary attests m_B. In this way, MixBuy achieves seller security.

Finally, the buyer in possession of aP_4 is guaranteed that publishing m_B will release the decryption key *pdk*. In this way, MixBuy achieves buyer security.

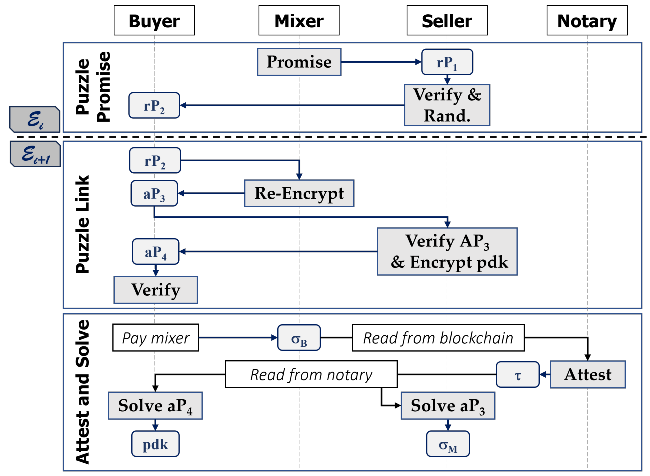

Figure 3: MixBuy execution phase. *Puzzle-promise* as in coin mixing (cf. Fig. 2). In *Puzzle-link*, mixer re-encrypts s_2 into aP_3 and seller encrypts the product decryption key *pdk* into aP_4. In *Attest-and-solve*, the oracle attests buyer's payment, so buyer and seller can solve aP_3 and aP_4.

Attest-and-solve. At this point, the buyer is in the unique position to trigger the final operations of the attest-and-solve step by submitting m_B. After m_B is published, the notary outputs its attestation that the payment occurred. Thereafter, the buyer can use the attestation to solve puzzle aP_4, retrieve the product decryption key *pdk* and get the product p. Likewise, the seller can use the attestation to solve puzzle aP_3, learn the solution to puzzle rP_2, recover the solution to the puzzle rP_1, and then submit m_M.

3 PRELIMINARIES

Notation. We denote by λ the security parameter. Symbol ($\xleftarrow{\$}$) denotes the sampling of an element at random from a uniform distribution, ($\leftarrow$) is used to store values from a probabilistic operation, (:=) is used to assign values from a deterministic operation, and ($\leftharpoondown$) is used to parse data from a variable. Furthermore, *ek*, *dk*, *vk*, and *sk* denote encryption, decryption, verification, and signing keys, respectively. We consider *probabilistic polynomial time* (PPT) and *deterministic polynomial time* (DPT) machines as efficient algorithms. In security games, adversaries are stateful.

Relation. We recall the notion of a relation. For that, let $\mathsf{R} \subseteq \mathcal{D}_S \times \mathcal{D}_w$ be a relation with statement/witness pairs $(\mathsf{X}, \mathsf{w}) \in \mathcal{D}_S \times \mathcal{D}_w$. We denote by $\mathcal{L}_\mathsf{R}$ the associated language defined as $\mathcal{L}_\mathsf{R} := \{\mathsf{X} \in \mathcal{D}_S \mid \exists \mathsf{w} \in \mathcal{D}_w \text{ s.t. } (\mathsf{X}, \mathsf{w}) \in \mathsf{R}\}$. For any relation that we consider in this paper, we require the following two properties: (i) There exists a PPT algorithm $\mathsf{createR}(1^\lambda)$ that computes $(\mathsf{X}, \mathsf{w}) \in \mathsf{R}$ (note that this implies that $|\mathsf{X}|, |\mathsf{w}| \le \mathsf{poly}(\lambda)$); and (ii) the relation is decidable in polynomial time. Furthermore, we say that R is a *hard relation* if for all PPT adversaries $\mathcal{A}$, the probability that on input X $\mathcal{A}$ outputs w such that $(\mathsf{X}, \mathsf{w}) \in \mathsf{R}$ is negligible, where the probability is taken over the coins of $\mathcal{A}$ and $(\mathsf{X}, \mathsf{w}) \leftarrow \mathsf{createR}(1^\lambda)$. We say that a relation is *linearly homomorphic* if there exist a pair of operations $(\otimes, +)$ such that for $(\mathsf{X}_1, \mathsf{w}_1) \in \mathsf{R}$, $(\mathsf{X}_2, \mathsf{w}_2) \in \mathsf{R}$ it holds that $(\mathsf{X}_1 \otimes \mathsf{X}_2, \mathsf{w}_1 + \mathsf{w}_2) \in \mathsf{R}$.

Digital Signature Scheme. We require a digital signature scheme [31] $\mathsf{DS} := (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$, where: (i) PPT algorithm KGen gets as input the security parameter 1^λ and outputs a verification/signing key pair (vk, sk); (ii) PPT algorithm Sig gets as input a signing key sk and a message m, and outputs a signature σ; and (iii) DPT algorithm Vf gets as input a verification key vk, a message m, and a signature σ, and outputs 1 if σ is a valid signature on m under vk, otherwise it outputs 0. We require a correct DS (i.e. it holds that $\Pr[\mathsf{Vf}(vk, m, \mathsf{Sig}(sk, m)) = 1] = 1$) and secure for existential unforgeability under chosen message attack (EUF-CMA).

Adaptor Signature Scheme. We require an adaptor signature scheme [4, 17] $\mathsf{ADP} := (\mathsf{PreSig}, \mathsf{PreVf}, \mathsf{Adapt}, \mathsf{Extract})$, defined with respect to a digital signature scheme DS and a relation R where: (i) PPT algorithm PreSig gets as input a signing key sk, a message m, and a public statement X, and outputs a pre-signature $\widetilde{\sigma}$; (ii) DPT algorithm PreVf gets as input a verification vk, a message m, a public statement X and a pre-signature $\widetilde{\sigma}$ and outputs 1 if $\widetilde{\sigma}$ is a valid pre-signature on m under vk and X, otherwise it outputs 0; (iii) DPT algorithm Adapt gets as input a pre-signature $\widetilde{\sigma}$ and a witness w, and outputs a signature σ; and (iv) DPT algorithm Extract gets as input a signature σ, a pre-signature $\widetilde{\sigma}$ and a public statement X, and outputs a witness w. We require a correct ADP, secure for full extractability and adaptability, as defined in [17].

Non-Interactive Zero Knowledge. Let R be a hard relation with corresponding $\mathcal{L} := \{\mathsf{X} \mid \exists \mathsf{w} \text{ s.t. } (\mathsf{X}, \mathsf{w}) \in \mathsf{R}\}$. We require a non-interactive zero-knowledge proof system [19] $\mathsf{NIZK} := (\mathsf{SetUp}, \mathsf{Prove}, \mathsf{Vf})$, for relation R, where: (i) PPT algorithm SetUp gets as input the security parameter 1^λ and outputs a common reference string crs and a trapdoor td; (ii) PPT algorithm Prove gets as input a crs, a public statement X and a witness w, and outputs a proof π; and (iii) DPT algorithm Vf gets as input a crs, a public statement X and a proof π, and outputs 1 if π is a valid proof, otherwise it outputs 0. We require three security properties, namely, completeness, zero-knowledge, and knowledge-soundness [7].

Witness Encryption based on Signatures. We require a witness encryption based on signatures scheme $\mathsf{WES} := (\mathsf{Enc}, \mathsf{Dec})$, defined with respect to a digital signature scheme $\widehat{\mathsf{DS}} = (\widehat{\mathsf{KGen}}, \widehat{\mathsf{Sig}}, \widehat{\mathsf{Vf}})$, where: (i) PPT algorithm Enc gets as input a tuple comprising a verification key $\widehat{vk}$ and a message $\hat{m}$, a plaintext m, and outputs a ciphertext c; and (ii) DPT algorithm Dec gets as input a signature $\hat{\sigma}$ and a ciphertext c, and outputs a plaintext m. We say that WES is correct if it holds that $\Pr[\mathsf{Dec}(\widehat{\mathsf{Sig}}(\widehat{sk}, \hat{m}), \mathsf{Enc}((\widehat{vk}, \hat{m}), m)) = m] = 1$, and we require the security notion of indistinguishability under chosen plaintext attack (IND-CPA) as defined in [49].

Verifiable Witness Encryption for a Relation. We require a verifiable witness encryption for a relation scheme $\mathsf{VWER} := (\mathsf{EncR}, \mathsf{VfEncR}, \mathsf{DecR})$, defined with respect to a relation R and a digital signature scheme $\widehat{\mathsf{DS}} = (\widehat{\mathsf{KGen}}, \widehat{\mathsf{Sig}}, \widehat{\mathsf{Vf}})$, where: (i) PPT algorithm EncR gets as input a tuple comprising a verification key $\widehat{vk}$ and a message $\hat{m}$, a a witness w, and outputs a ciphertext tuple, containing ciphertext and a proof (c, π); (ii) DPT algorithm VfEncR gets as input a ciphertext tuple, containing ciphertext and a proof (c, π), a tuple comprising a verification key $\widehat{vk}$ and a message $\hat{m}$ and a public statement X, and outputs 1 if it is a valid ciphertext, otherwise it outputs 0; and (iii) DPT algorithm Dec gets as input a signature $\hat{\sigma}$ and and tuple comprising a ciphertext c and a proof π, and outputs a witness w'. We require a VWER secure for one-wayness, which guarantees that w can be recovered from c only with a valid signature $\widehat{\sigma}$ on message $\widehat{m}$ under verification key $\widehat{vk}$; and verifiability, which guarantees that if π verifies, c encrypts w such that $(\mathsf{X}, \mathsf{w}) \in \mathsf{R}$. We provide formal definitions of VWER and its security properties in Appendix A. In Appendix C we provide a construction of VWER, together with the security proofs.

Linear-Only Homomorphic Encryption Scheme. We require a linear-only homomorphic encryption scheme $\mathsf{LHE} := (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ [34], where: (i) PPT algorithm KGen gets as input the security parameter 1^λ and outputs a encryption/description key pair (ek, dk); (ii) PPT algorithm Enc gets as input an encryption key ek and a plaintext m, and outputs a ciphertext c; and (iii) DPT algorithm Dec gets as input a decryption key dk and a ciphertext c, and outputs a plaintext m. We say that LHE is correct if it holds that $\Pr[\mathsf{Dec}(dk, \mathsf{Enc}(ek, m)) = m] = 1$ and we require the standard notion of indistinguishability under chosen plaintext attack (IND-CPA) [30]. An encryption scheme is *linearly homomorphic* if there exists a pair of operations $(\circ, +)$ such that $\mathsf{Enc}(ek, m_1) \circ \mathsf{Enc}(ek, m_2) = \mathsf{Enc}(ek, m_1 + m_2)$. We define an additional property called OMDL-LHE because it becomes useful to prove the security of our proposed construction in Section 5. We provide the intuition in the following, while the formal definition is in Appendix A. In OMDL-LHE, the challenger generates an encryption/decryption key pair and a list of $k+1$ (statement, witness) pairs. Then, encrypts all witnesses with the encryption key and provides the encryption key, the statements and ciphertexts to the adversary. The adversary has access to a decryption oracle. If the adversary is able to return more valid witnesses than queries to the decryption oracle, wins the game.

4 MIXBUY: OUR APPROACH FOR UCP

Environment. Protocol MixBuy involves a digital product p, and four parties: buyer B, mixer M, seller S, and notary N. The buyer owns a key pair (vk_B, sk_B) that controls α coins. The mixer owns a key pair (vk_M, sk_M) that controls β coins. The seller owns a key pair (vk_S, sk_S) that represents seller's address, The notary owns a key pair $(\widehat{vk}, \widehat{sk})$ and attest all transactions published on the blockchain. These attestations are disseminated through a public channel (e.g., a bulletin board or a blockchain). For ease of exposition, we describe the notary functionality as a single party, nevertheless we can distribute this functionality to a set of notaries (as discussed in Section 6). Finally, we assume the existence of a public inventory in the form of a key-value store that maps digital product p to its hash value h (i.e., $h := \mathsf{H}(p)$).

Threat Model. The three parties carrying out an unlinkable contingent payment, namely, the buyer, the mixer, and the seller are mutually distrustful. The notary is only trusted to correctly attest all transactions published on the blockchain. Moreover, we assume the blockchain accepts a transaction m only if it is accompanied by a digital signature σ that correctly verifies with the corresponding verification key vk. Finally, we assume that the communication between buyer and seller is not visible to the mixer, which is a common assumption in centralized coin mixing services [29, 37, 59].

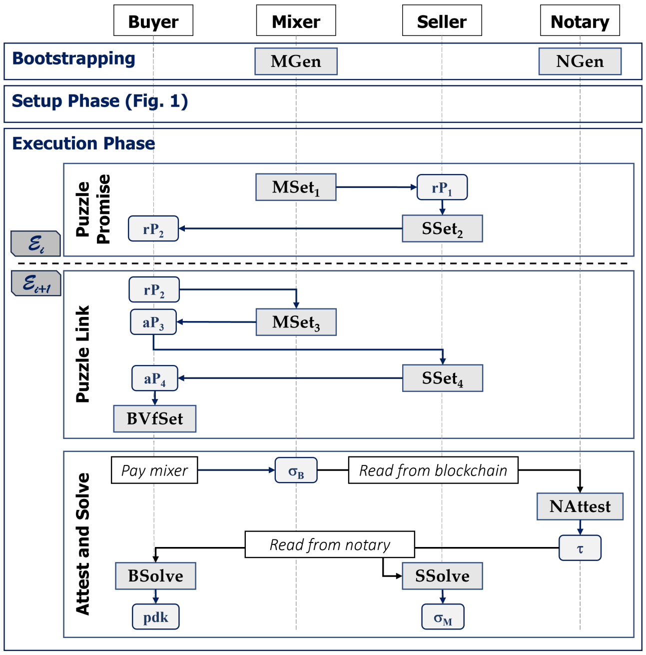

Figure 4: MixBuy protocol. *Bootstrapping:* mixer and notary generate the encryption/decryption key and the verification/decryption key. *Setup phase:* buyer, mixer and seller prepare the purchase (cf. Fig. 1). *Execution phase:* involves three steps, puzzle-promise, puzzle-link, and attest-and-solve.

4.1 Protocol Definition

In this section, we define *oracle-based unlinkable contingent payment* (O-UCP), our novel cryptographic protocol for MixBuy's execution phase. Thereafter, we show how O-UCP is used in MixBuy to execute unlinkable contingent payment. Finally, we formally describe the security and unlinkability properties of O-UCP.

Naming convention for the algorithms in Definition 1. The first letter indicates the party invoking the algorithm (e.g., seller S), the name of the algorithm follows (e.g., Set), and the subscript indicates the order of execution where appropriate. We denote randomizable puzzles by rP and attestation puzzles by aP.

Definition 1 (Oracle-based Unlinkable Contingent Payment)**.** *The oracle-based unlinkable contingent payment is defined w.r.t. a digital signature scheme* $\mathsf{DS} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ *and a relation* R. *It comprises 11 algorithms* $(\mathsf{MGen}, \mathsf{NGen}, \mathsf{MSet}_1, \mathsf{SSet}_2, \mathsf{MSet}_3, \mathsf{SSet}_4, \mathsf{BVfSet}, \mathsf{NAttest}, \mathsf{VfAttest}, \mathsf{SSolve}, \mathsf{BSolve})$*, defined bellow:*

- $\underline{(\overline{ek}, \overline{dk}) \leftarrow \mathsf{MGen}(1^\lambda)}$*: PPT algorithm invoked by mixer gets as input the security parameter* 1^λ *and outputs the keypair* $(\overline{ek}, \overline{dk})$.
- $\underline{(\widehat{vk}, \widehat{sk}) \leftarrow \mathsf{NGen}(1^\lambda)}$*: PPT algorithm invoked by notary, gets as input the security parameter* 1^λ *and outputs the notary verification/signing keypair* $(\widehat{vk}, \widehat{sk})$.
- $\underline{rP_1 \leftarrow \mathsf{MSet}_1(\overline{ek}, sk_M, m_M)}$*: PPT algorithm invoked by mixer, gets as input mixer's encryption and signing keys* $\overline{ek}$ *and* sk_M*, and a transaction* m_M *from mixer to seller , and outputs randomizable puzzle* rP_1.
- $\underline{\{(rP_2, st_S), \perp\} \leftarrow \mathsf{SSet}_2(\overline{ek}, vk_M, m_M, rP_1)}$*: PPT algorithm invoked by seller, gets as input mixer's encryption key* $\overline{ek}$*, mixer's verification key* vk_M*, a transaction from mixer to seller* m_M*, and a randomizable puzzle* rP_1*, and outputs either a tuple comprising randomizable puzzle* rP_2 *and seller's secret state* st_S*, or aborts* $(\perp)$.
- $\underline{aP_3 \leftarrow \mathsf{MSet}_3(\overline{dk}, \widehat{vk}, m_B, rP_2)}$*: PPT algorithm invoked by mixer, gets as input mixer's decryption key* $\overline{dk}$*, notary's verification key* $\widehat{vk}$*, a transaction from buyer to mixer* m_B*, and randomizable puzzle* rP_2*, and outputs attestation puzzle* aP_3.
- $\underline{\{aP_4, \perp\} \leftarrow \mathsf{SSet}_4(\widehat{vk}, m_B, pdk, aP_3, st_S)}$*: PPT algorithm invoked by seller, gets as input notary's verification key* $\widehat{vk}$*, a transaction from buyer to mixer* m_B*, product's decryption key* pdk*, attestation puzzle* aP_3*, and seller's secret state* st_S*, and outputs either attestation puzzle* aP_4*, or aborts* $(\perp)$.
- $\underline{1/0 \leftarrow \mathsf{BVfSet}(\widehat{vk}, m_B, pek, aP_4)}$*: DPT algorithm invoked by buyer, gets as input notary's verification key* $\widehat{vk}$*, a transaction from buyer to mixer* m_B*, product's encryption key* pek*, and attestation puzzle* aP_4*, and outputs* 1 *if puzzle* aP_4 *hides the corresponding product's decryption key* pdk*, otherwise it outputs* 0.
- $\underline{\tau \leftarrow \mathsf{NAttest}(\widehat{sk}, m_B)}$*: PPT algorithm invoked by notary, gets as input notary's signing key* $\widehat{sk}$ *and a transaction from buyer to mixer* m_B*, and outputs the attestation token* τ.
- $\underline{1/0 \leftarrow \mathsf{VfAttest}(\widehat{vk}, m_B, \tau)}$*: DPT algorithm gets as input notary's verification key* $\widehat{vk}$*, a transaction from buyer to mixer* m_B*, and an attestation token* τ*, and returns* 1 *if* τ *is a valid attestation on* m_B *under the key* $\widehat{vk}$*, otherwise it outputs* 0.
- $\underline{\sigma_M \leftarrow \mathsf{SSolve}(\tau, rP_1, aP_3, st_S)}$*: DPT algorithm invoked by seller, gets as input an attestation token* τ*, puzzle* rP_1*, attestation puzzle* aP_3*, and seller's secret state* st_S*, and outputs a signature* σ_M.
- $\underline{pdk \leftarrow \mathsf{BSolve}(\tau, aP_4)}$*: DPT algorithm invoked by buyer, gets as input an attestation token* τ *and attestation puzzle* aP_4*, and outputs product's decryption key* pdk.

O-UCP in MixBuy. Hereby, we show how O-UCP is used in MixBuy to execute an unlinkable contingent payment. The protocol is divided in three phases, namely, bootstrapping, setup phase, and execution phase (cf. Fig. 4).

<u>Bootstrapping.</u> During bootstrapping, the mixer and the notary invoke algorithms MGen and NGen, respectively, in order to generate their key pairs $(\overline{ek}, \overline{dk})$ and $(\widehat{vk}, \widehat{sk})$. Bootstrapping is executed only once at the time of deploying MixBuy.

<u>Setup phase.</u> The setup phase in MixBuy is identical to the one described in Section 2.1 with the addition of the dissemination of notary's verification key $\widehat{vk}$.

<u>Execution phase.</u> The execution phase in MixBuy is run in epochs and consists of three steps, namely, *puzzle-promise*, *puzzle-link*, and *attest-and-solve*:

- *Puzzle-promise.* In epoch $\mathcal{E}_i$, mixer invokes MSet_1 to create rP_1 and sends it to the seller. In turn, seller invokes SSet_2 to check if rP_1 is well-formed and randomizes it into rP_2. Finally, seller sends rP_2 to the buyer, who holds it until the end of epoch $\mathcal{E}_i$.

- *Puzzle-link.* At the beginning of epoch $\mathcal{E}_{i+1}$, the buyer sends rP_2 and transaction m_B to the mixer. Thereafter, the mixer invokes MSet_3 that outputs attestation puzzle aP_3, which can be solved with notary's attestation τ on m_B. The mixer sends aP_3 to the buyer, who in turn forwards it to the seller. The seller invokes SSet_4 that verifies that aP_3 is well-formed and outputs attestation puzzle aP_4, which encrypts the product decryption key pdk and can be solved with notary's attestation τ on m_B. The seller sends aP_4 to the buyer, who runs BVfSet to check if aP_4 is well-formed.
- *Attest-and-solve.* The attest-and-solve step is triggered with the submission of transaction m_B by the buyer. Thereafter, the notary invokes NAttest to create attestation τ, which is disseminated through a public channel. Finally, the buyer and the seller use τ to invoke BSolve and SSolve, respectively, in order to get the product decryption key pdk and the authorization σ_M.

Definition 2 (O-UCP Correctness)**.** *A O-UCP is said to be* correct *if for all $\lambda \in \mathbb{N}$, all $(\widehat{vk}, \widehat{sk}) \in \mathsf{NGen}(1^\lambda)$, all $(\overline{ek}, \overline{dk}) \in \mathsf{MGen}(1^\lambda)$, all $(vk_M, sk_M) \in \mathsf{KGen}(1^\lambda)$, all $(vk_B, sk_B) \in \mathsf{KGen}(1^\lambda)$, all $(pek, pdk) \in \mathsf{R}$, and all pairs of messages (m_B, m_M), it holds that:*

$$\Pr\left[\begin{array}{c|c} & rP_1 \leftarrow \mathsf{MSet}_1(\overline{ek}, sk_M, m_M) \\ & (rP_2, st_S) \leftarrow \mathsf{SSet}_2(\overline{ek}, vk_M, m_M, rP_1) \\ & aP_3 \leftarrow \mathsf{MSet}_3(\overline{dk}, \widehat{vk}, m_B, rP_2) \\ b_0 = 1 & aP_4 \leftarrow \mathsf{SSet}_4(\widehat{vk}, m_B, pdk, aP_3, st_S) \\ \wedge\ b_1 = 1 & \sigma_B \leftarrow \mathsf{Sig}(sk_B, m_B);\ \tau \leftarrow \mathsf{NAttest}(\widehat{sk}, m_B) \\ \wedge\ b_2 = 1 & \sigma_M \leftarrow \mathsf{SSolve}(\tau, rP_1, aP_3, st_S) \\ \wedge\ b_3 = 1 & pdk' \leftarrow \mathsf{BSolve}(\tau, aP_4) \\ \wedge\ b_4 = 1 & b_0 := \mathsf{BVfSet}(\widehat{vk}, m_B, pek, aP_4) \\ & b_1 := \mathsf{Vf}(vk_B, m_B, \sigma_B);\ b_2 := \mathsf{Vf}(vk_M, m_M, \sigma_M) \\ & b_3 := \mathsf{VfAttest}(\widehat{vk}, m_B, \tau);\ b_4 := (pek, pdk') \in \mathsf{R} \end{array}\right] = 1$$

Mixer Security. This property protects the balance of the mixer such that if the mixer pays to the seller, the former will be paid by the buyer. When interacting with an mixer in O-UCP, an adversary might stop when reaching MSet_1, MSet_3, or at the end. We model this with $\mathcal{O}\mathsf{MSet}_1$, $\mathcal{O}\mathsf{MSet}_3$, and $\mathcal{O}\mathsf{Full}$. Note that for a given transaction m_B, the adversary may choose to pay (hence, attestation exists) or not to pay (hence, attestation does not exist). Regardless of adversary's decision, the mixer will give only one attestation puzzle per transaction to the adversary (i.e., $\mathcal{O}\mathsf{MSet}_3$ and $\mathcal{O}\mathsf{Full}$ are mutually exclusive). The adversary returns a set of tuples comprising mixer's verification keys vk^i_M, messages m^i_M and signatures σ^i_M. The set contains one tuple more than the number of completed interactions with the mixer (i.e., the number of $\mathcal{O}\mathsf{Full}$ calls). We model two scenarios in which the adversary wins. If one of the tuples contains a valid forgery for a message that was not queried in $\mathcal{O}\mathsf{MSet}_1$ (condition b_0), the adversary wins. Alternatively, the adversary wins if all tuples contain different messages m^i_M queried in $\mathcal{O}\mathsf{MSet}_1$ and all signatures σ^i_M are valid (conditions b_1 and b_2). The second winning condition implies that the adversary managed to obtain information from rP_1 or aP_3 without an attestation.

Definition 3 (Mixer Security)**.** *A O-UCP offers mixer security if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all $\lambda \in \mathbb{N}$ and for all PPT adversaries $\mathcal{A}$ it holds that* $\Pr[\mathsf{ExpM}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$, *with* ExpM *defined in Fig. 5.*

ExpM

$$\begin{array}{l} Q_1 := \emptyset;\ Q_2 := \emptyset;\ \mathsf{q} := 0 \\ (\widehat{vk}, \widehat{sk}) \leftarrow \mathsf{NGen}(1^\lambda) \\ (\overline{ek}, \overline{dk}) \leftarrow \mathsf{MGen}(1^\lambda) \\ \{(vk^i_M, m^i_M, \sigma^i_M)\}_{i\in[0,\mathsf{q}]} \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{MSet}_1, \mathcal{O}\mathsf{MSet}_3, \mathcal{O}\mathsf{Full}}(\overline{ek}, \widehat{vk}) \\ b_0 := \exists i \in [0,\mathsf{q}] \text{ s.t. } (vk^i_M, \cdot) \in Q_1 \\ \quad \wedge (vk^i_M, m^i_M) \notin Q_1 \wedge \mathsf{Vf}(vk^i_M, m^i_M, \sigma^i_M) = 1 \\ b_1 := \forall i \in [0,\mathsf{q}], (vk^i_M, m^i_M) \in Q_1 \wedge \mathsf{Vf}(vk^i_M, m^i_M, \sigma^i_M) = 1 \\ b_2 := \forall i, j \in [0,\mathsf{q}], i \neq j, (vk^i_M, m^i_M, \sigma^i_M) \neq (vk^j_M, m^j_M, \sigma^j_M) \\ \textbf{return } b_0 \vee (b_1 \wedge b_2) \end{array}$$

$$\begin{array}{l|l} \underline{\mathcal{O}\mathsf{MSet}_1(m_M)} & \underline{\mathcal{O}\mathsf{Full}(m_B, rP_2, \sigma, vk)} \\ (vk_M, sk_M) \leftarrow \mathsf{KGen}(1^\lambda) & \textbf{if } m_B \in Q_2 \textbf{ abort} \\ rP_1 \leftarrow \mathsf{MSet}_1(\overline{ek}, sk_M, m_M) & \mathsf{q} = \mathsf{q} + 1 \\ Q_1 := Q_1 \cup (vk_M, m_M) & Q_2 := Q_2 \cup (m_B) \\ \textbf{return } (rP_1, vk_M) & aP_3 \leftarrow \mathsf{MSet}_3(\overline{dk}, \widehat{vk}, m_B, rP_2) \\ \underline{\mathcal{O}\mathsf{MSet}_3(m_B, rP_2)} & \textbf{if } \mathsf{Vf}(vk, m_B, \sigma) = 0 \textbf{ abort} \\ \textbf{if } m_B \in Q_2 \textbf{ abort} & \tau \leftarrow \mathsf{NAttest}(\widehat{sk}, m_B) \\ Q_2 := Q_2 \cup (m_B) & \textbf{return } (aP_3, \tau) \\ aP_3 \leftarrow \mathsf{MSet}_3(\overline{dk}, \widehat{vk}, m_B, rP_2) & \\ \textbf{return } (aP_3) & \end{array}$$

Figure 5: Definition of the experiment ExpM**.**

Seller Security. This property ensures that the adversary can only get the product if the seller is paid. Here, the adversary has access to an attestation oracle $\mathcal{O}\mathsf{NAttest}$, that models payments from the adversary to the mixer. The adversary generates all the mixer setup information, two messages m_M, m_B, as well as puzzle rP_1. Then, the challenger provides the adversary with rP_2, and the adversary produces aP_3. Finally, the challenger produces aP_4, which encrypts the product decryption key pdk and sends it to the adversary, who replies with a decryption key pdk'. We model two scenarios in which the adversary wins. The adversary wins if it did not use the attestation oracle on m_B (i.e. the buyer did not paid), but the decryption key pdk' is correct (condition b_0). This winning condition implies that the adversary managed to get the product without paying. Alternatively, if the adversary wins if it used the attestation oracle on m_B (i.e. the buyer did paid), but the seller fails to extract signature σ_M for the payment m_M(conditions b_1 and b_2). In this case the adversary was able to trick the seller with ill-formed rP_1 or aP_3 that prevents the seller to obtain σ_M.

Definition 4 (Seller Security)**.** *A O-UCP is said to offer seller security if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all $\lambda \in \mathbb{N}$ and for all PPT adversaries $\mathcal{A}$ it holds that* $\Pr[\mathsf{ExpS}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$, *where* ExpS *is defined in Fig. 6.*

Buyer Security. This property ensures that the adversary cannot prevent the buyer from getting the product if the buyer pays for it. We model the property by providing the adversary access to

ExpS	$\mathcal{O}$NAttest(σ, m, vk)
$Q := [\,]$	**if** $\mathsf{Vf}(vk, m, \sigma) = 0$ **abort**
$(\widehat{vk}, \widehat{sk}) \leftarrow \mathsf{NGen}(1^\lambda)$	$\tau \leftarrow \mathsf{NAttest_N}(\widehat{sk}, m)$
$(pek, pdk) \leftarrow \mathsf{createR}(1^\lambda)$	$Q[m] := \tau$
$(\overline{ek}, vk_M, m_B, m_M, rP_1) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{NAttest}}(\widehat{vk}, pek)$	**return** τ
$\{(rP_2, st_S), \perp\} \leftarrow \mathsf{SSet}_2(\overline{ek}, vk_M, m_M, rP_1)$	
if $\perp$ **abort**	
$aP_3 \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{NAttest}}(rP_2)$	
$\{aP_4, \perp\} \leftarrow \mathsf{SSet}_4(\widehat{vk}, m_B, pdk, aP_3, st_S)$	
if $\perp$ **abort**	
$pdk' \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{NAttest}}(aP_4)$	
if $Q[m_B] = \perp$	
$\quad b_0 := (pek, pdk') \in \mathsf{R}$	
else	
$\quad \tau := Q[m_B]$	
$\quad \sigma_M \leftarrow \mathsf{SSolve}(\tau, rP_1, aP_3, st_S)$	
$\quad b_1 := \mathsf{VfAttest}(\widehat{vk}, m_B, \tau) = 1$	
$\quad b_2 := \mathsf{Vf}(vk_M, m_M, \sigma_M) = 0$	
return $b_0 \vee (b_1 \wedge b_2)$	

Figure 6: Definition of the experiment ExpS.

an attestation oracle $\mathcal{O}$SigNAttest that models the signature generation from the buyer and the notary. The adversary can query oracle $\mathcal{O}$SigNAttest with messages of their choice. Then, the adversary outputs a tuple of buyer signature σ_B^*, product encryption key *pek*, transaction from buyer to mixer m_B and puzzle aP_4, which encrypts the product decryption key *pdk*. We model two scenarios. If the transaction m_B provided by the adversary was not queried in $\mathcal{O}$SigNAttest, but the forged signature σ_B^* is valid, the adversary wins (condition b_0). Here the adversary was successful in stealing money from buyer's account. Alternatively, the adversary wins if the oracle was queried, aP_4 verifies, but the challenger is unable to extract a valid *pek* from aP_4 (conditions b_1, b_2 and b_3). In this scenario the adversary tricks the buyer with an ill-formed aP_4 that does not contain the product decryption key *pdk*.

Definition 5 (Buyer Security). *A O-UCP is said to offer buyer security if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all* $\lambda \in \mathbb{N}$ *and for all PPT adversaries* $\mathcal{A}$ *it holds that* $\Pr[\mathsf{ExpB}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$*, where* ExpB *is defined in Fig. 7.*

Unlinkability. This property models the impossibility for the mixer to distinguish between two concurrent O-UCP executions. We model this property by completing two interactions with an adversarial mixer. These two interactions start sequentially requesting rP_1 from the adversary. Thereafter, the challenger runs algorithm SSet_2 for each received puzzle. At this point, the challenger flips a coin that defines the order in which puzzles rP_2 are sent to the adversary: i.e., in the same or the reversed order as puzzles rP_1 were received from the adversary. Once the full interaction is completed, if both or one of the operations fail, the challenger forwards $\perp$ to the

ExpB	$\mathcal{O}$SigNAttest(m)
$Q := [\,]$	$\sigma_B \leftarrow \mathsf{Sig}(sk_B, m)$
$(\widehat{vk}, \widehat{sk}) \leftarrow \mathsf{NGen}(1^\lambda)$	$\tau \leftarrow \mathsf{NAttest}(\widehat{sk}, m)$
$(vk_B, sk_B) \leftarrow \mathsf{KGen}(1^\lambda)$	$Q[m] := \tau$
$(\sigma_B^*, pek, m_B, aP_4)$	**return** (σ_B, τ)
$\qquad \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{SigNAttest}}(vk_B, \widehat{vk})$	
if $Q[m_B] = \perp$	
$\quad b_0 := (\mathsf{Vf}(vk_B, m_B, \sigma_B^*) = 1)$	
else	
$\quad \tau := Q[m_B]$	
$\quad pdk \leftarrow \mathsf{BSolve}(\tau, aP_4)$	
$\quad b_1 := \mathsf{BVfSet}(\widehat{vk}, m_B, pek, aP_4)$	
$\quad b_2 := \mathsf{VfAttest}(\widehat{vk}, m_B, \tau)$	
$\quad b_3 := (pek, pdk) \notin \mathsf{R}$	
return $b_0 \vee (b_1 \wedge b_2 \wedge b_3)$	

Figure 7: Definition of the experiment ExpB.

adversary, otherwise the resulting signatures are forwarded. The adversary wins if they can guess if the order of rP_2 was reversed with a probability better than random guess.

Definition 6 (Unlinkability). *A O-UCP is unlinkable if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all* $\lambda \in \mathbb{N}$ *and for all PPT adversaries* $\mathcal{A}$ *it holds that* $\Pr[\mathsf{ExpLink}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$*, where* ExpLink *is defined in Fig. 8.*

5 OUR CRYPTOGRAPHIC CONSTRUCTION

As described in Section 2, we remark that for the preparation of the product delivery, we follow the construction in zkCP and thus refer the reader to [12, 14, 27, 52] for a more complete description, security analysis and performance evaluation. In this section, we focus on describing the cryptographic construction, security analysis and performance evaluation of O-UCP.

Building Blocks. We require a digital signature scheme ($\widehat{\mathsf{DS}}$), an adaptor signature scheme (ADP), a linear only encryption scheme (LHE), a witness encryption based on signatures (WES), verifiable witness encryption for a relation (VWER), and a NIZK, with the properties described in Section 3. Regarding the NIZK, we require two different languages. Language $\mathcal{L}_1$ is used for MSet_1 while $\mathcal{L}_2$ is used for MSet_3.

$$\mathcal{L}_1 := \{(c, \overline{ek}, \mathsf{X}) \mid \exists\, \mathsf{w} \text{ s.t. } c \leftarrow \mathsf{LHE.Enc}(\overline{ek}, \mathsf{w}) \wedge (\mathsf{X}, \mathsf{w}) \in \mathsf{R}\}$$

$$\mathcal{L}_2 := \{(c, \widehat{vk}, m_B, \mathsf{X}) \mid \exists\, \mathsf{w} \text{ s.t. } c \leftarrow \mathsf{WES.Enc}((\widehat{vk}, m_B), \mathsf{w}) \wedge (\mathsf{X}, \mathsf{w}) \in \mathsf{R}\}$$

Overview. We present a high level overview of our construction, and the formal description is given in Fig. 9.
Bootstrapping. MGen and NGen are instantiated as the key generation algorithm of the LHE scheme and the signature scheme $\widehat{\mathsf{DS}}$ used in WES and VWER, respectively.

$$\begin{array}{l}
\underline{\mathsf{ExpLink}} \\
(vk_B^0, sk_B^0) \leftarrow \mathsf{KGen}(1^\lambda)\ ;\ (vk_B^1, sk_B^1) \leftarrow \mathsf{KGen}(1^\lambda) \\
(\overline{ek}, \widehat{vk}, vk_M^0, vk_M^1, rP_1^0, rP_1^1, (m_M^0, m_B^0), (m_M^1, m_B^1)) \leftarrow \mathcal{A}(vk_B^0, vk_B^1) \\
b \leftarrow \{0, 1\} \\
(pek^0, pdk^0) \leftarrow \mathsf{createR}(1^\lambda)\ ;\ (pek^1, pdk^1) \leftarrow \mathsf{createR}(1^\lambda) \\
\{(rP_2^0, st_S^0), \perp\} \leftarrow \mathsf{SSet}_2(\overline{ek}, vk_M^0, m_M^0, rP_1^0) \\
\{(rP_2^1, st_S^1), \perp\} \leftarrow \mathsf{SSet}_1(\overline{ek}, vk_M^1, m_M^1, rP_1^1) \\
(aP_3^0, aP_3^1) \leftarrow \mathcal{A}(rP_2^{0\oplus b}, rP_2^{1\oplus b}) \\
\{aP_4^0, \perp\} \leftarrow \mathsf{SSet}_4(\widehat{vk}, m_B^0, pdk^0, aP_3^0, st_S^{0\oplus b}) \\
\{aP_4^1, \perp\} \leftarrow \mathsf{SSet}_4(\widehat{vk}, m_B^1, pdk^1, aP_3^1, st_S^{1\oplus b}) \\
\sigma_B^0 \leftarrow \mathsf{Sig}(sk_B^0, m_B^0) \\
\sigma_B^1 \leftarrow \mathsf{Sig}(sk_B^1, m_B^1) \\
(\tau^0, \tau^1) \leftarrow \mathcal{A}(\sigma_B^0, \sigma_B^1) \\
\sigma_M^{0\oplus b} \leftarrow \mathsf{SSolve}(\tau^0, rP_1^{0\oplus b}, aP_3^0, st_S^{0\oplus b}) \\
\sigma_M^{1\oplus b} \leftarrow \mathsf{SSolve}(\tau^1, rP_1^{1\oplus b}, aP_3^1, st_S^{1\oplus b}) \\
\textbf{if } (\mathsf{Vf}(vk_M^0, m_M^0, \sigma_M^0) = 0) \lor (\mathsf{Vf}(vk_M^1, m_M^1, \sigma_M^1) = 0) \\
\quad \sigma_M^0 = \sigma_M^1 = \perp \\
b' \leftarrow \mathcal{A}(\sigma_M^0, \sigma_M^1) \\
\textbf{return } (b = b')
\end{array}$$

Figure 8: Definition of the experiment ExpLink. Note that in order to improve readability, we have not explicitly stated the conditions in which the challenger aborts: if any of the algorithms returns ⊥, the challenger aborts the game.

Puzzle-promise. MSet_1 starts with the generation of public statement/witness pair $(\mathsf{X}_1, \mathsf{w}_1) \in \mathsf{R}$. This is followed with the generation of a pre-signature $\widetilde{\sigma}$ of transaction m_M with statement X_1. The witness w_1 is encrypted using LHE resulting in ciphertext c_1 and a NIZK proof π_1 for language $\mathcal{L}_1$ is generated. Finally, the randomizable puzzle rP_1 is set to $(\widetilde{\sigma}, c_1, \pi_1, \mathsf{X}_1)$. Algorithm SSet_2 verifies that pre-signature $\widetilde{\sigma}$ and proof π_1 are valid. Thereafter, a public statement/witness pair $(\mathsf{X}_r, \mathsf{w}_r) \in \mathsf{R}$ is generated and used to randomize X_1 to X_2 and ciphertext c_1 into c_2, using the homomorphic properties of R and LHE. Finally, puzzle rP_2 is set to (c_2, X_2).

Puzzle-link. MSet_3 decrypts c_2 and re-encrypts the witness w_2 using WES resulting in ciphertext c_3, which can be decrypted with notary's attestation on transaction m_B. A NIZK proof π_3 for $\mathcal{L}_2$ is generated and the attestation puzzle aP_3 is set to (c_3, π_3). Algorithm SSet_4 first verifies that the proof π_3 is valid and then encrypts pdk using VWER resulting in ciphertext/proof tuple (c_4, π_4). c_4 can be decrypted with notary's attestation on transaction m_B. Finally, the attestation puzzle aP_4 is set to (c_4, π_4). Algorithm BVfSet is instantiated as the verification algorithm of the VWER scheme.

Attest-and-solve. NAttest and VfAttest are instantiated as the signature generation and verification of the signature scheme $\widehat{\mathsf{DS}}$, respectively. SSolve decrypts the WES ciphertext c_4 to get w_2 and then obtains w_1 by removing the randomization factor w_r from w_2. Finally, uses witness w_1 to adapt the pre-signature $\widetilde{\sigma}$ into signature σ_M. BSolve is instantiated as the decryption algorithm of VWER.

5.1 Security Analysis

In the following, we state our claims and provide intuitions on the security and privacy of our construction. We refer the reader to Appendix B for the full proofs.

Theorem 1 (Mixer Security). *Assume that NIZK is zero knowledge, that* WES *is IND-CPA, that adaptor signature is full extractable and the linear only encryption scheme is OMDL-*LHE*. Then, our construction offers mixer security according to Definition 3.*

In mixer security, the adversary attempts to generate a signature σ_M on a transaction m_M without notary's attestation τ on transaction m_B. Note that an attestation on m_B means that the mixer was paid. In a successful attack the adversary produces σ_M either from the randomizable puzzle rP_1 or the attestation puzzle aP_3 without an attestation. Puzzle rP_1 comprises a pre-signature $\widetilde{\sigma}$ on transaction m_B under the verification key vk_B and public statement X_1, an LHE ciphertext c_1 encrypting w_1, a NIZK π_1 for language $\mathcal{L}_1$, and X_1. Given that *OMDL*-LHE holds, the adversary cannot extract any information about w_1 from c_1. Likewise, given that a NIZK proof for $\mathcal{L}_1$ is zero knowledge, π_1 does not leak any information about w_1. Finally, given that the adaptor signature scheme satisfies the strong full extractability notion, the adversary cannot forge a valid signature σ_M using the pre-signature $\widetilde{\sigma}$ and public statement X_1. As regards to puzzle aP_3, it comprises a WES ciphertext c_3 encrypting w_2 and a NIZK π_3 for language $\mathcal{L}_2$. Given that WES is IND-CPA secure, the adversary cannot extract any information about w_2 from c_3. Similarly, given that a NIZK proof for $\mathcal{L}_2$ is zero knowledge, π_3 does not leak any information about w_2. Therefore, the adversary cannot produce σ_M from puzzles rP_1 and aP_3 without notary's attestation, hence mixer security holds.

Theorem 2 (Seller Security). *Assume the VWER is one way, NIZK is secure under soundness-knowledge and adaptor signature scheme is secure under adaptability. Then, our construction offers seller security according to Definition 4.*

In seller security, the adversary wants to obtain pdk without the seller getting a valid σ_M (i.e., without paying). In a successful attack the adversary: (i) extracts pdk from aP_4, containing a VWER ciphertext/proof pair (c_4, π_4); (ii) forges NIZK proofs π_1 for language $\mathcal{L}_1$ or π_3 for language $\mathcal{L}_2$, convincing the seller that rP_1 hides w_1 or aP_3 hides w_2, respectively; or (iii) produces a valid pre-signature $\widetilde{\sigma}$ on message m_M under the verification key vk_M and public statement X_1, such that it cannot be adapted to a valid signature σ_M using witness w_1. Concerning (i), given that VWER satisfies one-wayness, the adversary cannot extract pdk from (c_4, π_4) without notary's attestation τ. As regards to (ii), given that NIZK proofs for languages $\mathcal{L}_1$, $\mathcal{L}_2$ satisfy the soundness-knowledge notion, the adversary cannot forge either π_1 or π_3 without correctly hiding w_1, w_2 in puzzles rP_1, aP_3, respectively. Finally about (iii), given that the adaptor signature scheme satisfies the adaptability notion, a valid $\widetilde{\sigma}$ can always be adapted to a valid σ_M using w_1. Therefore, the adversary cannot obtain the product decryption key pdk without the seller receiving a payment, hence seller security holds.

$\underline{\mathsf{MGen}(1^\lambda)}$
$(\overline{ek}, \overline{dk}) \leftarrow \mathsf{LHE.KGen}(1^\lambda)$
$\textbf{return } (\overline{ek}, \overline{dk})$

$\underline{\mathsf{NGen}(1^\lambda)}$
$(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{KGen}}(1^\lambda)$
$\textbf{return } (\widehat{vk}, \widehat{sk})$

$\underline{\mathsf{NAttest}(\widehat{sk}, m_B)}$
$\tau \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{Sig}}(\widehat{sk}, m_B)$
$\textbf{return } \tau$

$\underline{\mathsf{VfAttest}(\widehat{vk}, m_B, \tau)}$
$\textbf{return } \widehat{\mathsf{DS}}.\widehat{\mathsf{Vf}}(\widehat{vk}, m_B, \tau)$

$\underline{\mathsf{BVfSet}(\widehat{vk}, m_B, pek, aP_4)}$
$(c_4, \pi_4) \leftharpoondown aP_4$
$\textbf{return } \mathsf{VWER.VfEncR}(c_4, \pi_4, (\widehat{vk}, m_B), pek)$

$\underline{\mathsf{MSet}_1(\overline{ek}, sk_M, m_M)}$
$(\mathsf{X}_1, \mathsf{w}_1) \leftarrow \mathsf{createR}(1^\lambda)$
$\widetilde{\sigma} \leftarrow \mathsf{ADP.PreSig}(sk_M, m_M, \mathsf{X}_1)$
$c_1 \leftarrow \mathsf{LHE.Enc}(\overline{ek}, \mathsf{w}_1)$
$\mathsf{y} := (c_M, \overline{ek}, \mathsf{X}_1)$
$\pi_1 \leftarrow \mathsf{NIZK.Prove}_{\mathcal{L}_1}(\mathsf{crs}, \mathsf{y}, \mathsf{w}_1)$
$rP_1 := (\widetilde{\sigma}, c_1, \pi_1, \mathsf{X}_1)$
$\textbf{return } rP_1$

$\underline{\mathsf{SSet}_4(\widehat{vk}, m_B, pdk, aP_3, st_S)}$
$(c_3, \pi_3) \leftharpoondown aP_3$
$(\mathsf{X}_2, \cdot, \cdot) \leftharpoondown st_S$
$\mathsf{y} := (c_3, \widehat{vk}, m_B, \mathsf{X}_2)$
$\textbf{if } \mathsf{NIZK.Vf}_{\mathcal{L}_2}(\mathsf{crs}, \mathsf{y}, \pi_3) = 0 \textbf{ abort}$
$(c_4, \pi_4) \leftarrow \mathsf{VWER.EncR}((\widehat{vk}, m_B), pdk)$
$aP_4 := (c_4, \pi_4)$
$\textbf{return } aP_4$

$\underline{\mathsf{SSet}_2(\overline{ek}, m_M, vk_M, rP_1)}$
$(\widetilde{\sigma}, c_1, \pi_1, \mathsf{X}_1) \leftharpoondown rP_1$
$\mathsf{y} := (c_1, \overline{ek}, \mathsf{X}_1)$
$a := \mathsf{NIZK.Vf}_{\mathcal{L}_1}(\mathsf{crs}, \mathsf{y}, \pi_1)$
$b := \mathsf{ADP.PreVf}(vk_M, m_M, \mathsf{X}_1, \widetilde{\sigma})$
$\textbf{if } (a = 0) \vee (b = 0) \textbf{ abort}$
$(\mathsf{X}_r, \mathsf{w}_r) \leftarrow \mathsf{createR}(1^\lambda)$
$\mathsf{X}_2 := \mathsf{X}_r \otimes \mathsf{X}_1$
$c_r \leftarrow \mathsf{LHE.Enc}(\overline{ek}, \mathsf{w}_r)$
$c_2 := c_1 \circ c_r$
$rP_2 := (c_2, \mathsf{X}_2)$
$st_S := (\mathsf{X}_2, \mathsf{X}_r, \mathsf{w}_r)$
$\textbf{return } (rP_2, st_S)$

$\underline{\mathsf{BSolve}(\tau, aP_4)}$
$(c_4, \pi_4) \leftharpoondown aP_4$
$pdk := \mathsf{VWER.DecR}(\tau, c_4, \pi_4)$
$\textbf{return } pdk$

$\underline{\mathsf{MSet}_3(\overline{dk}, \widehat{vk}, m_B, rP_2)}$
$(c_2, \mathsf{X}_2) \leftharpoondown rP_2$
$\mathsf{w}_2 \leftarrow \mathsf{LHE.Dec}(\overline{dk}, c_2)$
$c_3 \leftarrow \mathsf{WES.Enc}((\widehat{vk}, m_B), \mathsf{w}_2)$
$\mathsf{y} := (c_3, \widehat{vk}, m_B, \mathsf{X}_2)$
$\pi_3 \leftarrow \mathsf{NIZK.Prove}_{\mathcal{L}_2}(\mathsf{crs}, \mathsf{y}, \mathsf{w}_2)$
$aP_3 := (c_3, \pi_3)$
$\textbf{return } aP_3$

$\underline{\mathsf{SSolve}(\tau, rP_1, aP_3, st_S)}$
$(\widetilde{\sigma}, \cdot, \cdot, \cdot) \leftharpoondown rP_1$
$(c_3, \pi_3) \leftharpoondown aP_3$
$(\cdot, \cdot, \mathsf{w}_r) \leftharpoondown st_S$
$\mathsf{w}_2 \leftarrow \mathsf{WES.Dec}(\tau, c_3)$
$\mathsf{w}_1 = \mathsf{w}_2 - \mathsf{w}_r$
$\sigma_M := \mathsf{ADP.Adapt}(\widetilde{\sigma}, \mathsf{w}_1)$
$\textbf{return } \sigma_M$

Figure 9: Our cryptographic construction for O-UCP.

Theorem 3 (Buyer Security). *Assume the signature scheme is EUF-CMA and VWER provides VWER verifiability. Then, our construction offers buyer security according to Definition 5.*

In buyer security, the adversary attempts to obtain signature σ_B on m_B without the buyer getting the product decryption key *pdk*. In a successful attack the adversary: (i) forges a signature σ_B; or (ii) produces aP_4, comprising a VWER ciphertext/proof pair (c_4, π_4), such that either c_4 does not encrypt *pdk* or it cannot be decrypted using notary's attestation τ, yet π_4 convinces the buyer that c_4 is well-formed. Concerning (i), given that the DS is EUF-CMA, the adversary cannot produce such a forgery. As regards to (ii), given that VWER satisfies verifiability, the adversary cannot produce such a pair (c_4, π_4). Therefore, the adversary cannot obtain σ_B without the buyer getting *pdk*, hence buyer security holds.

Theorem 4 (Unlinkability). *Assume that* createR *samples at random from a uniform distribution. Then, our construction offers unlinkability according to Definition 6.*

In unlinkability, the adversary attempts to distinguish if buyer^0 interacted with seller^0 or with seller^1. However, the adversary only knows $\mathsf{w}_1^0, \mathsf{w}_1^1, \mathsf{w}_2^{0\oplus b}$ and $\mathsf{w}_2^{1\oplus b}$. In order to compute $\mathsf{w}_2^{0\oplus b}$ and $\mathsf{w}_2^{1\oplus b}$, the challenger sampled at random from a uniform distribution two values w_r^0 and w_r^1 and added them to $\mathsf{w}_1^0, \mathsf{w}_1^1$. Then, the challenger flipped a coin and provided the values to the adversary according to the random outcome. Note that w_2^0 and w_2^1 are indistinguishable from elements sampled at random from the same distribution as w_r^0 and w_r^1. Hence, in order to distinguish if buyer^0 interacted with seller^0 or with seller^1, the adversary would need to identify the order in which two elements were sampled at random from a uniform distribution. Since the adversary cannot do this with a probability greater than $1/2 + \mathsf{negl}(\lambda)$, unlinkability holds.

5.2 Performance Evaluation

We evaluate our implementation for O-UCP for puzzle-promise, puzzle-link and attest-and-solve.

Puzzle-promise. Algorithms MSet_1 and SSet_2 rely on the implementation of A2L [24]. As such, this step is implemented in C and relies on RELIC [2], GMP [32] and PARI [60]. We rely on the Schnorr ADP for curve secp256k1. The LHE is instantiated with HSM-CL [15, 16] encryption scheme for 128-bit security level.

Puzzle-link. MSet_3, SSet_4 and BVfSet are based on the implementation made available with the paper *Cryptographic Oracle-based Conditional Payments* [48, 49]. The oracle implementation is written in Rust with the crates Ristretto [42] and zkp [36]. In particular, MSet_3 runs the decryption algorithm of HSM-CL encryption scheme discussed in the previous paragraph (in C) to obtain w_2, followed by its re-encryption using the oracle encryption of [48, 49] (in Rust). SSet_4 runs the verification of the previous encryption and followed by the oracle encryption applied to *pdk*. Finally, BVfSet is implemented exactly as the verification algorithm of [48, 49].

Attest-and-solve. SSolve and BSolve use building blocks from [24, 48, 49] SSolve is implemented as the decryption algorithm of [48, 49]. BSolve runs the decryption algorithm of [48, 49], followed by the de-randomization and Adapt algorithms of [24].

NIZKs in MSet_1 and MSet_3 are instantiated with Σ protocols [18] made non interactive with the Fiat-Shamir heuristic [26]. We omit from the evaluation MGen, NGen and NAttest since their implementation is key and signature generation.

Table 2: Running time and message size of MixBuy.

Algorithm	Time (ms)
MSet_1	0.2 ± 0.1
SSet_2	500 ± 300
MSet_3	200 ± 100
SSet_4	20 ± 1
BVfSet	10 ± 1
SSolve	2 ± 1
BSolve	2 ± 1

Message	Size (kB)
rP_1	4.8
rP_2	2.2
aP_3	6.3
aP_4	6.3

Optimizations. MSet_1 and SSet_2 compute $(\mathsf{X}_1, \mathsf{w}_1)$ and $(\mathsf{X}_r, \mathsf{w}_r)$, respectively. The computation of these statement/witness pairs is pre-computed in advance. MSet_3 and SSet_4 require to run cut-and-choose to perform the proofs. The random values required by the cut-and-choose technique are pre-computed as in [49].

Testbed and Results. We conducted our experiments in an Ubuntu 22.04.3 virtual machine with 4GB of RAM and 2 processors. In our experiments, all four parties run on the same machine and communicate via localhost. We measured the average runtimes over 100 runs each, taking into consideration the optimizations mentioned above. We also measure the size of the messages exchanged between parties. Note that the messages considered are rP_1, rP_2, aP_3 and aP_4. Our findings (cf. Table 2) show that SSet_2 and MSet_3 take significantly longer than the rest of the algorithms. The reason for this is the use of the computationally heavy HSM-CL encryption: SSet_2 randomizes a HSM-CL ciphertext and MSet_3 decrypts it. The message sizes is relatively small, of a few kB, while the total execution time is under a second. The results of this proof of concept show that O-UCP is practical in commodity hardware.

6 DISCUSSION

Deploying MixBuy. In MixBuy, the notary can only attest transactions that are publicly accessible (i.e., on-chain transactions). We consider three environments to deploy MixBuy: (i) buyer and seller operate in the same cryptocurrency and mixer provides unlinkability; (ii) buyer and seller operate in different cryptocurrencies and the mixer also acts as an exchange platform; and (iii) the buyer operates on-chain, the seller off-chain, and the mixer runs a submarine swap [46] service. A submarine swap is an exchange between on-chain and off-chain liquidity. Moreover, MixBuy requires compatibility with shared addresses, either through programmability (e.g., HashTimeLock and multisignature [55]), or cryptographic protocols (e.g., two-party adaptor signatures [23] and timed verifiable signatures [61]), and hence is compatible with most blockchains.

Reducing Trust in the Notary. A single notary constitutes a single point of failure, hence buyer and seller might prefer to distribute the transaction attestation among a set of notaries. Hence, transaction m_B is attested only when threshold number of notaries have attested with their respective signing keys. MixBuy requires minimal changes for such setting: convert algorithms (i) MSet_3 and SSet_4 (cf. Fig. 9) to encrypt w_2 and pdk under a set of notaries' verification keys; and (ii) SSolve and BSolve (cf. Fig. 9) to decrypt c_3 and c_4 using a set of attestations τ, as described in [49].

Variable Amounts. In MixBuy all buyer transactions are of value α, while all seller transactions are of value β. Hence, unlikability is achieved for purchases products that have the same price. Nevertheless, as long as products have a price that is a multiple of β (e.g., $k \cdot \beta$), buyer and seller would need to run k times the puzzle-promise and puzzle-link steps. Instead of encrypting the decryption key of the product pdk in aP_4, the seller encrypts a k-share of pdk, such that all k of them are needed to reconstruct pdk. Once the buyer has all k-aP_4, the attest-and-solve step can start and the buyer sends the k payments to the mixer. The notary produces k attestations, allowing the buyer to get the product, and the seller to get the k payments from the mixer. However, setting up several instances of MixBuy for a purchase might be tedious for buyers. This problem is common to most centralized coin mixers [29, 35, 37, 59]. However, Accio [28] and Blindhub [57] achieve unlinkability for senders and receivers that are transferring different amounts. We see the extension of MixBuy to support purchases for products with different prices as interesting future work.

Griefing Attack. The mixer might be subject to griefing attacks [59], as it happens with centralized coin mixers [28, 29, 35, 37, 57, 59]. For MixBuy the attack results in the seller requesting rP_1, which makes the mixer lock funds in a shared account with the seller. If the attacker can lock the mixer's coins without a cost, a set of malicious buyers and sellers might collude to lock all mixer funds in shared addresses, resulting in a denial of service. In order to mitigate this attack, the adversary should only be able to lock mixer's coins at an equivalent cost. Note that in Fig. 1, the buyer needs to lock α coins in the shared address with the mixer before the mixer locks β coins with the seller. For simplicity, we omitted that after the buyer locks α coins, the mixer provides a blind signature, which the buyer forwards to the seller. Then, the seller presents the blind signature to the mixer. If it is valid and has not been used before, the mixer locks β coins. This approach is inspired by [59].

Breaking Unlinkability. The mixer might attempt to break the unlinkability by boycotting some of the transactions during the puzzle-promise step such that only one buyer receives rP_2 (e.g. by providing only one valid rP_1). If only one buyer has rP_2, when the buyer finalizes the purchase, the mixer can link the only buyer with the only seller. This attack affects most centralized coin mixing services [28, 29, 35, 37, 57, 59]. However, the business model of the mixer is to route a payment from a sender to a receiver in exchange for a fee. Therefore, mixer's cost for breaking unlinkability is twofold: (i) losing the fees for all but one of the payments; and (ii) losing credibility as an mixer, hence missing potential future users.

7 CONCLUSIONS

In this work, we presented MixBuy, a system that realizes unlinkable contingent payments (UCP). MixBuy relies on *oracle-based unlinkable contingent payment* (O-UCP), a novel four-party cryptographic protocol where the mixer pays the seller and the seller provides the buyer with the product only if a semi-trusted notary attests that the buyer has paid the mixer. We presented a provably secure and efficient cryptographic construction for O-UCP, and a proof of concept that demonstrates its practicality.

REFERENCES

[1] M. Adachi, A. Born, I. Gschossmann, and A. van der Kraaij, "The expanding functions and uses of stablecoins," ECB website, 2021, https://www.ecb.europa.eu/pub/financial-stability/fsr/focus/2021/html/ecb.fsrbox202111_04~45293c08fc.en.html.

[2] D. F. Aranha, "Relic is an efficient library for cryptography," 2020, accessed on 10.01.2024. [Online]. Available: https://dfaranha.github.io/project/relic/

[3] AT&T, "At&t now accepts bitpay," AT&T webpage, 2019, https://about.att.com/story/2019/att_bitpay.html.

[4] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostáková, M. Maffei, P. Moreno-Sanchez, and S. Riahi, "Generalized channels from limited blockchain scripts and adaptor signatures," in *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part II 27*. Springer, 2021, pp. 635–664.

[5] B. Bauer, G. Fuchsbauer, and A. Plouviez, "The one-more discrete logarithm assumption in the generic group model," in *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part IV 27*. Springer, 2021, pp. 587–617.

[6] Bellare, Namprempre, Pointcheval, and Semanko, "The one-more-rsa-inversion problems and the security of chaum's blind signature scheme," *Journal of Cryptology*, vol. 16, pp. 185–215, 2003.

[7] M. Bellare and O. Goldreich, "On defining proofs of knowledge," in *Advances in Cryptology — CRYPTO' 92*, E. F. Brickell, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 390–420.

[8] E. Ben Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE Symposium on Security and Privacy*, 2014, pp. 459–474.

[9] I. Bentov, Y. Ji, F. Zhang, L. Breidenbach, P. Daian, and A. Juels, "Tesseract: Real-time cryptocurrency exchange using trusted hardware," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1521–1538. [Online]. Available: https://doi.org/10.1145/3319535.3363221

[10] I. Bentov and R. Kumaresan, "How to use bitcoin to design fair protocols," in *Advances in Cryptology – CRYPTO 2014*, J. A. Garay and R. Gennaro, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 421–439.

[11] Bitpay, "Buy from microsoft," Bitpay merchan directory, 2023, https://bitpay.com/directory/microsoft/.

[12] S. Bursuc and S. Mauw, "Contingent payments from two-party signing and verification for abelian groups," in *2022 2022 IEEE 35th Computer Security Foundations Symposium (CSF) (CSF)*. Los Alamitos, CA, USA: IEEE Computer Society, aug 2022, pp. 195–210. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CSF54842.2022.9919654

[13] J. Camenisch and I. Damgård, "Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes," in *Advances in Cryptology — ASIACRYPT 2000*, T. Okamoto, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 331–345.

[14] M. Campanelli, R. Gennaro, S. Goldfeder, and L. Nizzardo, "Zero-knowledge contingent payments revisited: Attacks and payments for services," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 229–243. [Online]. Available: https://doi.org/10.1145/3133956.3134060

[15] G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker, "Two-party ecdsa from hash proof systems and efficient instantiations," in *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part III 39*. Springer, 2019, pp. 191–221.

[16] G. Castagnos and F. Laguillaumie, "Linearly homomorphic encryption from," in *Cryptographers' Track at the RSA Conference*. Springer, 2015, pp. 487–505.

[17] W. Dai, T. Okamoto, and G. Yamamoto, "Stronger security and generic constructions for adaptor signatures," in *Progress in Cryptology – INDOCRYPT 2022*, T. Isobe and S. Sarkar, Eds. Cham: Springer International Publishing, 2022, pp. 52–77.

[18] I. Damgård, "On σ-protocols," *Lecture Notes, University of Aarhus, Department for Computer Science*, p. 84, 2002.

[19] A. De Santis, S. Micali, and G. Persiano, "Non-interactive zero-knowledge proof systems," in *Advances in Cryptology—CRYPTO'87: Proceedings 7*. Springer, 1988, pp. 52–72.

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

[21] S. Dziembowski, L. Eckey, and S. Faust, "Fairswap: How to fairly exchange digital goods," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 967–984. [Online]. Available: https://doi.org/10.1145/3243734.3243857

[22] ECB Crypto-Assets Task Force, "Stablecoins: Implications for monetary policy, financial stability, market infrastructure and payments, and banking supervision in the euro area," European Central Bank Occasional Paper Series, 2020, https://www.ecb.europa.eu/pub/pdf/scpops/ecb.op247~fe3df92991.en.pdf.

[23] A. Erwig, S. Faust, K. Hostáková, M. Maitra, and S. Riahi, "Two-party adaptor signatures from identification schemes," in *IACR International Conference on Public-Key Cryptography*. Springer, 2021, pp. 451–480.

[24] Etairi, "Etairi/a2l: Implementation of anonymous atomic locks described in https://eprint.iacr.org/2019/589," 2022, https://github.com/etairi/A2L.

[25] European Central Bank and Bank of Japan, "Synchronized cross-border payments," STELLA project, 2019, accessed on 10.01.2024. [Online]. Available: https://www.ecb.europa.eu/paym/intro/publications/pdf/ecb.miptopical190604.en.pdf

[26] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Conference on the theory and application of cryptographic techniques*. Springer, 1986, pp. 186–194.

[27] G. Fuchsbauer, "Wi is not enough: Zero-knowledge contingent (service) payments revisited," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 49–62. [Online]. Available: https://doi.org/10.1145/3319535.3354234

[28] Z. Ge, J. Gu, C. Wang, Y. Long, X. Xu, and D. Gu, "Accio: Variable-amount, optimized-unlinkable and nizk-free off-chain payments via hubs," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 1541–1555.

[29] N. Glaeser, M. Maffei, G. Malavolta, P. Moreno-Sanchez, E. Tairi, and S. A. K. Thyagarajan, "Foundations of coin mixing services," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1259–1273. [Online]. Available: https://doi.org/10.1145/3548606.3560637

[30] S. Goldwasser and S. Micali, "Probabilistic encryption," *Journal of Computer and System Sciences*, vol. 28, no. 2, pp. 270–299, 1984. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0022000084900709

[31] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM J. Comput.*, vol. 17, no. 2, p. 281–308, apr 1988. [Online]. Available: https://doi.org/10.1137/0217017

[32] T. Granlund, "the gmp development team: Gnu mp. the gnu multiple precision arithmetic library." 2019, accessed on 10.01.2024. [Online]. Available: https://gmplib.org/

[33] M. Green and I. Miers, "Bolt: Anonymous payment channels for decentralized currencies," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 473–489. [Online]. Available: https://doi.org/10.1145/3133956.3134093

[34] J. Groth, "Rerandomizable and replayable adaptive chosen ciphertext attack secure cryptosystems," in *Theory of Cryptography Conference*. Springer, 2004, pp. 152–170.

[35] L. Hanzlik, J. Loss, S. A. Thyagarajan, and B. Wagner, "Sweep-uc: Swapping coins privately," *Cryptology ePrint Archive*, 2022.

[36] hdevalence, "zkp 0.8.0: a toolkit for schnorr proofs," 2020, accessed on 10.01.2024. [Online]. Available: https://docs.rs/zkp/latest/zkp/

[37] E. Heilman, L. Alshenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg, "Tumblebit: An untrusted bitcoin-compatible anonymous payment hub," in *Network and distributed system security symposium*, 2017.

[38] M. Herlihy, "Atomic cross-chain swaps," *CoRR*, vol. abs/1801.09515, 2018. [Online]. Available: http://arxiv.org/abs/1801.09515

[39] ——, "Atomic cross-chain swaps," in *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, ser. PODC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 245–254. [Online]. Available: https://doi.org/10.1145/3212734.3212736

[40] M. Herlihy, B. Liskov, and L. Shrira, "Cross-chain deals and adversarial commerce," *CoRR*, vol. abs/1905.09743, 2019. [Online]. Available: http://arxiv.org/abs/1905.09743

[41] S. Imoto, Y. Sudo, H. Kakugawa, and T. Masuzawa, "Atomic cross-chain swaps with improved space, time and local time complexities," *Information and Computation*, vol. 292, p. 105039, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0890540123000408

[42] isislovecruf and github:dalek-cryptography:curve-maintainers, "curve25519 dalek ristretto 4.0.0," 2022, accessed on 10.01.2024. [Online]. Available: https://crates.io/crates/curve25519-dalek

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

[44] KPMG, "Frontiers in finance: Innovating through platforms and ecosystems," KPMG, 2022, https://assets.kpmg.com/content/dam/kpmg/xx/pdf/2022/05/frontiers-in-finance.pdf.

[45] Lightning Labs, "Lsat: Lightning service authentication token," Lightning Labs, 2021, https://lsat.tech/. [Online]. Available: https://lsat.tech/

[46] ——, "Understanding submarine swaps," Lightning Labs, 2021, https://docs.lightning.engineering/the-lightning-network/multihop-payments/understanding-submarine-swaps. [Online]. Available: https://docs.lightning.engineering/the-lightning-network/multihop-

payments/understanding-submarine-swaps
[47] B. Liu, P. Szalachowski, and J. Zhou, "A first look into defi oracles," in *IEEE International Conference on Decentralized Applications and Infrastructures, DAPPS 2021, Online Event, August 23-26, 2021*. IEEE, 2021, pp. 39–48. [Online]. Available: https://doi.org/10.1109/DAPPS52256.2021.00010
[48] LLFourn, "Llfourn/dlc-verifiable-encryption-non-pairing," 2023, https://github.com/LLFourn/dlc-verifiable-encryption-non-pairing.
[49] V. Madathil, S. A. K. Thyagarajan, D. Vasilopoulos, L. Fournier, G. Malavolta, and P. Moreno-Sanchez, "Cryptographic oracle-based conditional payments," in *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society, 2023. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/cryptographic-oracle-based-conditional-payments/
[50] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and S. Ravi, "Concurrency and privacy with payment-channel networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 455–471. [Online]. Available: https://doi.org/10.1145/3133956.3134096
[51] G. Malavolta, P. A. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, "Anonymous multi-hop locks for blockchain scalability and interoperability," *Proceedings 2019 Network and Distributed System Security Symposium*, 2019.
[52] K. Nguyen, M. Ambrona, and M. Abe, "Wi is almost enough: Contingent payment all over again," ser. CCS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 641–656. [Online]. Available: https://doi.org/10.1145/3372297.3417888
[53] S. Noether, A. Mackenzie *et al.*, "Ring confidential transactions," *Ledger*, vol. 1, pp. 1–18, 2016.
[54] O. Osuntokun, "Lsat: Your ticket aboard the internet's money rails," The lightning Conference, 2019, https://www.youtube.com/watch?v=qfFESA961mk. [Online]. Available: https://www.youtube.com/watch?v=qfFESA961mk
[55] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," bitconlightning.com, 2016, accessed on 06.05.2022. [Online]. Available: https://www.bitcoinlightning.com/wp-content/uploads/2018/03/lightning-network-paper.pdf
[56] PriceWaterhouseCoopers, "El salvador's law: a meaningful test for bitcoin," PWC webpage, 2021, https://www.pwc.com/gx/en/financial-services/pdf/el-salvadors-law-a-meaningful-test-for-bitcoin.pdf.
[57] X. Qin, S. Pan, A. Mirzaei, Z. Sui, O. Ersoy, A. Sakzad, M. Esgin, J. K. Liu, J. Yu, and T. Yuen, "Blindhub: Bitcoin-compatible privacy-preserving payment channel hubs supporting variable amounts," in *2023 2023 IEEE Symposium on Security and Privacy (SP) (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2023, pp. 2020–2038. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP46215.2023.00116
[58] Shopify, "Shopify help center: cryptocurrencies," Shopify webpage, 2023, https://help.shopify.com/en/manual/payments/additional-payment-methods/cryptocurrency.
[59] E. Tairi, P. Moreno-Sanchez, and M. Maffei, "A 2 l: Anonymous atomic locks for scalability in payment channel hubs," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1834–1851.
[60] The PARI Group, Univ. Bordeaux, "Pari/gp version 2.12.0," 2019, accessed on 10.01.2024. [Online]. Available: https://pari.math.u-bordeaux.fr/
[61] S. A. K. Thyagarajan, A. Bhat, G. Malavolta, N. Döttling, A. Kate, and D. Schröder, "Verifiable timed signatures made practical," in *CCS*, 2020, pp. 1733–1750.
[62] S. A. K. Thyagarajan, G. Malavolta, and P. Moreno-Sanchez, "Universal atomic swaps: Secure exchange of coins across all blockchains," in *IEEE Symposium on Security and Privacy, SP*. IEEE, 2022, pp. 1299–1316. [Online]. Available: https://doi.org/10.1109/SP46214.2022.9833731
[63] S. M. Werner, D. Perez, L. Gudgeon, A. Klages-Mundt, D. Harz, and W. J. Knottenbelt, "Sok: Decentralized finance (defi)," *CoRR*, vol. abs/2101.08778, 2021. [Online]. Available: https://arxiv.org/abs/2101.08778
[64] B. Wiki, "Multi-signature in bitcoin," https://en.bitcoin.it/wiki/Multi-signature.
[65] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. ACM, 2016, pp. 270–282. [Online]. Available: https://doi.org/10.1145/2976749.2978326

A EXTENDED PRELIMINARIES

To facilitate the reader the games to which we make our reductions in Appendix B, we restate the games required by the security properties of EUF-CMA for digital signatures [31], strong full extractability and adaptability for adaptor signatures [17], the IND-CPA [49] security property of witness encryption based on signatures, correctness, one-wayness and verifiability for VWER and the zero-knowledge [19] and knowledge soundness [7] for NIZK. We also define the additional security property for the linear-only encryption scheme, OMDL-LHE. We also restate the one more discrete logarithm assumption, needed to prove OMDL-LHE

A.1 Digital Signature

Definition 7 (EUF-CMA). *An digital signature scheme is said to offer EUF-CMA if for all $\lambda \in \mathbb{N}$, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all PPT adversaries $\mathcal{A}$, it holds that* $\Pr[\mathsf{EUF-CMA}(\lambda) = 1] \leq \mathsf{negl}$*, where* $\mathsf{EUF-CMA}$ *is defined in Fig. 10.*

A.2 Adaptor Signatures

Regarding (strong) full extractability, note that we have added condition b_2, which does not exist in [17]. The reason for this is that we consider an attack that the adversary is able to forge a signature without querying the presignature oracle.

Definition 8 ((Strong) Full Extractability). *An adaptor signature scheme is said to offer (strong) full extractability if for all $\lambda \in \mathbb{N}$, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all PPT adversaries $\mathcal{A}$, it holds that* $\Pr[(\mathsf{s})\mathsf{fext}(\lambda) = 1] \leq \mathsf{negl}$*, where* $(\mathsf{s})\mathsf{fext}$ *is defined in Fig. 11.*

Definition 9 (Pre-Signature Adaptability). *An adaptor signature scheme is said to offer pre-signature adaptability if for all $\lambda \in \mathbb{N}$, any message $m \in \{0,1\}^*$, any statement and witness pair $(\mathsf{X}, \mathsf{w}) \in \mathsf{R}$, any public key such that $vk \in \mathsf{SUPP}(\mathsf{KGen})$ and any pre-signature $\widetilde{\sigma} \in \{0,1\}^*$ that satisfies $\mathsf{PreVf}(vk, m, \mathsf{X}, \widetilde{\sigma})$, we have that*
$$\Pr[\mathsf{Vf}(vk, m, \mathsf{Adapt}(\widetilde{\sigma}, \mathsf{w})) = 1] = 1.$$

A.3 Witness Encryption based on Signatures

Definition 10 (IND-CPA). *A witness encryption based on signatures scheme is said to offer IND-CPA if for all $\lambda \in \mathbb{N}$, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all PPT adversaries $\mathcal{A}$, it holds that* $\Pr[\mathsf{IND\text{-}CPA}(\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}$*, where* $\mathsf{IND\text{-}CPA}$ *is defined in Fig. 12.*

$\mathsf{EUF-CMA}$	$\mathsf{Sig}\mathcal{O}(m)$
$Q := \emptyset$	$\sigma \leftarrow \mathsf{Sig}(sk, m)$
$(vk, sk) \leftarrow \mathsf{KGen}(1^\lambda)$	$Q := Q \cup m$
$(m, \sigma) \leftarrow \mathcal{A}^{\mathsf{Sig}\mathcal{O}}(vk)$	**return** σ
return $\mathsf{Vf}(vk, m, \sigma) \wedge m \notin Q$	

Figure 10: Experiment for *EUF-CMA*.

$\mathsf{fext}(\lambda), \mathsf{sfext}(\lambda)$

$Q_{fext} := \emptyset \,;\ Q_{sfext} := \emptyset \,;\ Q_{st} := \emptyset$

$Q_{pS} := []$

$(vk, sk) \leftarrow \mathsf{KGen}(1^{\lambda})$

$(m^*, \sigma^*) \leftarrow \mathcal{A}^{O\mathsf{Sig}, O\mathsf{PreSig}, O\mathsf{newX}}(vk)$

$\mathsf{fext}:\ \textbf{assert}\ m^* \notin Q_{fext}$

$\mathsf{sfext}:\ \textbf{assert}\ (m^*, \sigma^*) \notin Q_{sfext}$

$b_0 := \mathsf{Vf}(vk, m^*, \sigma^*)$

$b_1 := \forall (\mathsf{X}, \widetilde{\sigma}) \in Q_{pS}[m^*]\ s.t.\ \mathsf{X} \notin Q_{st}$

$\quad (\mathsf{X}, \mathsf{Extract}(\sigma^*, \widetilde{\sigma}, \mathsf{X}) \notin \mathsf{R}$

$b_2 := Q_{pS}[m^*] = \bot$

$\textbf{return}\ b_0 \wedge (b_1 \vee b_2)$

$O\mathsf{Sig}(m)$

$\sigma \leftarrow \mathsf{Sig}(sk, m)$

$Q_{fext} := Q_{fext} \cup \{m\}$

$Q_{sfext} := Q_{sfext} \cup \{(m, \sigma)\}$

$\textbf{return}\ \sigma$

$O\mathsf{PreSig}(m, \mathsf{X})$

$\widetilde{\sigma} \leftarrow \mathsf{PreSig}(sk, m, \mathsf{X})$

$Q_{pS}[m] := Q_{pS}[m] \cup \{(\mathsf{X}, \widetilde{\sigma})\}$

$\textbf{return}\ \widetilde{\sigma}$

$O\mathsf{newX}()$

$(\mathsf{X}, \mathsf{w}) \leftarrow \mathsf{createR}(1^{\lambda})$

$Q_{st} := Q_{st} \cup \{\mathsf{X}\}$

$\textbf{return}\ \mathsf{X}$

Figure 11: Experiments for full extractability (fext(λ)) and strong full extractability (sfext(λ))

$\mathsf{IND\text{-}CPA}(\lambda)$

$Q := \emptyset;$

$(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{KGen}}(1^{\lambda})$

$(\widehat{m}^*, m_0, m_1) \leftarrow \mathcal{A}^{O\widehat{\mathsf{Sig}}}(vk)$

$b \xleftarrow{\$} \{0, 1\}$

$c_b \leftarrow \mathsf{Enc}((\widehat{vk}, \widehat{m}^*), m_b)$

$b' \leftarrow \mathcal{A}^{O\widehat{\mathsf{Sig}}}(c_b)$

$b_0 := (b = b')$

$b_1 := \widehat{m}^* \notin Q$

$\textbf{return}\ b_0 \wedge b_1$

$O\widehat{\mathsf{Sig}}(m)$

$\widehat{\sigma} \leftarrow \widehat{\mathsf{Sig}}(\widehat{sk}, m)$

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

$\textbf{return}\ \widehat{\sigma}$

Figure 12: Experiment IND-CPA for witness encryption based on signatures.

A.4 Verifiable Witness Encryption for a Relation

Here we present a variation of the primitive verifiable witness encryption based on threshold signatures (VWETS) introduced in [49]. We perform the following simplifications with respect to the original primitive: (i) the encrypted value is not a signature, but the logarithm of an element in a group where the discrete logarithm problem is computationally hard; and (ii) we consider a single oracle. This is done in order to facilitate the description of UCP. In section Section 6 we discuss how to decentralize the trust in the notary of UCP.

$\mathsf{ExpOW}_{\mathcal{A}}(\lambda)$

$Q := \emptyset$

$(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{KGen}}(1^{\lambda})$

$(\mathsf{X}, \mathsf{w}) \leftarrow \mathsf{createR}(1^{\lambda})$

$\mathsf{w}^* \leftarrow \mathcal{A}^{O\widehat{\mathsf{Sig}}, O\mathsf{EncR}}(\widehat{vk}, \mathsf{X})$

$b := (\mathsf{X}, \mathsf{w}^*) \in \mathsf{R}$

$\textbf{return}\ b$

$O\widehat{\mathsf{Sig}}(\widehat{m})$

$\textbf{if}\ \widehat{m} \in Q\ \textbf{abort}$

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

$\widehat{\sigma} \leftarrow \widehat{\mathsf{Sig}}(\widehat{sk}, \widehat{m})$

$\textbf{return}$

$O\mathsf{EncR}(\widehat{m})$

$\textbf{if}\ \widehat{m} \in Q\ \textbf{abort}$

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

$(c, \pi) \leftarrow \mathsf{EncR}((\widehat{vk}, \widehat{m}), \mathsf{w})$

$\textbf{return}\ (c, \pi)$

$\mathsf{ExpVer}_{\mathcal{A}}(\lambda)$

$(\widehat{m}, \widehat{vk}, \widehat{\sigma}, c, \pi, \mathsf{X}) \leftarrow \mathcal{A}(1^{\lambda})$

$\mathsf{w}^* \leftarrow \mathsf{DecR}(\widehat{\sigma}, (c, \pi))$

$b_0 := \mathsf{VfEncR}((c, \pi), (\widehat{vk}, \widehat{m}), \mathsf{X}) = 1$

$b_1 := \widehat{\mathsf{Vf}}(\widehat{vk}, \widehat{m}, \widehat{\sigma}) = 1$

$b_2 := (\mathsf{X}, \mathsf{w}^*) \notin \mathsf{R}$

$\textbf{return}\ b_0 \wedge b_1 \wedge b_2$

Figure 13: Definition of the experiments ExpOW **and** ExpVer**.**

Definition 11 (Verifiable Witness Encryption for a Relation (VWER))**.** *A Verifiable witness encryption for a relation is defined w.r.t. a relation* R *and a signature scheme,* $\widehat{\mathsf{DS}} = (\widehat{\mathsf{KGen}}, \widehat{\mathsf{Sig}}, \widehat{\mathsf{Vf}})$*. It comprises three algorithms (*EncR, VfEncR *and* DecR*), defined bellow:*

- $\underline{(c, \pi) \leftarrow \mathsf{EncR}((\widehat{vk}, \widehat{m}), \mathsf{w})}$: *PPT algorithm* EncR *gets as input a tuple, comprising a verification key* $\widehat{vk}$ *and a message* $\widehat{m}$*, and a witness* w*, and outputs the ciphertext tuple, containing ciphertext and a proof* (c, π)*.*
- $\underline{1/0 \leftarrow \mathsf{VfEncR}((c, \pi), (\widehat{vk}, \widehat{m}), \mathsf{X})}$: *DPT algorithm* VfEncR *gets as input a tuple comprising a ciphertext* c *and a proof* π*, a tuple comprising a public key* $\widehat{vk}$ *and a message* $\widehat{m}$*, and a public statement* X*, and outputs* 1 *if it is a valid ciphertext, otherwise it outputs* 0*.*
- $\underline{\mathsf{w}' \leftarrow \mathsf{DecR}(\widehat{\sigma}, (c, \pi))}$: *DPT algorithm* DecR *gets as input a signature* $\widehat{\sigma}$ *and tuple comprising a ciphertext* c *and a proof* π*, and outputs a witness* w'*.*

Definition 12 (VWER Correctness)**.** *A VWER is said to be* correct *if for all* $\lambda \in \mathbb{N}$*, all keys* $\widehat{vk} \in \mathsf{SUPP}(\widehat{\mathsf{KGen}}(1^{\lambda}))$*, all messages* $\widehat{m}$*, all statement and witness* $(\mathsf{X}, \mathsf{w}) \in \mathsf{R}$*, the following holds:*

(1) $\Pr[\mathsf{VfEncR}(\mathsf{EncR}((\widehat{vk}, \widehat{m}), \mathsf{w}), (\widehat{vk}, \widehat{m}), \mathsf{X}) = 1] = 1$

(2) If $\widehat{\mathsf{Vf}}(\widehat{vk}, \widehat{m}, \widehat{\sigma}) = 1$*, then:*

$$\Pr[(\mathsf{X}, \mathsf{DecR}(\widehat{\sigma}, \mathsf{EncR}((\widehat{vk}, \widehat{m}), \mathsf{w}))) \in \mathsf{R}] = 1$$

Definition 13 (VWER One Wayness)**.** *A VWER is said to be one way if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all* $\lambda \in \mathbb{N}$ *and* *all PPT adversaries $\mathcal{A}$ it holds that* $\Pr[\mathsf{ExpOW}_{\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$*, where* $\mathsf{ExpOW}_{\mathcal{A}}$ *is defined in Fig. 13.*

Definition 14 (VWER Verifiability). *A VWER is said to be verifiable if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all* $\lambda \in \mathbb{N}$ *and all PPT adversaries* $\mathcal{A}$ *it holds that* $\Pr[\mathsf{ExpVer}_{\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$*, where* $\mathsf{ExpVer}_{\mathcal{A}}$ *is defined in Fig. 13.*

A.5 NIZK

Definition 15 (Zero Knowledge). *A non interactive zero knowledge proof is said to offer zero knowledge if for all $\lambda \in \mathbb{N}$, there exists a negligible function* $\mathsf{negl}(\lambda)$ *and a PPT simulator $\mathcal{S}$ such that for all PPT adversaries $\mathcal{A}$, it holds that*

$$\Pr\left[b = b^* \middle| \begin{array}{c} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{SetUp}(1^\lambda) \\ (\mathsf{X}, \mathsf{w}) \leftarrow \mathcal{A}(\mathsf{crs}) \\ b \leftarrow_\$ \{0,1\} \\ \textbf{if } b = 0 : \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{X}, \mathsf{w}) \\ \textbf{if } b = 1 : \pi \leftarrow \mathcal{S}(\mathsf{crs}, \mathsf{X}, \mathsf{td}) \\ b^* \leftarrow \mathcal{A}(\mathsf{X}, \pi) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}$$

Definition 16 (Knowledge Soundness). *A non interactive zero knowledge proof is said to offer knowledge soundness if for all $\lambda \in \mathbb{N}$, there exists a negligible function* $\mathsf{negl}(\lambda)$ *and a extractor $\mathcal{E}$ such that for all PPT adversaries $\mathcal{A}$, it holds that*

$$\Pr\left[\begin{array}{c} b_0 := 1 \\ \wedge\, b_1 := 1 \end{array} \middle| \begin{array}{c} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{SetUp}(1^\lambda) \\ (\mathsf{X}, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) \\ b_0 := \mathsf{Vf}(\mathsf{crs}, \mathsf{X}, \pi) = 1 \\ b_1 := (\mathsf{X}, \mathcal{E}(\mathsf{td}, \mathsf{X}, \pi)) \notin \mathsf{R} \end{array} \right] \leq \mathsf{negl}$$

A.6 Linear-Only Homomorphic Encryption Scheme.

We define an additional property called OMDL-LHE. Here, the challenger generates an encryption/decryption key pair and a list of $k+1$ (statement, witness) pairs. Then, encrypts all witnesses with the encryption key and provides the encryption key, the statements and ciphertexts to the adversary. The adversary has access to a decryption oracle. If the adversary is able to return more valid witnesses than queries to the decryption oracle, wins the game. As stated in Lemma 1 a linear only encryption achieves OMDL-LHE if OMDL holds. We formally prove Lemma 1 in Appendix D. We introduce Lemma 1 because it becomes useful to prove the security of our proposed construction in Section 5.

Definition 17 (OMDL-LHE). *An encryption scheme is said to offer OMDL-*LHE *security if for all $\lambda \in \mathbb{N}$, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all PPT adversaries $\mathcal{A}$, it holds that* $\Pr[\mathit{OMDL}\text{-}\mathsf{LHE}(\lambda) = 1] \leq \mathsf{negl}$*, where the experiment OMDL-*LHE *is defined in Fig. 14.*

One-More Discrete Logarithm Assumption. We recall the one-more discrete logarithm (OMDL) [5, 6] assumption.

Definition 18 (One-More Discrete Logarithm (OMDL) Assumption). *Let $\mathbb{G}$ be a uniformly sampled cyclic group of prime order p and let g be a random generator of $\mathbb{G}$. The OMDL assumption states that for all $\lambda \in \mathbb{N}$, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all PPT adversaries $\mathcal{A}$ making at most q queries to $\mathcal{O}$*DL*, it holds*

$\mathsf{OMDL\text{-}LHE}_{\mathcal{A}}(\lambda)$	$\mathcal{O}\mathsf{Dec}(c, \mathsf{X})$
$\mathsf{q} := 0$	$\mathsf{q} := \mathsf{q} + 1$
$(\overline{ek}, \overline{dk}) \leftarrow \mathsf{LHE.KGen}(1^\lambda)$	$\mathsf{w} := \mathsf{LHE.Dec}(\overline{dk}, c)$
$\{(\mathsf{X}_i, \mathsf{w}_i)\}_{i \in [0,k]} \leftarrow \mathsf{createR}^{(k+1)}(1^\lambda)$	**if** $(\mathsf{X}, \mathsf{w}) \in \mathsf{R}$
for $i \in [0, k]$:	**return** w
$c_i \leftarrow \mathsf{LHE.Enc}(\overline{ek}, \mathsf{w}_i)$	**else return** $\perp$
$\{\mathsf{w}'_i\}_{i \in [0,k]} \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{Dec}}(\overline{ek}, \{(\mathsf{X}_i, c_i)\}_{i \in [0,k]})$	
$b_0 := \forall i,\ \mathsf{w}'_i = \mathsf{w}_i$	
$b_1 := \mathsf{q} < k$	
return $b_0 \wedge b_1$	

Figure 14: Definition of the OMDL-LHE experiment.

that:

$$\Pr\left[\forall i : x_i = r_i \middle| \begin{array}{c} r_1 ... r_{q+1} \xleftarrow{\$} \mathcal{Z}_q \\ \forall i \in [1, q+1], h_i \leftarrow g_i^{r_i} \\ \{x_i\}_{i \in [1,q+1]} \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{DL}}(\{h_i\}_{i \in [1,q+1]}) \end{array} \right] = 1$$

*where $\mathcal{O}$*DL *takes as input $h \in \mathbb{G}$ and outputs x s.t. $h = g^x$.*

Lemma 1. *Let* LHE *be a linear-only homomorphic encryption scheme. Assuming the hardness of the OMDL assumption,* LHE *is secure under OMDL-*LHE.

B ORACLE-BASED UNLINKABLE CONTINGENT PAYMENT CORRECTNESS, SECURITY AND PRIVACY PROOFS

Theorem 5 (O-UCPCorrectness). *Assume the adaptor signature scheme is correct, assume the* WES *encryption is correct, assume that VWER is correct and that the linear only encryption scheme is correct. Then, our protocol in Fig. 9 offers O-UCPcorrectness according to Definition 2.*

Proof. We have to prove that (i) $\mathsf{BVfSet}(\widehat{vk}, m_B, pek, aP_4)) = 1$; (ii) $\mathsf{Vf}(vk_B, m_B, \sigma_B) = 1$; (iii) $\mathsf{Vf}(vk_M, m_M, \sigma_M) = 1$; (iv) $\mathsf{VfAttest}(\widehat{vk}, m_B, \tau) = 1$; and (v) $(pek, pdk') \in \mathsf{R}$.

As described in Definition 2, we need to prove the previous conditions in the following setting: $\lambda \in \mathbb{N}$, $(\widehat{vk}, \widehat{sk}) \in \mathsf{NGen}(1^\lambda)$, $(\overline{ek}, \overline{dk}) \in \mathsf{MGen}(1^\lambda)$, $(vk_M, sk_M) \in \mathsf{KGen}(1^\lambda)$, $(vk_B, sk_B) \in \mathsf{KGen}(1^\lambda)$, $(pek, pdk) \in \mathsf{R}$, and a pair of messages (m_B, m_M).

<u>*Case* $\mathsf{BVfSet}(\widehat{vk}, m_B, pek, aP_4)) = 1$</u>: As defined in BVfSet, we have that

$$\begin{aligned} \mathsf{BVfSet}(\widehat{vk}, m_B, pek, aP_4)) &= \\ \mathsf{VWER.VfEncR}(aP_4, (\widehat{vk}, m_B), pek) &= \\ \mathsf{VWER.VfEncR}(\mathsf{VWER.EncR}((\widehat{vk}, m_B), pdk), (\widehat{vk}, m_B), pek) &= 1 \end{aligned}$$

<u>*Case* $\mathsf{Vf}(vk_B, m_B, \sigma_B) = 1$</u>: This trivially holds from the correctness of the digital signature scheme, namely

$$\mathsf{Vf}(vk_B, m_B, \mathsf{Sig}(sk_B, m_B)) = 1$$

Case $\mathsf{Vf}(vk_M, m_M, \sigma_M) = 1$: We analyze this case in two steps. First, assume that the value w_1 obtained in SSolve is the same value w_1 used in MSet_1. Then, it holds that:

$$\mathsf{Vf}(vk_M, m_M, \sigma_M) =$$
$$\mathsf{Vf}(vk_M, m_M, \mathsf{ADP.Adapt}(\widetilde{\sigma}, w_1)) =$$
$$\mathsf{Vf}(vk_M, m_M, \mathsf{ADP.Adapt}(\mathsf{ADP.PreSig}(sk_M, m_M, \mathsf{X}_1), w_1) = 1$$

Now, we show that indeed the value w_1 obtained in SSolve is the same value w_1 used in MSet_1.

$$w_1 = w_2 - w_r$$
$$w_1 = \mathsf{WES.Dec}(\tau, c_3) - w_r$$
$$w_1 = \mathsf{WES.Dec}(\widehat{\mathsf{DS}}.\widehat{\mathsf{Sig}}(\widehat{sk}, m_B), c_3) - w_r$$
$$w_1 = \mathsf{WES.Dec}(\widehat{\mathsf{DS}}.\widehat{\mathsf{Sig}}(\widehat{sk}, m_B), \mathsf{WES.Enc}((\widehat{vk}, m_B), w_2^*)) - w_r$$
$$w_1 = w_2^* - w_r$$
$$w_1 = \mathsf{LHE.Dec}(\overline{dk}, c_2) - w_r$$
$$w_1 = \mathsf{LHE.Dec}(\overline{dk}, c_1 \circ c_r) - w_r$$
$$w_1 = \mathsf{LHE.Dec}(\overline{dk}, \mathsf{LHE.Enc}(\overline{ek}, w_1) \circ \mathsf{LHE.Enc}(\overline{ek}, w_r)) - w_r$$
$$w_1 = \mathsf{LHE.Dec}(\overline{dk}, \mathsf{LHE.Enc}(\overline{ek}, w_1 + w_r) - w_r$$
$$w_1 = w_1 + w_r - w_r$$

Case $\mathsf{VfAttest}(\widehat{vk}, m_B, \tau) = 1$: This trivially holds from the correctness of the digital signature scheme used for attestations, namely

$$\widehat{\mathsf{DS}}.\widehat{\mathsf{Vf}}(\widehat{vk}, m_B, \widehat{\mathsf{DS}}.\widehat{\mathsf{Sig}}(\widehat{sk}, m_B)) = 1$$

Case $(pek, pdk') \in \mathsf{R}$: Recall that in the initial setting we have that $(pek, pdk) \in \mathsf{R}$. For this case, we prove that $pdk' = pdk$, which trivially implies that $(pek, pdk') \in \mathsf{R}$.

$$pdk' = \mathsf{VWER.DecR}(\tau, c_4, \pi_4)$$
$$pdk' = \mathsf{VWER.DecR}(\widehat{\mathsf{DS}}.\widehat{\mathsf{Sig}}(\widehat{sk}, m_B), \mathsf{VWER.EncR}((\widehat{vk}, m_B), pdk))$$
$$pdk' = pdk$$

□

Theorem 1 (Mixer Security). *Assume that NIZK is zero knowledge, that* WES *is IND-CPA, that adaptor signature is full extractable and the linear only encryption scheme is OMDL-*LHE. *Then, our construction offers mixer security according to Definition 3.*

Proof. We require the following game hops in order to prove our claim:

Game ExpM^{G_0}: This game, formally defined in Fig. 15, corresponds to the original game for ExpM defined in Definition 3 The game is expanded with the interactions described in our implementation.

Game ExpM^{G_1}: This game, formally defined in Fig. 16, works exactly as G_0 but with the highlighted grey line. The challenger uses a simulator instead of the Prove algorithm to generate the proof for MSet_1.

Game ExpM^{G_2}: This game, formally defined in Fig. 17, works exactly as G_1 but with the highlighted grey line. The challenger uses a simulator instead of the Prove algorithm to generate the proof for MSet_3.

Game ExpM^{G_3}: This game, formally defined in Fig. 18, works exactly as G_2 but with the highlighted grey line. Instead of encrypting w_2, the challenger encrypts 0 in $\mathcal{O}\mathsf{MSet}_3$.

Game ExpM^{G_4}: This game, formally defined in Fig. 19, works exactly as G_3 but with the highlighted grey lines. The challenger has an additional memory Q_1' to keep track of the presignatures and statements provided in $\mathcal{O}\mathsf{MSet}_1$. The game aborts if the adversary wins with a signature on message that was not queried in $\mathcal{O}\mathsf{MSet}_1$ or with a signature on a message queried in $\mathcal{O}\mathsf{MSet}_1$ such that the corresponding presignature does not provide the witness to the statement.

Claim 1. *Let* Bad_1 *be the event that:*

$$\left| \begin{matrix} \Pr[\mathsf{ExpM}^{G_0}(\lambda) = 1] \\ - \Pr[\mathsf{ExpM}^{G_1}(\lambda) = 1] \end{matrix} \right| > \mathsf{negl}$$

Assume that the NIZK used for $\mathcal{L}_1$ *is zero knowledge. Then* $\Pr[\mathsf{Bad}_1(1^\lambda) = 1] \le \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that $\Pr[\mathsf{Bad}_1(1^\lambda)] > \mathsf{negl}(\lambda)$, then there exists PPT distinguisher $\mathcal{A}$ such that:

$$\Pr\left[b = b^* \middle| \begin{matrix} b \xleftarrow{\$} \{0,1\} \\ \mathsf{ExpM}^{G_b}(\lambda) \\ b^* \leftarrow \mathcal{A}() \end{matrix} \right] > \frac{1}{2} + \mathsf{negl}$$

We can construct adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break zero knowledge of $\mathcal{L}_1$ with the following steps:

- $\mathcal{B}$ initializes the challenger, who will flip a bit and decide if it uses Prove or the simulator $\mathcal{S}$. The simulator sets the crs that will be used for the proofs related to $\mathcal{L}_1$. $\mathcal{B}$ initializes the crs for $\mathcal{L}_2$.
- $\mathcal{B}$ runs $(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{KGen}}(1^\lambda)$ and $(\overline{ek}, \overline{dk}) \leftarrow \mathsf{LHE.KGen}(1^\lambda)$.
- $\mathcal{B}$ invokes $\mathcal{A}$ on input $\widehat{vk}$ and $\overline{ek}$ to obtain $\left\{vk_M^i, m_M^i, \sigma_M^i\right\}_{i \in [0,q]}$.
- $\mathcal{B}$ receives the guess b^* from $\mathcal{A}$, which $\mathcal{B}$ forwards to the challenger.

Regarding oracles $\mathcal{O}\mathsf{MSet}_3$ and $\mathcal{O}\mathsf{Full}$, $\mathcal{B}$ knows all the private information required to run them. However, regarding $\mathcal{O}\mathsf{MSet}_1$, instead of running either $\mathcal{S}$ or Prove, $\mathcal{B}$ will forward the statement y and w_1 to the challenger, who will provide the proof π_1. Then, $\mathcal{B}$ will place this proof in rP_1.

Our adversary $\mathcal{B}$ perfectly simulates ExpM^{G_0} and ExpM^{G_1} to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. If the adversary can distinguish between the two games with probability higher than $\frac{1}{2}+\mathsf{negl}(\lambda)$, since the only difference between both games is whether the challenger decided to use Prove or $\mathcal{S}$ when it was initialized by $\mathcal{B}$, the guess b^* also wins the zero knowledge game with the same probability. However, this contradicts the assumption that the NIZK for $\mathcal{L}_1$ is zero knowledge. Thus, $\Pr[\mathsf{Bad}_1(1^\lambda)] \le \mathsf{negl}(\lambda)$ and this claim has been proven. Therefore, we can conclude that $\mathsf{ExpM}^{G_0} \approx \mathsf{ExpM}^{G_1}$ □

Claim 2. *Let* Bad_2 *be the event that:*

$$\left| \begin{matrix} \Pr[\mathsf{ExpM}^{G_1}(\lambda) = 1] \\ - \Pr[\mathsf{ExpM}^{G_2}(\lambda) = 1] \end{matrix} \right| > \mathsf{negl}$$

ExpM^{G_0}

$Q_1 := \emptyset$; $Q_2 := \emptyset$; $\mathsf{q} := 0$
$(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{KGen}}(1^\lambda)$
$(\overline{ek}, \overline{dk}) \leftarrow \mathsf{LHE.KGen}(1^\lambda)$
$\{vk_M^i, m_M^i, \sigma_M^i\}_{i\in[0,\mathsf{q}]} \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{MSet}_1, \mathcal{O}\mathsf{MSet}_3, \mathcal{O}\mathsf{Full}}(\overline{ek}, \widehat{vk})$
$b_0 := \exists i \in [0, \mathsf{q}]$ s.t. $(vk_M^i, \cdot) \in Q_1$
$\quad \wedge (vk_M^i, m_M^i) \notin Q_1 \wedge \mathsf{Vf}(vk_M^i, m_M^i, \sigma_M^i) = 1$
$b_1 := \forall i \in [0, \mathsf{q}], (vk_M^i, m_M^i) \in Q_1 \wedge \mathsf{Vf}(vk_M^i, m_M^i, \sigma_M^i) = 1$
$b_2 := \forall i, j \in [0, \mathsf{q}], i \neq j, (vk_M^i, m_M^i, \sigma_M^i) \neq (vk_M^j, m_M^j, \sigma_M^j)$
return $b_0 \vee (b_1 \wedge b_2)$

$\mathcal{O}\mathsf{MSet}_1(m_M)$

$(vk_M, sk_M) \leftarrow \mathsf{KGen}(1^\lambda)$
$(\mathsf{X}_1, \mathsf{w}_1) \leftarrow \mathsf{createR}(1^\lambda)$
$\widetilde{\sigma} \leftarrow \mathsf{ADP.PreSig}(sk_M, m_M, \mathsf{X}_1)$
$c_1 \leftarrow \mathsf{LHE.Enc}(\overline{ek}, \mathsf{w}_1)$
$\mathsf{y} := (c_1, \overline{ek}, \mathsf{X}_1)$
$\pi_1 \leftarrow \mathsf{NIZK.Prove}_{\mathcal{L}_1}(\mathsf{crs}, \mathsf{y}, \mathsf{w}_1)$
$rP_1 := (\widetilde{\sigma}, c_1, \pi_1, \mathsf{X}_1)$
$Q_1 := Q_1 \cup (vk_M, m_M)$
return (rP_1, vk_M)

$\mathcal{O}\mathsf{MSet}_3(m_B, rP_2)$

if $m_B \in Q_2$ **abort**
$Q_2 := Q_2 \cup (m_B)$
$(c_2, \mathsf{X}_2) \leftharpoondown rP_2$
$\mathsf{w}_2 \leftarrow \mathsf{LHE.Dec}(\overline{dk}, c_2)$
$c_3 \leftarrow \mathsf{WES.Enc}((\widehat{vk}, m_B), \mathsf{w}_2)$
$\mathsf{y} := (c_3, \widehat{vk}, m_B, \mathsf{X}_2)$
$\pi_3 \leftarrow \mathsf{NIZK.Prove}_{\mathcal{L}_2}(\mathsf{crs}, \mathsf{y}, \mathsf{w}_2)$
$aP_3 := (c_3, \pi_3)$
return (aP_3)

$\mathcal{O}\mathsf{Full}(m_B, rP_2, \sigma, vk)$

if $m_B \in Q_2$ **abort**
$\mathsf{q} := \mathsf{q} + 1$
$Q_2 := Q_2 \cup (m_B)$
$(c_2, \mathsf{X}_2) \leftharpoondown rP_2$
$\mathsf{w}_2 \leftarrow \mathsf{LHE.Dec}(\overline{dk}, c_2)$
$c_3 \leftarrow \mathsf{WES.Enc}((\widehat{vk}, m_B), \mathsf{w}_2)$
$\mathsf{y} := (c_3, \widehat{vk}, m_B, \mathsf{X}_2)$
$\pi_3 \leftarrow \mathsf{NIZK.Prove}_{\mathcal{L}_2}(\mathsf{crs}, \mathsf{y}, \mathsf{w}_2)$
$aP_3 := (c_3, \pi_3)$
if $\mathsf{Vf}(vk, m_B, \sigma) = 0$ **abort**
$\tau \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{Sig}}(\widehat{sk}, m_B)$
return (aP_3, τ)

Figure 15: The mixer security game expanded with our implementation.

Assume that the NIZK used for $\mathcal{L}_2$ is zero knowledge. Then $\Pr[\mathsf{Bad}_2(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that $\Pr[\mathsf{Bad}_2(1^\lambda)] > \mathsf{negl}(\lambda)$, then there exists PPT distinguisher $\mathcal{A}$ such that:

$$\Pr\left[b = b^* \middle| \begin{array}{c} b \xleftarrow{\$} \{0,1\} \\ \mathsf{ExpM}^{G_{1+b}}(\lambda) \\ b^* \leftarrow \mathcal{A}() \end{array}\right] > \frac{1}{2} + \mathsf{negl}$$

We can construct adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break zero knowledge of $\mathcal{L}_2$ with the following steps:

- $\mathcal{B}$ initializes the challenger, who will flip a bit and decide if it uses Prove or the simulator $\mathcal{S}$. The simulator sets the crs that will be used for the proofs related to $\mathcal{O}\mathsf{MSet}_3$. $\mathcal{B}$ initializes the crs for $\mathcal{L}_1$.
- $\mathcal{B}$ runs $(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{KGen}}(1^\lambda)$ and $(\overline{ek}, \overline{dk}) \leftarrow \mathsf{LHE.KGen}(1^\lambda)$.
- $\mathcal{B}$ invokes $\mathcal{A}$ on input $\widehat{vk}$ and $\overline{ek}$ to obtain $\{vk_M^i, m_M^i, \sigma_M^i\}_{i\in[0,\mathsf{q}]}$.
- $\mathcal{B}$ receives the guess b^* from $\mathcal{A}$, which $\mathcal{B}$ forwards to the challenger.

Regarding oracle $\mathcal{O}\mathsf{MSet}_1$, $\mathcal{B}$ knows all the private information required to run them. However, regarding $\mathcal{O}\mathsf{MSet}_3$ and $\mathcal{O}\mathsf{Full}$, instead of running either $\mathcal{S}$ or Prove, $\mathcal{B}$ will forward the statement y and w_2 to the challenger, who will provide the proof π_3. Then, $\mathcal{B}$ will place this proof in aP_3.

Our adversary $\mathcal{B}$ perfectly simulates ExpM^{G_1} and ExpM^{G_2} to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. If the adversary can distinguish between the two games with probability higher than $\frac{1}{2}$+$\mathsf{negl}(\lambda)$, since the only difference between both games is whether the challenger decided to use Prove or $\mathcal{S}$ when it was initialized by $\mathcal{B}$, the guess b^* also wins the zero knowledge game with the same probability. However, this contradicts the assumption that the NIZK used in $\mathcal{L}_2$ is zero knowledge. Thus, $\Pr[\mathsf{Bad}_2(1^\lambda)] \leq \mathsf{negl}(\lambda)$ and this claim has been proven. Therefore, we can conclude that $\mathsf{ExpM}^{G_1} \approx \mathsf{ExpM}^{G_2}$ □

Claim 3. *Let* Bad_3 *be the event that:*

$$\left| \begin{array}{l} \Pr[\mathsf{ExpM}^{G_2}(\lambda) = 1] \\ - \Pr[\mathsf{ExpM}^{G_3}(\lambda) = 1] \end{array} \right| > \mathsf{negl}$$

Assume that WES *used in* $\mathcal{O}\mathsf{MSet}_3$ *is IND-CPA secure. Then* $\Pr[\mathsf{Bad}_3(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. Let $q_2 := |Q_2|$ denote the number of queries to oracle $\mathcal{O}\mathsf{MSet}_3$. We consider q_2 sub-games such that for sub-game $i \in [1, q_2]$ queries 1 to $i-1$ are answered by oracle $\mathcal{O}\mathsf{MSet}_3$ of game ExpM^{G_3}, while queries $i+1$ to q_2 are answered by oracle $\mathcal{O}\mathsf{MSet}_3$

ExpM^{G_1}

$Q_1 := \emptyset$; $Q_2 := \emptyset$; $\mathsf{q} := 0$

$(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{KGen}}(1^\lambda)$

$(\overline{ek}, \overline{dk}) \leftarrow \mathsf{LHE.KGen}(1^\lambda)$

$\{vk^i_M, m^i_M, \sigma^i_M\}_{i\in[0,\mathsf{q}]} \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{MSet}_1, \mathcal{O}\mathsf{MSet}_3, \mathcal{O}\mathsf{Full}}(\overline{ek}, \widehat{vk})$

$b_0 := \exists i \in [0, \mathsf{q}]$ s.t. $(vk^i_M, \cdot) \in Q_1$

$\quad \wedge (vk^i_M, m^i_M) \notin Q_1 \wedge \mathsf{Vf}(vk^i_M, m^i_M, \sigma^i_M) = 1$

$b_1 := \forall i \in [0, \mathsf{q}], (vk^i_M, m^i_M) \in Q_1 \wedge \mathsf{Vf}(vk^i_M, m^i_M, \sigma^i_M) = 1$

$b_2 := \forall i, j \in [0, \mathsf{q}], i \neq j, (vk^i_M, m^i_M, \sigma^i_M) \neq (vk^j_M, m^j_M, \sigma^j_M)$

return $b_0 \vee (b_1 \wedge b_2)$

$\mathcal{O}\mathsf{MSet}_1(m_M)$

$(vk_M, sk_M) \leftarrow \mathsf{KGen}(1^\lambda)$

$(\mathsf{X}_1, \mathsf{w}_1) \leftarrow \mathsf{createR}(1^\lambda)$

$\widetilde{\sigma} \leftarrow \mathsf{ADP.PreSig}(sk_M, m_M, \mathsf{X}_1)$

$c_1 \leftarrow \mathsf{LHE.Enc}(\overline{ek}, \mathsf{w}_1)$

$\mathsf{y} := (c_1, \overline{ek}, \mathsf{X}_1)$

$\pi_1 \leftarrow \mathcal{S}_{\mathcal{L}_1}(\mathsf{y})$

$rP_1 := (\widetilde{\sigma}, c_1, \pi_1, \mathsf{X}_1)$

$Q_1 := Q_1 \cup (vk_M, m_M)$

return (rP_1, vk_M)

$\mathcal{O}\mathsf{MSet}_3(m_B, rP_2)$

if $m_B \in Q_2$ **abort**

$Q_2 := Q_2 \cup (m_B)$

$(c_2, \mathsf{X}_2) \leftharpoondown rP_2$

$\mathsf{w}_2 \leftarrow \mathsf{LHE.Dec}(\overline{dk}, c_2)$

$c_3 \leftarrow \mathsf{WES.Enc}((\widehat{vk}, m_B), \mathsf{w}_2)$

$\mathsf{y} := (c_3, \widehat{vk}, m_B, \mathsf{X}_2)$

$\pi_3 \leftarrow \mathsf{NIZK.Prove}_{\mathcal{L}_2}(\mathsf{crs}, \mathsf{y}, \mathsf{w}_2)$

$aP_3 := (c_3, \pi_3)$

return (aP_3)

$\mathcal{O}\mathsf{Full}(m_B, rP_2, \sigma, vk)$

if $m_B \in Q_2$ **abort**

$\mathsf{q} := \mathsf{q} + 1$

$Q_2 := Q_2 \cup (m_B)$

$(c_2, \mathsf{X}_2) \leftharpoondown rP_2$

$\mathsf{w}_2 \leftarrow \mathsf{LHE.Dec}(\overline{dk}, c_2)$

$c_3 \leftarrow \mathsf{WES.Enc}((\widehat{vk}, m_B), \mathsf{w}_2)$

$\mathsf{y} := (c_3, \widehat{vk}, m_B, \mathsf{X}_2)$

$\pi_3 \leftarrow \mathsf{NIZK.Prove}_{\mathcal{L}_2}(\mathsf{crs}, \mathsf{y}, \mathsf{w}_2)$

$aP_3 := (c_3, \pi_3)$

if $\mathsf{Vf}(vk, m_B, \sigma) = 0$ **abort**

$\tau \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{Sig}}(\widehat{sk}, m_B)$

return (aP_3, τ)

Figure 16: The mixer security game, identical to ExpM^{G_0}, except for the highlighted grey lines. Instead of running Prove for the proof of MSet_1, the challenger runs a simulator $\mathcal{S}$.

of game ExpM^{G_2}. The intuition is that if $\Pr[\mathsf{Bad}_3(1^\lambda)] > \mathsf{negl}(\lambda)$, then there exists some PPT distinguisher $\mathcal{A}_i$, for $i \in [1, q_2]$, that it can determine with non-negligible probability whether it plays game ExpM^{G_2} or game ExpM^{G_3} base on the i^{th} answer of oracle $\mathcal{O}\mathsf{MSet}_3$.

More precisely, assume by contradiction that $\Pr[\mathsf{Bad}_3(1^\lambda)] > \mathsf{negl}(\lambda)$, then there exists PPT distinguisher $\mathcal{A}_{i^*}$ such that:

$$\Pr\left[b = b^* \middle| \begin{array}{c} b \xleftarrow{\$} \{0,1\} \\ \mathsf{ExpM}^{subG_{i^*}}(\lambda) \\ b^* \leftarrow \mathcal{A}_{i^*}() \end{array} \right] > \frac{1}{2} + \mathsf{negl}$$

We can construct adversary $\mathcal{B}$ that uses $\mathcal{A}_{i^*}$ to break IND-CPA the encryption used in $\mathcal{O}\mathsf{MSet}_3$ with the following steps:

- $\mathcal{B}$ initializes the challenger, who sends $\widehat{vk}$.
- $\mathcal{B}$ runs $(\overline{ek}, \overline{dk}) \leftarrow \mathsf{LHE.KGen}(1^\lambda)$.
- $\mathcal{B}$ invokes $\mathcal{A}$ on input $\widehat{vk}$ and $\overline{ek}$.
- $\mathcal{O}\mathsf{MSet}_3$ queries are treated in the following manner: (i) for $j \in [1, i^*-1]$, $\mathcal{B}$ answers with of $c_{3,j} \leftarrow \mathsf{WES.Enc}((\widehat{vk}, m_B), 0)$; (ii) for $j \in [i^*+1, q_2]$, $\mathcal{B}$ answers with $c_{3,j} \leftarrow \mathsf{WES.Enc}((\widehat{vk}, m_B), \mathsf{w}_2)$; and (iii) for $j = i^*$, $\mathcal{B}$ sets $\hat{m}^* := m_B$, $m_0 := \mathsf{w}_2$ and $m_1 := 0$ and forwards the tuple $(\hat{m}^*, m_0, m_1)$ to the challenger to obtain c_b which in turn $\mathcal{B}$ forwards to $\mathcal{A}_{i^*}$ as $c_{3,j}$.
- Thereafter $\mathcal{A}_{i^*}$ outputs $\{vk^i_M, m^i_M, \sigma^i_M\}_{i\in[0,\mathsf{q}]}$.
- $\mathcal{B}$ receives the guess b^* from $\mathcal{A}_{i^*}$, which $\mathcal{B}$ forwards to the challenger.

Regarding oracle $\mathcal{O}\mathsf{MSet}_1$, $\mathcal{B}$ knows all the private information required to run it. Regarding $\mathcal{O}\mathsf{Full}$, $\mathcal{B}$ can run up to $\widehat{\mathsf{Sig}}$. When arriving at this line, $\mathcal{B}$ forwards the query to $\mathcal{O}\mathsf{Sig}$ of the WES IND-CPA oracle, which returns τ. Note that this means that memory Q_3 and the memory of IND-CPA are synchronized. As already described, $\mathcal{B}$ knows all the private information required to run oracle $\mathcal{O}\mathsf{MSet}_3$.

Our adversary $\mathcal{B}$ perfectly simulates the sub-game $\mathsf{ExpM}^{subG_{i^*}}$ to $\mathcal{A}_{i^*}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. If adversary $\mathcal{A}_{i^*}$ can win the sub-game $\mathsf{ExpM}^{subG_{i^*}}$ with probability higher than $\frac{1}{2} + \mathsf{negl}(\lambda)$, since the only difference between games ExpM^{G_2} and ExpM^{G_3} is the i^*th query of $\mathcal{O}\mathsf{MSet}_3$ that was forwarded to the challenger and since Q_2 and Q_3 intersection has to be empty, $\mathcal{A}_{i^*}$ has not made a query to the same message of the challenger ciphertext in $\mathcal{O}\mathsf{Full}$, which satisfies that the sign oracle was not queried on the same message of the challenge. Therefore, the bit forwarded by $\mathcal{A}_{i^*}$ can also be used to differentiate in the

ExpM^{G_2}

$Q_1 := \emptyset$; $Q_2 := \emptyset$; $\mathsf{q} := 0$

$(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{KGen}}(1^\lambda)$

$(\overline{ek}, \overline{dk}) \leftarrow \mathsf{LHE.KGen}(1^\lambda)$

$\{vk^i_M, m^i_M, \sigma^i_M\}_{i\in[0,\mathsf{q}]} \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{MSet}_1, \mathcal{O}\mathsf{MSet}_3, \mathcal{O}\mathsf{Full}}(\overline{ek}, \widehat{vk})$

$b_0 := \exists i \in [0, \mathsf{q}]$ s.t. $(vk^i_M, \cdot) \in Q_1$

$\quad \wedge (vk^i_M, m^i_M) \notin Q_1 \wedge \mathsf{Vf}(vk^i_M, m^i_M, \sigma^i_M) = 1$

$b_1 := \forall i \in [0, \mathsf{q}], (vk^i_M, m^i_M) \in Q_1 \wedge \mathsf{Vf}(vk^i_M, m^i_M, \sigma^i_M) = 1$

$b_2 := \forall i, j \in [0, \mathsf{q}], i \neq j, (vk^i_M, m^i_M, \sigma^i_M) \neq (vk^j_M, m^j_M, \sigma^j_M)$

return $b_0 \vee (b_1 \wedge b_2)$

$\mathcal{O}\mathsf{MSet}_1(m_M)$

$(vk_M, sk_M) \leftarrow \mathsf{KGen}(1^\lambda)$

$(\mathsf{X}_1, \mathsf{w}_1) \leftarrow \mathsf{createR}(1^\lambda)$

$\widetilde{\sigma} \leftarrow \mathsf{ADP.PreSig}(sk_M, m_M, \mathsf{X}_1)$

$c_1 \leftarrow \mathsf{LHE.Enc}(\overline{ek}, \mathsf{w}_1)$

$\mathsf{y} := (c_1, \overline{ek}, \mathsf{X}_1)$

$\pi_1 \leftarrow \mathcal{S}_{\mathcal{L}_1}(\mathsf{y})$

$rP_1 := (\widetilde{\sigma}, c_1, \pi_1, \mathsf{X}_1)$

$Q_1 := Q_1 \cup (vk_M, m_M)$

return (rP_1, vk_M)

$\mathcal{O}\mathsf{MSet}_3(m_B, rP_2)$

if $m_B \in Q_2$ **abort**

$Q_2 := Q_2 \cup (m_B)$

$(c_2, \mathsf{X}_2) \leftharpoondown rP_2$

$\mathsf{w}_2 \leftarrow \mathsf{LHE.Dec}(\overline{dk}, c_2)$

$c_3 \leftarrow \mathsf{WES.Enc}((\widehat{vk}, m_B), \mathsf{w}_2)$

$\mathsf{y} := (c_3, \widehat{vk}, m_B, \mathsf{X}_2)$

$\pi_3 \leftarrow \mathcal{S}_{\mathcal{L}_2}(\mathsf{y})$

$aP_3 := (c_3, \pi_3)$

return (aP_3)

$\mathcal{O}\mathsf{Full}(m_B, rP_2, \sigma, vk)$

if $m_B \in Q_2$ **abort**

$\mathsf{q} := \mathsf{q} + 1$

$Q_2 := Q_2 \cup (m_B)$

$(c_2, \mathsf{X}_2) \leftharpoondown rP_2$

$\mathsf{w}_2 \leftarrow \mathsf{LHE.Dec}(\overline{dk}, c_2)$

$c_3 \leftarrow \mathsf{WES.Enc}((\widehat{vk}, m_B), \mathsf{w}_2)$

$\mathsf{y} := (c_3, \widehat{vk}, m_B, \mathsf{X}_2)$

$\pi_3 \leftarrow \mathcal{S}_{\mathcal{L}_2}(\mathsf{y})$

$aP_3 := (c_3, \pi_3)$

if $\mathsf{Vf}(vk, m_B, \sigma) = 0$ **abort**

$\tau \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{Sig}}(\widehat{sk}, m_B)$

return (aP_3, τ)

Figure 17: The mixer security game, identical to ExpM^{G_1}, except for the highlighted grey lines. Instead of running Prove for the proof of MSet_2, the challenger runs a simulator $\mathcal{S}$.

IND-CPA game. However, this contradicts the assumption that the WES used is IND-CPA.

Our adversary $\mathcal{B}$ chooses which sub-game i^* to play with probability $\frac{1}{q_2}$. Thus, $\Pr[\mathsf{Bad}_3(1^\lambda)] \leq \frac{\mathsf{negl}(\lambda)}{q_2} \leq \mathsf{negl}(\lambda)$ and this claim has been proven. Therefore, we can conclude that $\mathsf{ExpM}^{G_2} \approx \mathsf{ExpM}^{G_3}$ □

Claim 4. *Let Bad_4 be the event that ExpM^{G_4} aborts because b_0 or b_3 is satisfied. Assume that the adaptor signature scheme provides full extractability. Then $\Pr[\mathsf{Bad}_4(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$.*

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{Bad}_4(1^\lambda)] > \mathsf{negl}(\lambda)$. We can construct adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break full extractability of the adaptor signature used in $\mathcal{O}\mathsf{MSet}_1$ with the following steps:

- $\mathcal{B}$ runs $(\overline{ek}, \overline{dk}) \leftarrow \mathsf{LHE.KGen}(1^\lambda)$ and $(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{KGen}}(1^\lambda)$.
- $\mathcal{B}$ initializes the challenger and obtains the public key of the challenger, vk.
- $\mathcal{B}$ invokes $\mathcal{A}$ on input $\widehat{vk}$ and $\overline{ek}$ to obtain $\{vk^i_M, m^i_M, \sigma^i_M\}_{i\in[0,\mathsf{q}]}$.
- $\mathcal{B}$ searches for a triplet of $vk^i_M, m^i_M, \sigma^i_M$ such that b_0 or b_4 hold. If $vk^i_M = vk$, then $\mathcal{B}$ forwards (m^i_M, σ^i_M) to the challenger. If $vk^i_M \neq vk$, $\mathcal{B}$ samples a signature from the signature space and forwards it to the challenger.

Regarding oracle $\mathcal{O}\mathsf{MSet}_3$ and $\mathcal{O}\mathsf{Full}$, $\mathcal{B}$ knows all the private information required to run them. Regarding $\mathcal{O}\mathsf{MSet}_1$, $\mathcal{B}$ samples fresh keys with each query. However, for one the queries, $\mathcal{B}$ generates a public statement X and uses it together with the message from $\mathcal{A}$ as input for $\mathcal{O}\mathsf{PreSig}$ of the fext game to obtain a presignature. The rest of $\mathcal{O}\mathsf{MSet}_1$ runs normally.

Our adversary $\mathcal{B}$ perfectly simulates ExpM^{G_4} to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, the only differences between ExpM^{G_3} and ExpM^{G_4} are the change in the memory of $\mathcal{B}$ and the abort condition. Since we assume that $\mathcal{A}$ is successful in aborting ExpM^{G_4}, this means that $\mathcal{A}$ satisfies either b_0 or b_3. If $\mathcal{A}$ satisfies b_0, this means that one of the signatures was done for a message that was not queried in $\mathcal{O}\mathsf{MSet}_1$. If the forgery is valid for the challenger's public key vk, but not for the other keys generated with $\mathcal{O}\mathsf{MSet}_1$, it holds that: (i) $\mathcal{B}$ did not queried $\mathcal{O}\mathsf{Sig}$ of fext; (ii) the signature verifies for vk; and (iii) m^i_M was not queried in $\mathcal{O}\mathsf{PreSig}$. Therefore, if $\mathcal{A}$ forges a signature for vk without querying $\mathcal{O}\mathsf{MSet}_1$ for m^i_M, $\mathcal{B}$ wins fext. Alternatively, if $\mathcal{A}$ satisfies b_3, this means that m^i_M was queried in $\mathcal{O}\mathsf{MSet}_1$, but the signature and the presignature do not output a valid witness. If the forgery is

ExpM^{G_3}

$Q_1 := \emptyset$; $Q_2 := \emptyset$; $\mathsf{q} := 0$

$(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{KGen}}(1^\lambda)$

$(\overline{ek}, \overline{dk}) \leftarrow \mathsf{LHE.KGen}(1^\lambda)$

$\{vk^i_M, m^i_M, \sigma^i_M\}_{i\in[0,\mathsf{q}]} \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{MSet}_1, \mathcal{O}\mathsf{MSet}_3, \mathcal{O}\mathsf{Full}}(\overline{ek}, \widehat{vk})$

$b_0 := \exists i \in [0, \mathsf{q}]$ s.t. $(vk^i_M, \cdot) \in Q_1$

$\quad \wedge (vk^i_M, m^i_M) \notin Q_1 \wedge \mathsf{Vf}(vk^i_M, m^i_M, \sigma^i_M) = 1$

$b_1 := \forall i \in [0, \mathsf{q}], (vk^i_M, m^i_M) \in Q_1 \wedge \mathsf{Vf}(vk^i_M, m^i_M, \sigma^i_M) = 1$

$b_2 := \forall i, j \in [0, \mathsf{q}], i \neq j, (vk^i_M, m^i_M, \sigma^i_M) \neq (vk^j_M, m^j_M, \sigma^j_M)$

return $b_0 \vee (b_1 \wedge b_2)$

$\mathcal{O}\mathsf{MSet}_1(m_M)$

$(vk_M, sk_M) \leftarrow \mathsf{KGen}(1^\lambda)$

$(\mathsf{X}_1, \mathsf{w}_1) \leftarrow \mathsf{createR}(1^\lambda)$

$\widetilde{\sigma} \leftarrow \mathsf{ADP.PreSig}(sk_M, m_M, \mathsf{X}_1)$

$c_1 \leftarrow \mathsf{LHE.Enc}(\overline{ek}, \mathsf{w}_1)$

$\mathsf{y} := (c_1, \overline{ek}, \mathsf{X}_1)$

$\pi_1 \leftarrow \mathcal{S}_{\mathcal{L}_1}(\mathsf{y})$

$rP_1 := (\widetilde{\sigma}, c_1, \pi_1, \mathsf{X}_1)$

$Q_1 := Q_1 \cup (vk_M, m_M)$

return (rP_1, vk_M)

$\mathcal{O}\mathsf{MSet}_3(m_B, rP_2)$

if $m_B \in Q_2$ **abort**

$Q_2 := Q_2 \cup (m_B)$

$(c_2, \mathsf{X}_2) \leftharpoondown rP_2$

$\mathsf{w}_2 \leftarrow \mathsf{LHE.Dec}(\overline{dk}, c_2)$

$c_3 \leftarrow \mathsf{WES.Enc}((\widehat{vk}, m_B), 0)$

$\mathsf{y} := (c_3, \widehat{vk}, m_B, \mathsf{X}_2)$

$\pi_3 \leftarrow \mathcal{S}_{\mathcal{L}_2}(\mathsf{y})$

$aP_3 := (c_3, \pi_3)$

return (aP_3)

$\mathcal{O}\mathsf{Full}(m_B, rP_2, \sigma, vk)$

if $m_B \in Q_2$ **abort**

$\mathsf{q} := \mathsf{q} + 1$

$Q_2 := Q_2 \cup (m_B)$

$(c_2, \mathsf{X}_2) \leftharpoondown rP_2$

$\mathsf{w}_2 \leftarrow \mathsf{LHE.Dec}(\overline{dk}, c_2)$

$c_3 \leftarrow \mathsf{WES.Enc}((\widehat{vk}, m_B), \mathsf{w}_2)$

$\mathsf{y} := (c_3, \widehat{vk}, m_B, \mathsf{X}_2)$

$\pi_3 \leftarrow \mathcal{S}_{\mathcal{L}_2}(\mathsf{y})$

$aP_3 := (c_3, \pi_3)$

if $\mathsf{Vf}(vk, m_B, \sigma) = 0$ **abort**

$\tau \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{Sig}}(\widehat{sk}, m_B)$

return (aP_3, τ)

Figure 18: The mixer security game, identical to ExpM^{G_2}, except for the highlighted grey line. Instead of running encrypting w_3 in $\mathcal{O}\mathsf{MSet}_2$, the challenger encrypts 0.

valid for the challenger's public key vk, but not for the other keys generated with $\mathcal{O}\mathsf{MSet}_1$, it holds that: (i) $\mathcal{B}$ did not queried $\mathcal{O}\mathsf{Sig}$ of fext; (ii) the signature verifies for vk; (iii) the public statement was not queried in $\mathcal{O}\mathsf{newX}$; and (iv) extract gives a witness not in the relation. Therefore, if $\mathcal{A}$ satisfies b_3, $\mathcal{B}$ also breaks fext. We only need to quantify the probability that $\mathcal{A}$ sends a forgery for vk instead of any other key generated with queries to $\mathcal{O}\mathsf{MSet}_1$. $\mathcal{A}$ is a polynomial-time adversary, which means that the k queries made to $\mathcal{O}\mathsf{MSet}_1$ are polynomially bounded. We assume that $\mathcal{A}$ satisfies b_0 or b_3 with a non negligible probability ϵ. Then, the probability that the forgery presented to $\mathcal{B}$ is on vk is ϵ/k. Therefore, if $\mathcal{B}$ forwards the forgery to the challenger, the probability of winning fext is $\Pr[\mathsf{Bad}_4(1^\lambda) = 1]/k$. Since k is polynomial and $\Pr[\mathsf{Bad}_4(1^\lambda) = 1]$ is non negligible, $\mathcal{B}$ wins with non negligible probability. However, this contradicts the assumption that the adaptor signature scheme offers full extractability. Thus, $\Pr[\mathsf{Bad}_4(1^\lambda)] \leq \mathsf{negl}(\lambda)$ and this claim has been proven. Therefore, we can conclude that $\mathsf{ExpM}^{G_3} \approx \mathsf{ExpM}^{G_4}$

□

Claim 5. *Assume the encryption scheme is OMDL-*LHE*. Then* $\Pr[\mathsf{ExpM}^{G_4}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{ExpM}^{G_4}(1^\lambda)] > \mathsf{negl}(\lambda)$. We can construct adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break OMDL-LHE of the encryption used in $\mathcal{O}\mathsf{MSet}_1$ with the following steps:

- $\mathcal{B}$ initializes the challenger, who provides $\mathcal{B}$ with $\overline{ek}$ and $\{(\mathsf{X}_i, c_i)\}_{i\in[0,k]}$.
- $\mathcal{B}$ runs $(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{KGen}}(1^\lambda)$.
- $\mathcal{B}$ invokes $\mathcal{A}$ on input $\widehat{vk}$ and $\overline{ek}$ to obtain $\{vk^i_M, m^i_M, \sigma^i_M\}_{i\in[0,\mathsf{q}]}$.
- Since we assume that $\mathcal{A}$ wins ExpM^{G_4}, then b_0 and b_3 are not satisfied.
- For each of the tuples $vk^i_M, m^i_M, \sigma^i_M$, $\mathcal{B}$ gets $\widetilde{\sigma}^i$ and X^i_1 from Q'_1 and extracts w^i_1. Note that $(\mathsf{X}^i_1, \mathsf{w}^i_1) \in \mathsf{R}$, as otherwise $\mathcal{B}$ would abort because of b_3. Also note that due to conditions b_1 and b_2, all witnesses are different. Parameter k from OMDL-LHE corresponds to the number of queries for $\mathcal{O}\mathsf{MSet}_1$, while Parameter q from OMDL-LHE corresponds to the number of queries for $\mathcal{O}\mathsf{Full}$. For the witnesses remaining between the q+1 witnesses obtained from $\mathcal{A}$ until the k+1 that must be forwarded to the challenger, $\mathcal{B}$ calls k-q times $\mathcal{O}\mathsf{Dec}$ of OMDL-LHE. Finally, $\mathcal{B}$ sends all k+1 witnesses to the challenger.

$\underline{\mathsf{ExpM}^{G_4}}$

$Q_1 := \emptyset\ ;\ Q_2 := \emptyset\ ;\ \mathsf{q} := 0$

$Q_1' := [\,]$

$(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{KGen}}(1^\lambda)$

$(\overline{ek}, \overline{dk}) \leftarrow \mathsf{LHE.KGen}(1^\lambda)$

$\{vk^i_M, m^i_M, \sigma^i_M\}_{i\in[0,\mathsf{q}]} \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{MSet}_1, \mathcal{O}\mathsf{MSet}_3, \mathcal{O}\mathsf{Full}}(\overline{ek}, \widehat{vk})$

$b_0 := \exists i \in [0,\mathsf{q}]$ s.t. $(vk^i_M, \cdot) \in Q_1$

$\quad \wedge (vk^i_M, m^i_M, \cdot) \notin Q_1 \wedge \mathsf{Vf}(vk^i_M, m^i_M, \sigma^i_M) = 1$

$b_1 := \forall i \in [0,\mathsf{q}], (vk^i_M, m^i_M) \in Q_1 \wedge \mathsf{Vf}(vk^i_M, m^i_M, \sigma^i_M) = 1$

$b_2 := \forall i, j \in [0,\mathsf{q}], i \neq j, (vk^i_M, m^i_M, \sigma^i_M) \neq (vk^j_M, m^j_M, \sigma^j_M)$

$b_3 := \exists i \in [0,\mathsf{q}]$ s.t. $(vk^i_M, m^i_M) \in Q_1 \wedge (\widetilde{\sigma}^i, \mathsf{X}^i_1) \leftharpoondown Q_1'[m^i_M]$

$\quad \wedge \mathsf{Vf}(vk^i_M, m^i_M, \sigma^i_M) = 1 \wedge (\mathsf{X}^i_1, \mathsf{Extract}(\sigma^i_M, \widetilde{\sigma}^i, \mathsf{X}^i_1)) \notin \mathsf{R}$

if $b_0 \vee b_3$ **abort**

return $b_0 \vee (b_1 \wedge b_2)$

$\underline{\mathcal{O}\mathsf{MSet}_1(m_M)}$

$(vk_M, sk_M) \leftarrow \mathsf{KGen}(1^\lambda)$

$(\mathsf{X}_1, \mathsf{w}_1) \leftarrow \mathsf{createR}(1^\lambda)$

$\widetilde{\sigma} \leftarrow \mathsf{ADP.PreSig}(sk_M, m_M, \mathsf{X}_1)$

$c_1 \leftarrow \mathsf{LHE.Enc}(\overline{ek}, \mathsf{w}_1)$

$\mathsf{y} := (c_1, \overline{ek}, \mathsf{X}_1)$

$\pi_1 \leftarrow \mathcal{S}_{\mathcal{L}_1}(\mathsf{y})$

$rP_1 := (\widetilde{\sigma}, c_1, \pi_1, \mathsf{X}_1)$

$Q_1 := Q_1 \cup (vk_M, m_M)$

$Q_1'[m_M] := (\widetilde{\sigma}, \mathsf{X}_1)$

return (rP_1, vk_M)

$\underline{\mathcal{O}\mathsf{MSet}_3(m_B, rP_2)}$

if $m_B \in Q_2$ **abort**

$Q_2 := Q_2 \cup (m_B)$

$(c_2, \mathsf{X}_2) \leftharpoondown rP_2$

$\mathsf{w}_2 \leftarrow \mathsf{LHE.Dec}(\overline{dk}, c_2)$

$c_3 \leftarrow \mathsf{WES.Enc}((\widehat{vk}, m_B), 0)$

$\mathsf{y} := (c_3, \widehat{vk}, m_B, \mathsf{X}_2)$

$\pi_3 \leftarrow \mathcal{S}_{\mathcal{L}_2}(\mathsf{y})$

$aP_3 := (c_3, \pi_3)$

return (aP_3)

$\underline{\mathcal{O}\mathsf{Full}(m_B, rP_2, \sigma, vk)}$

if $m_B \in Q_2$ **abort**

$\mathsf{q} := \mathsf{q} + 1$

$Q_2 := Q_2 \cup (m_B)$

$(c_2, \mathsf{X}_2) \leftharpoondown rP_2$

$\mathsf{w}_2 \leftarrow \mathsf{LHE.Dec}(\overline{dk}, c_2)$

$c_3 \leftarrow \mathsf{WES.Enc}((\widehat{vk}, m_B), \mathsf{w}_2)$

$\mathsf{y} := (c_3, \widehat{vk}, m_B, \mathsf{X}_2)$

$\pi_3 \leftarrow \mathcal{S}_{\mathcal{L}_2}(\mathsf{y})$

$aP_3 := (c_3, \pi_3)$

if $\mathsf{Vf}(vk, m_B, \sigma) = 0$ **abort**

$\tau \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{Sig}}(\widehat{sk}, m_B)$

return (aP_3, τ)

Figure 19: The mixer security security game, works exactly as G_3 but with the highlighted grey lines. The challenger has an additional memory Q_1' to keep track of the presignatures and statements provided in $\mathcal{O}\mathsf{MSet}_1$. The game aborts if the adversary wins with a signature on message that was not queried in $\mathcal{O}\mathsf{MSet}_1$ or with a signature on a message queried in $\mathcal{O}\mathsf{MSet}_1$ such that the corresponding presignature does not provide the witness to the statement.

Regarding oracle $\mathcal{O}\mathsf{MSet}_3$, since c_3 encrypts zero, there is no need to decrypt rP_2, so $\mathcal{B}$ can run the oracle with the information in their hands. Regarding $\mathcal{O}\mathsf{Full}$, $\mathcal{B}$ forwards the decryption query to the decryption oracle of OMDL-LHE, while the rest of the oracle remains the same. Note that this ensures that the counter for both oracles is the same. Finally, regarding $\mathcal{O}\mathsf{MSet}_1$, $\mathcal{B}$ uses for each query a different pair of X_i, c_i received from the challenger to make $\widetilde{\sigma}$ and c_1.

Our adversary $\mathcal{B}$ perfectly simulates ExpM^{G_4} to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Since we assume that $\mathcal{A}$ is successful in winning ExpM^{G_4}, this implies that the adversary is able to produce one signature more than the q signatures he has had access to. Since the adversary wins the game, it does not abort on b_0 or b_4, which ensures that all of the witnesses extracted for all $i \in [0,\mathsf{q}]$ are valid. Finally, since the messages of all tuples sent by $\mathcal{A}$ are in the memory of Q_1, and they are one more in number that the counter of q, this implies that the decryption oracle has not been called for at least one of the witnesses extracted by $\mathcal{B}$. Note that the counters of both games are synchronized and that $\mathcal{B}$ is only using the pairs X_i, c_i sent by the challenger to run $\mathcal{O}\mathsf{MSet}_1$. Therefore, when $\mathcal{B}$ forwards all the witnesses to the challenger, the set of witnesses also wins the OMDL-LHE. However, this contradicts the assumption that the encryption scheme satisfies OMDL-LHE, and so $\mathcal{A}$ does not exist. □

We have proved that $\mathsf{ExpM}^{G_0} \approx \mathsf{ExpM}^{G_4}$ and that $\Pr[\mathsf{ExpM}^{G_4}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$. Therefore, Theorem 1 has been proven. □

Theorem 2 (Seller Security). *Assume the VWER is one way, NIZK is secure under soundness-knowledge and adaptor signature scheme is secure under adaptability. Then, our construction offers seller security according to Definition 4.*

$$
\begin{array}{l}
\underline{\mathsf{ExpS}^{G_0}} \\
Q := [\,] \\
(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{KGen}}(1^\lambda) \\
(pek, pdk) \leftarrow \mathsf{createR}(1^\lambda) \\
(\overline{ek}, vk_M, m_B, m_M, rP_1) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{NAttest}}(\widehat{vk}, pek) \\
(\widetilde{\sigma}, c_1, \pi_1, \mathsf{X}_1) \leftharpoondown rP_1 \\
\mathsf{y} := (c_1, \overline{ek}, \mathsf{X}_1) \\
\textbf{if } \mathsf{NIZK.Vf}_{\mathcal{L}_1}(\mathsf{crs}, \mathsf{y}, \pi_1) = 0 \textbf{ abort} \\
\textbf{if } \mathsf{ADP.PreVf}(vk_M, m_M, \mathsf{X}_1, \widetilde{\sigma}) = 0 \textbf{ abort} \\
(\mathsf{X}_r, \mathsf{w}_r) \leftarrow \mathsf{createR}(1^\lambda) \\
\mathsf{X}_2 := \mathsf{X}_r \otimes \mathsf{X}_1 \\
c_r \leftarrow \mathsf{LHE.Enc}(\overline{ek}_M, \mathsf{w}_r) \\
c_2 := c_1 \circ c_r \\
rP_2 := (c_2, \mathsf{X}_2) \\
st_S := (\mathsf{X}_2, \mathsf{X}_r, \mathsf{w}_r) \\
aP_3 \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{NAttest}}(rP_2) \\
(c_3, \pi_3) \leftharpoondown aP_3 \\
\mathsf{y} := (c_3, \widehat{vk}, m_B, \mathsf{X}_2) \\
\textbf{if } \mathsf{NIZK.Vf}_{\mathcal{L}_2}(\mathsf{crs}, \mathsf{y}, \pi_3) = 0 \textbf{ abort} \\
(c_4, \pi_4) \leftarrow \mathsf{VWER.EncR}((\widehat{vk}, m_B), pdk) \\
aP_4 := (c_4, \pi_4) \\
pdk' \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{NAttest}}(aP_4) \\
\textbf{if } Q[m_B] = \bot \\
\quad b_0 := (pek, pdk') \in \mathsf{R} \\
\textbf{else} \\
\quad \tau \leftarrow Q[m_B] \\
\quad \mathsf{w}_2 \leftarrow \mathsf{WES.Dec}(\tau, c_3) \\
\quad \mathsf{w}_1 := \mathsf{w}_2 - \mathsf{w}_r \\
\quad \sigma_M \leftarrow \mathsf{ADP.Adapt}(\widetilde{\sigma}, \mathsf{w}_1) \\
\quad b_1 := \widehat{\mathsf{DS}}.\widehat{\mathsf{Vf}}(\widehat{vk}, m_B, \tau) = 1 \\
\quad b_2 := \mathsf{Vf}(vk_M, m_M, \sigma_M) = 0 \\
\textbf{return } b_0 \vee (b_1 \wedge b_2) \\
\\
\underline{\mathcal{O}\mathsf{NAttest}(vk, m, \sigma)} \\
\textbf{if } \mathsf{Vf}(vk, m, \sigma) = 0 \textbf{ abort} \\
\tau \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{Sig}}(\widehat{sk}, m_B) \\
Q[m] := \tau \\
\textbf{return } \tau
\end{array}
$$

Figure 20: Seller security expanded with the interactions described in our implementation.

Proof. We require the following game hops in order to prove our theorem:

Game ExpS^{G_0}: This game, formally defined in Fig. 20, corresponds to the original game for ExpS defined in Definition 4 The game is expanded with the interactions described in our implementation.

$$
\begin{array}{l}
\underline{\mathsf{ExpS}^{G_1}} \\
Q := [\,] \\
(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{KGen}}(1^\lambda) \\
(pek, pdk) \leftarrow \mathsf{createR}(1^\lambda) \\
(\overline{ek}, vk_M, m_B, m_M, rP_1) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{NAttest}}(\widehat{vk}, pek) \\
(\widetilde{\sigma}, c_1, \pi_1, \mathsf{X}_1) \leftharpoondown rP_1 \\
\mathsf{y} := (c_1, \overline{ek}, \mathsf{X}_1) \\
\textbf{if } \mathsf{NIZK.Vf}_{\mathcal{L}_1}(\mathsf{crs}, \mathsf{y}, \pi_1) = 0 \textbf{ abort} \\
\textbf{if } \mathsf{ADP.PreVf}(vk_M, m_M, \mathsf{X}_1, \widetilde{\sigma}) = 0 \textbf{ abort} \\
(\mathsf{X}_r, \mathsf{w}_r) \leftarrow \mathsf{createR}(1^\lambda) \\
\mathsf{X}_2 := \mathsf{X}_r \otimes \mathsf{X}_1 \\
c_r \leftarrow \mathsf{LHE.Enc}(\overline{ek}_M, \mathsf{w}_r) \\
c_2 := c_1 \circ c_r \\
rP_2 := (c_2, \mathsf{X}_2) \\
st_S := (\mathsf{X}_2, \mathsf{X}_r, \mathsf{w}_r) \\
aP_3 \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{NAttest}}(rP_2) \\
(c_3, \pi_3) \leftharpoondown aP_3 \\
\mathsf{y} := (c_3, \widehat{vk}, m_B, \mathsf{X}_2) \\
\textbf{if } \mathsf{NIZK.Vf}_{\mathcal{L}_2}(\mathsf{crs}, \mathsf{y}, \pi_3) = 0 \textbf{ abort} \\
(c_4, \pi_4) \leftarrow \mathsf{VWER.EncR}((\widehat{vk}, m_B), pdk) \\
aP_4 := (c_4, \pi_4) \\
pdk' \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{NAttest}}(aP_4) \\
\textbf{if } Q[m_B] = \bot \\
\quad b_0 := (pek, pdk') \in \mathsf{R} \\
\quad \textbf{if } b_0 \textbf{ abort} \\
\textbf{else} \\
\quad \tau \leftarrow Q[m_B] \\
\quad \mathsf{w}_2 \leftarrow \mathsf{WES.Dec}(\tau, c_3) \\
\quad \mathsf{w}_1 := \mathsf{w}_2 - \mathsf{w}_r \\
\quad \sigma_M \leftarrow \mathsf{ADP.Adapt}(\widetilde{\sigma}, \mathsf{w}_1) \\
\quad b_1 := \widehat{\mathsf{DS}}.\widehat{\mathsf{Vf}}(\widehat{vk}, m_B, \tau) = 1 \\
\quad b_2 := \mathsf{Vf}(vk_M, m_M, \sigma_M) = 0 \\
\textbf{return } b_0 \vee (b_1 \wedge b_2)
\end{array}
$$

Figure 21: Seller security game, identical to ExpS^{G_0}, except for the highlighted grey lines. If condition b_0 is satisfied, the game aborts. The oracle is the same as in Fig. 20.

Game ExpS^{G_1}: This game, formally defined in Fig. 21, works exactly as G_0 but with the highlighted grey line. If condition b_0 is satisfied, the game aborts.

Game ExpS^{G_2}: This game, formally defined in Fig. 22, works exactly as G_1 but with the highlighted grey line. If $(\mathsf{X}_2, \mathsf{w}_2) \notin \mathsf{R}$ the game aborts.

ExpS^{G_2}

$Q := []$
$(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{KGen}}(1^\lambda)$
$(pek, pdk) \leftarrow \mathsf{createR}(1^\lambda)$
$(\overline{ek}, vk_M, m_B, m_M, rP_1) \leftarrow \mathcal{A}^{\mathsf{ONAttest}}(\widehat{vk}, pek)$
$(\widetilde{\sigma}, c_1, \pi_1, \mathsf{X}_1) \leftharpoondown rP_1$
$\mathsf{y} := (c_1, \overline{ek}, \mathsf{X}_1)$
if $\mathsf{NIZK.Vf}_{\mathcal{L}_1}(\mathsf{crs}, \mathsf{y}, \pi_1) = 0$ **abort**
if $\mathsf{ADP.PreVf}(vk_M, m_M, \mathsf{X}_1, \widetilde{\sigma}) = 0$ **abort**
$(\mathsf{X}_r, \mathsf{w}_r) \leftarrow \mathsf{createR}(1^\lambda)$
$\mathsf{X}_2 := \mathsf{X}_r \otimes \mathsf{X}_1$
$c_r \leftarrow \mathsf{LHE.Enc}(\overline{ek}_M, \mathsf{w}_r)$
$c_2 := c_1 \circ c_r$
$rP_2 := (c_2, \mathsf{X}_2)$
$st_S := (\mathsf{X}_2, \mathsf{X}_r, \mathsf{w}_r)$
$aP_3 \leftarrow \mathcal{A}^{\mathsf{ONAttest}}(rP_2)$
$(c_3, \pi_3) \leftharpoondown aP_3$
$\mathsf{y} := (c_3, \widehat{vk}, m_B, \mathsf{X}_2)$
if $\mathsf{NIZK.Vf}_{\mathcal{L}_2}(\mathsf{crs}, \mathsf{y}, \pi_3) = 0$ **abort**
$(c_4, \pi_4) \leftarrow \mathsf{VWER.EncR}((\widehat{vk}, m_B), pdk)$
$aP_4 := (c_4, \pi_4)$
$pdk' \leftarrow \mathcal{A}^{\mathsf{ONAttest}}(aP_4)$
if $Q[m_B] = \bot$
  $b_0 := (pek, pdk') \in \mathsf{R}$
  if b_0 **abort**
else
  $\tau \leftarrow Q[m_B]$
  $\mathsf{w}_2 \leftarrow \mathsf{WES.Dec}(\tau, c_3)$
  if $(\mathsf{X}_2, \mathsf{w}_2) \notin \mathsf{R}$ **abort**
  $\mathsf{w}_1 := \mathsf{w}_2 - \mathsf{w}_r$
  $\sigma_M \leftarrow \mathsf{ADP.Adapt}(\widetilde{\sigma}, \mathsf{w}_1)$
  $b_1 := \widehat{\mathsf{DS}}.\widehat{\mathsf{Vf}}(\widehat{vk}, m_B, \tau) = 1$
  $b_2 := \mathsf{Vf}(vk_M, m_M, \sigma_M) = 0$
return $b_0 \vee (b_1 \wedge b_2)$

Figure 22: Seller security game, identical to ExpS^{G_1}, except for the highlighted grey lines. If w_1 is not the R of X_1, the game aborts.The oracle is the same as in Fig. 20.

ExpS^{G_3}

$Q := []$
$(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{KGen}}(1^\lambda)$
$(pek, pdk) \leftarrow \mathsf{createR}(1^\lambda)$
$(\overline{ek}, vk_M, m_B, m_M, rP_1) \leftarrow \mathcal{A}^{\mathsf{ONAttest}}(\widehat{vk}, pek)$
$(\widetilde{\sigma}, c_1, \pi_1, \mathsf{X}_1) \leftharpoondown rP_1$
$\mathsf{y} := (c_1, \overline{ek}, \mathsf{X}_1)$
if $\mathsf{NIZK.Vf}_{\mathcal{L}_1}(\mathsf{crs}, \mathsf{y}, \pi_1) = 0$ **abort**
if $\mathsf{ADP.PreVf}(vk_M, m_M, \mathsf{X}_1, \widetilde{\sigma}) = 0$ **abort**
$(\mathsf{X}_r, \mathsf{w}_r) \leftarrow \mathsf{createR}(1^\lambda)$
$\mathsf{X}_2 := \mathsf{X}_r \otimes \mathsf{X}_1$
$c_r \leftarrow \mathsf{LHE.Enc}(\overline{ek}_M, \mathsf{w}_r)$
$c_2 := c_1 \circ c_r$
$rP_2 := (c_2, \mathsf{X}_2)$
$st_S := (\mathsf{X}_2, \mathsf{X}_r, \mathsf{w}_r)$
$aP_3 \leftarrow \mathcal{A}^{\mathsf{ONAttest}}(rP_2)$
$(c_3, \pi_3) \leftharpoondown aP_3$
$\mathsf{y} := (c_3, \widehat{vk}, m_B, \mathsf{X}_2)$
if $\mathsf{NIZK.Vf}_{\mathcal{L}_2}(\mathsf{crs}, \mathsf{y}, \pi_3) = 0$ **abort**
$(c_4, \pi_4) \leftarrow \mathsf{VWER.EncR}((\widehat{vk}, m_B), pdk)$
$aP_4 := (c_4, \pi_4)$
$pdk' \leftarrow \mathcal{A}^{\mathsf{ONAttest}}(aP_4)$
if $Q[m_B] = \bot$
  $b_0 := (pek, pdk') \in \mathsf{R}$
  if b_0 **abort**
else
  $\tau \leftarrow Q[m_B]$
  $\mathsf{w}_2 \leftarrow \mathsf{WES.Dec}(\tau, c_3)$
  if $(\mathsf{X}_2, \mathsf{w}_2) \notin \mathsf{R}$ **abort**
  $\mathsf{w}_1 := \mathsf{w}_2 - \mathsf{w}_r$
  if $(\mathsf{X}_1, \mathsf{w}_1) \notin \mathsf{R}$ **abort**
  $\sigma_M \leftarrow \mathsf{ADP.Adapt}(\widetilde{\sigma}, \mathsf{w}_1)$
  $b_1 := \widehat{\mathsf{DS}}.\widehat{\mathsf{Vf}}(\widehat{vk}, m_B, \tau) = 1$
  $b_2 := \mathsf{Vf}(vk_M, m_M, \sigma_M) = 0$
return $b_0 \vee (b_1 \wedge b_2)$

Figure 23: Seller security game, identical to ExpS^{G_1}, except for the highlighted grey lines. If w_3 is not the R of X_3, the game aborts.The oracle is the same as in Fig. 20.

Game ExpS^{G_3}: This game, formally defined in Fig. 23, works exactly as G_2 but with the highlighted grey line. If $(\mathsf{X}_1, \mathsf{w}_1) \notin \mathsf{R}$ the game aborts.

Claim 6. *Let* Bad_1 *be the event that* ExpS^{G_1} *aborts because* b_0 *is satisfied. Assume that the VWER is one way. Then* $\Pr[\mathsf{Bad}_1(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{Bad}_1(1^\lambda)] > \mathsf{negl}(\lambda)$. We can construct adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break one wayness of VWER with the following steps:

- $\mathcal{B}$ initializes the challenger of $\mathsf{ExpOW}_{\mathcal{A}}$ game and obtains $\widehat{vk}$ and pek.
- $\mathcal{B}$ invokes $\mathcal{A}$ on input $\widehat{vk}$ and pek to obtain $(\overline{ek}, vk_M, m_B, m_M, rP_1)$. $\mathcal{B}$ parses rP_1 as $(\widetilde{\sigma}, c_1, \pi_1, \mathsf{X}_1)$ and sets $\mathsf{y} := (c_1, \overline{ek}, \mathsf{X}_1)$.

- $\mathcal{B}$ checks the NIZK and the presignature. Since we assume that $\mathcal{A}$ aborts on b_0 and has all the information required to generate valid proofs, $\mathcal{B}$ does not abort here.
- $\mathcal{B}$ runs $(\mathsf{X}_r, \mathsf{w}_r) \leftarrow \mathsf{createR}(1^\lambda)$, $\mathsf{X}_2 := \mathsf{X}_2 \otimes \mathsf{X}_1$, $c_r \leftarrow \mathsf{LHE.Enc}(\overline{ek}_M, \mathsf{w}_r)$ and $c_2 := c_r \circ c_2$ and then sets $rP_2 := (c_2, \mathsf{X}_2)$ and $st_S := (\mathsf{X}_2, \mathsf{X}_r, \mathsf{w}_r)$.
- $\mathcal{B}$ invokes $\mathcal{A}$ on input rP_2 to obtain aP_3, which $\mathcal{B}$ parses to obtain (c_3, π_3).
- $\mathcal{B}$ checks the NIZK. Since we assume that $\mathcal{A}$ aborts on b_0 and has all the information required to generate valid proofs, $\mathcal{B}$ does not abort here.
- $\mathcal{B}$ queries $\mathcal{O}\mathsf{EncR}$ on message m_B to obtain (c_4, π_4), which is assigned as aP_4.
- $\mathcal{B}$ invokes $\mathcal{A}$ on input aP_4 to obtain pdk', which is forwarded to the challenger.

Regarding $\mathcal{O}\mathsf{NAttest}$, $\mathcal{B}$ does not know $\widehat{sk}$ and cannot run $\widehat{\mathsf{DS}}.\widehat{\mathsf{Sig}}$. Therefore, $\mathcal{B}$ forwards these queries to $\mathcal{O}\widehat{\mathsf{Sig}}$. Note that this ensures that the messages queried in both oracles are the same.

Our adversary $\mathcal{B}$ perfectly simulates ExpS^{G_1} to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ is successful in aborting in b_0, this means that m_B is not on the memory of Q, which implies that it is also not in the memory of $\mathcal{O}\widehat{\mathsf{Sig}}$. In addition, only m_B is on the memory of $\mathcal{O}\mathsf{EncR}$. This ensures that the intersection of the two memories is an empty set. Note that condition b_0 is equivalent to condition b_1 of $\mathsf{ExpOW}_{\mathcal{A}}$. Since our assumption is that $\mathcal{A}$ aborts with no negligible probability, this means that $\mathcal{B}$ wins $\mathsf{ExpOW}_{\mathcal{A}}$ with the same probability. However, this contradicts our assumption that VWER is one way, so this adversary does not exist. This claim has been proven and we can conclude that $\mathsf{ExpS}^{G_0} \approx \mathsf{ExpS}^{G_1}$

□

Claim 7. *Let* Bad_2 *be the event that* ExpS^{G_2} *aborts because* $(\mathsf{X}_2, \mathsf{w}_2) \notin \mathsf{R}$. *Assume that the NIZK for* $\mathcal{L}_2$ *is secure under knowledge-soundness. Then* $\Pr[\mathsf{Bad}_2(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{Bad}_2(1^\lambda)] > \mathsf{negl}(\lambda)$. We can construct adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break break knowledge-soundness of NIZK for $\mathcal{L}_2$ with the following steps:

- $\mathcal{B}$ initializes the challenger who sets the crs.
- $\mathcal{B}$ runs $(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{KGen}}(1^\lambda)$ and $(pek, pdk) \leftarrow \mathsf{createR}(1^\lambda)$.
- $\mathcal{B}$ invokes $\mathcal{A}$ on input $\widehat{vk}$ and pek to obtain $(\overline{ek}, vk_M, m_B, m_M, rP_1)$. $\mathcal{B}$ parses rP_1 as $(\widetilde{\sigma}, c_1, \pi_1, \mathsf{X}_1)$ and sets $\mathsf{y} := (c_1, \overline{ek}, \mathsf{X}_1)$.
- $\mathcal{B}$ checks the NIZK and the presignature. Since we assume that $\mathcal{A}$ aborts on the highlighted grey line and has all the information required to generate valid proofs, $\mathcal{B}$ does not abort here.
- $\mathcal{B}$ runs $(\mathsf{X}_r, \mathsf{w}_r) \leftarrow \mathsf{createR}(1^\lambda)$, $\mathsf{X}_2 := \mathsf{X}_r \otimes \mathsf{X}_1$, $c_r \leftarrow \mathsf{LHE.Enc}(\overline{ek}_M, \mathsf{w}_r)$ and $c_2 := c_r \circ c_2$ and then sets $rP_2 := (c_2, \mathsf{X}_2)$ and $st_S := (\mathsf{X}_2, \mathsf{X}_r, \mathsf{w}_r)$.
- $\mathcal{B}$ invokes $\mathcal{A}$ on input rP_2 to obtain aP_3, which $\mathcal{B}$ parses to obtain (c_3, π_3).
- $\mathcal{B}$ checks the NIZK. Since we assume that $\mathcal{A}$ aborts on b_0 and has all the information required to generate valid proofs, $\mathcal{B}$ does not abort here.
- $\mathcal{B}$ runs $(c_4, \pi_4) \leftarrow \mathsf{VWER.EncR}((\widehat{vk}, m_B), pdk)$, which is assigned as aP_4.
- $\mathcal{B}$ invokes $\mathcal{A}$ on input aP_4 to obtain pdk'.
- $\mathcal{B}$ extracts τ from $Q[m_B]$ and uses it to decrypt c_3 and obtain w_2.
- $\mathcal{B}$ forwards c_3, $\widehat{vk}$, m_B, X_2 and π_3 to the challenger.

Regarding $\mathcal{O}\mathsf{NAttest}$, $\mathcal{B}$ has all the information required to simulate it to $\mathcal{A}$. Our adversary $\mathcal{B}$ perfectly simulates ExpS^{G_2} to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ makes the challenger abort on the grey line with non-negligible probability, this means that the zero knowledge proof for $\mathcal{L}2$ was done for $((c_3, \widehat{vk}, m_B, \mathsf{X}_2), \mathsf{w}_2) \notin \mathcal{L}2$, while $\mathsf{NIZK.Vf}_{\mathcal{L}_2}(\mathsf{crs}, (c_3, \widehat{vk}, m_B, \mathsf{X}_2), \pi_3) = 1$. However, this contradicts our assumption that the NIZK used for $\mathcal{L}2$ is knowledge sound, so this adversary does not exist. This claim has been proven and we can conclude that $\mathsf{ExpS}^{G_1} \approx \mathsf{ExpS}^{G_2}$. □

Claim 8. *Let* Bad_3 *be the event that* ExpS^{G_3} *aborts because* $(\mathsf{X}_1, \mathsf{w}_1) \notin \mathsf{R}$. *Assume that the NIZK for* $\mathcal{L}1$ *is secure under knowledge-soundness. Then* $\Pr[\mathsf{Bad}_3(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{Bad}_3(1^\lambda)] > \mathsf{negl}(\lambda)$. We can construct adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break Knowledge soundness of NIZK for $\mathcal{L}1$ with the following steps:

- $\mathcal{B}$ initializes the challenger who sets the crs.
- $\mathcal{B}$ runs $(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{KGen}}(1^\lambda)$ and $(pek, pdk) \leftarrow \mathsf{createR}(1^\lambda)$.
- $\mathcal{B}$ invokes $\mathcal{A}$ on input $\widehat{vk}$ and pek to obtain $(\overline{ek}, vk_M, m_B, m_M, rP_1)$. $\mathcal{B}$ parses rP_1 as $(\widetilde{\sigma}, c_1, \pi_1, \mathsf{X}_1)$ and assigns $\mathsf{y} := (c_1, \overline{ek}, \mathsf{X}_1)$.
- $\mathcal{B}$ checks the NIZK for $\mathcal{L}1$ and the presignature. Since we assume that $\mathcal{A}$ aborts on the highlighted grey line and has all the information required to generate valid proofs, $\mathcal{B}$ does not abort here.
- $\mathcal{B}$ runs $(\mathsf{X}_r, \mathsf{w}_r) \leftarrow \mathsf{createR}(1^\lambda)$, $\mathsf{X}_2 := \mathsf{X}_r \otimes \mathsf{X}_1$, $c_r \leftarrow \mathsf{LHE.Enc}(\overline{ek}_M, \mathsf{w}_r)$ and $c_2 := c_r \circ c_1$ and then sets $rP_2 := (c_2, \mathsf{X}_2)$ and $st_S := (\mathsf{X}_2, \mathsf{X}_r, \mathsf{w}_r)$.
- $\mathcal{B}$ invokes $\mathcal{A}$ on input rP_2 to obtain aP_3, which $\mathcal{B}$ parses to obtain (c_3, π_3).
- $\mathcal{B}$ checks the NIZK for $\mathcal{L}2$. Since we assume that $\mathcal{A}$ aborts on the grey line and has all the information required to generate valid proofs, $\mathcal{B}$ does not abort here.
- $\mathcal{B}$ runs $(c_4, \pi_4) \leftarrow \mathsf{VWER.EncR}((\widehat{vk}, m_B), pdk)$, which is assigned as aP_4.
- $\mathcal{B}$ invokes $\mathcal{A}$ on input aP_4 to obtain pdk'.
- $\mathcal{B}$ extracts τ from the memory using m_B as key. Uses τ as decryption key for c_3 to obtain w_2. Since we assume that $\mathcal{B}$ aborts because $(\mathsf{X}_1, \mathsf{w}_1) \notin \mathsf{R}$, this means that $(\mathsf{X}_2, \mathsf{w}_2) \in \mathsf{R}$.
- $\mathcal{B}$ computes $\mathsf{w}_1 := \mathsf{w}_2 - \mathsf{w}_r$. Note that w_1 is the same as $\mathcal{A}$ encrypted in c_1 for the same reasons as outlined in Theorem 5 since X_2, c_2 and X_r are created by $\mathcal{B}$ honestly.
- $\mathcal{B}$ forwards c_1, $\overline{ek}$, X_1 and π_1 to the challenger.

Regarding $\mathcal{O}\mathsf{NAttest}$, $\mathcal{B}$ has all the information required to simulate it to $\mathcal{A}$. Our adversary $\mathcal{B}$ perfectly simulates ExpS^{G_3} to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ makes the challenger abort on the grey line with non-negligible probability, this means that the zero knowledge proof for $\mathcal{L}1$ was done for $((c_1, \overline{ek}, \mathsf{X}_1), \mathsf{w}_1) \notin \mathcal{L}1$, while $\mathsf{NIZK.Vf}_{\mathcal{L}_1}(\mathsf{crs}, (c_1, \overline{ek}, \mathsf{X}_1), \pi_1) = 1$. However, this contradicts our assumption that the NIZK used for $\mathcal{L}1$ is knowledge sound, so this adversary does not exist. This claim has been proven and we can conclude that $\mathsf{ExpS}^{G_2} \approx \mathsf{ExpS}^{G_3}$. □

Claim 9. *Assume that the adaptor signature scheme is secure under adaptability. Then* $\Pr[\mathsf{ExpS}^{G_3}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{ExpS}^{G_3}(1^\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break adaptability of the adaptor signature scheme with the following steps:

- $\mathcal{B}$ runs $(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{KGen}}(1^\lambda)$ and $(pek, pdk) \leftarrow \mathsf{createR}(1^\lambda)$.
- $\mathcal{B}$ invokes $\mathcal{A}$ on input $\widehat{vk}$ and pek to obtain $(\overline{ek}, vk_M, m_B, m_M, rP_1)$. $\mathcal{B}$ parses rP_1 as $(\widetilde{\sigma}, c_1, \pi_1, \mathsf{X}_1)$ and sets $\mathsf{y} := (c_1, \overline{ek}, \mathsf{X}_1)$.
- $\mathcal{B}$ checks the NIZK for $\mathcal{L}1$ and the presignature. Since we assume that $\mathcal{A}$ wins the game and has all the information required to generate valid proofs, $\mathcal{B}$ does not abort here.
- $\mathcal{B}$ runs $(\mathsf{X}_r, \mathsf{w}_r) \leftarrow \mathsf{createR}(1^\lambda)$, $\mathsf{X}_2 := \mathsf{X}_r \otimes \mathsf{X}_1$, $c_r \leftarrow \mathsf{LHE.Enc}(\overline{ek}_M, \mathsf{w}_r)$ and $c_2 := c_r \circ c_1$ and then sets $rP_2 := (c_2, \mathsf{X}_2)$ and $st_S := (\mathsf{X}_2, \mathsf{X}_r, \mathsf{w}_r)$.
- $\mathcal{B}$ invokes $\mathcal{A}$ on input rP_2 to obtain aP_3, which $\mathcal{B}$ parses to obtain (c_3, π_3).
- $\mathcal{B}$ checks the NIZK for $\mathcal{L}2$. Since we assume that $\mathcal{A}$ wins the game and has all the information required to generate valid proofs, $\mathcal{B}$ does not abort here.
- $\mathcal{B}$ runs $(c_4, \pi_4) \leftarrow \mathsf{VWER.EncR}((\widehat{vk}, m_B), pdk)$, which is assigned as aP_4.
- $\mathcal{B}$ invokes $\mathcal{A}$ on input aP_4 to obtain pdk'.
- $\mathcal{B}$ extracts τ from the memory using m_B as key. Uses τ as decryption key for c_3 to obtain w_2
- $\mathcal{B}$ obtains w_1 using w_r and w_2.
- $\mathcal{B}$ forwards $\mathsf{X}_1, \mathsf{w}_1, \widetilde{\sigma}, m_M, vk_M$ to the challenger.

Regarding $\mathcal{O}\mathsf{NAttest}$, $\mathcal{B}$ has all the information required to simulate it to $\mathcal{A}$. Our adversary $\mathcal{B}$ perfectly simulates ExpS^{G_3} to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, the presignature is valid, as otherwise $\mathcal{B}$ would have aborted. However, since $\mathcal{A}$ wins the game with non-negligible probability, this means that $\mathsf{Adapt}(\widetilde{\sigma}, \mathsf{w}_1)$ produces a signature that does not verify for m_M and vk_M. Therefore, if $\mathcal{B}$ forwards $\mathsf{X}_1, \mathsf{w}_1, \widetilde{\sigma}, m_M, vk_M$ to the challenger, this wins the adaptability game with non-negligible probability. However, this contradicts our assumption that the adaptor signature scheme guarantees adaptability, so this adversary does not exist. This claim has been proven. □

We have proved that $\mathsf{ExpS}^{G_0} \approx \mathsf{ExpS}^{G_3}$ and that $\Pr[\mathsf{ExpS}^{G_3}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$. Therefore, Theorem 2 has been proven. □

Theorem 3 (Buyer Security). *Assume the signature scheme is EUF-CMA and VWER provides VWER verifiability. Then, our construction offers buyer security according to Definition 5.*

Proof. We consider the following game hops:

Game ExpB^{G_0}: This game, formally defined in Fig. 24, corresponds to the original game for ExpB defined in Definition 5. The game is expanded with the interactions described in our implementation.

$$
\begin{array}{l}
\underline{\mathsf{ExpB}^{G_0}} \\
Q := [\,] \\
(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{KGen}}(1^\lambda) \\
(vk_B, sk_B) \leftarrow \mathsf{KGen}(1^\lambda) \\
(\sigma_B^*, pek, m_B, aP_4) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{SigNAttest}}(vk_B, \widehat{vk}) \\
\textbf{if } Q[m_B] = \bot \\
\quad b_0 := (\mathsf{Vf}(vk_B, m_B, \sigma_B^*) = 1) \\
\textbf{else} \\
\quad \tau \leftarrow Q[m_B] \\
\quad (c_4, \pi_4) \leftarrow aP_4 \\
\quad pdk := \mathsf{VWER.DecR}(\tau, c_4, \pi_4) \\
\quad b_1 := \mathsf{VWER.VfEncR}(c_4, \pi_4, (\widehat{vk}, m_B), pek) \\
\quad b_2 := \widehat{\mathsf{DS}}.\widehat{\mathsf{Vf}}(\widehat{vk}, m_B, \tau) \\
\quad b_3 := (pek, pdk) \notin \mathsf{R} \\
\textbf{return } b_0 \vee (b_1 \wedge b_2 \wedge b_3) \\
\\
\underline{\mathcal{O}\mathsf{SigNAttest}(m)} \\
\sigma_B \leftarrow \mathsf{Sig}(sk_B, m) \\
\tau \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{Sig}}(\widehat{sk}, m_B) \\
Q[m] := \tau \\
\textbf{return } (\sigma_B, \tau)
\end{array}
$$

Figure 24: Buyer security expanded with the interactions described in our implementation.

Game ExpB^{G_1}: This game, formally defined in Fig. 25, works exactly as G_0 but with highlighted grey line. If the adversary satisfies condition b_0, the game aborts.

Claim 10. *Let* Bad_1 *be the event that* ExpB^{G_1} *aborts on the highlighted grey line. Assume that the digital signature scheme is unforgeable. Then* $\Pr[\mathsf{Bad}_1(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that there is a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{Bad}_1(1^\lambda)] > \mathsf{negl}(\lambda)$, then we can construct a PPT adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break unforgeability of digital signature with the following steps:

- $\mathcal{B}$ receives vk_B from challenger.
- $\mathcal{B}$ runs $(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{KGen}}(1^\lambda)$.
- $\mathcal{B}$ invokes $\mathcal{A}$ on input $\widehat{vk}$ and vk_B to obtain a $(\sigma_B^*, pek, m_B, aP_4)$.
- $\mathcal{B}$ forwards σ_B^* and m_B to the challenger.

To simulate $\mathcal{O}\mathsf{SigNAttest}$, $\mathcal{B}$ needs to invoke oracle $\mathcal{O}\mathsf{Sig}$ of the EUF-CMA challenger. This ensures that Q and the memory of EUF-CMA are synchronized. For the other signature, $\widehat{\mathsf{DS}}.\widehat{\mathsf{Sig}}$, $\mathcal{B}$ can generate the τ locally.

Our adversary $\mathcal{B}$ perfectly simulates ExpB^{G_1} to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ has $\Pr[\mathsf{Bad}_1(1^\lambda)] > \mathsf{negl}(\lambda)$, this means that $\mathsf{Vf}(vk_B, m_B, \sigma_B^*) = 1$ and that m_B has not been queried in $\mathcal{O}\mathsf{SigNAttest}$. Since the memories of oracles

ExpB^{G_1}

$Q := []$
$(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{KGen}}(1^\lambda)$
$(vk_B, sk_B) \leftarrow \mathsf{KGen}(1^\lambda)$
$(\sigma_B^*, pek, m_B, aP_4) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{SigNAttest}}(vk_B, \widehat{vk})$
if $Q[m_B] = \bot$
 $b_0 := (\mathsf{Vf}(vk_B, m_B, \sigma_B^*) = 1)$
 if b_0 **abort**
else
 $\tau \leftarrow Q[m_B]$
 $(c_4, \pi_4) \leftharpoondown aP_4$
 $pdk := \mathsf{VWER.DecR}(\tau, c_4, \pi_4)$
 $b_1 := \mathsf{VWER.VfEncR}(c_4, \pi_4, (\widehat{vk}, m_B), pek)$
 $b_2 := \widehat{\mathsf{DS}}.\widehat{\mathsf{Vf}}(\widehat{vk}, m_B, \tau)$
 $b_3 := (pek, pdk) \notin R$
return $b_0 \vee (b_1 \wedge b_2 \wedge b_3)$

$\mathcal{O}\mathsf{SigNAttest}(m)$

$\sigma_B \leftarrow \mathsf{Sig}(sk_B, m)$
$\tau \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{Sig}}(\widehat{sk}, m_B)$
$Q[m] := \tau$
return (σ_B, τ)

Figure 25: Buyer security game, identical to ExpB^{G_0}, except for the highlighted grey line. If condition b_0 is satisfied, the game aborts.

$\mathcal{O}$SigNAttest and $\mathcal{O}$Sig are synchronized, these two conditions are equivalent to the wining conditions of EUF-CMA game. However, this contradicts our assumption that the signature scheme is EUF-CMA secure, so $\mathcal{A}$ does not exist and this claim has been proven. We can conclude that $\mathsf{ExpB}^{G_0} \approx \mathsf{ExpB}^{G_1}$ □

Claim 11. *Assume that VWER satisfies VWER Verifiability. Then* $\Pr[\mathsf{ExpB}^{G_1}(1^\lambda) = 1] \leq \mathsf{negl}$.

Proof. Assume by contradiction that there is a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{ExpB}^{G_1}(1^\lambda) = 1] > \mathsf{negl}(\lambda)$, then we can construct a PPT adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break VWER Verifiability of VWER with the following steps:

- $\mathcal{B}$ runs $(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{DS}}.\widehat{\mathsf{KGen}}(1^\lambda)$.
- $\mathcal{B}$ runs $(vk_B, sk_B) \leftarrow \mathsf{KGen}(1^\lambda)$.
- $\mathcal{B}$ invokes $\mathcal{A}$ on input $\widehat{vk}$ and vk_B to obtain a $(\sigma_B^*, pek, m_B, aP_4)$.
- $\mathcal{B}$ extracts (c_4, π_4) from aP_4 and τ from $Q[m_B]$.
- $\mathcal{B}$ runs $pdk \leftarrow \mathsf{VWER.DecR}(\tau, c_4, \pi_4)$.
- $\mathcal{B}$ forwards $(m_B, \widehat{vk}, \tau, c_4, \pi_4, pek)$ to the challenger.

To simulate $\mathcal{O}$SigNAttest, $\mathcal{B}$ uses $\widehat{sk}$ and sk_B.

Our adversary $\mathcal{B}$ perfectly simulates ExpB^{G_1} to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ has $\Pr[\mathsf{ExpB}^{G_1}(1^\lambda) = 1] > \mathsf{negl}(\lambda)$, this means that $\mathsf{VWER.VfEncR}(c_4, \pi_4, (\widehat{vk}, m_B), pek) = 1$, $\widehat{\mathsf{Vf}}(\widehat{vk}, m_B, \tau) = 1$ and $(pek, pdk) \notin R$. Note that these three conditions are the same conditions as those in $\mathsf{ExpVer}_{\mathcal{A}}$, therefore, winning ExpB^{G_1} with no negligible probability implies winning $\mathsf{ExpVer}_{\mathcal{A}}$ also with no negligible proability. However, this contradicts our assumption that the VWER achieves VWER verifiability, so $\mathcal{A}$ does not exist and this claim has been proven. □

$\mathsf{ExpLink}^{G_0}$

$(vk_B^0, sk_B^0) \leftarrow \mathsf{KGen}(1^\lambda)$; $(vk_B^1, sk_B^1) \leftarrow \mathsf{KGen}(1^\lambda)$
$(\overline{ek}, \widehat{vk}, vk_M^0, vk_M^1, rP_1^0, rP_1^1, (m_M^0, m_B^0), (m_M^1, m_B^1)) \leftarrow \mathcal{A}(vk_B^0, vk_B^1)$
$b \leftarrow \{0, 1\}$
$(\widetilde{\sigma}^0, c_1^0, \pi_1^0, \mathsf{X}_1^0) \leftharpoondown rP_1^0$; $(\widetilde{\sigma}^1, c_1^1, \pi_1^1, \mathsf{X}_1^1) \leftharpoondown rP_1^1$
if $\mathsf{NIZK.Vf}_{\mathcal{L}_1}(\mathsf{crs}, (c_1^0, \overline{ek}, \mathsf{X}_1^0), \pi_1^0) = 0$ **abort**
if $\mathsf{NIZK.Vf}_{\mathcal{L}_1}(\mathsf{crs}, (c_1^1, \overline{ek}, \mathsf{X}_1^1), \pi_1^1) = 0$ **abort**
if $\mathsf{ADP.PreVf}(vk_M^0, m_M^0, \mathsf{X}_1^0, \widetilde{\sigma}^0) = 0$ **abort**
if $\mathsf{ADP.PreVf}(vk_M^1, m_M^1, \mathsf{X}_1^1, \widetilde{\sigma}^1) = 0$ **abort**
$(\mathsf{X}_r^0, \mathsf{w}_r^0) \leftarrow \mathsf{createR}(1^\lambda)$; $(\mathsf{X}_r^1, \mathsf{w}_r^1) \leftarrow \mathsf{createR}(1^\lambda)$
$\mathsf{X}_2^0 := \mathsf{X}_r^0 \otimes \mathsf{X}_1^0$; $\mathsf{X}_2^1 := \mathsf{X}_r^1 \otimes \mathsf{X}_1^1$
$c_r^0 \leftarrow \mathsf{Enc}(\overline{ek}_M, \mathsf{w}_r^0)$; $c_r^1 \leftarrow \mathsf{Enc}(\overline{ek}_M, \mathsf{w}_r^1)$
$c_2^0 := c_1^0 \circ c_2^0$; $c_2^1 := c_1^1 \circ c_2^1$
$rP_2^0 := (c_2^0, \mathsf{X}_2^0)$; $rP_2^1 := (c_2^1, \mathsf{X}_2^1)$
$st_S^0 := (\mathsf{X}_2^0, \mathsf{X}_r^0, \mathsf{w}_r^0)$; $st_S^1 := (\mathsf{X}_2^1, \mathsf{X}_r^1, \mathsf{w}_r^1)$
$(aP_3^0, aP_3^1) \leftarrow \mathcal{A}(rP_2^{0\oplus b}, rP_2^{1\oplus b})$
$(c_3^0, \pi_3^0) \leftharpoondown aP_3^0$; $(c_3^1, \pi_3^1) \leftharpoondown aP_3^1$
if $\mathsf{NIZK.Vf}_{\mathcal{L}_2}(\mathsf{crs}, (c_3^0, \widehat{vk}, m_B^0, \mathsf{X}_2^{0\oplus b}, \pi_3^{0\oplus b}) = 0)$ **abort**
if $\mathsf{NIZK.Vf}_{\mathcal{L}_2}(\mathsf{crs}, (c_3^1, \widehat{vk}, m_B^1, \mathsf{X}_2^{1\oplus b}, \pi_3^{1\oplus b}) = 0$ **abort**
$\sigma_B^0 \leftarrow \mathsf{Sig}(sk_B^0, m_B^0)$; $\sigma_B^1 \leftarrow \mathsf{Sig}(sk_B^1, m_B^1)$
$(\tau^0, \tau^1) \leftarrow \mathcal{A}(\sigma_B^0, \sigma_B^1)$
$\mathsf{w}_2^{0\oplus b} \leftarrow \mathsf{WES.Dec}(\tau^0, c_3^0)$; $\mathsf{w}_2^{1\oplus b} \leftarrow \mathsf{WES.Dec}(\tau^1, c_3^1)$
$\mathsf{w}_1^{0\oplus b} := \mathsf{w}_2^{0\oplus b} - \mathsf{w}_r^{0\oplus b}$; $\mathsf{w}_1^{1\oplus b} := \mathsf{w}_2^{1\oplus b} - \mathsf{w}_r^{1\oplus b}$
$\sigma_M^{0\oplus b} \leftarrow \mathsf{ADP.Adapt}(\widetilde{\sigma}^{0\oplus b}, \mathsf{w}_1^{0\oplus b})$
$\sigma_M^{1\oplus b} \leftarrow \mathsf{ADP.Adapt}(\widetilde{\sigma}^{1\oplus b}, \mathsf{w}_1^{1\oplus b})$
if $(\mathsf{Vf}(vk_M^0, m_M^0, \sigma_M^0) = 0) \vee (\mathsf{Vf}(vk_M^1, m_M^1, \sigma_M^1) = 0)$
 $\sigma_M^0 = \sigma_M^1 = \bot$
$b' \leftarrow \mathcal{A}(\sigma_M^0, \sigma_M^1)$
return $(b = b')$

Figure 26: unlinkability property expanded with the interactions described in our implementation.

We have proved that $\mathsf{ExpB}^{G_0} \approx \mathsf{ExpB}^{G_1}$ and that $\Pr[\mathsf{ExpB}^{G_1}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$. Therefore, Theorem 3 has been proven. □

Theorem 4 (Unlinkability). *Assume that* createR *samples at random from a uniform distribution. Then, our construction offers unlinkability according to Definition 6.*

$\mathsf{ExpLink}^{G_1}$

$(vk_B^0, sk_B^0) \leftarrow \mathsf{KGen}(1^\lambda)$; $(vk_B^1, sk_B^1) \leftarrow \mathsf{KGen}(1^\lambda)$
$(\overline{ek}, \widehat{vk}, vk_M^0, vk_M^1, rP_1^0, rP_1^1, (m_M^0, m_B^0), (m_M^1, m_B^1)) \leftarrow \mathcal{A}(vk_B^0, vk_B^1)$
$b \leftarrow \{0,1\}$
$(\widetilde{\sigma}^0, c_1^0, \pi_1^0, \mathsf{X}_1^0) \leftharpoondown rP_1^0$; $(\widetilde{\sigma}^1, c_1^1, \pi_1^1, \mathsf{X}_1^1) \leftharpoondown rP_1^1$
if $\mathsf{NIZK.Vf}_{\mathcal{L}_1}(\mathsf{crs}, (c_1^0, \overline{ek}, \mathsf{X}_1^0), \pi_1^0) = 0$ **abort**
if $\mathsf{NIZK.Vf}_{\mathcal{L}_1}(\mathsf{crs}, (c_1^1, \overline{ek}, \mathsf{X}_1^1), \pi_1^1) = 0$ **abort**
if $\mathsf{ADP.PreVf}(vk_M^0, m_M^0, \mathsf{X}_1^0, \widetilde{\sigma}^0) = 0$ **abort**
if $\mathsf{ADP.PreVf}(vk_M^1, m_M^1, \mathsf{X}_1^1, \widetilde{\sigma}^1) = 0$ **abort**
$(\mathsf{X}_r^0, \mathsf{w}_r^0) \leftarrow \mathsf{createR}(1^\lambda)$; $(\mathsf{X}_r^1, \mathsf{w}_r^1) \leftarrow \mathsf{createR}(1^\lambda)$
$\mathsf{X}_2^0 := \mathsf{X}_r^0 \otimes \mathsf{X}_1^0$; $\mathsf{X}_2^1 := \mathsf{X}_r^1 \otimes \mathsf{X}_1^1$
$c_r^0 \leftarrow \mathsf{Enc}(\overline{ek}_M, \mathsf{w}_r^0)$; $c_r^1 \leftarrow \mathsf{Enc}(\overline{ek}_M, \mathsf{w}_r^1)$
$c_2^0 := c_1^0 \circ c_2^0$; $c_2^1 := c_1^1 \circ c_2^1$
$rP_2^0 := (c_2^0, \mathsf{X}_2^0)$; $rP_2^1 := (c_2^1, \mathsf{X}_2^1)$
$st_S^0 := (\mathsf{X}_2^0, \mathsf{X}_r^0, \mathsf{w}_r^0)$; $st_S^1 := (\mathsf{X}_2^1, \mathsf{X}_r^1, \mathsf{w}_r^1)$
$(aP_3^0, aP_3^1) \leftarrow \mathcal{A}(rP_2^{0\oplus b}, rP_2^{1\oplus b})$
$(c_3^0, \pi_3^0) \leftharpoondown aP_3^0$; $(c_3^1, \pi_3^1) \leftharpoondown aP_3^1$
if $\mathsf{NIZK.Vf}_{\mathcal{L}_2}(\mathsf{crs}, (c_3^0, \widehat{vk}, m_B^0, \mathsf{X}_2^{0\oplus b}, \pi_3^{0\oplus b}) = 0)$ **abort**
if $\mathsf{NIZK.Vf}_{\mathcal{L}_2}(\mathsf{crs}, (c_3^1, \widehat{vk}, m_B^1, \mathsf{X}_2^{1\oplus b}, \pi_3^{1\oplus b}) = 0$ **abort**
$\sigma_B^0 \leftarrow \mathsf{Sig}(sk_B^0, m_B^0)$; $\sigma_B^1 \leftarrow \mathsf{Sig}(sk_B^1, m_B^1)$
$b' \leftarrow \mathcal{A}(\sigma_B^0, \sigma_B^1)$
return $(b = b')$

Figure 27: unlinkability game, identical to $\mathsf{ExpLink}^{G_0}$, except for the highlighted grey lines: the adversary provides the bit after receiving the signatures from the buyer.

$\mathsf{ExpLink}^{G_2}$

$(vk_B^0, sk_B^0) \leftarrow \mathsf{KGen}(1^\lambda)$; $(vk_B^1, sk_B^1) \leftarrow \mathsf{KGen}(1^\lambda)$
$(\overline{ek}, \widehat{vk}, vk_M^0, vk_M^1, rP_1^0, rP_1^1, (m_M^0, m_B^0), (m_M^1, m_B^1)) \leftarrow \mathcal{A}(vk_B^0, vk_B^1)$
$b \leftarrow \{0,1\}$
$(\widetilde{\sigma}^0, c_1^0, \pi_1^0, \mathsf{X}_1^0) \leftharpoondown rP_1^0$; $(\widetilde{\sigma}^1, c_1^1, \pi_1^1, \mathsf{X}_1^1) \leftharpoondown rP_1^1$
if $\mathsf{NIZK.Vf}_{\mathcal{L}_1}(\mathsf{crs}, (c_1^0, \overline{ek}, \mathsf{X}_1^0), \pi_1^0) = 0$ **abort**
if $\mathsf{NIZK.Vf}_{\mathcal{L}_1}(\mathsf{crs}, (c_1^1, \overline{ek}, \mathsf{X}_1^1), \pi_1^1) = 0$ **abort**
if $\mathsf{ADP.PreVf}(vk_M^0, m_M^0, \mathsf{X}_1^0, \widetilde{\sigma}^0) = 0$ **abort**
if $\mathsf{ADP.PreVf}(vk_M^1, m_M^1, \mathsf{X}_1^1, \widetilde{\sigma}^1) = 0$ **abort**
$(\mathsf{X}_r^0, \mathsf{w}_r^0) \leftarrow \mathsf{createR}(1^\lambda)$; $(\mathsf{X}_r^1, \mathsf{w}_r^1) \leftarrow \mathsf{createR}(1^\lambda)$
$(\mathsf{X}_2^0, \mathsf{w}_2^0) \leftarrow \mathsf{createR}(1^\lambda)$; $(\mathsf{X}_2^1, \mathsf{w}_2^1) \leftarrow \mathsf{createR}(1^\lambda)$
$c_2^0 \leftarrow \mathsf{Enc}(\overline{ek}_M, \mathsf{w}_2^0)$; $c_2^1 \leftarrow \mathsf{Enc}(\overline{ek}_M, \mathsf{w}_2^1)$
$rP_2^0 := (c_2^0, \mathsf{X}_2^0)$; $rP_2^1 := (c_2^1, \mathsf{X}_2^1)$
$st_S^0 := (\mathsf{X}_2^0, \mathsf{X}_r^0, \mathsf{w}_r^0)$; $st_S^1 := (\mathsf{X}_2^1, \mathsf{X}_r^1, \mathsf{w}_r^1)$
$(aP_3^0, aP_3^1) \leftarrow \mathcal{A}(rP_2^{0\oplus b}, rP_2^{1\oplus b})$
$(c_3^0, \pi_3^0) \leftharpoondown aP_3^0$; $(c_3^1, \pi_3^1) \leftharpoondown aP_3^1$
if $\mathsf{NIZK.Vf}_{\mathcal{L}_2}(\mathsf{crs}, (c_3^0, \widehat{vk}, m_B^0, \mathsf{X}_2^{0\oplus b}, \pi_3^{0\oplus b}) = 0)$ **abort**
if $\mathsf{NIZK.Vf}_{\mathcal{L}_2}(\mathsf{crs}, (c_3^1, \widehat{vk}, m_B^1, \mathsf{X}_2^{1\oplus b}, \pi_3^{1\oplus b}) = 0$ **abort**
$\sigma_B^0 \leftarrow \mathsf{Sig}(sk_B^0, m_B^0)$; $\sigma_B^1 \leftarrow \mathsf{Sig}(sk_B^1, m_B^1)$
$b' \leftarrow \mathcal{A}(\sigma_B^0, \sigma_B^1)$
return $(b = b')$

Figure 28: unlinkability game, identical to $\mathsf{ExpLink}^{G_1}$, except for the highlighted grey lines. Instead of randomizing the ciphertext received by the adversary, the new plaintext that are encrypted and sent are sampled directly from a uniform distribution.

Proof. We consider the following game hops:

Game $\mathsf{ExpLink}^{G_0}$: This game, formally defined in Fig. 26, corresponds to the original game for unlinkability defined in Definition 6 The game is expanded with the interactions described in our implementation.

Game $\mathsf{ExpLink}^{G_1}$: This game, formally defined in Fig. 27, works exactly as G_0 but the adversary provides the bit after receiving the signatures from the buyer.

Game $\mathsf{ExpLink}^{G_2}$: This game, formally defined in Fig. 28, works exactly as G_1 but with highlighted grey lines. Instead of randomizing the ciphertexts with a randomly sampled witnesses, c_2 is directly calculated as the encryption of a randomly sampled element from a uniform distribution.

Claim 12. *Let* Bad_1 *be the event that:*

$$\left| \begin{array}{l} \Pr[\mathsf{ExpLink}^{G_0}(\lambda) = 1] \\ -\Pr[\mathsf{ExpLink}^{G_1}(\lambda) = 1] \end{array} \right| > \mathsf{negl}$$

Proof. The difference between the two games is that in $\mathsf{ExpLink}^{G_0}$ the challenger provides the pair (σ_M^0, σ_M^1) or $\perp$ to the adversary, while in $\mathsf{ExpLink}^{G_1}$, this information is not shared with the adversary. However, note that the adversary knows w_1^0 and w_1^1 and has generated the presignatures using X_1^0 and X_1^1. Therefore, in both games the adversary is able to generate on its own the same pair (σ_M^0, σ_M^1) that the challenger would have provided. Therefore, the adversary in $\mathsf{ExpLink}^{G_0}$ and $\mathsf{ExpLink}^{G_1}$ has the same information, so $\mathsf{ExpLink}^{G_0} \approx \mathsf{ExpLink}^{G_1}$. □

Claim 13. *Let* Bad_2 *be the event that:*

$$\left| \begin{array}{l} \Pr[\mathsf{ExpLink}^{G_1}(\lambda) = 1] \\ -\Pr[\mathsf{ExpLink}^{G_2}(\lambda) = 1] \end{array} \right| > \mathsf{negl}$$

Assume that createR *randomly samples from a uniform distribution. Then* $\Pr[\mathsf{Bad}_2(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. The difference between the two games is whether w_2 was randomly sampled from a uniform distribution or if it is w_1 masked with w_r, which is randomly sampled from the same uniform distribution. If we assume that createR samples from a uniform distribution, both instances are statistically indistinguishable. Therefore, $\Pr[\mathsf{Bad}_2(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$ and $\mathsf{ExpLink}^{G_1} \approx \mathsf{ExpLink}^{G_2}$. □

Since $\mathsf{ExpLink}^{G_0} \approx \mathsf{ExpLink}^{G_2}$, we only have left to quantify the probability of winning $\mathsf{ExpLink}^{G_2}$. The probability of winning $\mathsf{ExpLink}^{G_2}$ is equivalent to distinguish in which order to uniformly random elements were sampled from a uniform distribution. Therefore, $\Pr[\mathsf{ExpLink}^{G_2} = 1] \leq \frac{1}{/}2 + \mathsf{negl}(\lambda)$, which satisfies the unlinkability notion as defined in Definition 6. This concludes the proof for Theorem 4. □

C CONSTRUCTION AND SECURITY PROOFS OF VWER

Here we present a concrete construction of VWER encrypting the discrete logarithm of a group element. Our construction relies on the following cryptographic blocks:

- A digital signature scheme $\widehat{\mathsf{DS}} = (\widehat{\mathsf{KGen}}, \widehat{\mathsf{Sig}}, \widehat{\mathsf{Vf}})$ instantiated as the BLS digital signature scheme.
- A witness encryption based on signatures WES := (Enc, Dec) presented in [49].

We provide the details of the construction in Fig. 29. H denotes the random oracle used in The Fiat-Shamir heuristic, γ is the statistical parameter defining the numbers of ciphertexts required by the cut-and-choose techinque, S_{op} and S_{unop} denote the set of opened and unopened values outputted by algorithm EncR, respectively.

C.1 Correctness and Security Proofs

THEOREM 6. *Our VWER construction is correct according to Definition 12.*

PROOF. Let $(c, \pi) \leftarrow \mathsf{EncR}((\widehat{vk}, \widehat{m}), \mathsf{w})$. To prove correctness we first need to show that

$$\Pr[\mathsf{VfEncR}(c, \pi, (\widehat{vk}, \widehat{m}), \mathsf{X}) = 1] = 1$$

Note that algorithm VfEncR will output 0 if one of the following occurs.

(1) If $b_i = 1$ and $c_i \neq \mathsf{WES.Enc}((\widehat{vk}, \widehat{m}), r_i; r'_i)$. Provided the encryption is done correctly, this cannot occur.
(2) If $b_i = 0$ and $g^{s_i} \neq R_i \otimes \mathsf{X}$. By construction we have $s_i := r_i + \mathsf{w}$. This implies $g^{s_i} = g^{r_i} \otimes g^{\mathsf{w}} = R_i \otimes \mathsf{X}$ and therefore this case never occurs.

Next we need to show that if we have $\widehat{\mathsf{Vf}}(\widehat{vk}, \widehat{m}, \widehat{\sigma}) = 1$, then

$$\Pr[(\mathsf{X}, \mathsf{DecR}(\widehat{\sigma}, c, \pi)) \in \mathsf{R}] = 1$$

We are given that $\widehat{\mathsf{Vf}}(\widehat{vk}, \widehat{m}, \widehat{\sigma}) = 1$. For all $b_i = 0$ we have $r_i := \mathsf{WES.Dec}(\widehat{\sigma}, c_i)$. By the correctness property of WES we can correctly compute all r_i. Each r_i is associated to a tuple (i, s_i, c_i). By construction it is guaranteed that $R_i = g^{r_i}$. Pick any r_i and let's call it r_a, since by construction $s_a := r_a + \mathsf{w}$, we can always compute $\mathsf{w}^* := s_a - r_a$. Therefore, $(\mathsf{X}, \mathsf{w}^*) \notin \mathsf{R}$ never occurs. □

THEOREM 7. *Assume that* WES *is IND-CPA and the discrete logarithm problem is hard. Then our protocol offers VWER one wayness according to Definition 13.*

PROOF. We require the following game hops in order to prove our claim:

Public parameters: $(\mathbb{G}, g, q, \gamma, H)$

$\mathsf{EncR}((\widehat{vk}, \widehat{m}), \mathsf{w})$

$S_{\text{op}} := \emptyset$; $S_{\text{unop}} := \emptyset$
for $i \in [1, \gamma]$:
  $r_i \xleftarrow{\$} \mathbb{Z}_q$; $R_i := g^{r_i}$
  $c_i := \mathsf{WES.Enc}((\widehat{vk}, \widehat{m}), r_i; r'_i)$
    / where r'_i are the random coins used in WES.Enc.
$(b_1, b_2, ..., b_\gamma) := H((c_i, R_i)_{i \in [1,\gamma]})$
for $i \in [1, \gamma]$:
  if $b_i = 1$ **then**
    $S_{\text{op}} := S_{\text{op}} \cup \{(i, r_i, r'_i)\}$
  if $b_i = 0$ **then**
    $s_i := r_i + \mathsf{w}$
    $S_{\text{unop}} := S_{\text{unop}} \cup \{(i, s_i, c_i)\}$
return $c := \{c_i\}_{i \in [1,\gamma]}, \pi := \{S_{\text{op}}, S_{\text{unop}}, \{R_i\}_{i \in [1,\gamma]}\}$

$\mathsf{VfEncR}(c, \pi, (\widehat{vk}, \widehat{m}), \mathsf{X})$

$\{c_i\}_{i \in [1,\gamma]} \leftharpoondown c$; $\{S_{\text{op}}, S_{\text{unop}}, \{R_i\}_{i \in [1,\gamma]}\} \leftharpoondown \pi$
$(b_1, b_2, ..., b_\gamma) := H((c_i, R_i)_{i \in [1,\gamma]})$
for $i \in [1, \gamma]$:
  if $b_i = 1$ **then**
    Check that $(i, r_i, r'_i) \in S_{\text{op}}$
    Check that $c_i = \mathsf{WES.Enc}((\widehat{vk}, \widehat{m}), r_i; r'_i)$
  if $b_i = 0$ **then**
    Check that $(i, s_i, c_i) \in S_{\text{unop}}$
    Check that $g^{s_i} = R_i \otimes \mathsf{X}$
if Any of the checks fail **return** 0, **else return** 1

$\mathsf{DecR}(\widehat{\sigma}, c, \pi)$

$\{c_i\}_{i \in [1,\gamma]} \leftharpoondown c$; $\{S_{\text{op}}, S_{\text{unop}}, \{R_i\}_{i \in [1,\gamma]}\} \leftharpoondown \pi$
foreach $(i, s_i, c_i) \in S_{\text{unop}}$
  $r_i := \mathsf{WES.Dec}(\widehat{\sigma}, c_i)$
There exists at least one r_a s.t. $R_a = g^{r_a}$
$\mathsf{w}^* := s_a - r_a$
return w^*

Figure 29: Construction for VWER.

Game $\mathsf{ExpOW}^{G_0}_{\mathcal{A}}$: This game, formally defined in Fig. 30, corresponds to the original game for VWER one wayness defined in Definition 13. The game is expanded with the interactions described in our construction.

Game $\mathsf{ExpOW}^{G_1}_{\mathcal{A}}$: This game, formally defined in Fig. 31, works exactly as G_0 but with the highlighted grey line. For the oracle query $\mathcal{O}\mathsf{EncR}$ the random oracle H is simulated by lazy sampling, a random bit string $(b_1, b_2, ..., b_\gamma)$ is sampled and the output of the random oracle on the ciphertexts c_i and R_i is set to it. Since the

$\mathsf{ExpOW}_{\mathcal{A}}^{G_0}$

$Q := Q := \emptyset$

$(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{KGen}}(1^\lambda)$

$(\mathsf{X}, \mathsf{w}) \leftarrow \mathsf{createR}(1^\lambda)$

$\mathsf{w}^* \leftarrow \mathcal{A}^{O\widehat{\mathsf{Sig}}, O\mathsf{EncR}}(\widehat{vk}, \mathsf{X})$

$b := (\mathsf{X}, \mathsf{w}^*) \in \mathsf{R}$

return b

$O\widehat{\mathsf{Sig}}(\widehat{m})$

if $\widehat{m} \in Q$ **abort**

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

$\widehat{\sigma} \leftarrow \widehat{\mathsf{Sig}}(\widehat{sk}, \widehat{m})$

return

$O\mathsf{EncR}(\widehat{m})$

if $\widehat{m} \in Q$ **abort**

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

$S_{\mathsf{op}} = S_{\mathsf{unop}} := \emptyset$

for $i \in [0, \gamma]$:

  $r_i \xleftarrow{\$} \mathbb{Z}_q$; $R_i := g^{r_i}$

  $c_i := \mathsf{WES.Enc}((\widehat{vk}, \widehat{m}), r_i; r_i')$

$(b_1, b_2, ..., b_\gamma) := H((c_i, R_i)_{i \in [0,\gamma]})$

for $i \in [0, \gamma]$:

  if $b_i = 1$ **then**

    $S_{\mathsf{op}} := S_{\mathsf{op}} \cup \{(i, r_i, r_i')\}$

  if $b_i = 0$ **then**

    $s_i := r_i + \mathsf{w}$

    $S_{\mathsf{unop}} := S_{\mathsf{unop}} \cup \{(i, s_i, c_i)\}$

$c := \{c_i\}_{i \in [0,\gamma]}$

$\pi := \{S_{\mathsf{op}}, S_{\mathsf{unop}}, \{R_i\}_{i \in [0,\gamma]}\}$

return (c, π)

Figure 30: Definition of the experiment $\mathsf{ExpOW}_{\mathcal{A}}^{G_0}$.

$\mathsf{ExpOW}_{\mathcal{A}}^{G_1}$

$Q := Q := \emptyset$

$(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{KGen}}(1^\lambda)$

$(\mathsf{X}, \mathsf{w}) \leftarrow \mathsf{createR}(1^\lambda)$

$\mathsf{w}^* \leftarrow \mathcal{A}^{O\widehat{\mathsf{Sig}}, O\mathsf{EncR}}(\widehat{vk}, \mathsf{X})$

$b := (\mathsf{X}, \mathsf{w}^*) \in \mathsf{R}$

return b

$O\widehat{\mathsf{Sig}}(\widehat{m})$

if $\widehat{m} \in Q$ **abort**

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

$\widehat{\sigma} \leftarrow \widehat{\mathsf{Sig}}(\widehat{sk}, \widehat{m})$

return

$O\mathsf{EncR}(\widehat{m})$

if $\widehat{m} \in Q$ **abort**

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

$S_{\mathsf{op}} = S_{\mathsf{unop}} := \emptyset$

for $i \in [0, \gamma]$:

  $r_i \xleftarrow{\$} \mathbb{Z}_q$; $R_i := g^{r_i}$

  $c_i := \mathsf{WES.Enc}((\widehat{vk}, \widehat{m}), r_i; r_i')$

$(b_1, b_2, ..., b_\gamma) \leftarrow \{0,1\}^\gamma$

for $i \in [0, \gamma]$:

  if $b_i = 1$ **then**

    $S_{\mathsf{op}} := S_{\mathsf{op}} \cup \{(i, r_i, r_i')\}$

  if $b_i = 0$ **then**

    $s_i := r_i + \mathsf{w}$

    $S_{\mathsf{unop}} := S_{\mathsf{unop}} \cup \{(i, s_i, c_i)\}$

$c := \{c_i\}_{i \in [0,\gamma]}$

$\pi := \{S_{\mathsf{op}}, S_{\mathsf{unop}}, \{R_i\}_{i \in [0,\gamma]}\}$

return (c, π)

Figure 31: Definition of the experiment $\mathsf{ExpOW}_{\mathcal{A}}^{G_1}$.

output of the random oracle is supposed to be random, $\mathsf{ExpOW}_{\mathcal{A}}^{G_0}$ and $\mathsf{ExpOW}_{\mathcal{A}}^{G_1}$ are indistinguishable.

Game $\mathsf{ExpOW}_{\mathcal{A}}^{G_2}$: This game, formally defined in Fig. 32, works exactly as G_0 but with the highlighted grey line. For the oracle query $O\mathsf{EncR}$, for the ciphertexts c_i of S_{unop} (i.e., $b_i = 0$) are replaced by encryptions of 0.

Game $\mathsf{ExpOW}_{\mathcal{A}}^{G_3}$: This game, formally defined in Fig. 33, works exactly as G_1 but with the highlighted grey line. Fore the oracle query $O\mathsf{EncR}$, $b_i = 0$ the variables s_i are randomly smapled as $s_i \leftarrow \mathbb{Z}_q$ and R_i is computed as $R_i := \frac{g^{s_i}}{\mathsf{X}}$. The distribution of s_i and R_i are identical to the previous hybrid and therefore $\mathsf{ExpOW}_{\mathcal{A}}^{G_2}$ and $\mathsf{ExpOW}_{\mathcal{A}}^{G_2}$ are indistinguishable.

Claim 14. *Let* Bad_1 *be the event that:*

$$\left| \begin{array}{l} \Pr[\mathsf{ExpOW}_{\mathcal{A}}^{G_1}(\lambda) = 1] \\ - \Pr[\mathsf{ExpOW}_{\mathcal{A}}^{G_2}(\lambda) = 1] \end{array} \right| > \mathsf{negl}$$

Assume that WES *used in* $O\mathsf{EncR}$ *is IND-CPA secure. Then* $\Pr[\mathsf{Bad}_1(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. Let $q_E := |Q|$ denote the number of queries to oracle $O\mathsf{EncR}$. We consider q_E sub-games such that in sub-game $j \in [1, q_E]$, for queries 1 to $j-1$ to oracle $O\mathsf{EncR}$ ciphertexts c_i, for $i \in [1, \gamma]$, of S_{unop} encrypt 0 (i.e., as in game $\mathsf{ExpOW}_{\mathcal{A}}^{G_2}$); while for queries $j+1$ to q_E ciphertexts c_i for $i \in [1, \gamma]$ of S_{unop} encrypt r_i (i.e., as in game $\mathsf{ExpOW}_{\mathcal{A}}^{G_2}$). The intuition is that if $\Pr[\mathsf{Bad}_1(1^\lambda)] > \mathsf{negl}(\lambda)$, then there exixts some PPT distinguisher $\mathcal{A}_i$, for $i \in$

$\mathsf{ExpOW}_{\mathcal{A}}^{G_2}$

$Q := Q := \emptyset$

$(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{KGen}}(1^\lambda)$

$(\mathsf{X}, \mathsf{w}) \leftarrow \mathsf{createR}(1^\lambda)$

$\mathsf{w}^* \leftarrow \mathcal{A}^{\mathcal{O}\widehat{\mathsf{Sig}}, \mathcal{O}\mathsf{EncR}}(\widehat{vk}, \mathsf{X})$

$b := (\mathsf{X}, \mathsf{w}^*) \in \mathsf{R}$

return b

$\mathcal{O}\widehat{\mathsf{Sig}}(\widehat{m})$

if $\widehat{m} \in Q$ **abort**

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

$\widehat{\sigma} \leftarrow \widehat{\mathsf{Sig}}(\widehat{sk}, \widehat{m})$

return

$\mathcal{O}\mathsf{EncR}(\widehat{m})$

if $\widehat{m} \in Q$ **abort**

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

$S_{\mathrm{op}} = S_{\mathrm{unop}} := \emptyset$

for $i \in [0, \gamma]$:

  $r_i \xleftarrow{\$} \mathbb{Z}_q$; $R_i := g^{r_i}$

$(b_1, b_2, ..., b_\gamma) \leftarrow \{0,1\}^\gamma$

for $i \in [0, \gamma]$:

  if $b_i = 1$ **then**

    $c_i := \mathsf{WES.Enc}((\widehat{vk}, \widehat{m}), r_i; r'_i)$

    $S_{\mathrm{op}} := S_{\mathrm{op}} \cup \{(i, r_i, r'_i)\}$

  if $b_i = 0$ **then**

    $s_i := r_i + \mathsf{w}$

    $c_i := \mathsf{WES.Enc}((\widehat{vk}, \widehat{m}), 0)$

    $S_{\mathrm{unop}} := S_{\mathrm{unop}} \cup \{(i, s_i, c_i)\}$

$c := \{c_i\}_{i \in [0,\gamma]}$

$\pi := \{S_{\mathrm{op}}, S_{\mathrm{unop}}, \{R_i\}_{i \in [0,\gamma]}\}$

return (c, π)

Figure 32: Definition of the experiment $\mathsf{ExpOW}_{\mathcal{A}}^{G_2}$**.**

$\mathsf{ExpOW}_{\mathcal{A}}^{G_3}$

$Q := Q := \emptyset$

$(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{KGen}}(1^\lambda)$

$(\mathsf{X}, \mathsf{w}) \leftarrow \mathsf{createR}(1^\lambda)$

$\mathsf{w}^* \leftarrow \mathcal{A}^{\mathcal{O}\widehat{\mathsf{Sig}}, \mathcal{O}\mathsf{EncR}}(\widehat{vk}, \mathsf{X})$

$b := (\mathsf{X}, \mathsf{w}^*) \in \mathsf{R}$

return b

$\mathcal{O}\widehat{\mathsf{Sig}}(\widehat{m})$

if $\widehat{m} \in Q$ **abort**

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

$\widehat{\sigma} \leftarrow \widehat{\mathsf{Sig}}(\widehat{sk}, \widehat{m})$

return

$\mathcal{O}\mathsf{EncR}(\widehat{m})$

if $\widehat{m} \in Q$ **abort**

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

$S_{\mathrm{op}} = S_{\mathrm{unop}} := \emptyset$

for $i \in [0, \gamma]$:

$(b_1, b_2, ..., b_\gamma) \leftarrow \{0,1\}^\gamma$

for $i \in [0, \gamma]$:

  if $b_i = 1$ **then**

    $r_i \xleftarrow{\$} \mathbb{Z}_q$; $R_i := g^{r_i}$

    $c_i := \mathsf{WES.Enc}((\widehat{vk}, \widehat{m}), r_i; r'_i)$

    $S_{\mathrm{op}} := S_{\mathrm{op}} \cup \{(i, r_i, r'_i)\}$

  if $b_i = 0$ **then**

    $s_i \xleftarrow{\$} \mathbb{Z}_q$; $R_i := \frac{g^{s_i}}{\mathsf{X}}$

    $c_i := \mathsf{WES.Enc}((\widehat{vk}, \widehat{m}), 0)$

    $S_{\mathrm{unop}} := S_{\mathrm{unop}} \cup \{(i, s_i, c_i)\}$

$c := \{c_i\}_{i \in [0,\gamma]}$

$\pi := \{S_{\mathrm{op}}, S_{\mathrm{unop}}, \{R_i\}_{i \in [0,\gamma]}\}$

return (c, π)

Figure 33: Definition of the experiment $\mathsf{ExpOW}_{\mathcal{A}}^{G_3}$**.**

$[1, q_E]$, that it can determine with non-negligible probability whether it plays game $\mathsf{ExpOW}_{\mathcal{A}}^{G_1}$ or game $\mathsf{ExpOW}_{\mathcal{A}}^{G_2}$ based on the i^{th} answer of oracle $\mathcal{O}\mathsf{EncR}$.

More specifically, assume by contradiction that $\Pr[\mathsf{Bad}_1(1^\lambda)] > \mathsf{negl}(\lambda)$, then there exists PPT distinguisher $\mathcal{A}_{j^*}$ such that:

$$\Pr\left[b = b^* \middle| \begin{array}{c} b \xleftarrow{\$} \{0,1\} \\ \mathsf{ExpOW}_{\mathcal{A}}^{subG_{j^*}}(\lambda) \\ b^* \leftarrow \mathcal{A}_{j^*}() \end{array} \right] > \frac{1}{2} + \mathsf{negl}$$

We can construct adversary $\mathcal{B}$ that uses $\mathcal{A}_{i^*}$ to break IND-CPA the encryption used in $\mathcal{O}\mathsf{EncR}$ with the following steps:

- $\mathcal{B}$ initializes the challenger, who sends $\widehat{vk}$.
- $\mathcal{B}$ runs $(\mathsf{X}, \mathsf{w}) \leftarrow \mathsf{createR}(1^\lambda)$.
- $\mathcal{B}$ invokes $\mathcal{A}_{i^*}$ on input $\widehat{vk}$ and X.
- $\mathcal{O}\mathsf{EncR}$ queries are treated in the following manner: (i) for $j \in [1, j^*-1]$, $\mathcal{B}$ answers with $c_i := \mathsf{WES.Enc}((\widehat{vk}, \widehat{m}), r_j; 0)$ for $b_i = 0$; (ii) for $j \in [j^*+1, q_E]$, $\mathcal{B}$ answers with $c_i := \mathsf{WES.Enc}((\widehat{vk}, \widehat{m}), r_j; r'_j)$; and (iii) for $j = j^*$, $\mathcal{B}$ chooses at random i^* such that $b_{i^*} = 0$ and sets $\hat{m}^* := \hat{m}$, $m_0 := r_{i^*}$ and $m_1 := 0$ and forwards the tuple $(\hat{m}^*, m_0, m_1)$ to the challenger to obtain c_b which in turn $\mathcal{B}$ forwards to $\mathcal{A}_{j^*}$ as c_{i^*}.
- Thereafter $\mathcal{A}_{j^*}$ outputs w^*.
- $\mathcal{B}$ receives the guess b^* from $\mathcal{A}_{j^*}$.

- $\mathcal{B}$ forwards b^* to the challenger.

As already described, $\mathcal{B}$ knows all the private information required to run oracle $\mathcal{O}$EncR. Regarding oracle $\mathcal{O}\widehat{\mathsf{Sig}}$, $\mathcal{B}$ forwards the query to $\mathcal{O}$Sig of the WES oracle, which returns $\widehat{\sigma}$. Note that this means that memory Q and the memory of WES oracle are synchronized.

Our adversary $\mathcal{B}$ perfectly simulates the sub-game $\mathsf{ExpOW}_{\mathcal{A}}^{subG_{j^*}}$ to $\mathcal{A}_{j^*}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. If adversary $\mathcal{A}_{j^*}$ outputs $b^* = b$ with probability higher than $\frac{1}{2} + \mathsf{negl}(\lambda)$, since the only difference between games $\mathsf{ExpOW}_{\mathcal{A}}^{G_1}$ and $\mathsf{ExpOW}_{\mathcal{A}}^{G_2}$ is the ciphertext c_{i^*} of the j^*th query to $\mathcal{O}$EncR that was forwarded to the challenger, the bit forwarded by $\mathcal{A}$ can also be used to differentiate in the IND-CPA game. However, this contradicts the assumption that the WES used is IND-CPA.

Our adversary $\mathcal{B}$ chooses which sub-game j^* to play with probability $\frac{1}{q_E}$. Moreover, $\mathcal{B}$ chooses which ciphertext c_{i^*} to forward to the challenger with probability $\frac{1}{\gamma}$ Thus, $\Pr[\mathsf{Bad}_1(1^\lambda)] \leq \frac{\mathsf{negl}(\lambda)}{\gamma q_E} \leq \mathsf{negl}(\lambda)$ and this claim has been proven. Therefore, we can conclude that $\mathsf{ExpOW}_{\mathcal{A}}^{G_1} \approx \mathsf{ExpOW}_{\mathcal{A}}^{G_1}$ □

Claim 15. *Assume that the discrete logarithm problem is hard. Then* $\Pr[\mathsf{ExpOW}_{\mathcal{A}}^{G_3}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that there exists PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{ExpOW}_{\mathcal{A}}^{G_3}(1^\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct adversary $\mathcal{B}$ that uses $\mathcal{A}$ to solve the discrete logarithm problem with the following steps:

- $\mathcal{B}$ initializes the challenger, who sends X.
- $\mathcal{B}$ runs $(\widehat{vk}, \widehat{sk}) \leftarrow \widehat{\mathsf{KGen}}(1^\lambda)$.
- $\mathcal{B}$ invokes $\mathcal{A}$ on input $\widehat{vk}$ and X to obtain w^*.
- $\mathcal{B}$ forwards w^* to the challenger.

Regarding oracles $\mathcal{O}\widehat{\mathsf{Sig}}$ and $\mathcal{O}$EncR, $\mathcal{B}$ knows all the private information required to simulate them.

Our adversary $\mathcal{B}$ perfectly simulates $\mathsf{ExpOW}_{\mathcal{A}}^{G_3}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now if $\mathcal{A}$ wins with $\Pr[\mathsf{ExpOW}_{\mathcal{A}}^{G_3}(1^\lambda) = 1] > \mathsf{negl}(\lambda)$, this means that $(\mathsf{X}, \mathsf{w}^*) \in \mathsf{R}$, therefore winning $\mathsf{ExpOW}_{\mathcal{A}}^{G_3}$ with non-negligible probability implies solving the discrete logarithm problem with non-negligible probability. However, this contradicts the assumption that the discrete logarithm problem is hard, thus such an $\mathcal{A}$ cannot exist and this claim has been proven. □

We have shown that $\mathsf{ExpOW}_{\mathcal{A}}^{G_0} \approx \mathsf{ExpOW}_{\mathcal{A}}^{G_3}$ and that $\Pr[\mathsf{ExpOW}_{\mathcal{A}}^{G_3}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$. Therefore Theorem 7 has been proven. □

Theorem 8. *Assume* $\widehat{\mathsf{DS}}$ *is signature schemes that satisfy unforgeability and* WES *be a secure witness encryption based on signatures scheme. Then, our protocol offers Verifiable witness encryption for a relation according to Definition 14.*

Proof. Assume that an adversary $\mathcal{A}$ breaks the verifiability of the protocol. This implies that $\mathcal{A}$ message $\widehat{m}$ outputs oracle verification key $\widehat{vk}$, oracle signature $\widehat{\sigma}$ on message $\widehat{m}$, outputs (c, π) of EncR and a public statement X such that:

(1) $\widehat{\sigma}$ is a valid signature, i.e., $\widehat{\mathsf{Vf}}(\widehat{vk}, \widehat{m}, \widehat{\sigma}) = 1$.
(2) The output of EncR is valid, i.e., $\mathsf{VfEncR}(c, \pi, (\widehat{vk}, \widehat{m}), \mathsf{X}) = 1$.
(3) The final outputted witness $\mathsf{w}^* \leftarrow \mathsf{DecR}(\widehat{\sigma}, c, \pi)$ is not in a hard relation with the public statement X, i.e., $(\mathsf{X}, \mathsf{w}^*) \notin \mathsf{R}$.

We will now show that if the first and second conditions hold true, then algorithm DecR will output a witness w^* so that it holds that $(\mathsf{w}^*, \mathsf{X}) \in \mathsf{R}$ except with negligible probability.

Recall that $(\widehat{m}, \widehat{vk})$ is associated with γ-many ciphertexts $(c_1, c_2, ..., c_\gamma)$ that encrypt random values $(r_1, r_2, ..., r_\gamma)$. Note that algorithm DecR decrypts these ciphertexts in order to get the encrypted values $(r_1, r_2, ..., r_\gamma)$.

Next, recall that since algorithm VfEncR outputs 1, we are guaranteed that: $g^{s_i} = R_i \otimes \mathsf{X}$, for $i \in [0, \gamma]$, where $R_i = g^{r_i}$. Thus, the following equation is satisfied in the exponent, $s_i = r_i + \mathsf{w}$.

Setting the total number of ciphertexts γ sufficiently large, then the probability of all $(r_1, r_2, ..., r_\gamma)$ be invalid is negligible according to theorem 2 of [13]. More precisely, we are guaranteed that there exists at least one r_i such that $c_i := \mathsf{WES.Enc}((\widehat{vk}, \widehat{m}), r_i; r_i')$ and $R_i = g^{r_i}$ (recall that R_i was part of π). This implies that a valid witness w^* can be computed as $\mathsf{w}^* = s_i - r_i$. Hence, giving the property of verifiability. □

OM-CCA-A2L
$\mathsf{q} := 0$
$(\overline{ek}, \overline{dk}) \leftarrow \overline{\mathsf{KGen}}(1^\lambda)$
$(\mathsf{X}_i, \mathsf{w}_i) \leftarrow \mathsf{createR}(1^\lambda)$
$c_i \leftarrow \mathsf{Enc}(\overline{ek}, \mathsf{w}_i)$
$\{\mathsf{w}_i'\}_{i\in[0,k]} \leftarrow \mathcal{A}^{\mathcal{O}\mathrm{A2L}}(\overline{ek}, \{(\mathsf{X}_i, c_i)\}_{i\in[0,k]})$
$b_0 := \forall i, \mathsf{w}_i' = \mathsf{w}_i$
$b_1 := \mathsf{q} < k$
return $b_0 \wedge b_1$
$\mathcal{O}\mathrm{A2L}(vk, m, \mathsf{X}, c, \hat{\sigma})$
if $vk \notin (\mathsf{ADP.KGen}(1^\lambda))$ **abort**
$\mathsf{w} \leftarrow \mathsf{Dec}(\overline{dk}, c)$
if $\mathsf{PreVf}(\mathsf{X}, m, vk, \hat{\sigma}) \wedge (\mathsf{X}, \mathsf{w}) \in \mathsf{R}$
$\mathsf{q} := \mathsf{q} + 1$
return w
else return $\perp$

Figure 34: One more CCA-A2L

D PROOF OF LEMMA 1

In [29], Lemma 4.8 is very similar to Lemma 1. They prove that the following property, called one more CCA A2L (OM-CCA-A2L) (Fig. 34) holds if the OMDL assumption holds. Instead of proving directly against OMDL, we will prove Lemma 1 by contradiction against OM-CCA-A2L.

Claim 16. *Assume that OM-CCA-A2L holds. Then* $\Pr[\mathit{OMDL}\text{-}\mathsf{LHE}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{OMDL\text{-}LHE}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$. We can construct adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break OM-CCA-A2L with the following steps:

- $\mathcal{B}$ initializes the challenger, who provides $\mathcal{B}$ with $\overline{ek}$ and $\{(\mathsf{X}_i, c_i)\}_{i\in[0,k]}$.
- $\mathcal{B}$ invokes $\mathcal{A}$ on input $\overline{ek}$ and $\{(\mathsf{X}_i, c_i)\}_{i\in[0,k]}$ to obtain $\{\mathsf{w}'_i\}_{i\in[0,k]}$.
- $\mathcal{B}$ sends $\{\mathsf{w}'_i\}_{i\in[0,k]}$ to the challenger.

Regarding oracle OMDL-LHE, for every query that $\mathcal{B}$ receives, he will run $(vk, sk) \leftarrow \mathsf{ADP.KGen}$ and sample a message m. Then he generates a presignature using the X queried by $\mathcal{A}$. Now, he will run the query to oracle $\mathcal{O}$A2L using c and X as received from $\mathcal{A}$, together with the generated vk, message m and presignature. Since the presignature check of $\mathcal{O}$A2L will always pass, $\mathcal{O}$A2L will only return $\perp$ if c is not encrypting the DL of X. This ensures that q of both oracles is the same.

Our adversary $\mathcal{B}$ perfectly simulates OMDL-LHE to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, since the count of both oracles is synchronized and the k is the same in both games, if $\{\mathsf{w}'_i\}_{i\in[0,k]}$ wins OMDL-LHE, it also wins OM-CCA-A2L. However, this contradicts the assumption that OM-CCA-A2L holds. Therefore, $\mathcal{A}$ does not exist. $\square$