A Cryptographic Layer for the Interoperability of CBDC and Cryptocurrency Ledgers

Diego Castejon-Molina*†, Alberto Del Amo Pastelero*†, Dimitrios Vasilopoulos*, Pedro Moreno-Sanchez[1]*‡
* IMDEA Software Institute
† Universidad Politécnica de Madrid
‡ VISA Research

***Abstract*—Cryptocurrencies are used in several, distinct use cases, thereby sustaining the existence of many ledgers that are heterogeneous in terms of design and purpose. In addition, the interest of central banks in deploying Central Bank Digital Currency (CBDC) has spurred a blooming number of conceptually different proposals from central banks and academia. As a result of the diversity of cryptocurrency and CBDC ledgers, interoperability, i.e., the seamless transfer of value between users that operate on different ledgers, has become an interesting research problem. In fact, interoperability has been explored both in CBDC and cryptocurrencies, and numerous proposals exist. However, these proposals are tailored to the characteristics of the ledgers for which they are designed. For instance, some rely on trusted hardware, others rely on the scripting capabilities of the underlying ledger or on specific cryptographic assumptions of the transaction authorization mechanism (e.g., adaptor signatures), while others rely on a trusted entity. This fragmentation results in the repetitive development of interoperablity protocols that address the same applications across the various ledgers.**

In this work, we propose an alternative approach: decouple the transaction authorization details of each ledger from the definition of cross-ledger applications. To do so, we define a middle layer that abstracts the core functionality for authorizing transactions in any ledger and the security notions of interest. This middle layer serves two purposes: (i) it becomes the main cryptographic building block to (re)define cross-ledger applications in a ledger agnostic manner; (ii) for any ledger that exists (and the new to come), it suffices to prove that it is a secure instance of the middle layer to be compatible with cross-ledger protocols. We define two new primitives for our middle layer, the basic payment ledger (BL) and the conditional payment ledger (CL). We prove that the two most common transaction authorization mechanism (digital signatures and zero knowledge proofs) are secure constructions for BL. We also prove that common smart contracts (e.g., HTLC and PTLC) and the combination of adaptor signatures with verifiable timelock puzzles are also secure constructions for CL. Finally, we discuss how to design some popular applications (e.g. atomic swaps between ledgers) using our middle layer.

I. Introduction

Distributed ledgers (DLs) are gradually becoming an accepted technology. Since the inception of Bitcoin, a wide range of distributed ledgers have been proposed to cope with the peculiarities of different use cases. For instance, cryptocurrencies other than Bitcoin have been proposed to provide privacy-preserving payments (e.g., Monero and Zcash) or smart contracts (e.g., Ethereum, Tezos). To accommodate for these different goals, these ledgers expose distinct transaction authorization mechanisms: While Monero relies on unlinkable ring signatures and Zcash builds upon zero-knowledge proofs, ledgers with support for smart contracts accept more expressive transaction authorization policies encoded as a code excerpt.

More recently, several central banks have shown interest in creating a digital version of their physical currency, called *Central Bank Digital Currency* (CBDC) [2], [4]–[9], [11], [17], [18], [20], [23], [24], [27]–[29], [39], [44], [48], [51]–[54], [62]. Spurred from this interest, recent academic works have contributed further proposals for CBDC [33], [37], [39], [60], [65]. The different motivations behind each of the aforementioned works (i.e. transaction privacy, regulation adherence, etc.) has resulted in a heterogeneous set of CBDC proposals. For example, KSIcash [27] and Hamilton [39] build upon a custom-made ledger; Bakong [44] relies on Hyperledger Iroha; Itcoin [11], [62] is based on the payment channel network of a Bitcoin inspired custom ledger; TIPS+ [62] is based on a payment service of the Eurosystem [26]; while Platypus [65] and the Swiss National Bank CBDC [20] are based on eCash.

In summary, there is a wide range of ledgers that are heterogeneous in terms of design and purpose. In this state of affairs, interoperability across these ledgers has become an interesting research problem. Moreover, according to a public consultation about the digital euro [25], one of the main challenges for the digital euro is achieving interoperability.

Interoperability in the context of payments involves enabling the seamless flow of value between two users even if their accounts are held in different ledgers. Such a flow of value (or payment) between two users with accounts in different ledgers requires a protocol to synchronize the authorizations of two or more transactions across two or more ledgers. Examples of such payment protocols are atomic swaps between ledgers, coin mixing, and atomic multi-hop payments.

In the literature, there exist several instances of such payment protocols, each tailored to the concrete transaction authorization mechanism provided by one (or a small set) of available distributed ledgers. For instance, for the three mentioned applications, there exist protocols (i) tailored to ledgers that support HTLC smart contracts (e.g. [34], [41], [42], [49]); (ii) tailored to ledgers that support a digital signature compatible with adaptor signatures (e.g [3], [21], [30], [38], [47], [56], [59]); (iii) tailored to ledgers that support a digital signature not compatible with adaptor (e.g. [58]); (iv) tailored to ledgers that support advanced scripting capabilities (e.g. [1], [32], [67]); (v) based on trusted hardware (e.g. [12], [22], [61]).

[1] This work has been done while in employment of IMDEA Software Institute

This situation is undesirable from a scientific point of view because, for each payment application, it requires designing and analyzing several instances, one for each ledger setting.

Instead, the goal of this work is to define a framework that allows to decouple the definition of cross-ledger protocols (e.g. atomic swaps, coin mixing) from the specific authorization mechanisms provided by existing ledgers.

A. Where Existing Approaches Fall Short

In the following, we summarize existing interoperability solutions either for all CBDC or all cryptocurrencies.

CBDC-focused Approaches. There are reports from different institutions like the European Central Bank (ECB) [24], the Bank of International Settlements (BIS) [8], [10] or SWIFT [55] that explore interoperability for CBDC. However, these proposals do not contain many technical details and rely on trusted intermediaries to facilitate the CBDC exchange, similar to what happens currently with the banking sector. Relying on a trusted entity to perform cross-ledger protocols introduces a single point of failure, which could be mitigated by using secure decentralized protocols based on cryptographic assumptions. Furthermore, these CBDC interoperability proposals focus on the coordination of payments between multiple CBDC ledgers, but not between CBDC and a non CBDC ledgers (e.g., any cryptocurrency). Extending these proposals for cross-ledger payments based on cryptographic assumptions (instead of trusted entities) has not been explored yet.

Bridge Ledger. Some approaches, like Polkadot [1] or X-Claim [67], define cross-ledger payments with the aid of smart contracts that are enforced by a ledger that supports scripts. Therefore, if an atomic swap involves a ledger that does not support scripting capabilities, an additional ledger is used to provide the capabilities, like a bridge. In the case of X-Claim, this bridge ledger issues redeemable tokens fully backed in the ledger without scripting capabilities. Then, the cross-ledger protocol is performed in a smart contract in the bridge ledger. This approach could result in forcing users to operate in a third ledger, a setting which implies additional trust assumptions that are not always desirable. Moreover, this bridge ledger needs to be able to interact with any other ledger.

Universal Atomic Swaps. Authors in [59] proposed a swap protocol between any two ledgers. Such a generic construction relies on a general-purpose 2-party computation and verifiable timelock puzzles for discrete logarithms. Although such a generic approach is in line with the goal of this work, the concrete realization of the 2-party computation might lead to protocols inefficient in practice, as agreed by the authors. In fact, they also propose an alternative, efficient swap protocol that is however restricted to ledgers that authorize transactions using a digital signature scheme compatible with adaptor signatures. This approach cannot seamlessly be extended to cryptocurrency ledgers such as Zerocash, or CBDC ledgers such as UTT, PEReDi and Platypus, since they do not rely on digital signatures for authorizing transactions.

B. Our Approach

A more flexible and transparent approach would be to detach the authorization details of each ledger from the definition of cross-ledger applications. To do so, we define a middle layer that abstracts the core functionality for authorizing transactions in any ledger, along with the security notions of interest. For instance, in the case of transaction authorization for conditional payments, our middle layer models the complete functionality of creating the shared account and locking the coins, authorizing the transfer of coins to the receiver, or authorizing the transfer of coins to the sender, if the condition to pay the receiver is not met. Hence, this middle layer serves a two-fold purpose: (i) it can become the main cryptographic building block to (re)define cross-ledger applications, agnostic to the concrete ledger instance; and (ii) if a new ledger is defined (e.g., as currently being proposed for CBDC), it suffices to prove that it is a secure instance of this middle layer to benefit from these ledger-agnostic cross-ledger protocols.

As we explain later in Section IV-A, we identify two classes of ledger transaction authorization schemes according to the type of accounts and type of transactions that they support: ledgers with basic authorizations (BL) and ledgers with conditional authorizations (CL). Then, our approach consists in (i) proving that a given ledger is a secure instance of either BL, CL or both (Section IV); and (ii) (re)defining cross-ledger protocols using (several copies of) BL and CL as building block (Section V).

Contributions. In this work, we:

- Introduce two new primitives for ledger transaction authorizations and their security properties. They are called basic payment ledger (BL) and conditional payment ledger (CL) and capture the authorization mechanism of many ledgers.
- Introduce two constructions for BL and prove them secure. The first construction is based on digital signature schemes, while the second is based on NIZK. By doing this, we show that at least the following ledgers can be described as a BL [11], [27], [31], [35], [36], [39], [43]–[45], [50], [60], [62], [64], [66].
- Introduce three constructions for CL and prove them secure. The first construction assumes that the ledger supports evaluation of a hard relation (e.g. preimage of a hash) and has a timelock feature. The second construction assumes a ledger that only supports timelocks. The third construction assumes a BL whose authorization mechanism is compatible with adaptor signatures. By doing this, we show that at least the following ledgers can be described as a CL [11], [27], [31], [35], [36], [39], [43]–[45], [64], [66].
- Show how to use the framework to solve the atomic swaps, and other applications that we will explore in the future.

II. Preliminaries

Accounts. Coins are held in *accounts*. We differentiate between two types of accounts. First, we consider *simple accounts*. A simple account is identified by its public key *pk*, while the corresponding private key *sk* is used to spend the account's coins. We thereby represent, e.g., an output in the UTXO model (e.g., Bitcoin) whose spending condition is a signature with the corresponding private key *sk*, an externally owned account (EOA) in Ethereum or a CBDC account governed by the key pair (pk, sk).

Second, we consider *escrow accounts*. An escrow account is identified by a tuple $epk = ((pk_{\mathsf{red}}, \mathbb{C}), (pk_{\mathsf{ref}}, T))$ where $\mathbb{C}$ denotes an instance of a cryptographic hard problem (e.g., find the pre-image of a given hash value) and models co-ownership of coins between two users (sender S and receiver R) for a predefined amount of time T. Coins held in an escrow account can be spent in two paths. First, using the secret key sk_{red} corresponding to pk_{red} and the solution to the problem instance in $\mathbb{C}$ (redeem path). Second, using the secret key sk_{ref} after the timeout T expires (refund path). We thereby represent, e.g., an output of type hash timelock contract (HTLC) in Bitcoin, an instance of HTLC contract in Ethereum, or similar procedures such as authorization holds in credit/debit cards [63].

Transactions. A transaction transfers coins between accounts. We identify three types of transactions, each defined over its own message space. First, we consider *basic* (bsc) transactions. A basic transaction, denoted by tx_{bsc}, transfers coins from a simple account. We abstract away from the transaction creation details at each ledger and assume that there exists a function $tx_{\mathsf{bsc}} \leftarrow \mathsf{ctTx}_{\mathsf{bsc}}(pk, \left\{\ddot{pk}, \ddot{epk}\right\})$.

We then consider two additional types of transactions, corresponding to the two spending paths for an escrow account. A *redeem* transaction, denoted by tx_{red}, transfers coins from an escrow account through the redeem path. A *refund* transaction, denoted by tx_{ref}, transfers coins from an escrow account through the refund path. Similar to the creation of basic transactions, we assume the existence of two functions $tx_{\mathsf{red}} \leftarrow \mathsf{ctTx}_{\mathsf{red}}(epk, \ddot{pk}_{\mathsf{red}} := \left\{\ddot{pk}, \ddot{epk}\right\})$ and $tx_{\mathsf{ref}} \leftarrow \mathsf{ctTx}_{\mathsf{ref}}(epk, \ddot{pk}_{\mathsf{ref}} := \left\{\ddot{pk}, \ddot{epk}\right\})$ that create redeem and refund transactions, respectively. For the rest of the paper, every time we denote a transaction as tx_{bsc}, tx_{red} or tx_{ref}, we are referring to transactions in the *basic*, *redeem* and *refund* message spaces, accordingly.

Generic Ledger. A ledger (L) maintains a list of (ordered) transactions, denoted by TX_L. The addition of new transactions to TX_L is enforced by the operator of the ledger, which can be a small and fixed group of entities (e.g., as in CBDC or permissioned blockchains), or a large and dynamic group of entities running a permissionless distributed consensus protocol (e.g., as in Bitcoin or Ethereum).

A transaction *tx* that transfers coins from a sender account $pk_S := \{pk, epk\}$ to a receiver account $\ddot{pk}_R := \left\{\ddot{pk}, \ddot{epk}\right\}$ must satisfy a series of predicates before it is added to TX_L.

1) The sender's account must have enough coins to carry out the transaction, a predicate that we denote by $\{0,1\} \leftarrow \mathtt{IsFunded}(pk_S, TX_L)$.

2) The transaction must be well formed and not create any new coins, a predicate that we denote by $\{0,1\} \leftarrow \mathtt{IsValid}(tx)$.

3) The transaction must not have been previously included in the ledger, a predicate that we denote by $\{0,1\} \leftarrow \mathtt{IsUnique}(tx, TX_L)$.

4) The transaction must have been successfully authorized, a predicate that we denote by $\{0,1\} \leftarrow \mathtt{IsAuth}(tx, \sigma, pk_S)$. Considering the different types of accounts, we can further subdivide the transaction authorization predicate into: (i) $\{0,1\} \leftarrow \mathtt{IsAuth}_{\mathtt{Own}}(tx, \sigma, pk_S)$, verifying that for an account pk_S of any type, it correctly authorizes the asset transfer in *tx* with σ; (ii) $\{0,1\} \leftarrow \mathtt{IsAuth}_{\mathtt{Rel}}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, epk)$, verifying that for escrow account *epk*, the authorization σ_{red} provides a solution to the problem $\mathbb{C}$ specified in *epk*; and (iii) $\{0,1\} \leftarrow \mathtt{IsAuth}_{\mathtt{T}}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}, epk)$, that verifies for refund transactions if the required timeout T has expired.

Ledger Types. While existing ledgers follow the blueprint of a generic ledger, we can classify existing ledgers depending on: (i) the type of account supported (simple and escrow); and (ii) the type of transaction supported (basic, redeem and refund). Following this classification, we can distinguish between ledgers whose scripting capabilities only support basic transactions and simple accounts, and those whose scripting capabilities additionally support refund and redeem transactions, as well as escrow accounts. We refer to the former as *Basic Payment Ledger* (BL) and to the latter as *Conditional Payment Ledger* (CL). We next define two schemes for our cryptographic framework, BL and CL that describe how basic, redeem and refund transactions are authorized to spend coins from simple and escrow accounts.

III. Cryptographic Layer: Definitions for BL and CL

Preliminaries. First, we assume unitary transactions to simplify the notation. We discuss the extension to support arbitrary amounts in Section VI. Second, we assume the existence of an algorithm, called $\mathtt{createT}(\cdot)$ that, when called, returns an appropriate time parameter for the creation of an escrow account. Moreover, we let $\mathtt{readTime}(\cdot)$ be a function that returns the current ledger time (e.g., current block height if the time is measured as the number of blocks in the blockchain). Third, an escrow account is based on the notion of hard relation. We recall the notion of a hard relation $\mathcal{R} \subseteq \mathcal{D}_S \times \mathcal{D}_w$ with statement-witness pairs $(\mathbb{C}, w) \in \mathcal{D}_S \times \mathcal{D}_w$. We denote by $\mathcal{L}_{\mathcal{R}}$ the associated language defined as $\mathcal{L}_{\mathcal{R}} := \{\mathbb{C} \in \mathcal{D}_S \mid \exists w \in \mathcal{D}_w \text{ s.t. } (\mathbb{C}, w) \in \mathcal{R}\}$. We say that $\mathcal{R}$ is a hard relation if the following holds: (i) There exists a PPT algorithm $\mathsf{createR}(1^\lambda)$ that computes $(\mathbb{C}, w) \in \mathcal{R}$; (ii) the relation is decidable in polynomial time; and (iii) for all PPT adversaries $\mathcal{A}$, the probability that on input $\mathbb{C}$, $\mathcal{A}$ outputs w such that $(\mathbb{C}, w) \in \mathcal{R}$ is negligible.

Remark. *The cryptographic games defined in this section do not take into consideration the transaction correctness, its format or the balance of the account, only the cryptographic properties evaluated under* $\mathtt{IsAuth}$ *(e.g. ownership, hard relation, time). We thereby focus on the authorization of transactions as required for interoperability and assume that the rest of predicates (e.g.,* $\mathtt{IsValid}$*,* $\mathtt{IsFunded}$*,* $\mathtt{IsUnique}$*) are checked by the ledger operator before adding the transaction to the ledger.*

Therefore, the message space that the adversary $\mathcal{A}$ *has access in these games is larger than what it would actually be when attacking the ledger (e.g., an adversary could manage to forge the authorization of a transaction that spends from an account with zero balance and thus would be later rejected by the ledger operator).*

$\mathsf{bscForge}_{\Pi_{BL},\mathcal{A}}(\lambda)$	$\mathcal{O}\mathsf{Auth}_{\mathsf{bsc}}(tx_{\mathsf{bsc}})$
$\mathcal{Q} := \emptyset$	$\sigma_{\mathsf{bsc}} \leftarrow \mathsf{Auth}_{\mathsf{bsc}}(sk_S, tx_{\mathsf{bsc}})$
$(pk, sk) \leftarrow \mathsf{ctAcc}(1^\lambda)$	$\mathcal{Q} := \mathcal{Q} \cup \{tx_{\mathsf{bsc}}\}$
$(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}}) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{Auth}_{\mathsf{bsc}}}(pk)$	**return** σ_{bsc}
$b_0 := tx_{\mathsf{bsc}} \notin \mathcal{Q}$	
$b_1 := \mathsf{isAuth}_{\mathsf{BL}}(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}}, pk)$	
return $b_0 \wedge b_1$	

Fig. 1. BL unforgeability.

A. Basic Payment Ledger (BL)

Users interact with BL using the API described in Definition 1.

Definition 1 (Basic Payment Ledger (BL)). *A BL comprises the three algorithms* (ctAcc, $\mathsf{Auth}_{\mathsf{bsc}}$, $\mathsf{isAuth}_{\mathsf{BL}}$) *defined below:*

- $(pk, sk) \leftarrow \mathsf{ctAcc}(1^\lambda)$: *Account creation is a PPT algorithm, that takes as input the security parameter* λ*, and outputs the public key pk and private key sk of the account.*
- $\sigma_{\mathsf{bsc}} \leftarrow \mathsf{Auth}_{\mathsf{bsc}}(sk, tx_{\mathsf{bsc}})$: *Transaction authorization is a PPT algorithm, that takes as input the private key of the account sk and a basic transaction* tx_{bsc}*, and produces a basic transaction authorization,* σ_{bsc}.
- $1/0 \leftarrow \mathsf{isAuth}_{\mathsf{BL}}(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}}, pk)$: *Authorization verification for basic transactions is DPT algorithm that takes as input a basic transaction* tx_{bsc}*, a basic transaction authorization* σ_{bsc}*, and the public key of the sender pk, and outputs* 1 *if the authorization corresponds to the sender and* 0 *otherwise.*

Definition 2 (BL Correctness). *A BL is said to be* correct *if for all* $\lambda \in \mathbb{N}$*, all* $(pk, sk) \leftarrow \mathsf{ctAcc}(1^\lambda)$*, all* $(\ddot{pk}, \ddot{sk}) \leftarrow \mathsf{ctAcc}(1^\lambda)$*, all* $tx_{\mathsf{bsc}} \leftarrow \mathsf{ctTx}_{\mathsf{bsc}}(pk, \ddot{pk})$*, all* $\sigma_{\mathsf{bsc}} \leftarrow \mathsf{Auth}_{\mathsf{bsc}}(sk, tx_{\mathsf{bsc}})$ *it holds that:*

$$\Pr\left[\mathsf{isAuth}_{\mathsf{BL}}(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}}, pk) = 1\right] = 1$$

Security. The notion of security for BL is *BL unforgeability* and ensures that the adversary cannot successfully authorize transactions without access to the corresponding secret key.

Definition 3 (BL Unforgeability). *A BL is said to offer BL unforgeability 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* $\Pr[\mathsf{bscForge}_{\Pi_{BL},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$*, where* $\mathsf{bscForge}$ *is defined in Fig. 1.*

Interactive BL. Platypus [65] and PEReDi [37] are two ledgers supporting simple accounts, in which a full transaction authorization requires an interactive protocol between sender and receiver. In Appendix A we provide an extension of the BL, which we call Interactive BL (IBL), that contemplates interactive authorization of basic transactions between sender and receiver.

B. Conditional Payment Ledger (CL)

CL is an extension of BL that supports payments conditioned on the knowledge of the solution to some hard problem and a timeout based on ledger-dependent time. Users interact with CL using the API described in Definition 1, with the additions defined in Definition 4.

Definition 4 (Conditional Payment Ledger (CL)). *A conditional payment ledger defined w.r.t. a* BL *and a hard relation* $\mathcal{R}$*, extends* BL *with the algorithms (*$\mathsf{Auth}_{\mathsf{red}}$*,* $\mathsf{Auth}_{\mathsf{ref}}$*,* GWit*,* $\mathsf{isAuth}_{\mathsf{CL}}$*) and interactive protocol* ctEAcc *defined below:*

- $\left\langle \begin{matrix} \{(epk, \mathsf{aux}_S, tx_{\mathsf{red}}, tx_{\mathsf{ref}}), \perp\}, \\ \{(epk, \mathsf{aux}_R, tx_{\mathsf{red}}, tx_{\mathsf{ref}}), \perp\} \end{matrix} \right\rangle \leftarrow \mathsf{ctEAcc}\left\langle \begin{matrix} S(\mathbb{C}, T), \\ R(\mathbb{C}, T) \end{matrix} \right\rangle$: *The escrow account generation is a probabilistic protocol. Each party takes as input* $\mathbb{C}$ *and* T*. The protocol outputs the public identifier of the escrow account epk, private information* aux_S *(which allows to execute the refund path), private information* aux_R *(which allows to execute the redeem path), the redeem transaction* tx_{red}*, the refund transaction* tx_{ref}*, or* $\perp$.
- $\{\sigma_{\mathsf{red}}, \perp\} \leftarrow \mathsf{Auth}_{\mathsf{red}}(\mathsf{aux}_R, w, tx_{\mathsf{red}})$: *The redeem transaction authorization is a PPT algorithm, that takes as input the private information for redeem* aux_R*, the witness to the hard relation* w*, and the redeem transaction* tx_{red}*, and outputs the redeem transaction authorization* σ_{red} *or* $\perp$.
- $\{\sigma_{\mathsf{ref}}, \perp\} \leftarrow \mathsf{Auth}_{\mathsf{ref}}(\mathsf{aux}_S, tx_{\mathsf{ref}})$: *The refund transaction authorization PPT algorithm, that takes as input the private information for refund* aux_S *and the refund transaction* tx_{ref}*, and outputs the refund transaction authorization* σ_{ref} *or* $\perp$.
- $\{w, \perp\} \leftarrow \mathsf{GWit}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, \mathsf{aux}_S)$: *The Witness extraction DPT algorithm, takes as input the redeem transaction* tx_{red}*, the redeem transaction authorization* σ_{red}*, and the private information for refund* aux_S*, and outputs a witness to the hard relation* w *or* $\perp$.
- $1/0 \leftarrow \mathsf{isAuth}_{\mathsf{CL}}(\{(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}), (tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}})\}, epk)$: *The authorization verification DPT algorithm for redeem and refund transactions, takes as input a redeem* $(tx_{\mathsf{red}}, \sigma_{\mathsf{red}})$ *or refund* $(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}})$ *transaction-authorization pair and the public identifier of the escrow account, epk, and outputs* 1 *if the authorization is valid or* 0 *otherwise.*

The ctEAcc protocol models the redeem and refund paths for the escrow account, as well as the creation of the escrow account itself in which the coins must be locked with a basic transaction from the sender. This mechanism is shown in Section V. Note that there is no verification algorithm for ctEAcc. The reasons for this are two-fold: (i) the protocol creates the escrow account and sets up the redeem and refund process, but does not transfer coins to the escrow account; and (ii) the protocol can be aborted by any of the two parties, so we implicitly require that S and R check its validity during execution and abort it if needed.

Definition 5 (CL correctness). *A CL is said to be* correct *if for all* $\lambda \in \mathbb{N}$*, all* $(\mathbb{C}, w) \in \mathcal{R}$*, all* $T \leftarrow \mathtt{createT}(\cdot)$*, all* $\left\langle \begin{matrix} (epk, \mathsf{aux}_S, tx_{\mathsf{red}}, tx_{\mathsf{ref}}), \\ (epk, \mathsf{aux}_R, tx_{\mathsf{red}}, tx_{\mathsf{ref}}) \end{matrix} \right\rangle \leftarrow \mathsf{ctEAcc}\left\langle \begin{matrix} S(\mathbb{C}, T), \\ R(\mathbb{C}, T) \end{matrix} \right\rangle$*, all* $\sigma_{\mathsf{red}} \leftarrow \mathsf{Auth}_{\mathsf{red}}(\mathsf{aux}_R, w, tx_{\mathsf{red}})$*, all* $w' \leftarrow \mathsf{GWit}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, \mathsf{aux}_S)$ *the following condition is satisfied:*

$$\Pr\left[\begin{matrix} \mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, epk) = 1 \\ (\mathbb{C}, w') \in \mathcal{R} \end{matrix}\right] = 1$$

$\mathsf{redForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda)$

$(T, \mathsf{st}_0) \leftarrow \mathcal{A}(1^\lambda)$
$(\mathbb{C}, w) \leftarrow \mathsf{createR}(1^\lambda)$
$\left\langle \begin{matrix} (epk, \mathsf{aux}_{\mathcal{A}}, tx_{\mathsf{red}}, tx_{\mathsf{ref}}, \mathsf{st}_1), \\ (epk, \mathsf{aux}_R, \mathsf{aux}_{\mathcal{A}}, tx_{\mathsf{red}}, tx_{\mathsf{ref}}) \end{matrix} \right\rangle \leftarrow \mathsf{ctEAcc} \left\langle \begin{matrix} \mathcal{A}(\mathbb{C}, \mathsf{st}_0), \\ R(\mathbb{C}, T) \end{matrix} \right\rangle$
$(tx^*_{\mathsf{red}}, \sigma^*_{\mathsf{red}}) \leftarrow \mathcal{A}(\mathsf{st}_1)$
$\tau \leftarrow \mathtt{readTime}(\cdot)$
$b_0 := \mathsf{isAuth}_{\mathsf{CL}}(tx^*_{\mathsf{red}}, \sigma^*_{\mathsf{red}}, epk)$
$b_1 := \tau < T$
$b_2 := tx^*_{\mathsf{red}} \neq tx_{\mathsf{red}}$
$\mathbf{return}\ b_0 \wedge b_1 \wedge b_2$

$\mathsf{ExpRedeem}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda)$

$(\mathbb{C}, w, T, \mathsf{st}_0) \leftarrow \mathcal{A}(1^\lambda)$
$\left\langle \begin{matrix} (epk, \mathsf{aux}_{\mathcal{A}}, tx_{\mathsf{red}}, tx_{\mathsf{ref}}, \mathsf{st}_1), \\ (epk, \mathsf{aux}_R, \mathsf{aux}_{\mathcal{A}}, tx_{\mathsf{red}}, tx_{\mathsf{ref}}) \end{matrix} \right\rangle \leftarrow \mathsf{ctEAcc} \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_0), \\ R(\mathbb{C}, T) \end{matrix} \right\rangle$
$\sigma_{\mathsf{red}} \leftarrow \mathsf{Auth}_{\mathsf{red}}(\mathsf{aux}_R, w, tx_{\mathsf{red}})$
$b_0 := \mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, epk) = 0$
$b_1 := (\mathbb{C}, w) \in \mathcal{R}$
$\mathbf{return}\ b_0 \wedge b_1$

$\mathsf{ExpExtract}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda)$

$(\mathbb{C}, T, \mathsf{st}_0) \leftarrow \mathcal{A}(1^\lambda)$
$\left\langle \begin{matrix} (epk, \mathsf{aux}_S, \mathsf{aux}_{\mathcal{A}}, tx_{\mathsf{red}}, tx_{\mathsf{ref}}), \\ (epk, \mathsf{aux}_{\mathcal{A}}, tx_{\mathsf{red}}, tx_{\mathsf{ref}}, \mathsf{st}_1) \end{matrix} \right\rangle \leftarrow \mathsf{ctEAcc} \left\langle \begin{matrix} S(\mathbb{C}, T), \\ \mathcal{A}(\mathsf{st}_0) \end{matrix} \right\rangle$
$\sigma_{\mathsf{red}} \leftarrow \mathcal{A}(\mathsf{st}_1)$
$w \leftarrow \mathsf{GWit}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, \mathsf{aux}_S)$
$b_0 := \mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, epk)$
$b_1 := (\mathbb{C}, w) \notin \mathcal{R}$
$\mathbf{return}\ b_0 \wedge b_1$

$\mathsf{refForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda)$

$(\mathbb{C}, T, \mathsf{st}_0) \leftarrow \mathcal{A}(1^\lambda)$
$\left\langle \begin{matrix} (epk, \mathsf{aux}_S, \mathsf{aux}_{\mathcal{A}}, tx_{\mathsf{red}}, tx_{\mathsf{ref}}), \\ (epk, \mathsf{aux}_{\mathcal{A}}, tx_{\mathsf{red}}, tx_{\mathsf{ref}}, \mathsf{st}_1) \end{matrix} \right\rangle \leftarrow \mathsf{ctEAcc} \left\langle \begin{matrix} S(\mathbb{C}, T), \\ \mathcal{A}(\mathsf{st}_0) \end{matrix} \right\rangle$
$(tx^*_{\mathsf{ref}}, \sigma^*_{\mathsf{ref}}) \leftarrow \mathcal{A}(\mathsf{st}_1)$
$b_0 := tx^*_{\mathsf{ref}} \notin \{tx_{\mathsf{red}}, tx_{\mathsf{ref}}\}$
$b_1 := \mathsf{isAuth}_{\mathsf{CL}}(tx^*_{\mathsf{ref}}, \sigma^*_{\mathsf{ref}}, epk)$
$\mathbf{return}\ b_0 \wedge b_1$

$\mathsf{ExpRefund}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda)$

$(T, \mathsf{st}_0) \leftarrow \mathcal{A}(1^\lambda)$
$(\mathbb{C}, w) \leftarrow \mathsf{createR}(1^\lambda)$
$\left\langle \begin{matrix} (epk, \mathsf{aux}_S, \mathsf{aux}_{\mathcal{A}}, tx_{\mathsf{red}}, tx_{\mathsf{ref}}), \\ (epk, \mathsf{aux}_{\mathcal{A}}, tx_{\mathsf{red}}, tx_{\mathsf{ref}}, \mathsf{st}_1) \end{matrix} \right\rangle \leftarrow \mathsf{ctEAcc} \left\langle \begin{matrix} S(\mathbb{C}, T), \\ \mathcal{A}(\mathbb{C}, \mathsf{st}_0) \end{matrix} \right\rangle$
$\sigma_{\mathsf{ref}} \leftarrow \mathsf{Auth}_{\mathsf{ref}}(\mathsf{aux}_S, tx_{\mathsf{ref}})$
$\mathbf{return}\ \mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}, epk) = 0$

Fig. 2. Experiments for CL redeem unforgeability, CL redeemability, CL extractability, CL refund unforgeability, and CL refundability.

and, in addition to previous preconditions, for all $\sigma_{\mathsf{ref}} \leftarrow \mathsf{Auth}_{\mathsf{ref}}(\mathsf{aux}_S, tx_{\mathsf{ref}})$*, all* $\tau \leftarrow \mathtt{readTime}(\cdot)$ *such that* $\tau \geq T$ *the following is satisfied:*

$$\Pr\left[\mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}, epk) = 1\right] = 1$$

Security. We consider the additional security goals of CL with respect to BL, formally described in Fig. 2.

While the objective of the security in BL is to protect the simple accounts, here the main goal is to guarantee that the escrow account can only be spent either by the redeem or refund paths. Therefore, the properties must guarantee that the private information generated in protocol ctEAcc, namely, aux_S and aux_R, are the only method that enables the redeem and refund paths. Note that in an escrow account, only one of a redeem authorization σ_{red} or refund authorization σ_{ref} will be added to the ledger's transaction list TX_L. Therefore, we model the escrow account as a one time account.

CL redeem unforgeability ensures that an honest receiver of an escrow account is the only party that can generate a valid authorization for a redeem transaction $(tx_{\mathsf{red}}, \sigma_{\mathsf{red}})$. We model this property by allowing $\mathcal{A}$ to choose the timeout (T), while the challenger generates the instance of the hard relation $(\mathbb{C}, w)$. The adversary wins if she can forge a redeem authorization for a redeem transaction different to the one outputted by ctEAcc before the timeout expires. Note that if the forged authorization is in the same redeem transaction generated by ctEAcc, the adversary is not stealing coins from the honest party. We explicitly introduce the time constraint since there are some instances of CL in which tx_{red} and tx_{ref} are in the same message space (see Section IV-F). In this setting, without the time constraint, a trivial way for the adversary to win this game would be to wait until T expires, make a refund transaction and its authorization, and use them as a forgery for a redeem transaction. In order to correctly model the security property, we force the adversary to come with a forgery *before* the timeout T expires.

CL redeemability ensures that an honest receiver of an escrow account can always generate a valid authorization for a redeem transaction $(tx_{\mathsf{red}}, \sigma_{\mathsf{red}})$. We model this property by allowing $\mathcal{A}$ to choose all the inputs of ctEAcc, including a valid instance of the hard relation $(\mathbb{C}, w) \in \mathcal{R}$ and then engage with the challenger in ctEAcc to generate *epk*, aux, and transactions tx_{red} and tx_{ref}. The adversary wins if the honest party fails to generate a valid redeem authorization σ_{red}, while the witness and statement provided by the adversary are in the hard relation.

CL extractability ensures that an honest sender of an escrow account can always obtain a w from $(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, \mathsf{aux}_S)$ such that $(\mathbb{C}, w) \in \mathcal{R}$. We model this property by allowing $\mathcal{A}$ to choose all the inputs of ctEAcc and then engage with the challenger to generate *epk*, aux, and transactions tx_{red} and tx_{ref}. The adversary wins if she can provide a redeem authorization for the redeem transaction outputted by the ctEAcc protocol such that the honest party cannot obtain a witness for the statement committed in the escrow account *epk*.

CL refund unforgeability ensures that an honest sender of an escrow account is the only party that can generate a valid authorization for a refund transaction $(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}})$. We model this property by allowing $\mathcal{A}$ to choose a timeout T and a condition $\mathbb{C}$. Then, they both engage in ctEAcc to generate *epk*, aux, tx_{red} and tx_{ref}. The adversary wins if it is able to generate a refund authorization in a transaction different to

tx_{ref} or tx_{red}. Note that if the forged authorization is in the refund transaction generated by ctEAcc, the adversary is not stealing coins from the honest party. We explicitly exclude also tx_{red} as there are instances of CL in which tx_{red} and tx_{ref} belong to the same message space (see Section IV-F), which would allow the adversary to trivially win by simply forwarding the redeem transaction, since she knows the witness, w. We do not consider this as an attack, since an honest sender with knowledge of the w, should be able to generate a valid authorization for the redeem transaction.

CL refundability ensures that an honest sender of an escrow account can always generate a valid authorization for a refund transaction $(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}})$. We model this property by allowing $\mathcal{A}$ to choose all the inputs of ctEAcc. Then, they both engage in ctEAcc to generate *epk*, aux and the redeem and refund transactions. The adversary wins if the honest party fails to generate a valid refund authorization σ_{ref} for the refund transaction generated with ctEAcc.

The thoughtful reader might be considering if an honest sender can be assured that the receiver cannot generate a redeem transaction without knowledge of the witness. We answer to this affirmatively; such an adversary can be used to break the hard relation in which the escrow account is based, and as such, cannot exist if the hard relation assumption holds. Appendix J provides further details on this point.

We use the knowledge of secret key model (KOSK [13]) in which the adversary outputs its (possibly maliciously chosen) set of keys. For CL, $\mathcal{A}$ outputs $\mathsf{aux}_{\mathcal{A}}$ to the challenger after running ctEAcc. We assume that during the execution of ctEAcc protocol S and R generate some set of keys that later they will store in aux_S and aux_R. During the execution of the protocol, S and R prove that they know the secret key for the sets of keys that allow to spend coins from *epk*. However, we do not write these proofs explicitly in any of the constructions we shown in Section IV.

A CL is secure if all the security properties hold. We formally define CL security in Definition 6.

Definition 6 (CL security)**.** *Let* $G := \{\mathsf{redForge}, \mathsf{refForge}, \mathsf{ExpRedeem}, \mathsf{ExpExtract}, \mathsf{ExpRefund}\}$ *be the games defined in Fig. 2. A CL is secure if for every* $G_i \in G$ *and 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* $\Pr[G_i(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

IV. Constructions for BL and CL

In the following section we provide constructions for BL and CL assuming that the predicates portrayed in Table I are provided by the ledger operator. CL is an extension of BL. Therefore, it is expected that a ledger that is compatible with a CL construction is also compatible with another BL construction. Similarly, some ledgers, like Bitcoin or Ethereum, may support $\mathtt{IsAuth}_{\mathtt{Rel}}$ and $\mathtt{IsAuth}_{\mathtt{T}}$. In these ledgers, it is possible to build protocols that rely on those two predicates, that rely only on one of them ($\mathtt{IsAuth}_{\mathtt{T}}$) or that do not rely on any of those predicates. For example, the construction shown in Section IV-E is based on PTLC, which is regarded as a cheaper alternative (in terms of transaction cost) to HTLC transactions in Bitcoin. With the five constructions introduced in this section, we show that at least 15 popular ledgers can be described with the primitives defined in Section III.

A. Building Blocks

Digital Signature Scheme. We require a digital signature scheme $\Pi_{DS} :=$ (KGen, Sig, Vf), where: (i) KGen gets as input 1^λ and outputs a key pair (pk, sk); (ii) Sig gets as input *sk* and a message m, and outputs a signature σ; and (iii) Vf gets as input *pk*, the message m, and the signature σ, and outputs a bit b. We assume that Π_{DS} is correct (i.e. it holds that $\Pr[\mathsf{Vf}(pk, m, \mathsf{Sig}(sk, m)) = 1] = 1$) and secure under the standard notion of existential unforgeability under chosen message attack (EUF-CMA), which we restate in Appendix B.

Non Interactive Zero Knowledge. We require a Non Interactive Zero Knowledge (NIZK) system, which for input x in language $\mathcal{L}$, with witness ω, is a set of efficient algorithms $\Pi_{Vf} := (\mathsf{SetUp}, \mathsf{Prove}, \mathsf{Vf})$, where: (i) SetUp takes as input the security parameter 1^λ and outputs a random public string CRS; (ii) Prove takes as input x, ω, and CRS, and outputs a proof π; and (iii) Vf takes as input x, CRS, and π, and outputs $1/0$ to accept or reject the proof. We assume that Π_{Vf} is correct (i.e. it holds that $\Pr[\mathsf{Vf}(\mathsf{x}, \mathsf{Prove}(\mathsf{x}, \omega, \mathsf{CRS}), \mathsf{CRS}) = 1] = 1$ for all $\mathsf{CRS} \leftarrow \mathsf{SetUp}(1^\lambda)$). It is also secure under the notions of knowledge-soundness and zero knowledge.

Verifiable Timed Dlog (VTD). For a general description of the protocol, we require a verifiable timed commitment scheme for any hard relation. We define this scheme in Appendix D. However, instances of such a scheme do not exist for arbitrary hard relations. Efficient implementations are known when the hard relation is the discrete logarithm [57], [59], which are the ones we use in our construction in Section IV-F. We restate in the following the properties of the Verifiable Timed Dlog, for the group $\mathbb{G}$ with generator G and order q is a tuple $\Pi_{VTD} := (\mathsf{Commit}, \mathsf{Verify}, \mathsf{Open}, \mathsf{forceOpen})$: (i) Commit takes as input parameter T and the witness w, and outputs a puzzle P and a proof π; (ii) Verify takes as input the puzzle P, T, the proof π, and the public statement $\mathbb{C}$, and outputs $1/0$ to accept or reject the proof; (iii) Open is run by the creator of the puzzle, it takes as input the puzzle P of hardness T, and outputs the solution to the puzzle w and the randomness of the puzzle r; and (iv) forceOpen takes as input the puzzle P, and outputs the solution to the puzzle w. We assume Π_{VTD} to be correct (i.e. $\Pr[\mathsf{forceOpen}(\mathsf{Commit}(w, T)) = w] = 1$) and secure under the notions of Timed Privacy and Soundness. Timed privacy ensures that for all PRAM[1] algorithms whose running time is at most t (where $\mathrm{t} < T$) the probability of success in extracting w from the puzzle P is negligible. Soundness guarantees that a user holding a puzzle P of hardness T will be able to run forceOpen to obtain w in time T. For a detailed description of the security properties, we refer the reader to [57], [59] or to Appendix D, where we restate the properties, and generalize them for any hard relation.

Two-party Signature with Aggregatable Public Keys. We require a two-party signature scheme with aggregatable public keys Π_{DSA} :=(Setup, KGen, Π_{Sig}, KAgg, Vf), where: (i) Setup takes as input the security parameter 1^λ and outputs public parameters pp; (ii) KGen takes as input pp, and outputs a key pair (pk, sk); (iii) Π_{Sig} is an interactive protocol between all signers, which takes as input their private keys and the

[1]PRAM algorithms are polynomial time algorithms that have polynomially bounded amount of parallel computing power.

TABLE I. CONSTRUCTIONS PREDICATES SUPPORTED BY THE LEDGER AND EXAMPLES. UNDERLINED LEDGERS ARE NOT CBDC.

Construction	`IsAuth`	Predicate Assumptions: `IsFunded`	`IsValid`	`IsUnique`	$\texttt{IsAuth}_{\texttt{Own}}$	$\texttt{IsAuth}_{\texttt{Rel}}$	$\texttt{IsAuth}_{\texttt{T}}$	Ledgers compatible with constructions
Section IV-B	BL	✓	✓	✓	✓	×	×	KSICash [27]; TIPS+ [62]; Project Hamilton [39]; Monero [45];Bakong [44]; itCoin [11], [62]; Iberpay Smart Money [35]; Bitcoin [43]; Ethereum [64]; Hyperledger Iroha [36]; XRP ledger [66]; Tezos [31]
Section IV-C	BL	✓	✓	✓	✓	×	×	UTT [60]; Zcash [50]; Tezos [31]
Section IV-D	CL	✓	✓	✓	✓	✓	✓	Bakong [44]; itCoin [11], [62]; Iberpay Smart Money [35]; Bitcoin [43]; Ethereum [64]; Hyperledger Iroha [36]; XRP ledger [66]; Tezos [31]
Section IV-E	CL	✓	✓	✓	✓	✓	×	Project Hamilton [39]; Monero [45];Bakong [44]; itCoin [11], [62]; Iberpay Smart Money [35]; Bitcoin [43]; Ethereum [64]; Hyperledger Iroha [36]; XRP ledger [66]; Tezos [31]
Section IV-F	CL	✓	✓	✓	✓	×	×	KSICash [27]; Project Hamilton [39]; Monero [45];Bakong [44]; itCoin [11], [62]; Iberpay Smart Money [35]; Bitcoin [43]; Ethereum [64]; Hyperledger Iroha [36]; XRP ledger [66]; Tezos [31]

messages, and outputs a signature (σ); (iv) KAgg takes as input the public keys of all signers and generates the aggregated public key; and (v) Vf takes as input the aggregated public key, the message m, and the signature σ, and outputs a bit b. We assume that Π_{DSA} is correct (i.e. it holds that $\Pr[\mathsf{Vf}(\mathsf{KAgg}(pk_0, pk_1), m, \Pi_{\mathsf{Sig}} < (sk_0, m), (sk_1, m) >) = 1] = 1$) and secure under the notion of two party existential unforgeability under chosen message attack (2-EUF-CMA), as described in [21].

Two-party Adaptor Signature with Aggregatable Public Keys. We require a two-party adaptor signature scheme with aggregatable public keys defined over a two-party signature scheme with aggregatable public keys Π_{aDSA} :=(Π_{pSig}, pVf, Adapt, Extract), where: (i) Π_{pSig} is an interactive protocol between all signers, which takes as input their private keys, the message, and the public statement, and outputs a presignature ($\hat{\sigma}$); (ii) pVf takes as input the presignature $\hat{\sigma}$, the public statement, the message, and the aggregated public key, and outpus a bit b; (iii) Adapt takes as input the presignature $\hat{\sigma}$ and the witness w, and outputs a signature σ; (iv) Extract takes as input the signature σ, the presignature $\hat{\sigma}$, and the public statement, and returns a witness w. We assume that Π_{aDSA} is correct and secure under the notions of two party existential presignature unforgeability under chosen message attack (2-aEUF-CMA), two party presignature adaptability and two party presignature extractability, as described in [21]. We restate these properties in Appendix C.

B. BL Authorization from Digital Signatures

We now provide a generic construction of `IsAuth` for BL w.r.t. a digital signature scheme and prove that it satisfies BL unforgeability. Note that to give a complete construction of a ledger with BL authorization, it is also required to provide a construction for `IsFunded`, `IsUnique` and `IsValid`.

Building Blocks. We require a digital signature scheme that is correct and existentially unforgeable under chosen message attacks, as described in Section IV-A. We assume that the message space of the digital signature scheme coincides with $\mathrm{Supp}(\mathsf{ctTx}_{\mathsf{bsc}}(\cdot,\cdot))$.

Construction. Our construction is depicted in Fig. 3. In particular, transactions are authorized by a signature created with the sender's signing key. Accordingly, the transaction validation (i.e., `IsAuth` predicate) checks whether the signature accompanying a transaction successfully verifies with respect to sender's public key.

$\underline{\mathsf{ctAcc}(1^\lambda):}$	$\underline{\mathsf{ctAcc}(1^\lambda):}$
$(pk, sk) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$	$(pk, sk) \leftarrow \mathsf{createR}(1^\lambda)$
$\textbf{return}\ \ (pk, sk)$	$\textbf{return}\ \ (pk, sk)$
$\underline{\mathsf{Auth}_{\mathsf{bsc}}(sk, tx_{\mathsf{bsc}}):}$	$\underline{\mathsf{Auth}_{\mathsf{bsc}}(sk, tx_{\mathsf{bsc}}):}$
$\sigma \leftarrow \Pi_{DS}.\mathsf{Sig}(sk, tx_{\mathsf{bsc}})$	$\mathsf{x} := \mathit{ParseSt}(tx_{\mathsf{bsc}});\ \omega := sk$
$\textbf{return}\ \sigma_{\mathsf{bsc}} := \sigma$	$\pi \leftarrow \Pi_{NIZK}.\mathsf{Prove}(\mathsf{x}, \omega, \mathsf{CRS})$
	$\textbf{return}\ \sigma_{\mathsf{bsc}} := \pi$
$\underline{\mathsf{isAuth}_{\mathsf{BL}}(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}}, pk):}$	$\underline{\mathsf{isAuth}_{\mathsf{BL}}(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}}, pk):}$
$\sigma := \sigma_{\mathsf{bsc}}$	$\mathsf{x} := \mathit{ParseSt}(tx_{\mathsf{bsc}});\ \pi := \sigma_{\mathsf{bsc}}$
$\textbf{return}\ \ \Pi_{DS}.\mathsf{Vf}(tx_{\mathsf{bsc}}, \sigma, pk)$	$\textbf{return}\ \ \Pi_{NIZK}.\mathsf{Vf}(\mathsf{x}, \pi, \mathsf{CRS})$

Fig. 3. BL authentication from a digital signature scheme (left) and a NIZK (right). We assume that CRS is a ledger parameter set at ledger setup and is known by all participants.

Security. We state below our formal security claims and defer the formal proof to Appendix E.

Theorem 1 (BL Unforgeability)**.** *Assume that the digital signature scheme is existentially unforgeable against chosen message attacks. Then, the construction in Fig. 3 offers BL unforgeabilty according to Definition 3.*

C. BL Authorization from NIZK

In this section, we provide a generic construction of `IsAuth` for BL w.r.t. a NIZK and prove that it satisfies BL unforgeability.

Building Blocks. We require a NIZK that is correct and provides soundness and zero-knowledge, as described in Section IV-A, and a hard relation, as described in Section III.

Construction. Our construction is depicted in Fig. 3. In our construction, we assume that the public statement required by the NIZK can be extracted from the transaction itself, and we denote it by $\mathsf{x} := \mathit{ParseSt}(tx_{\mathsf{bsc}})$. Later, we discuss how to realize this assumption with Zcash and UTT. Given that, our construction uses the Prove algorithm to authorize a transaction and Vf to check the validity of such authorization. This construction assumes that SetUp is run at ledger setup, and the CRS is publicly available.

Security. We state below our formal security claims and defer the formal proof to Appendix F.

Theorem 2 (BL Unforgeability)**.** *Assume that the NIZK provides knowledge soundness. Then, the construction in Fig. 3 offers BL unforgeabilty according to Definition 3.*

Examples. Both in UTT [60] and Zcash [50] transactions consist on a set of old coins, that are being burnt or destroyed, and newly minted coins. Transactions are accompanied by a proof for many statements (Zcash) or a set of proofs, one per statement (UTT). The statements refer to proving that (i) the input coins have sufficient coins (`IsFunded`), (ii) the input and output coins are well formed ($tx_{\mathsf{bsc}} \in TX_{\mathsf{bsc}}$), (iii) balance is preserved (`IsValid`), and (iv) the input coins belong to the sender (`IsAuth`). The ledger also verifies if the coins have been spent (`IsUnique`).

Note that the statement (iv) is the only one related to the transaction authorization (i.e., predicate `IsAuth`). Therefore, to map UTT and Zcash to our framework, we only consider the concrete statement-witness relation used in (iv).

D. CL from a Ledger with Condition and TimeLock Support

Bitcoin [43], Ethereum [64], Hyperledger Iroha [36], XRP Ledger [66], Tezos [31] and their CBDC counterparts, itCoin [11], [62]- based on Bitcoin's lightning network-, Iberpay Smart Money [35] -based on Ethereum contracts- and Bakong [36], [44] - based on Hyperledger Iroha-, are examples of ledgers with scripting capabilities that support $\texttt{IsAuth}_{\texttt{Rel}}$ and $\texttt{IsAuth}_{\texttt{T}}$.

The conditional payment capabilities are embodied by Hash TimeLock Contracts (HTLC) [49]. A HTLC allows to create a transaction in which a sender and a receiver agree to lock coins in an escrow address, under a $\mathbb{C}$ (a hash value) and a timelock. In order to spend the coins from this escrow account, there are two possible paths: (i) the witness of the public statement (e.g. the hash preimage) is provided, together with a signature of transaction valid under the receiver's secret key, or (ii) after the timelock, a signature under the sender's secret key is provided.

Building Blocks. For this construction, we require a digital signature scheme that is correct and existentially unforgeable under chosen message attacks, as described in Section IV-A. We assume that the message space of the digital signature scheme coincides with $\mathrm{Supp}(\mathsf{ctTx}_{\mathsf{bsc}}(\cdot,\cdot)) \cup \mathrm{Supp}(\mathsf{ctTx}_{\mathsf{red}}(\cdot,\cdot)) \cup \mathrm{Supp}(\mathsf{ctTx}_{\mathsf{ref}}(\cdot,\cdot))$. Furthermore, we assume the existence of a scripting language to enforce the *epk* paths for redeem and refund transactions. Note that the construction presented in this section does not rely only on a hash as the hard relation: any other hard relation can be supported (e.g. discrete logarithm).

Construction. In Section IV-B we proved that a ledger based on a digital signature algorithm is a secure BL. Given that, we focus here only on the specific functions that CL adds on top of BL. This is presented in Fig. 4.

Security. We now analyze the security properties of our construction. For properties CL redeem unforgeability and CL refund unforgeability we state below our formal security claims and defer the formal proof to Appendix G. Regarding CL redeemability, CL refundability and CL extractability, it is trivial to see that they hold. Both $\mathsf{Auth}_{\mathsf{red}}$ and $\mathsf{Auth}_{\mathsf{ref}}$ result in running $\Pi_{DS}.\mathsf{Sig}$ with knowledge of the appropriate secret key (and the witness for redeem). Due to the correctness of Π_{DS}, both algorithms will generate valid signatures, ensuring that CL redeemability and CL refundability hold. In the specific case of CL redeemability, the adversary provides the pair $(\mathbb{C}, w)$. Since the signature is valid due to correctness of Π_{DS}, the authorization will be valid as long as the statement and witness provided by the adversary are in the hard relation. For CL extractability, it is easy to see that a valid σ_{red} contains a valid witness w, due to the definition of σ_{red} (see Figure 4). Therefore, any valid σ_{red} in this setting provides a valid witness and CL extractability holds.

$\underline{\mathsf{ctEAcc}_S(\mathbb{C}, T)}$
$(pk_S, sk_S) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$
$(\ddot{pk}_S, \ddot{sk}_S) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$
$pk_R \leftarrow \mathsf{Exchange}(pk_S)$
$epk := (pk_R, \mathbb{C}), (pk_S, T)$
$tx_{\mathsf{ref}} \leftarrow \mathsf{ctTx}_{\mathsf{ref}}(epk, \ddot{pk}_S)$
$tx_{\mathsf{red}} \leftarrow \mathsf{Exchange}(tx_{\mathsf{ref}})$
$\mathsf{aux}_S := (sk_S, \ddot{sk}_S)$
$\textbf{return } (epk, \mathsf{aux}_S, tx_{\mathsf{red}}, tx_{\mathsf{ref}})$

$\underline{\mathsf{ctEAcc}_R(\mathbb{C}, T)}$
$(pk_R, sk_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$
$(\ddot{pk}_R, \ddot{sk}_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$
$pk_S \leftarrow \mathsf{Exchange}(pk_R)$
$epk := (pk_R, \mathbb{C}), (pk_S, T)$
$tx_{\mathsf{red}} \leftarrow \mathsf{ctTx}_{\mathsf{red}}(epk, \ddot{pk}_R)$
$tx_{\mathsf{ref}} \leftarrow \mathsf{Exchange}(tx_{\mathsf{red}})$
$\mathsf{aux}_R := (sk_R, \ddot{sk}_R)$
$\textbf{return } (epk, \mathsf{aux}_R, tx_{\mathsf{red}}, tx_{\mathsf{ref}})$

$\underline{\mathsf{Auth}_{\mathsf{red}}(\mathsf{aux}_R, w, tx_{\mathsf{red}})}$
$(sk_R, \cdot) \leftarrow \mathsf{aux}_R$
$\sigma \leftarrow \Pi_{DS}.\mathsf{Sig}(sk_R, tx_{\mathsf{red}})$
$\textbf{return } (\sigma, w)$

$\underline{\mathsf{Auth}_{\mathsf{ref}}(\mathsf{aux}_R, tx_{\mathsf{ref}})}$
$(sk_S, \cdot) \leftarrow \mathsf{aux}_S$
$\textbf{return } \Pi_{DS}.\mathsf{Sig}(sk_S, tx_{\mathsf{ref}})$

$\underline{\mathsf{GWit}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, \mathsf{aux}_S)}$
$(\cdot, w) \leftarrow \sigma_{\mathsf{red}}$
$\textbf{return } w$

$\underline{\mathsf{isAuth}_{\mathsf{CL}}(\{(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}), (tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}})\}, epk)}$
$(pk_R, \mathbb{C}), (pk_S, T) \leftarrow epk$
$\textbf{if } tx_{\mathsf{red}}$
$\quad (\sigma, w) \leftarrow \sigma_{\mathsf{red}}$
$\quad \textbf{return } \texttt{IsAuth}_{\texttt{Rel}}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, epk) \wedge \Pi_{DS}.\mathsf{Vf}(tx_{\mathsf{red}}, \sigma, pk_R)$
$\textbf{else if } tx_{\mathsf{ref}}$
$\quad \textbf{return } \texttt{IsAuth}_{\texttt{T}}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}, epk) \wedge \Pi_{DS}.\mathsf{Vf}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}, pk_S)$
$\textbf{else return } 0$

Fig. 4. Implementation of CL functionalities using HTLC. We denote the exchange of information between S and R with Exchange().

Theorem 3 (CL redeem unforgeability)**.** *Assume that that the digital signature scheme is unforgeable. Then, the our protocol offers CL redeem unforgeability according to Definition 6.*

Theorem 4 (CL refund unforgeability)**.** *Assume that that the digital signature scheme is unforgeable. Then, our protocol offers CL refund unforgeability according to Definition 6.*

E. CL from a Ledger with TimeLock Support

The ledgers discussed in Section IV-D can also support contracts that do not explicitly check $\texttt{IsAuth}_{\texttt{Rel}}$ and only use the scripting capabilities to ensure $\texttt{IsAuth}_{\texttt{T}}$. An example of this type of contract is Point TimeLock Contracts (PTLC) [46]. A PTLC allows to create a transaction in which a sender and a receiver agree to lock coins in an escrow address, under a condition $\mathbb{C}$ (a point in the elliptic curve) and a timelock. In order to spend the coins from this escrow account there are two possible paths: (i) the witness of the public statement (e.g. the discrete log of the point) is used to adapt a presignature on the escrow address into a valid signature, or (ii) after the timelock, a signature under the sender's secret key is provided.

Building Blocks. We require a digital signature scheme, a two party digital signature scheme with aggregatable public keys, and a two party adaptor signature scheme with aggregatable public keys, as described in Section IV-A. We assume that the message space of the digital signature scheme coincides with $\mathrm{Supp}(\mathsf{ctTx}_{\mathsf{bsc}}(\cdot,\cdot)) \cup \mathrm{Supp}(\mathsf{ctTx}_{\mathsf{red}}(\cdot,\cdot)) \cup \mathrm{Supp}(\mathsf{ctTx}_{\mathsf{ref}}(\cdot,\cdot))$. Finally, we need a timelock scripting capability to prevent the refund path of the escrow account until T has expired.

Construction. In Section IV-B we proved that a ledger based on a digital signature algorithm is a secure BL. In Section IV-D we proved that a ledger based on a digital signature algorithm and some scripting capabilities to encode the escrow account is a secure CL. We present in Figure 5 our construction. Note that $\mathsf{Auth}_{\mathsf{ref}}$ is equivalent to that of Section IV-D.

Security. For properties CL redeem unforgeability, CL redeemability and CL extractability we state below our formal security claims and defer the formal proof to Appendix H. Regarding CL refund unforgeability and CL refundability since $\mathsf{Auth}_{\mathsf{ref}}$ is the same as in Section IV-D, these properties hold for the same reasons. Note that the security reduction is the same because pk_S^{ref} is a different key to the one used for the redeem path $pk_{S,R}$.

Theorem 5 (CL redeem unforgeability)**.** *Assume that the two party adaptor signature scheme satisfies 2-aEUF-CMA Security. Then, our protocol offers CL redeem unforgeability according to Definition 6.*

Theorem 6 (CL redeemability)**.** *Assume that the two party adaptor signature scheme satisfies two-party pre-signature adaptability. Then, our protocol offers CL redeemability according to Definition 6.*

Theorem 7 (CL extractability)**.** *Assume that the two party adaptor signature scheme satisfies two-party witness extractability. Then, our protocol offers CL extractability according to Definition 6.*

F. Emulating a CL with a BL

We now consider how to create a CL when neither $\texttt{IsAuth}_{\texttt{Rel}}$ and $\texttt{IsAuth}_{\texttt{T}}$ are available. Effectively, this means that the underlying ledger is of type BL. Therefore, in this section we *emulate* the properties of a CL building protocols on top of a BL. In a way, the emulation of a CL from a BL could be considered an extension of Section IV-E that allows to cryptographically timelock the generation of the refund authorization.

$\underline{\mathsf{ctEAcc}_S(\mathbb{C},T)}$	$\underline{\mathsf{ctEAcc}_R(\mathbb{C},T)}$
$(pk_S, sk_S) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$	$(pk_R, sk_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$
$(pk_S^{\mathsf{ref}}, sk_S^{\mathsf{ref}}) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$	
$(\ddot{pk}_S, \ddot{sk}_S) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$	$(\ddot{pk}_R, \ddot{sk}_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$
$pk_R \leftarrow \mathsf{Exchange}(pk_S, pk_S^{\mathsf{ref}})$	$pk_S, pk_S^{\mathsf{ref}} \leftarrow \mathsf{Exchange}(pk_R)$
$pk_{S,R} \leftarrow \mathsf{KAgg}(pk_S, pk_R)$	$pk_{S,R} \leftarrow \mathsf{KAgg}(pk_S, pk_R)$
$epk := ((pk_{S,R}), (pk_S^{\mathsf{ref}}, T))$	$epk := ((pk_{S,R}), (pk_S^{\mathsf{ref}}, T))$
$tx_{\mathsf{ref}} \leftarrow \mathsf{ctTx}_{\mathsf{ref}}(pk_{S,R}, \ddot{pk}_S)$	$tx_{\mathsf{red}} \leftarrow \mathsf{ctTx}_{\mathsf{red}}(pk_{S,R}, \ddot{pk}_R)$
$tx_{\mathsf{red}} \leftarrow \mathsf{Exchange}(tx_{\mathsf{ref}})$	$tx_{\mathsf{ref}} \leftarrow \mathsf{Exchange}(tx_{\mathsf{red}})$
$\hat{\sigma} \leftarrow \Pi_{\mathsf{pSig}}(sk_S, tx_{\mathsf{red}}, \mathbb{C})$	$\hat{\sigma} \leftarrow \Pi_{\mathsf{pSig}}(sk_R, tx_{\mathsf{red}}, \mathbb{C})$
$a_S := \mathsf{pVf}(tx_{\mathsf{red}}, \mathbb{C}, \hat{\sigma}) = 0$	$a_R := \mathsf{pVf}(tx_{\mathsf{red}}, \mathbb{C}, \hat{\sigma}) = 0$
if a_S **abort**	**if** a_R **abort**
$\mathsf{aux}_S := (sk_S, sk_S^{\mathsf{ref}}, \ddot{sk}_S, \hat{\sigma}, \mathbb{C})$	$\mathsf{aux}_R := (sk_R, \ddot{sk}_R, \hat{\sigma})$
return $(epk, \mathsf{aux}_S, tx_{\mathsf{red}}, tx_{\mathsf{ref}})$	**return** $(epk, \mathsf{aux}_R, tx_{\mathsf{red}}, tx_{\mathsf{ref}})$

$\underline{\mathsf{Auth}_{\mathsf{red}}(\mathsf{aux}_R, w, tx_{\mathsf{red}})}$
$(\cdot,\cdot,\hat{\sigma}) \leftarrow \mathsf{aux}_R$
return $\mathsf{Adapt}(\hat{\sigma}, w)$

$\underline{\mathsf{Auth}_{\mathsf{ref}}(\mathsf{aux}_R, tx_{\mathsf{ref}})}$
$(\cdot, sk_S^{\mathsf{ref}}, \cdot,\cdot,\cdot) \leftarrow \mathsf{aux}_S$
return $\Pi_{DS}.\mathsf{Sig}(sk_S^{\mathsf{ref}}, tx_{\mathsf{ref}})$

$\underline{\mathsf{GWit}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, \mathsf{aux}_S)}$
$(\cdot,\cdot,\cdot,\hat{\sigma},\mathbb{C}) \leftarrow \mathsf{aux}_S$
return $\mathsf{Extract}(\sigma_{\mathsf{red}}, \hat{\sigma}, \mathbb{C})$

$\underline{\mathsf{isAuth}_{\mathsf{CL}}(\{(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}), (tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}})\}, epk)}$
$((pk_{S,R}), (pk_S^{\mathsf{ref}}, T)) \leftarrow epk$
if tx_{red}
  return $\Pi_{DS}.\mathsf{Vf}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, pk_{S,R})$
else if tx_{ref}
  return $\texttt{IsAuth}_{\texttt{T}}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}, epk) \wedge \Pi_{DS}.\mathsf{Vf}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}, pk_S^{\mathsf{ref}})$
else return 0

Fig. 5. Implementation CL functionalities of PTLC. Note that $\mathsf{Auth}_{\mathsf{ref}}$ is exactly the same as in HTLC (see Figure 4). We denote the exchange of information between S and R with Exchange().

Building Blocks. For this construction, we require a digital signature scheme, a two party digital signature scheme with aggregatable public keys, a two party adaptor signature scheme with aggregatable public keys, and a verifiable timed dlog as described in Section IV-A. We assume that the message space of all three type of transactions is the same. Therefore, the message space of the digital signature scheme coincides with $\mathrm{Supp}(\mathsf{ctTx}_{\mathsf{bsc}}(\cdot,\cdot)) = \mathrm{Supp}(\mathsf{ctTx}_{\mathsf{red}}(\cdot,\cdot)) = \mathrm{Supp}(\mathsf{ctTx}_{\mathsf{ref}}(\cdot,\cdot))$.

Construction. In Section IV-B we proved that a ledger based on a digital signature algorithm is a secure BL. In Section IV-E we proved the CL security of a ledger based on an identification scheme. This scheme is similar to the PTLC, but with the added complexity of having two presignatures. We present our construction in Figure 6.

$\underline{\mathsf{ctEAcc}_S(\mathbb{C}, T):}$	$\underline{\mathsf{ctEAcc}_R(\mathbb{C}, T):}$
$(pk_S, sk_S) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$	$(pk_R, sk_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$
$(\ddot{pk}_S, \ddot{sk}_S) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$	$(\ddot{pk}_R, \ddot{sk}_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$
$pk_R \leftarrow \mathsf{Exchange}(pk_S)$	$pk_S \leftarrow \mathsf{Exchange}(pk_R)$
$pk_{S,R} \leftarrow \mathsf{KAgg}(pk_S, pk_R)$	$pk_{S,R} \leftarrow \mathsf{KAgg}(pk_S, pk_R)$
$epk := pk_{S,R}$	$epk := pk_{S,R}$
$tx_{\mathsf{ref}} \leftarrow \mathsf{ctTx}_{\mathsf{ref}}(pk_{S,R}, \ddot{pk}_S)$	$tx_{\mathsf{red}} \leftarrow \mathsf{ctTx}_{\mathsf{red}}(pk_{S,R}, \ddot{pk}_R)$
$tx_{\mathsf{red}} \leftarrow \mathsf{Exchange}(tx_{\mathsf{ref}})$	$tx_{\mathsf{ref}} \leftarrow \mathsf{Exchange}(tx_{\mathsf{red}})$
$\hat{\sigma}_{\mathsf{red}} \leftarrow \Pi_{\mathsf{pSig}}(sk_S, tx_{\mathsf{red}}, \mathbb{C})$	$\hat{\sigma}_{\mathsf{red}} \leftarrow \Pi_{\mathsf{pSig}}(sk_R, tx_{\mathsf{red}}, \mathbb{C})$
$a_S := \mathsf{pVf}(tx_{\mathsf{red}}, \mathbb{C}, \hat{\sigma}_{\mathsf{red}})$	$a_R := \mathsf{pVf}(tx_{\mathsf{red}}, \mathbb{C}, \hat{\sigma}_{\mathsf{red}})$
if $a_S = 0$ **abort**	**if** $a_R = 0$ **abort**
	$(\mathbb{C}^{\mathsf{ref}}, w^{\mathsf{ref}}) \leftarrow \mathsf{createR}(1^\lambda)$
	$(P, \pi) \leftarrow \mathsf{Commit}(T, w^{\mathsf{ref}})$
$\mathsf{Receive}(P, \pi, \mathbb{C}^{\mathsf{ref}})$	$\mathsf{Send}(P, \pi, \mathbb{C}^{\mathsf{ref}})$
$a_S := \mathsf{Vf}(P, \pi, \mathbb{C}^{\mathsf{ref}}, T)$	
if $a_S = 0$ **abort**	
$\hat{\sigma}_{\mathsf{ref}} \leftarrow \Pi_{\mathsf{pSig}}(sk_S, tx_{\mathsf{ref}}, \mathbb{C}^{\mathsf{ref}})$	$\hat{\sigma}_{\mathsf{ref}} \leftarrow \Pi_{\mathsf{pSig}}(sk_R, tx_{\mathsf{ref}}, \mathbb{C}^{\mathsf{ref}})$
$a_S := \mathsf{pVf}(tx_{\mathsf{ref}}, \mathbb{C}^{\mathsf{ref}}, \hat{\sigma}_{\mathsf{ref}})$	$a_R := \mathsf{pVf}(tx_{\mathsf{ref}}, \mathbb{C}^{\mathsf{ref}}, \hat{\sigma}_{\mathsf{ref}})$
if $a_S = 0$ **abort**	**if** $a_R = 0$ **abort**
$\mathsf{aux}_S := (sk_S, \ddot{sk}_S, \hat{\sigma}_{\mathsf{red}}$ $\hat{\sigma}_{\mathsf{ref}}, \mathbb{C}, P)$	$\mathsf{aux}_R := (sk_R, \ddot{sk}_R, \hat{\sigma}_{\mathsf{red}})$
return $(epk, \mathsf{aux}_S, tx_{\mathsf{red}}, tx_{\mathsf{ref}})$	**return** $(epk, \mathsf{aux}_R, tx_{\mathsf{red}}, tx_{\mathsf{ref}})$

$\underline{\mathsf{Auth}_{\mathsf{red}}(\mathsf{aux}_R, w, tx_{\mathsf{red}})}$
$(\cdot, \cdot, \hat{\sigma}_{\mathsf{red}}) \leftarrow \mathsf{aux}_R$
return $\mathsf{Adapt}(\hat{\sigma}_{\mathsf{red}}, w)$

$\underline{\mathsf{Auth}_{\mathsf{ref}}(\mathsf{aux}_S, tx_{\mathsf{ref}})}:$
$(\cdot, \cdot, \hat{\sigma}_{\mathsf{ref}}, \cdot, P) \leftarrow \mathsf{aux}_S$
$w^{\mathsf{ref}} \leftarrow \mathsf{forceOpen}(P)$
return $\mathsf{Adapt}(\hat{\sigma}_{\mathsf{ref}}, w^{\mathsf{ref}})$

$\underline{\mathsf{GWit}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, \mathsf{aux}_S)}$
$(\cdot, \cdot, \hat{\sigma}, \mathbb{C}, \cdot) \leftarrow \mathsf{aux}_S$
return $\mathsf{Extract}(\sigma_{\mathsf{red}}, \hat{\sigma}, \mathbb{C})$

$\underline{\mathsf{isAuth}_{\mathsf{CL}}(\{(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}), (tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}})\}, epk)}:$
$pk_{S,R} \leftarrow epk$
return $\Pi_{DS}.\mathsf{Vf}(tx_i, \sigma_i, pk_{S,R})$

Fig. 6. Emulation of a CL from a BL. $\mathsf{Auth}_{\mathsf{red}}$ and GWit are implemented as in Figure 5. We denote the bilateral exchange of information $\mathsf{Exchange}()$ and unilateral sharing of information with $\mathsf{Send}()$ and $\mathsf{Receive}()$.

Security. We now analyze the security properties of our construction. For properties CL redeem unforgeability, CL refund unforgeability, CL redeemability, CL refundability and CL extractability we state below our formal security claims and defer the formal proof to Appendix I.

Theorem 8 (CL redeem unforgeability)**.** *Assume that the two party adaptor signature scheme satisfies 2-aEUF-CMA security and the verifiable timed dlog guarantees privacy. Then, our protocol offers CL redeem unforgeability according to Definition 6.*

Theorem 9 (CL refund unforgeability)**.** *Assume that the two party adaptor signature scheme satisfies two-party pre-signature adaptability. Then, our protocol offers CL redeemability according to Definition 6.*

Theorem 10 (CL redeemability)**.** *Assume that the two party adaptor signature scheme satisfies two-party pre-signature adaptability. Then, our protocol offers CL redeemability according to Definition 6.*

Theorem 11 (CL refundability)**.** *Assume that verifiable timed dlog is sound and that the two party adaptor signature scheme satisfies two-party pre-signature adaptability. Then, our protocol offers CL refundability according to Definition 6.*

Theorem 12 (CL extractability)**.** *Assume that the two party adaptor signature scheme satisfies two-party witness extractability. Then, our protocol offers CL extractability according to Definition 6.*

V. Atomic Cross-Ledger Coin Swaps

In this section, we showcase how to use our cryptographic framework to describe cross-ledger applications, such as atomic coin swaps.

A cross-ledger swap involves two ledgers L_0, L_1 and two users, Alice and Bob. Alice holds α coins on L_0 controlled by the account (sk_A, pk_A) whereas Bob holds β coins on L_1 controlled by the account (sk_B, pk_B). The swap problem consists in ensuring that Alice transfers α to Bob in L_0 if and only if Bob transfers β to Alice in L_1. Such a swap is *atomic* if it results in one of the following outcomes (i) Bob owning α in L_0 and Alice owning β in L_1 (i.e., coin swap); or (ii) Alice recovering α in L_0 and Bob recovering β in L_1 (i.e., coin refund).

Instead, we hereby show how to use our framework to design a financial application (i.e., a cross-ledger coin swap in this case) that is agnostic to the concrete transaction authorization mechanism of the underlying ledger. This demonstrates the utility of our framework in designing cross-ledger protocols. Since we have defined two types of ledger, we discuss the coin swap between two CL (Section V-A) and also between a BL and a CL (Section V-B).

A. Atomic Cross-Ledger Swap between CL Ledgers

Protocol. The protocol is detailed in Fig. 7. Note that in blue we detail the interactions of the users with the CL ledgers. The protocol starts with a commit phase where parties agree on the instance of the hard relation, the timeouts and the escrow accounts parameters for the coin swap. This phase ends with both parties sending coins to the escrow accounts in both ledgers with the $tx^0_{\mathsf{bsc}}, tx^1_{\mathsf{bsc}}$ transactions, respectively.

At this point, the protocol can enter either in the release or the refund phase. In the former, Alice can authorize and submit tx^1_{red} to L_1 and then Bob can retrieve the coins from L_0 using the CL functionality. In the latter, Alice and Bob can both recover their coins from the previously created escrow accounts authorizing the corresponding refund transactions, after their timeouts, T^0 and T^1 have expired.

Security intuition. Alice pays in L_0 in exchange for a payment from Bob in L_1. During the commit phase, Alice engages with Bob to generate and fund two escrow accounts, one in L_0 and the other in L_1.

Regarding Alice, if they proceed to the release phase, we know due to the property CL redeemability that Alice will generate a valid authorization for the redeem transaction in L_1. Furthermore, due to the property CL redeem unforgeability, she is the only party that can generate a redeem transaction authorization. Alternatively, if they proceed to the refund stage, due to the property CL refundability, she will generate a valid authorization for the refund transaction in L_0. Additionally, due to CL refund unforgeability, she is the only party that can generate a refund transaction authorization.

Regarding Bob, if they proceed to the release phase, we know due to the property CL extractability, he will obtain a correct witness w if Alice is able to generate a valid redeem transaction in L_1. We also know that due to CL redeemability, Bob will generate a valid authorization for the redeem transaction in L_0, since he knows the correct witness. Furthermore, due to the property CL redeem unforgeability, he is the only party that can generate a redeem transaction authorization. On the other hand, if they proceed to the refund stage, due to the property CL refundability, he will generate a valid authorization for the refund transaction in L_1. Moreover, due to CL refund unforgeability, he is the only party that can generate a refund transaction authorization.

Discussion. CL provides the functionality to commit coins into an escrow account between two parties, which can then be either released or refunded. Atomic swap protocols built on top of CL would only require to argue about: (i) how to link the different conditions $\mathbb{C}$ for the created escrow accounts, and (ii) how to set the timeouts, T, in order to guarantee that either all parties fulfil the payment, or all parties can be refunded.

For example, in the running example of the coin swap, since there are only two ledgers, there is no threat of a wormhole attack [42]. As such, both escrow accounts could be created with the same $(\mathbb{C}, w) \in \mathcal{R}$. If the two ledgers are defined w.r.t different hard relations, then the sender should provide two statements, $\mathbb{C}$, $\mathbb{C}'$, one for each hard relation, and prove that both have the same witness w. Finally, the timeouts need to be set in such a way that, even if the sender waits until the last moment to trigger the redeem path, there is enough time to run GWit to learn the witness w and use it in the second ledger. Hence, the timeout T of the second ledger should be larger than the one of the first ledger.

B. Atomic Cross-Ledger Swap between CL and BL Ledgers

Building Blocks. Aside from both ledgers, we require an additional building block that allows to (i) verifiably encrypt the witness of a statement in a given hard relation; and (ii) allow to decrypt the data if and only if an oracle has attested that a given event has occurred. Hereafter, we define two security primitives that allow us to meet the above requirements:

Witness Encryption Based on Signatures. A witness encryption based on signatures [40] is a cryptographic primitive defined with respect to signature scheme $\Pi_{DS} := (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$.

$\mathsf{Alice}(sk_A, pk_A, \Delta)$	$\mathsf{Bob}(sk_B, pk_B, \Delta)$
Commit Phase	
$(\mathbb{C}, w) \leftarrow \mathsf{createR}(1^\lambda)$	$T^1 \leftarrow \mathtt{createT}(\cdot)$
$\mathsf{Send}(\mathbb{C})$	$\mathsf{Receive}(\mathbb{C})$
$\mathsf{Receive}(T^1)$	$\mathsf{Send}(T^1)$
$T^0 := T^1 + \Delta$	$T^0 := T^1 + \Delta$
$(epk^0, \mathsf{aux}^0_S, tx^0_{\mathsf{red}}, tx^0_{\mathsf{ref}})$	$(epk^0, \mathsf{aux}^0_R, tx^0_{\mathsf{red}}, tx^0_{\mathsf{ref}})$
$\leftarrow \mathsf{ctEAcc}^{L_0} \langle S(\mathbb{C}, T^0) \rangle$	$\leftarrow \mathsf{ctEAcc}^{L_0} \langle R(\mathbb{C}, T^0) \rangle$
$(epk^1, \mathsf{aux}^1_R, tx^1_{\mathsf{red}}, tx^1_{\mathsf{ref}})$	$(epk^1, \mathsf{aux}^1_S, tx^1_{\mathsf{red}}, tx^1_{\mathsf{ref}})$
$\leftarrow \mathsf{ctEAcc}^{L_1} \langle S(\mathbb{C}, T^1) \rangle$	$\leftarrow \mathsf{ctEAcc}^{L_1} \langle R(\mathbb{C}, T^1) \rangle$
$tx^0_{\mathsf{bsc}} \leftarrow \mathsf{ctTx}^{L_0}(pk_A, epk^1)$	$tx^1_{\mathsf{bsc}} \leftarrow \mathsf{ctTx}^{L_1}(pk_B, epk^1)$
$\sigma^0_{\mathsf{bsc}} \leftarrow \mathsf{Auth}^{\mathsf{L_0}}_{\mathsf{bsc}}(sk_A, tx^0_{\mathsf{bsc}})$	$\sigma^1_{\mathsf{bsc}} \leftarrow \mathsf{Auth}^{\mathsf{L_1}}_{\mathsf{bsc}}(sk_B, tx^1_{\mathsf{bsc}})$
$\mathsf{Send}(tx^0_{\mathsf{bsc}})$	$\mathsf{Receive}(tx^0_{\mathsf{bsc}})$
$\mathsf{Receive}(tx^1_{\mathsf{bsc}})$	$\mathsf{Send}(tx^1_{\mathsf{bsc}})$
$\mathsf{subTx}^{L_0}(tx^0_{\mathsf{bsc}}, \sigma^0_{\mathsf{bsc}})$	
	if $tx^0_{\mathsf{bsc}} \notin L_0$ **abort**
	$\mathsf{subTx}^{L_1}(tx^1_{\mathsf{bsc}}, \sigma^1_{\mathsf{bsc}})$
Release phase (before T^1)	
$\sigma^1_{\mathsf{red}} \leftarrow \mathsf{Auth}^{\mathsf{L_1}}_{\mathsf{red}}(\mathsf{aux}^1_R, w, tx^1_{\mathsf{red}})$	
$\mathsf{subTx}^{L_1}(tx^1_{\mathsf{red}}, \sigma^1_{\mathsf{red}})$	
	$w \leftarrow \mathsf{GWit}^{L_1}(tx^1_{\mathsf{red}}, \sigma^1_{\mathsf{red}}, \mathsf{aux}^1_S)$
	$\sigma^0_{\mathsf{red}} \leftarrow \mathsf{Auth}^{\mathsf{L_0}}_{\mathsf{red}}(\mathsf{aux}^0_R, w, tx^0_{\mathsf{red}})$
	$\mathsf{subTx}^{L_0}(tx^0_{\mathsf{red}}, \sigma^0_{\mathsf{red}})$
Refund phase (after T^1)	
$\sigma^0_{\mathsf{ref}} \leftarrow \mathsf{Auth}^{\mathsf{L_0}}_{\mathsf{ref}}(\mathsf{aux}^0_S, tx^0_{\mathsf{ref}})$	$\sigma^1_{\mathsf{ref}} \leftarrow \mathsf{Auth}^{\mathsf{L_1}}_{\mathsf{ref}}(\mathsf{aux}^1_S, tx^1_{\mathsf{ref}})$
$\mathsf{subTx}^{L_0}(tx^0_{\mathsf{ref}}, \sigma^0_{\mathsf{ref}})$	$\mathsf{subTx}^{L_1}(tx^1_{\mathsf{ref}}, \sigma^1_{\mathsf{ref}})$

Fig. 7. Atomic cross-ledger swap between CL ledgers. subTx refers to the action of sending a transaction to the ledger.

It consists of two PPT algorithms (Enc, Dec), which we restate in the following:

- $c \leftarrow \mathsf{Enc}((\overline{pk}, \overline{m}), m))$. The encryption algorithm takes as input a verification key pk of the signature scheme, a message $\hat{m}$, and the message to be encrypted m. It outputs a ciphertext c.
- $m \leftarrow \mathsf{Dec}(\overline{\sigma}, c)$. The decryption algorithm takes as input a signature $\overline{\sigma}$ (s.t. $\mathsf{Vf}(\overline{pk}, \overline{m}, \overline{\sigma}) = 1$) and the ciphertext c. It outputs a message m.

We assume it is correct and IND-CPA secure as defined in [40].

NIZK. A NIZK (as described in Section IV-A) for the following language:

$$\{(c, \mathbb{C}, \overline{pk}, \overline{tx}_{\mathsf{bsc}}) \in \mathcal{L} : \exists\ w \text{ s.t.}$$
$$c \leftarrow \mathsf{Enc}((\overline{pk}, \overline{tx}_{\mathsf{bsc}}), w) \wedge (\mathbb{C}, w) \in \mathcal{R}\}$$

Assumptions. We assume an oracle service, similar to those already deployed in practice (e.g., [15], [19], [40]), that check whether a transaction tx_{bsc} has appeared in the BL ledger and attest it with a signature $\overline{\sigma} \leftarrow \mathsf{Sig}(\overline{sk}, \overline{tx}_{\mathsf{bsc}})$. This oracle publishes these signatures in a bulletin board.

$\underline{\mathsf{Alice}}(sk_A, pk_A)$	$\underline{\mathsf{Bob}}(sk_B, pk_B)$
Commit Phase	
$(pk'_A, sk'_A) \leftarrow \mathsf{KGen}^{\mathsf{BL}}(1^\lambda)$	$T \leftarrow \texttt{createT}(\cdot)$
$\mathsf{Send}(pk_A, pk'_A)$	$\mathsf{Receive}(pk_A, pk'_A)$
$\mathsf{Receive}(pk_B, T)$	$\mathsf{Send}(pk_B, T)$
$(\mathbb{C}, w) \leftarrow \mathsf{createR}(1^\lambda)$	
$tx^{\mathsf{BL}}_{\mathsf{bsc}} \leftarrow \mathsf{ctTx}^{\mathsf{BL}}(pk_B, pk'_A)$	
$c \leftarrow \mathsf{Enc}((pk_{\mathcal{O}}, tx^{\mathsf{BL}}_{\mathsf{bsc}}), w)$	
$\pi \leftarrow \mathsf{Prove}(c, pk_{\mathcal{O}}, tx^{\mathsf{BL}}_{\mathsf{bsc}}, w)$	
$\mathsf{Send}(tx^{\mathsf{BL}}_{\mathsf{bsc}}, c, \pi, \mathbb{C})$	$\mathsf{Receive}(tx^{\mathsf{BL}}_{\mathsf{bsc}}, c, \pi, \mathbb{C})$
	$a := \mathsf{Vf}(\pi, c, \mathbb{C}, pk_{\mathcal{O}}, tx^{\mathsf{BL}}_{\mathsf{bsc}})$
	if $a = 0$ **abort**
$(epk^{\mathsf{CL}}, \mathsf{aux}^{\mathsf{CL}}_S, tx^{\mathsf{CL}}_{\mathsf{red}}, tx^{\mathsf{CL}}_{\mathsf{ref}})$ $\leftarrow \mathsf{ctEAcc}^{\mathsf{CL}}\langle S(\mathbb{C}, T)\rangle$	$(epk^{\mathsf{CL}}, \mathsf{aux}^{\mathsf{CL}}_R, tx^{\mathsf{CL}}_{\mathsf{red}}, tx^{\mathsf{CL}}_{\mathsf{ref}})$ $\leftarrow \mathsf{ctEAcc}^{\mathsf{CL}}\langle R(\mathbb{C}, T)\rangle$
$tx^{\mathsf{CL}}_{\mathsf{bsc}} \leftarrow \mathsf{ctTx}^{\mathsf{CL}}(pk_A, epk)$	
$\mathsf{Send}(tx^{\mathsf{CL}}_{\mathsf{bsc}})$	$\mathsf{Receive}(tx^{\mathsf{CL}}_{\mathsf{bsc}})$
$\sigma^{\mathsf{CL}}_{\mathsf{bsc}} \leftarrow \mathsf{Auth}^{\mathsf{CL}}_{\mathsf{bsc}}(sk_A, tx^{\mathsf{CL}}_{\mathsf{bsc}})$	
$\mathsf{subTx}^{\mathsf{CL}}(tx^{\mathsf{CL}}_{\mathsf{bsc}}, \sigma^{\mathsf{CL}}_{\mathsf{bsc}})$	
	if $tx^{\mathsf{CL}}_{\mathsf{bsc}} \notin TX_{\mathsf{CL}}$ **abort**
Release phase (before T)	
	$\sigma^{\mathsf{BL}}_{\mathsf{bsc}} \leftarrow \mathsf{Auth}^{\mathsf{BL}}_{\mathsf{bsc}}(sk_B, tx^{\mathsf{BL}}_{\mathsf{bsc}})$
	$\mathsf{subTx}^{\mathsf{BL}}(tx^{\mathsf{BL}}_{\mathsf{bsc}}, \sigma^{\mathsf{BL}}_{\mathsf{bsc}})$
	$\sigma_{\mathcal{O}} \leftarrow \mathsf{Read}\mathcal{O}(tx^{\mathsf{BL}}_{\mathsf{bsc}})$
	$w \leftarrow \mathsf{Dec}(\sigma_{\mathcal{O}}, c)$
	$\sigma^{\mathsf{CL}}_{\mathsf{red}} \leftarrow \mathsf{Auth}^{\mathsf{CL}}_{\mathsf{red}}(\mathsf{aux}^{\mathsf{CL}}_R, w, tx^{\mathsf{CL}}_{\mathsf{red}})$
	$\mathsf{subTx}^{\mathsf{CL}}(tx^{\mathsf{CL}}_{\mathsf{red}}, \sigma^{\mathsf{CL}}_{\mathsf{red}})$
Refund phase (after T)	
$\sigma^{\mathsf{CL}}_{\mathsf{ref}} \leftarrow \mathsf{Auth}^{\mathsf{CL}}_{\mathsf{ref}}(\mathsf{aux}^{\mathsf{CL}}_S, tx^{\mathsf{CL}}_{\mathsf{ref}})$	
$\mathsf{subTx}^{\mathsf{CL}}(tx^{\mathsf{CL}}_{\mathsf{ref}}, \sigma^{\mathsf{CL}}_{\mathsf{ref}})$	

Fig. 8. Payment synchronization protocol between BL and CL. subTx refers to the action of sending a transaction to the ledger, while $\mathsf{Read}\mathcal{O}$ refers to reading information from the bulletin board of the oracle.

Protocol. Assuming the existence of the oracle service, we can synchronize a payment between BL and CL. The idea is to encrypt w and make its decryption possible if and only if the oracle has attested that a specific transaction in BL has happened. We show the synchronization protocol in Figure 8. In blue, we mark the interactions of the users with the bulletin board of the oracle, the CL and the BL. In this manner, parties can read the attestation of the oracle ($\mathsf{Read}\mathcal{O}$), submit transactions to CL and BL ledgers (subTx), and read the ledger history (TX_{BL}, TX_{CL}). All parties know the public key of the oracle service $pk_{\mathcal{O}}$.

Security intuition. Alice pays in CL in exchange for a payment from Bob in BL. During the commit phase, Alice engages with Bob to generate one escrow account and agree on the transaction that she will receive from Bob in BL.

Regarding Alice, if they proceed to the release phase, we know due to the IND-CPA property of the encryption scheme, Bob should not be able to learn the witness w without sending the required transaction in BL. Furthermore, Bob is not able to redeem without knowledge of the w, as this implies he can break the hard relation. Alternatively, if they proceed to the refund stage Alice will generate a valid authorization for the refund transaction in L_0, due to the property CL refundability. Moreover, she is the only party that can generate a refund transaction authorization, due to CL refund unforgeability.

As regards to Bob, if they proceed to the release phase Alice should not be able to convince him that the cipher-text contains a valid witness w when in fact it does not, due to the knowledge-soundness property of the NIZK. Similarly, Bob, who knows a valid witness, will generate a valid authorization for the redeem transaction, due to CL redeemability. Additionally, he is the only party that can generate a redeem transaction authorization, due to the property CL redeem unforgeability. On the other hand, if they proceed to the refund stage, due to the property BL unforgeability, we know that Alice should not be able to spend from Bob's account without knowledge of the secret key of Bob.

Discussion. To the best of our knowledge, a construction that realizes the functionality of witness encryption based on signatures together with the corresponding NIZK has not been formally defined before. However, Section II of [40], outlines the design of an efficient cryptographic primitive that allows for the verifiable encryption of a discrete logarithm. This design relies on the Boneh-Franklin identity-based encryption [14] to realize the witness encryption based on signatures, and adopts ideas from the cut-and-choose technique used in the verifiable encryption scheme of Camenisch et al. [16] to achieve efficient verifiability for the encrypted message.

C. Other Applications

In this section, we showed as an example how to use our framework for atomic swaps. However, it can also be used as a cryptographic building blocks for other applications. We provide two more examples.

Coin Mixers. Some coin mixer protocols like the primitive blinded conditional signatures [30] are based on ledgers compatible with adaptor signatures, while others like Obscuro [61] or Teeseract [12] rely on trusted hardware. Again, the current situation seems to point to a heterogeneous set of protocols each tailored to the characteristics of some ledgers. As such, our cryptographic framework can be used to define coin mixing protocols using the CL definition, ensuring that any protocol devised with our framework could be compatible with any ledger that is proven a secure CL.

Collateralized Loans. Collateralized loans are a standard product offered by commercial banks, cryptocurrency exchanges, such as Binance, as well as standardized as an Ethereum smart contract in the ERC-3156. However, the aforementioned examples are tailored to the underlying assumptions. While the ERC-3156 is based on the scripting capabilities of the ethereum virtual machine, loans in an exchange or a bank are based on the assumption of a trusted party (although the regulation, guarantees and type of collateral required differs between commercial banks and cryptocurrency exchanges). We see as an interesting future work to model collateralized loans using the CL as a cryptographic building block, to provide a transversal protocol for any CL ledger.

VI. Discussion

Arbitrary Amounts. For readability, the definitions for BL and CL consider unitary transactions. Extending their definitions to consider arbitrary amounts is possible: the validation of the transaction amounts is considered in the predicates `IsFunded` and `IsValid`. Moreover, transactions with arbitrary amounts open the door for multiple receivers in basic, redeem and refund transactions. In particular, one could adapt Definition 1 and Definition 4 to allow for multiple receivers by defining the receiver accounts $\ddot{pk}$, as a list of accounts rather than a single one.

Nested Escrow Accounts. In CL (Definition 4), the creation of the escrow account protocol (ctEAcc) takes as input a public statement and a timeout. This implicitly requires parties to generate the receiving accounts ($\ddot{pk}$) for tx_{red} and tx_{ref} inside ctEAcc. However, inspection of our three CL constructions (c.f. Figs. 4 to 6) as well as the respective security proofs, shows that $\ddot{pk}$ is only used as a placeholder for the receiver of tx_{red} and tx_{ref}, and does not affect the security of our scheme. Hence, it can be also given as an input to ctEAcc. Therefore, if $\ddot{pk}$ are inputs to ctEAcc, it is possible to provide the escrow account of another execution of ctEAcc as input, allowing to combine multiple layers of conditions. This enables for more advance functionalities for CL, also in combination with multiple receiving addresses. We leave this as an interesting future work direction.

Multisignature Wallets. Some ledgers (e.g. Bitcoin) support multisignature wallets. The constructions shown in Section IV-E and Section IV-F are based on aggregatable keys, but can be slightly modified to support multisignature wallets. Instead of an aggregatable public key, the epk can consist of two public keys (one for sender and another for receiver). Therefore, a complete authorization on epk would require two signatures, one for each of the keys. The redeem path could be based on single-key adaptor signatures [3] that unlock the signature that the receiver cannot generate on its own with the proper witness. On the other hand, the refund path for Section IV-F would also require a simple adaptor signature on the signature of the receiver, with the witness to that path (w^{ref}) timelocked in a VTD puzzle.

Privacy. Privacy is an important feature to consider in any payment system. However, in this work we focused on the security aspects of the BL and CL, as our main goal is to enable a universal methodology to build secure protocols based on cryptographic assumptions to enable the interoperability of BL and CL. We leave the definition of a privacy extension to our cryptographic layer as future work.

Fungibility. A desirable property in many cross ledger protocols is fungibility, or the inability to pinpoint a conditional transaction from all the transactions in the ledger. Note that this property is also related to transaction privacy. With our framework, one can check if a ledger can provide fungibility by assessing (i) whether tx_{bsc}, tx_{red} and tx_{ref} belong to the same message space; and (ii) if they are distributed equally in such space. In particular, a BL that emulates a CL following the construction shown in Section IV-F should achieve this notion.

Restrictions by a Central Bank. A central bank might want to prevent citizens from performing cross-ledger payments unless they are facilitated by a commercial bank. The reason being compliance with anti-money laundering regulation. In order to achieve so, there are several design options for the CBDC easily identified with our framework. We discuss two options: (i) require that the commercial bank holds partially or in full the secret key of the user; and (ii) design the CBDC ledger as a BL not compatible with adaptor signatures, and limited view on the transactions accepted to the ledger (i.e. each user can only see those in which they participate). Design option (i) grants power to the commercial bank to prevent any transaction not compliant with regulation from taking place. Design option (ii) prevents users from building protocols to emulate a CL, as described in Section IV-F. Furthermore, it also prevents that an oracle service attesting accepted as we described in Section V-B can be created. By introducing such limitations, the central bank might run the risk of designing a CBDC with features that are not sufficiently attractive for citizens to use.

VII. Conclusions

Interoperability is a fundamental problem in CBDC and cryptocurrencies alike. Yet, the approaches that exist are tailored either to the specific ledger features, or rely on trusted parties to facilitate exchanges. In this work, we proposed an alternative approach: decouple the transaction authorization details of each ledger from the definition of cross-ledger applications. To do so, we defined a middle layer that abstracts the core functionality for authorizing transactions in any ledger and the security notions of interest. This middle layer servers two purposes: (i) it becomes the main cryptographic building block to (re)define cross-ledger applications in a ledger agnostic manner; (ii) for any ledger that exists (and the new to come), it suffices to prove that it is a secure instance of the middle layer to be compatible with cross-ledger protocols.

In this work, we defined two new primitives for our cryptographic layer, namely, BL and CL. We proved that digital signatures and zero knowledge proofs are secure constructions for BL. We also proved that HTLC, PTLC and the combination of adaptor signatures with verifiable timelock puzzles are also secure constructions for CL. Finally, we discussed how to define some popular applications (e.g. atomic swaps between ledgers) using our cryptographic layer.

Acknowledgements. This work has been partially supported by PRODIGY Project (TED2021-132464B-I00) funded by MCIN/AEI/10.13039/501100011033/ and the European Union NextGenerationEU/PRTR

References

[1] "Polkadot: Vision for a heterogeneous multi-chain framework," 2016.

[2] R. Auer and R. Boehme, "Central bank digital currency: the quest for minimally invasive technology," Bank for International Settlements, BIS Working Papers 948, Jun. 2021. [Online]. Available: https://ideas.repec.org/p/bis/biswps/948.html

[3] 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 *ASIACRYPT 2021*. Berlin, Heidelberg: Springer-Verlag, 2021, p. 635–664. [Online]. Available: https://doi.org/10.1007/978-3-030-92075-3_22

[4] Bank of Canada, "Contingency Planning for a Central Bank Digital Currency," Bank of Canada webpage, 2020, accessed on 10.05.2022. [Online]. Available: https://www.bankofcanada.ca/2020/02/contingency-planning-central-bank-digital-currency/

[5] Bank of England, "Central bank digital currency: Oportunities, challenges and design," Bank of England Discussion Paper, 2019, accessed on 26.04.2022. [Online]. Available: https://www.bankofengland.co.uk/-/media/boe/files/paper/2020/central-bank-digital-currency-opportunities-challenges-and-design.pdf?la=en&hash=DFAD18646A77C00772AF1C5B18E63E71F68E4593

[6] Bank of International Settlements, "Central bank digital currencies: foundational principles and core features," BIS other publications, 2020, accessed on 26.04.2022. [Online]. Available: https://www.bis.org/publ/othp33.pdf

[7] ——, "Central bank digital currencies: financial stability implications," BIS other publications, 2021, accessed on 26.04.2022. [Online]. Available: https://www.bis.org/publ/othp42_fin_stab.pdf

[8] ——, "Central bank digital currencies: System design and interoperability," BIS other publications, 2021, accessed on 26.04.2022. [Online]. Available: https://www.bis.org/publ/othp42_system_design.pdf

[9] ——, "Central bank digital currencies: user needs and adoption," BIS other publications, 2021, accessed on 26.04.2022. [Online]. Available: https://www.bis.org/publ/othp42_user_needs.pdf

[10] ——, "Options for access to and interoperability of CBDCs for cross-border payments: Report to G20," BIS Innovation Hub, Other, 2022, accessed on 29.05.2025. [Online]. Available: https://www.bis.org/publ/othp52.pdf

[11] M. Benedetti, F. D. Sclavis, M. Favorito, G. Galano, S. Giammusso, A. Muci, and M. Nardelli, "A pow-less bitcoin with certified byzantine consensus," *CoRR*, vol. abs/2207.06870, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2207.06870

[12] I. Bentov, Y. Ji, F. Zhang, Y. Li, X. Zhao, L. Breidenbach, P. Daian, and A. Juels, "Tesseract: Real-time cryptocurrency exchange using trusted hardware," *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019.

[13] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme," in *Public Key Cryptography — PKC 2003*, Y. G. Desmedt, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 31–46.

[14] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology — CRYPTO 2001*, J. Kilian, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 213–229.

[15] L. Breidenbach, C. Cachin, B. Chan, A. Coventry, S. Ellis, A. Juels, F. Koushanfar, A. Miller, B. Magauran, D. Moroz *et al.*, "Chainlink 2.0: Next steps in the evolution of decentralized oracle networks," *Chainlink Labs*, vol. 1, 2021.

[16] J. Camenisch and I. Damgård, "Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes," in *Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ser. ASIACRYPT '00. Berlin, Heidelberg: Springer-Verlag, 2000, p. 331–345.

[17] Central Bank of Nigeria, "Design Paper for the eNaira," eNaira webpage, 2021, accessed on 26.04.2022. [Online]. Available: https://enaira.gov.ng/download/eNaira_Design_Paper.pdf

[18] Central Bank of the Bahamas, "Project Sand Dollar: A Bahamas Payments System Modernisation Initiative," Central bank of bahamas documents, 2019, accessed on 26.04.2022. [Online]. Available: https://www.centralbankbahamas.com/viewPDF/documents/2019-12-25-02-18-11-Project-Sanddollar.pdf

[19] P. Chakka, S. Joshi, A. Kate, J. Tobkin, and D. Yang, "Dora: Distributed oracle agreement with simple majority," *arXiv preprint arXiv:2305.03903*, 2023.

[20] D. Chaum, C. Grothoff, and T. Moser, "How to issue a central bank digital currency," Swiss National Bank Working Papers, 2021, accessed on 26.04.2022. [Online]. Available: https://www.snb.ch/n/mmr/reference/working_paper_2021_03/source/working_paper_2021_03.n.pdf

[21] 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.

[22] A. Erwig, S. Faust, S. Riahi, and T. Stöckert, "Commitee: An efficient and secure commit-chain protocol using tees," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 1486, 2020.

[23] European Central Bank, "A Digital Euro," Digital Euro, 2020, accessed on 26.04.2022. [Online]. Available: https://www.ecb.europa.eu/pub/pdf/other/Report_on_a_digital_euro~4d7268b458.en.pdf

[24] ——, "Digital euro experimentation scope and key learnings," Digital Euro, 2021, accessed on 26.04.2022. [Online]. Available: https://www.ecb.europa.eu/pub/pdf/other/ecb.digitaleuroscopekeylearnings202107~564d89045e.en.pdf

[25] ——, "Eurosystem report on the public consultation on a digital euro," Digital Euro, 2021, accessed on 26.04.2022. [Online]. Available: https://www.ecb.europa.eu/pub/pdf/other/Eurosystem_report_on_the_public_consultation_on_a_digital_euro~539fa8cd8d.en.pdf

[26] ——, "What is TARGET Instant Payment Settlement (TIPS)?" European Central Bank main webpage, 2022, accessed on 26.04.2022. [Online]. Available: https://www.ecb.europa.eu/paym/target/tips/html/index.en.html

[27] European Central Bank, Banco de España, Eesti Pank, Bank of Greece, Deutsche Bundesbank, Central Bank of Ireland, Latvijas Banka, Banca d'Italia, and De Nederlandsche Bank, "Work stream 3: A New Solution Blockchain and eID," Eesti Pank Varia, 2021, accessed on 26.04.2022. [Online]. Available: https://haldus.eestipank.ee/sites/default/files/2021-07/Work%20stream%203%20-%20A%20New%20Solution%20-%20Blockchain%20and%20eID_1.pdf

[28] European Central Bank and Bank of Japan, "Balancing confidenciality and auditability in a distributed ledger environment," STELLA project, 2019, accessed on 26.04.2022. [Online]. Available: https://www.ecb.europa.eu/paym/intro/publications/pdf/ecb.miptopical200212.en.pdf

[29] ——, "Synchronized cross-border payments," STELLA project, 2019, accessed on 26.04.2022. [Online]. Available: https://www.ecb.europa.eu/paym/intro/publications/pdf/ecb.miptopical190604.en.pdf

[30] 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

[31] L. M. Goodman, "Tezos : A self-amending crypto-ledger position paper," 2014.

[32] 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

[33] J. Gross, J. Sedlmeir, M. Babel, A. Bechtel, and B. Schellinger, "Designing a central bank digital currency with support for cash-like privacy," 2021.

[34] 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.

[35] Iberpay, "SMART MONEY Initiative: Preparations for the possible launch of a digital euro or bank digital money by the Spanish financial sector," Iberpay webpage, 2021, accessed on 24.11.2022. [Online]. Available: https://www.iberpay.es/media/21555/iberpay-report-on-smart-money-initiative-by-spanish-banking-sector.pdf

[36] H. Iroha, "Iroha 2 documentation," 2023. [Online]. Available: https://hyperledger.github.io/iroha-2-docs/

[37] A. Kiayias, M. Kohlweiss, and A. Sarencheh, "Peredi: Privacy-enhanced, regulated and distributed central bank digital currencies," 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. 1739–1752. [Online]. Available: https://doi.org/10.1145/3548606.3560707

[38] Z. Liu, A. Yang, J. Weng, T. Li, H. Zeng, and X. Liang, "Gmhl: Generalized multi-hop locks for privacy-preserving payment channel

networks," Cryptology ePrint Archive, Report 2022/115, 2022, https://ia.cr/2022/115.

[39] J. Lovejoy, C. Fields, M. Virza, T. Frederick, D. Urness, K. Karwaski, A. Brownworth, and N. Narula, "A High Performance Payment Processing System Designed for Central Bank Digital Currencies," Federal Reserve of Boston and Digital Currency Initiative, 2022, accessed on 26.04.2022. [Online]. Available: https://dam-prod.media.mit.edu/x/2022/02/04/Hamilton-Whitepaper-2022.pdf

[40] V. Madathil, S. A. K. Thyagarajan, D. Vasilopoulos, L. Fournier, G. Malavolta, and P. Moreno-Sanchez, "Cryptographic oracle-based conditional payments," 2023. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/cryptographic-oracle-based-conditional-payments/

[41] 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*, 2017, pp. 455–471.

[42] 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.

[43] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[44] National Bank of Cambodia, "Project Bakong: Next Generation Payment System," Bakong Project webpage, 2020, accessed on 26.04.2022. [Online]. Available: https://bakong.nbc.org.kh/download/NBC_BAKONG_White_Paper.pdf

[45] S. Noether, A. Mackenzie *et al.*, "Ring confidential transactions," *Ledger*, vol. 1, pp. 1–18, 2016.

[46] B. Optech, "Bitcoin optech newsletter 163. preparing for taproot 10: Ptlc," 2021. [Online]. Available: https://bitcoinops.org/en/newsletters/2021/08/25/#preparing-for-taproot-10-ptlcs

[47] ——, "Bitcoin optech newsletter 170." 2021. [Online]. Available: https://bitcoinops.org/en/newsletters/2021/10/13/#multiple-proposed-ln-improvements

[48] People's Bank of China, "Progress of Research and Development of E-CNY in China," People's Bank of China Press Release, 2021, accessed on 26.04.2022. [Online]. Available: http://www.pbc.gov.cn/en/3688110/3688172/4157443/4293696/2021071614584691871.pdf

[49] 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

[50] E. B. 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*. IEEE, 2014, pp. 459–474.

[51] Sveriges Riksbank, "E-krona project, report 1," E-krona reports, 2017, accessed on 26.04.2022. [Online]. Available: https://www.riksbank.se/globalassets/media/rapporter/e-krona/2017/rapport_ekrona_uppdaterad_170920_eng.pdf

[52] ——, "E-krona project, report 2," E-krona reports, 2018, accessed on 26.04.2022. [Online]. Available: https://www.riksbank.se/globalassets/media/rapporter/e-krona/2018/the-riksbanks-e-krona-project-report-2.pdf

[53] ——, "E-krona pilot phase 1," E-krona reports, 2021, accessed on 26.04.2022. [Online]. Available: https://www.riksbank.se/globalassets/media/rapporter/e-krona/2021/e-krona-pilot-phase-1.pdf

[54] ——, "E-krona pilot phase 2," E-krona reports, 2022, accessed on 26.04.2022. [Online]. Available: https://www.riksbank.se/globalassets/media/rapporter/e-krona/2022/e-krona-pilot-phase-2.pdf

[55] Swift. Cbdcs interoperability: 5 key takeaways from our ground-breaking experiments. Accessed: 2023-05-29. [Online]. Available: https://www.swift.com/news-events/news/cbdcs-interoperability-5-key-takeaways-our-ground-breaking-experiments

[56] 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.

[57] 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.

[58] S. A. K. Thyagarajan and G. Malavolta, "Lockable signatures for blockchains: Scriptless scripts for all signatures," *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 937–954, 2021.

[59] 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

[60] A. Tomescu, A. Bhat, B. Applebaum, I. Abraham, G. Gueta, B. Pinkas, and A. Yanai, "Utt: Decentralized ecash with accountable privacy," Cryptology ePrint Archive, Paper 2022/452, 2022, https://eprint.iacr.org/2022/452. [Online]. Available: https://eprint.iacr.org/2022/452

[61] M. Tran, L. Luu, M. S. Kang, I. Bentov, and P. Saxena, "Obscuro: A bitcoin mixer using trusted execution environments," in *Proceedings of the 34th Annual Computer Security Applications Conference*, 2018, pp. 692–701.

[62] E. Urbinati, A. Belsito, D. Cani, A. Caporrini, M. Capotosto, S. Folino, G. Galano, G. Goretti, Giancarloand Marcelli, P. Tiberi, and A. Vita, "A digital euro: a contribution to the discussion on technical design choices," Banca d'Italia, 2021, accessed on 26.04.2022. [Online]. Available: https://www.bancaditalia.it/pubblicazioni/mercati-infrastrutture-e-sistemi-di-pagamento/questioni-istituzionali/2021-010/N.10-MISP.pdf?language_id=1

[63] VISA, "Authorization and reversal processing requirements for merchant," VISA webpage, 2020, accessed on 10.10.2022. [Online]. Available: https://usa.visa.com/content/dam/VCOM/global/support-legal/documents/best-practices-authorization-and-reversal-processing.pdf

[64] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[65] K. Wüst, K. Kostiainen, N. Delius, and S. Capkun, "Platypus: A central bank digital currency with unlinkable transactions and privacy-preserving regulation," New York, NY, USA, p. 2947–2960, 2022. [Online]. Available: https://doi.org/10.1145/3548606.3560617

[66] XRPL Foundation. Xrp ledger developer resources: Escrow. Accessed: 2023-05-18. [Online]. Available: https://xrpl.org/escrow.html

[67] A. Zamyatin, D. Harz, J. Lind, P. Panayiotou, A. Gervais, and W. Knottenbelt, "Xclaim: Trustless, interoperable, cryptocurrency-backed assets," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 193–210.

APPENDIX

A. Interactive-BL (IBL)

While in most ledgers the sender is able to authorize the transaction without the intervention of the receiver (i.e. intervention other than sharing a public key or some randomness.) and submit it to the ledger, we found a CBDC ledger, Platypus [65], in which sender and receiver must interact in order for the transaction to be properly authorized and submitted to the ledger. Platypus is a version of interactive and transferable eCash. As such, we extend the definition of BL to accommodate to other CBDC ledgers that might follow Platypus an also require interactive transaction authorization.

Definition 7 (Interactive-Basic Payment Ledger (IBL))**.** *Interactive-Basic Payment ledger comprises the PPT algorithms* (ctAcc, $\mathsf{isAuth}_{\mathsf{BL}}$) *and the protocol* $\Pi_{\mathsf{Auth}_{\mathsf{bsc}}}$ *defined below:*

- $\underline{(pk, sk) \leftarrow \mathsf{ctAcc}(1^\lambda)}$: *The account creation algorithm. Takes as input the security parameter,* 1^λ *and outputs the public key pk and private key sk of the account.*
- $\sigma_{\mathsf{bsc}} \leftarrow \Pi_{\mathsf{Auth}_{\mathsf{bsc}}} \left\langle \begin{matrix} S(sk_S, tx_{\mathsf{bsc}}), \\ R(sk_R, tx_{\mathsf{bsc}}) \end{matrix} \right\rangle$ *The transaction authorization protocol. Is an interactive protocol between* S *and* R*. Each user takes as input their private key of the account, (*sk_S *for S and* sk_R *for R) and a basic transaction,* tx_{bsc} *and produces a basic transaction authorization,* σ_{bsc}.
- $\underline{1/0 \leftarrow \mathsf{isAuth}_{\mathsf{BL}}(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}})}$: *The authorization verification algorithm for basic transactions. It takes as input a basic transaction* tx_{bsc}*, a basic transaction authorization* σ_{bsc} *and outputs* 1 *if the authorization corresponds to the sender and receiver and* 0 *otherwise.*

Definition 8 (IBL Correctness)**.** *A IBL is said to be* correct *if for all* $\lambda \in \mathbb{N}$*, all* $(pk_S, sk_S) \leftarrow \mathsf{ctAcc}(1^\lambda)$*, all* $(pk_R, sk_R) \leftarrow \mathsf{ctAcc}(1^\lambda)$*, all* $tx_{\mathsf{bsc}} \leftarrow \mathsf{ctTx}_{\mathsf{bsc}}(pk_S, pk_R)$*, all* $\sigma_{\mathsf{bsc}} \leftarrow \Pi_{\mathsf{Auth}_{\mathsf{bsc}}} \left\langle \begin{matrix} S(sk_S, tx_{\mathsf{bsc}}), \\ R(sk_R, tx_{\mathsf{bsc}}) \end{matrix} \right\rangle$*, it holds that:*

$$\Pr\left[\mathsf{isAuth}_{\mathsf{BL}}(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}}) = 1\right] = 1$$

Security. The security properties are directed towards ensuring that knowledge of the secret key is required to generate a transaction involving any of the parties, as defined in Definition 9.

Definition 9 (IBL security)**.** *Let* $G := \{S\mathsf{bscForge}, R\mathsf{bscForge}\}$ *be the games defined in Fig. 9. A IBL is secure if for every* $G_i \in G$ *and 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* $\Pr[G_i(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

B. Digital Signature Scheme

A digital signature scheme consists on the tuple $\Pi_{DS} :=$ (KGen, Sig, Vf), where: (i) KGen gets as input 1^λ and outputs a key pair (pk, sk); (ii) Sig gets as input sk and a message m and outputs a signature σ; and (iii) Vf gets as input pk, the message m and the signature σ, and outputs a bit b. We assume that Π_{DS} is correct (i.e. it holds that $\mathsf{Vf}(pk, m, \mathsf{Sig}(sk, m)) = 1$) and secure under the standard notion of existential unforgeability under chosen message attack (EUF-CMA), which we restate in Figure 10.

$S\mathsf{bscForge}_{\Pi_{BL},\mathcal{A}}(\lambda)$
$\mathcal{Q} := \emptyset$
$(pk_S, sk_S) \leftarrow \mathsf{ctAcc}(1^\lambda)$
$(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}}) \leftarrow \mathcal{A}^{\mathcal{O}S\Pi_{\mathsf{Auth}_{\mathsf{bsc}}}}(pk_S)$
$b_0 := tx_{\mathsf{bsc}} \notin \mathcal{Q}$
$b_1 := \mathsf{isAuth}_{\mathsf{BL}}(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}})$
$\textbf{return } b_0 \wedge b_1$
$\mathcal{O}S\Pi_{\mathsf{Auth}_{\mathsf{bsc}}}(tx_{\mathsf{bsc}}, sk_R)$
$\sigma_{\mathsf{bsc}} \leftarrow \Pi_{\mathsf{Auth}_{\mathsf{bsc}}} \left\langle \begin{matrix} S(sk_S, tx_{\mathsf{bsc}}), \\ R(sk_R, tx_{\mathsf{bsc}}) \end{matrix} \right\rangle$
$\mathcal{Q} := \mathcal{Q} \cup \{tx_{\mathsf{bsc}}\}$
$\textbf{return } \sigma_{\mathsf{bsc}}$

Fig. 9. IBL security. We only show $S\mathsf{bscForge}$, since $R\mathsf{bscForge}$ is the same game, but changing S for R.

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

Fig. 10. Experiment for *EUF-CMA*.

C. Two party adaptor signature scheme with aggregatable public keys

A two-party adaptor signature scheme with aggregatable public keys [21] is defined w.r.t. a hard relation $\mathcal{R}$ and a two party signature scheme with aggregatable public keys Π_{DSA} :=(Setup, KGen, Π_{Sig}, KAgg, Vf). It is run between parties P_0 and P_1 and consists on the tupleΠ_{aDSA} :=(Π_{pSig}, pVf, Adapt, $\mathsf{Extract}$), where:

- Π_{pSig} is an interactive protocol between all signers, which takes as input their private keys, the message and the public statement and outputs a presignature ($\hat{\sigma}$);
- pVf takes as input the presignature $\hat{\sigma}$, the public statement, the message and the aggregated public key and outpus a bit b
- Adapt takes as input the presignature $\hat{\sigma}$ and the witness w and outputs a signature σ ;
- $\mathsf{Extract}$ takes as input the signature σ, the presignature $\hat{\sigma}$ and the public statement and returns a witness, w.

We assume that Π_{aDSA} is correct and secure under the notions of two party existential pre signature unforgeability under chosen message attack (2-aEUF-CMA), two party presignature adaptability and two party presignature extractability.

Definition 10 (2-aEUF-CMA)**.** *A two-party adaptor signature scheme with aggregatable public keys is said to offer 2-aEUF-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* $\Pr[\mathsf{aSigForge}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$*, where* aSigForge *is defined in Fig. 11.*

$$
\begin{array}{|l|}
\hline
\underline{\mathsf{aSigForge}(\lambda)} \\
\mathcal{Q} := \emptyset;\ pp \leftarrow \mathsf{Setup}(1^\lambda) \\
(pk_{1-b}, sk_{1-b}) \leftarrow \mathsf{ctAcc}(pp) \\
(pk_b, sk_b) \leftarrow \mathcal{A}(pp, pk_{1-b}) \\
m^* \leftarrow \mathcal{A}^{\mathcal{O}\Pi_{\mathsf{Sig}}, \mathcal{O}\Pi_{\mathsf{pSig}}}(pk_{1-b}, pk_b, sk_b) \\
(\mathbb{C}, w) \leftarrow \mathsf{createR}(1^\lambda) \\
\hat{\sigma} \leftarrow \Pi_{\mathsf{pSig}_{sk_{1-b}}}(m^*, \mathbb{C}) \\
\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}\Pi_{\mathsf{Sig}}, \mathcal{O}\Pi_{\mathsf{pSig}}}(\hat{\sigma}, \mathbb{C}) \\
b_0 := m \notin \mathcal{Q} \\
b_1 := \mathsf{Vf}(m^*, \sigma^*, \mathsf{KAgg}(pk_0, pk_1)) \\
\textbf{return } b_0 \wedge b_1 \\
\underline{\mathcal{O}\Pi_{\mathsf{Sig}}(m)} \\
\sigma \leftarrow \Pi_{\mathsf{Sig}_{sk_{1-b}}}(pk_0, pk_1, m) \\
\mathcal{Q} := \mathcal{Q} \cup \{m\} \\
\textbf{return } \sigma \\
\underline{\mathcal{O}\Pi_{\mathsf{pSig}}(m, \mathbb{C})} \\
\hat{\sigma} \leftarrow \Pi_{\mathsf{pSig}_{sk_{1-b}}}(pk_0, pk_1, m, \mathbb{C}) \\
\mathcal{Q} := \mathcal{Q} \cup \{m\} \\
\textbf{return } \hat{\sigma} \\
\hline
\end{array}
$$

Fig. 11. Experiment for 2-aEUF-CMA.

Definition 11 (Two-Party Pre-Signature Adaptability)**.** *A two-party adaptor signature scheme with aggregatable public keys is said to offer Two-Party Pre-Signature Adaptability if for all* $\lambda \in \mathbb{N}$*, messages* $m \in \{0,1\}^*$*, statement and witness pairs* $(\mathbb{C}, w) \in \mathcal{R}$*, publick keys* pk_0 *and* pk_1 *and presignature* $\hat{\sigma} \in \{0,1\}^*$ *that satisfies* $\mathsf{pVf}(m, \mathbb{C}, \hat{\sigma}, \mathsf{KAgg}(pk_0, pk_1)$*, we have that* $\Pr[\mathsf{Vf}(m, \mathsf{Adapt}(\hat{\sigma}, w), \mathsf{KAgg}(pk_0, pk_1) = 1] = 1$.

Definition 12 (Two Party Witness Extractability)**.** *A two-party adaptor signature scheme with aggregatable public keys is said to offer Two Party Witness 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* $\Pr[\mathsf{aWitExt}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$*, where* aWitExt *is defined in Fig. 12.*

$$
\begin{array}{|l|}
\hline
\underline{\mathsf{aWitExt}(\lambda)} \\
\mathcal{Q} := \emptyset;\ pp \leftarrow \mathsf{Setup}(1^\lambda) \\
(pk_{1-b}, sk_{1-b}) \leftarrow \mathsf{ctAcc}(pp) \\
(pk_b, sk_b) \leftarrow \mathcal{A}(pp, pk_{1-b}) \\
(m^*, \mathbb{C}^*) \\
\quad \leftarrow \mathcal{A}^{\mathcal{O}\Pi_{\mathsf{Sig}}, \mathcal{O}\Pi_{\mathsf{pSig}}}(pk_{1-b}, pk_b, sk_b) \\
\hat{\sigma} \leftarrow \Pi_{\mathsf{pSig}_{sk_{1-b}}}(m^*, \mathbb{C}^*) \\
\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}\Pi_{\mathsf{Sig}}, \mathcal{O}\Pi_{\mathsf{pSig}}}(\hat{\sigma}, \mathbb{C}) \\
w^* \leftarrow \mathsf{Extract}(\sigma^*, \hat{\sigma}, \mathbb{C}^*) \\
b_0 := m \notin \mathcal{Q} \\
b_1 := \mathsf{Vf}(m^*, \sigma^*, \mathsf{KAgg}(pk_0, pk_1)) \\
b_2 := (\mathbb{C}^*, w^*) \notin \mathcal{R} \\
\textbf{return } b_0 \wedge b_1 \wedge b_2 \\
\underline{\mathcal{O}\Pi_{\mathsf{Sig}}(m)} \\
\sigma \leftarrow \Pi_{\mathsf{Sig}_{sk_{1-b}}}(pk_0, pk_1, m) \\
\mathcal{Q} := \mathcal{Q} \cup \{m\} \\
\textbf{return } \sigma \\
\underline{\mathcal{O}\Pi_{\mathsf{pSig}}(m, \mathbb{C})} \\
\hat{\sigma} \leftarrow \Pi_{\mathsf{pSig}_{sk_{1-b}}}(pk_0, pk_1, m, \mathbb{C}) \\
\mathcal{Q} := \mathcal{Q} \cup \{m\} \\
\textbf{return } \hat{\sigma} \\
\hline
\end{array}
$$

Fig. 12. Experiment for Two Party Witness Extractability.

D. Verifiable Timed Commitment

A verifiable timed commitment scheme, $\Pi_{VTC} :=$ (Commit, Verify, Open, forceOpen) is defined with respect to a hard relation, $\mathcal{R}$, where $(\mathbb{C}, w) \in \mathcal{R}$ and: (i) Commit takes as input parameter T and the witness w and outputs a puzzle, P and a proof π; (ii) Verify takes as input the puzzle P, T, the proof π and the public statement $\mathbb{C}$ and outputs $1/0$ to accept or reject the proof; (iii) Open is run by the creator of the puzzle, takes as input the puzzle P of hardness T and outputs the solution to the puzzle, w and the randomness of the puzzle r; and (iv) forceOpen takes as input the puzzle P and outputs the solution to the puzzle, w. We assume Π_{VTC} to be correct (i.e. $\Pr[\mathsf{SolP}(\mathsf{GenP}(T, s)) = s] = 1$) and secure under the notions of Timed Privacy and Soundness, which are defined in [59]. We restate them below.

Definition 13 (Soundness)**.** *A VTC scheme* $\Pi_{VTC} :=$ (Commit, Verify, Open, forceOpen) *is sound if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all PPT adversaries* $\mathcal{A}$*, it holds that*

$$
\Pr\left[\begin{array}{c|c} b_1 = 1 & (\mathbb{C}, P, \pi, T) \leftarrow \mathcal{A}(1^\lambda) \\ b_2 = 0 & w \leftarrow \mathsf{forceOpen}(P) \\ & b_1 := \mathsf{Vf}(\pi, P, \mathbb{C}, T) \\ & b_2 := (\mathbb{C}, w) \in \mathcal{R} \end{array}\right] \leq \mathsf{negl}(\lambda)
$$

We say that a VTC is simulation-sound if it is sound even when the prover has access to simulated proofs for (possibly false) statements of his choice; i.e., the prover must not be able to compute a valid proof for a fresh false statement of his choice.

Definition 14 (Timed Privacy)**.** *A VTC scheme* $\Pi_{VTC} :=$ (Commit, Verify, Open, forceOpen) *is private if there exists a negligible function* $\mathsf{negl}(\lambda)$*, a PPT simulator* $\mathcal{S}$*, and a polynomial* $\hat{T}$ *such that for all polynomials* $T > \hat{T}$*, all PRAM algorithms* $\mathcal{A}$ *whose running time is at most* $t < T$*, all messages* $m \in \{0,1\}^*$*, and all* $\lambda \in N$*, it holds that:*

$$
\Pr\left[\begin{array}{c|c} & (\mathbb{C}, w) \leftarrow \mathsf{createR}(1^\lambda) \\ & b \leftarrow_\$ \{0,1\} \\ b = b^* & \textbf{if } b = 0 : (P, \pi) \leftarrow \mathsf{Commit}(w, T) \\ & \textbf{if } b = 1 : (P, \pi) \leftarrow \mathcal{S}(\mathbb{C}, T) \\ & b^* \leftarrow \mathcal{A}(\mathbb{C}, P, \pi, T) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)
$$

E. Security proofs for BL based on digital signatures

Correctness. We now analyze the correctness of our construction.

Theorem 13 (BL correctness)**.** *Assume that the digital signature scheme is correct. Then, this construction is a correct BL according to Definition 2.*

Proof: We need to show that

$$\Pr\left[\mathsf{isAuth}_{\mathsf{BL}}(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}}, pk_S) = 1\right] = 1$$

By definition of $\mathsf{isAuth}_{\mathsf{BL}}$ we have that $\mathsf{isAuth}_{\mathsf{BL}}(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}}, pk_S) = \Pi_{DS}.\mathsf{Vf}(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}}, pk_S)$. By definition of $\mathsf{Auth}_{\mathsf{bsc}}$, we have that $\mathsf{isAuth}_{\mathsf{BL}}(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}}, pk_S) = \Pi_{DS}.\mathsf{Vf}(tx_{\mathsf{bsc}}, \Pi_{DS}.\mathsf{Sig}(sk_S, tx_{\mathsf{bsc}}), pk_S)$. Finally, assuming that the digital signature scheme Π_{DS} is correct, we have that $\mathsf{isAuth}_{\mathsf{BL}}(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}}, pk_S) = 1$. This concludes the proof. ■

Security. We now analyze the security of our construction.

Theorem 1 (BL Unforgeability)**.** *Assume that the digital signature scheme is existentially unforgeable against chosen message attacks. Then, the construction in Fig. 3 offers BL unforgeabilty according to Definition 3.*

Proof: Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{bscForge}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win the unforgeability of the signature scheme, with the following steps:

- The challenger produces they key pair (pk, sk) and shares pk with $\mathcal{B}$.
- $\mathcal{B}$ forwards pk to $\mathcal{A}$, which replies with a basic transaction, tx_{bsc} and transaction authorization σ_{bsc}.
- $\mathcal{B}$ forwards the pair $(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}})$ to the challenger.

The $\mathcal{O}\mathsf{Auth}_{\mathsf{bsc}}$ oracle of $\mathsf{bscForge}$ requires to call the $\mathcal{O}\mathsf{Sig}$ oracle of the signature unforgeability game, which guarantees that the memory of both oracles is synchronized.

Our adversary $\mathcal{B}$ perfectly simulates $\mathsf{bscForge}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. If $\mathsf{isAuth}_{\mathsf{BL}}$ returns 1 for $(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}})$, this implies that $\Pi_{DS}.\mathsf{Vf}(m, \sigma)$ returns 1, and σ will be a forgery. However, this contradicts the assumption that the digital signature scheme is EUF-CMA secure, so $\mathcal{A}$ cannot exist and this concludes the proof of Theorem 1. ■

F. Security proofs for BL based on NIZK

Correctness. We now analyze the correctness of our construction.

Theorem 14 (BL correctness)**.** *Assume that the NIZK is correct. Then, the NIZK construction in Figure 3 is a correct BL according to Definition 2.*

Proof: We need to show that

$$\Pr\left[\mathsf{isAuth}_{\mathsf{BL}}(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}}, pk_S) = 1\right] = 1$$

By definition of $\mathsf{isAuth}_{\mathsf{BL}}$ we have that $\mathsf{isAuth}_{\mathsf{BL}}(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}}, pk_S) = \Pi_{NIZK}.\mathsf{Vf}(\mathsf{x}, \pi, \mathsf{CRS})$. By definition of $\mathsf{Auth}_{\mathsf{bsc}}$, we have that $\mathsf{isAuth}_{\mathsf{BL}}(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}}, pk_S) = \Pi_{NIZK}.\mathsf{Vf}(\mathsf{x}, \Pi_{NIZK}.\mathsf{Prove}(\mathsf{x}, \omega, \mathsf{CRS}), \mathsf{CRS})$. The construction assumes that CRS is generated at ledger setup by running $\mathsf{CRS} \leftarrow \mathsf{SetUp}(1^\lambda)$. Finally, assuming that the NIZK Π_{NIZK} is correct, we have that $\mathsf{isAuth}_{\mathsf{BL}}(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}}, pk_S) = 1$. This concludes the proof. ■

Security. We now analyze the security of our construction.

Theorem 2 (BL Unforgeability)**.** *Assume that the NIZK provides knowledge soundness. Then, the construction in Fig. 3 offers BL unforgeabilty according to Definition 3.*

Proof: Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{bscForge}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win the knowledge soundness of the NIZK scheme, with the following steps:

- The challenger produces they CRS and shares CRS with $\mathcal{B}$. $\mathcal{B}$ and $\mathcal{A}$ use this string in the BL ledger.
- $\mathcal{B}$ runs $(pk, sk) \leftarrow \mathsf{ctAcc}$ and forwards pk to $\mathcal{A}$, which replies with a basic transaction, tx_{bsc} and transaction authorization σ_{bsc}.
- $\mathcal{B}$ forwards the pair $(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}})$ to the challenger, encoded as (x, π).

Our adversary $\mathcal{B}$ perfectly simulates $\mathsf{bscForge}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. If $\mathsf{isAuth}_{\mathsf{BL}}$ returns 1 for $(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}})$, this implies that $\Pi_{NIZK}.\mathsf{Vf}(\mathsf{x}, \pi, \mathsf{CRS})$ returns 1, while at the same time $\mathcal{A}$ did not have access to the witness, sk. Therefore, the challenger will not be able to extract sk such that $(tx_{\mathsf{bsc}}, sk) \in \mathcal{L}$. However, this contradicts the assumption that the NIZK scheme provides knowledge-soundness, so $\mathcal{A}$ cannot exist and this concludes the proof of Theorem 2. ■

G. Security proofs for ledger with condition and time-lock support

Correctness. We first analyze the correctness of our construction.

Theorem 15 (CL correctness)**.** *Assume that the digital signature scheme is correct. Then, the HTLC blueprint is a correct CL according to Definition 5.*

Proof: We need to show for the redeem path that

$$\Pr\begin{bmatrix}\mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, epk) = 1 \\ (\mathbb{C}, w') \in \mathcal{R}\end{bmatrix} = 1$$

and for the refund path that

$$\Pr\left[\mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}, epk) = 1\right] = 1$$

Redeem refers to prove that $\mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, epk) = 1$. By definition of $\mathsf{isAuth}_{\mathsf{CL}}$, we have that $\mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, epk) = \Pi_{DS}.\mathsf{Vf}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, pk_R) \wedge (\mathbb{C}, w) \in \mathcal{R}$. We now focus on the Vf side of the equation. By the definition of $\mathsf{Auth}_{\mathsf{red}}$, we have that $\Pi_{DS}.\mathsf{Vf}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, pk_R) = \Pi_{DS}.\mathsf{Vf}(tx_{\mathsf{red}}, \Pi_{DS}.\mathsf{Sig}(tx_{\mathsf{red}}, sk_R), pk_R)$. The key pair (pk_R, sk_R) was generated in ctEAcc running $\Pi_{DS}.\mathsf{KGen}(1^\lambda)$. Therefore, since we assume that the digital signature is correct, the Vf side of $\mathsf{isAuth}_{\mathsf{CL}}$ must return 1. Regarding the hard relation side of $\mathsf{isAuth}_{\mathsf{CL}}$, $\mathsf{Auth}_{\mathsf{red}}$ simply takes w and appends it to σ_{red}. Since our assumption is that $(\mathbb{C}, w) \in \mathcal{R}$, this side of $\mathsf{isAuth}_{\mathsf{CL}}$ also returns 1.

Get Witness refers to proving that $(\mathbb{C}, w') \in \mathcal{R}$. By construction, σ_{red} is the tuple (σ, w). Since we shown already that $\mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, epk) = 1$, this implies that w in σ_{red} is a valid witness from $\mathbb{C}$ in $\mathcal{R}$ and by definition, GWit obtains w' simply by taking w from σ_{red}, therefore $(\mathbb{C}, w') \in \mathcal{R}$.

Refund refers to prove that $\mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}, epk) = 1$. By definition of $\mathsf{isAuth}_{\mathsf{CL}}$, we have that $\mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}, epk) = \Pi_{DS}.\mathsf{Vf}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}, pk_S)$. By the definition of $\mathsf{Auth}_{\mathsf{ref}}$, we have that $\Pi_{DS}.\mathsf{Vf}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}, pk_S) = \Pi_{DS}.\mathsf{Vf}(tx_{\mathsf{ref}}, \Pi_{DS}.\mathsf{Sig}(tx_{\mathsf{red}}, sk_S), pk_S)$. The key pair (pk_S, sk_S) was generated in ctEAcc running $\Pi_{DS}.\mathsf{KGen}(1^\lambda)$. Therefore, since we assume that the digital signature is correct, $\mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, epk) = 1$. This concludes the proof of Theorem 15. ■

Security. We now analyze the security properties of our construction. Properties CL redeem unforgeability and CL refund unforgeability are proven below. Regarding CL redeemability, CL refundability and CL extractability, it is trivial to see that they hold. Both $\mathsf{Auth}_{\mathsf{red}}$ and $\mathsf{Auth}_{\mathsf{ref}}$ result in running $\Pi_{DS}.\mathsf{Sig}$ with knowledge of the appropriate secret key and the witness (for redeem). Due to the correctness of Π_{DS}, both algorithms will generate valid signatures, ensuring that CL redeemability and CL refundability hold. In the specific case of CL redeemability, the adversary provides the pair $(\mathbb{C}, w)$. Since the signature is valid due to correctness of Π_{DS}, the authorization will be valid as long as the statement and witness provided by the adversary is in the hard relation. For CL extractability, it is easy to see that a valid σ_{red} contains a valid witness w, due to the definition of σ_{red} (see Figure 4). Therefore, any valid σ_{red} in this setting provides a valid witness and CL extractability holds.

Theorem 3 (CL redeem unforgeability)**.** *Assume that that the digital signature scheme is unforgeable. Then, the our protocol offers CL redeem unforgeability according to Definition 6.*

Proof: Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{refForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win the unforgeability of the signature scheme, with the following steps:

- The challenger produces they key pair (pk, sk) and shares pk with $\mathcal{B}$.
- $\mathcal{B}$ receives T from $\mathcal{A}$ and runs createR to obtain $(\mathbb{C}, w) \in \mathcal{R}$.
- Then, $\mathcal{B}$ engages in ctEAcc with $\mathcal{A}$, providing $\mathbb{C}$ to the latter as input. When it is the turn of $\mathcal{B}$ to forward pk_R, instead $\mathcal{B}$ forwards pk to $\mathcal{A}$. The rest of the protocol continues as described in Fig. 4.
- $\mathcal{A}$ sends $\mathcal{B}$ a forged pair of redeem transaction, tx^*_{red} and transaction authorization σ_{red}.
- $\mathcal{B}$ renames the pair $(tx^*_{\mathsf{red}}, \sigma^*_{\mathsf{red}})$ as (m, σ) and forwards this to the challenger.

Our adversary $\mathcal{B}$ perfectly simulates $\mathsf{redForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. If $\mathsf{isAuth}_{\mathsf{CL}}$ returns 1 for $(tx^*_{\mathsf{red}}, \sigma^*_{\mathsf{red}}, epk)$, this means that $\Pi_{DS}.\mathsf{Vf}(tx^*_{\mathsf{red}}, \sigma^*_{\mathsf{red}}, pk_R) = 1$, then $\Pi_{DS}.\mathsf{Vf}(m, \sigma, pk)$ will also return 1, and σ will be a forgery. Furthermore, since $\mathcal{A}$ does not have access to an oracle, tx^*_{red} will not be in the memory of the challenger. However, this contradicts the assumption that the digital signature scheme is EUF-CMA secure, so $\mathcal{A}$ cannot exist and this concludes the proof of Theorem 3. ■

Theorem 4 (CL refund unforgeability)**.** *Assume that that the digital signature scheme is unforgeable. Then, our protocol offers CL refund unforgeability according to Definition 6.*

Proof: Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{refForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win the unforgeability of the signature scheme, with the following steps:

- The challenger produces they key pair (pk, sk) and shares pk with $\mathcal{B}$.
- $\mathcal{B}$ receives $\mathbb{C}$, T and st_0 from $\mathcal{A}$, and engages in ctEAcc with $\mathcal{A}$. When it is the turn of $\mathcal{B}$ to forward pk_S, instead $\mathcal{B}$ forwards pk to $\mathcal{A}$. The rest of the protocol continues as described in Fig. 4.
- $\mathcal{A}$ sends $\mathcal{B}$ a forged pair of refund transaction, tx^*_{ref} and transaction authorization σ^*_{ref}.
- $\mathcal{B}$ renames the pair $(tx^*_{\mathsf{ref}}, \sigma^*_{\mathsf{ref}})$ as (m, σ) and forwards this to the challenger.

Our adversary $\mathcal{B}$ perfectly simulates $\mathsf{refForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. If $\mathsf{isAuth}_{\mathsf{CL}}$ returns 1 for $(tx^*_{\mathsf{ref}}, \sigma^*_{\mathsf{ref}}, epk)$, this means that $\Pi_{DS}.\mathsf{Vf}(tx^*_{\mathsf{ref}}, \sigma^*_{\mathsf{ref}}, pk_S) = 1$, then $\Pi_{DS}.\mathsf{Vf}(m, \sigma, pk)$ will also return 1, and σ will be a forgery. Furthermore, since $\mathcal{A}$ does not have access to an oracle, tx^*_{ref} will not be in the memory of the challenger. However, this contradicts the assumption that the digital signature scheme is EUF-CMA secure, so $\mathcal{A}$ cannot exist and this concludes the proof of Theorem 4. ■

H. Security proofs for ledger with time-lock support

Correctness. We first analyze the correctness of our construction.

Theorem 16 (CL correctness)**.** *Assume that the digital signature scheme, the 2 party adaptor signature scheme and the script for epk are correct. Then, the PTLC blueprint is a correct CL according to Definition 5.*

Proof: We need to show for the redeem path that

$$\Pr\begin{bmatrix}\mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, epk) = 1 \\ (\mathbb{C}, w') \in \mathcal{R}\end{bmatrix} = 1$$

and for the refund path that

$$\Pr\left[\mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}, epk) = 1\right] = 1$$

Redeem refers to prove that $\mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, epk) = 1$. By definition of $\mathsf{isAuth}_{\mathsf{CL}}$ for the redeem path, we have that $\mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, epk) = \Pi_{DS}.\mathsf{Vf}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, pk_{S,R})$ By the definition of $\mathsf{Auth}_{\mathsf{red}}$, we have that $\Pi_{DS}.\mathsf{Vf}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, pk_{S,R}) = \Pi_{DS}.\mathsf{Vf}(tx_{\mathsf{red}}, \mathsf{Adapt}(\hat{\sigma}, w), pk_{S,R})$. By definition of ctEAcc, $\hat{\sigma} \leftarrow \Pi_{\mathsf{pSig}}$. Therefore, since we assume that the two party adaptor digital signature is correct, $\mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, epk) = 1$.

Get Witness refers to proving that $(\mathbb{C}, w') \in \mathcal{R}$. By construction, GWit executes $\mathsf{Extract}(\sigma_{\mathsf{red}}, \hat{\sigma}, \mathbb{C})$. We have already proven that σ_{red} is the output of Adapt, and that it is a valid signature. Therefore, since the two party adaptor signature scheme is correct, w' generated by $\mathsf{Extract}(\sigma_{\mathsf{red}}, \hat{\sigma}, \mathbb{C})$ must also be a valid witness to $\mathbb{C}$.

Refund refers to prove that $\mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}, epk) = 1$. Since $\mathsf{Auth}_{\mathsf{ref}}$ is the same as in Section IV-D with regards to keypair $(pk_S^{\mathsf{ref}}, sk_S^{\mathsf{ref}})$, the statement holds for the same reasons as in Theorem 15. And this concludes the proof of Theorem 16. ∎

Security. We now analyze the security properties of our construction. Properties CL redeem unforgeability, CL redeemability and CL extractability are proven below. Regarding CL refund unforgeability and CL refundability since $\mathsf{Auth}_{\mathsf{ref}}$ is the same as in Section IV-D, these properties hold for the same reasons. Note that the reduction is the same because $(pk_S^{\mathsf{ref}}, sk_S^{\mathsf{ref}})$ is a different key to the one used for the redeem path (pk_S, sk_S).

Theorem 5 (CL redeem unforgeability)**.** *Assume that the two party adaptor signature scheme satisfies 2-aEUF-CMA Security. Then, our protocol offers CL redeem unforgeability according to Definition 6.*

Proof: We show $\mathsf{redForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$, expanded with the interactions of the PTLC blueprint in Figure 13

PTLC: $\mathsf{redForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda)$

$$
\begin{aligned}
&(T, \mathsf{st}_0) \leftarrow \mathcal{A}(1^\lambda)\\
&(\mathbb{C}, w) \leftarrow \mathsf{createR}(1^\lambda)\\
&\mathsf{st}_a \leftarrow \mathcal{A}(\mathbb{C}, \mathsf{st}_0)\\
&(pk_R, sk_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)\\
&(\ddot{pk}_R, \ddot{sk}_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)\\
&(pk_S, sk_S, pk_S^{\mathsf{ref}}, \mathsf{st}_b) \leftarrow \mathcal{A}(pk_R, \mathsf{st}_a)\\
&pk_{S,R} \leftarrow \mathsf{KAgg}(pk_S, pk_R)\\
&epk := pk_{S,R} \vee (pk_S^{\mathsf{ref}} \wedge T)\\
&tx_{\mathsf{red}} \leftarrow \mathsf{ctTx}_{\mathsf{red}}(pk_{S,R}, \ddot{pk}_R)\\
&(tx_{\mathsf{ref}}, \mathsf{st}_c) \leftarrow \mathcal{A}(tx_{\mathsf{red}}, \mathsf{st}_b)\\
&(\hat{\sigma}, \mathsf{st}_1) \leftarrow \Pi_{\mathsf{pSig}} \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_c), \\ R(sk_R, tx_{\mathsf{red}}, \mathbb{C}) \end{matrix} \right\rangle\\
&\textbf{if } \mathsf{pVf}(tx_{\mathsf{red}}, \mathbb{C}, \hat{\sigma}) = 0 \textbf{ abort}\\
&(tx^*_{\mathsf{red}}, \sigma^*_{\mathsf{red}}) \leftarrow \mathcal{A}(\mathsf{st}_1)\\
&\tau \leftarrow \mathtt{readTime}(\cdot)\\
&b_0 := \Pi_{DS}.\mathsf{Vf}(tx^*_{\mathsf{red}}, \sigma^*_{\mathsf{red}}, pk'_{S,R})\\
&b_1 := \tau < T\\
&b_2 := tx^*_{\mathsf{red}} \neq tx_{\mathsf{red}}\\
&\textbf{return } b_0 \wedge b_1 \wedge b_2
\end{aligned}
$$

Fig. 13. PTLC: Experiment for $\mathsf{redForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$.

Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{redForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break 2-aEUF-CMA Security (see Appendix C) with the following steps:

- $\mathcal{B}$ receives pk_R from the challenger.
- $\mathcal{B}$ receives T from $\mathcal{A}$.
- $\mathcal{B}$ runs $(\mathbb{C}, w) \leftarrow \mathsf{createR}(1^\lambda)$ and shares $\mathbb{C}$ with $\mathcal{A}$.
- $\mathcal{B}$ runs $(pk_R, sk_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$, $(\ddot{pk}_R, \ddot{sk}_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$ and shares pk_R with $\mathcal{A}$.
- $\mathcal{A}$ returns $(pk_S, sk_S, pk_S^{\mathsf{ref}})$ to $\mathcal{B}$, who forwards (pk_S, sk_S) to the challenger.
- $\mathcal{B}$ runs $pk_{S,R} \leftarrow \mathsf{KAgg}(pk_S, pk_R)$, defines $epk := (pk_{S,R} \vee pk_S^{\mathsf{ref}} \& T)$ and creates tx_{red}. Sends tx_{red} to $\mathcal{A}$ to obtain tx_{ref}.
- $\mathcal{B}$ calls $\mathcal{O}\Pi_{\mathsf{pSig}}$ of the challenger to run Π_{pSig} with $\mathcal{A}$ and obtain $\hat{\sigma}$ using $\mathbb{C}$.
- $\mathcal{B}$ will check if $\mathsf{pVf}(tx_{\mathsf{red}}, \mathbb{C}, \hat{\sigma}) = 0$. However, this check will verify, as $\mathcal{B}$ used the challenger as an oracle.
- $\mathcal{A}$ sends $(tx^*_{\mathsf{red}}, \sigma^*_{\mathsf{red}})$ to $\mathcal{B}$.
- $\mathcal{B}$ sends tx^*_{red} to the challenger, and both engage in Π_{pSig} protocol, using a freshly created $\mathbb{C}'$ by the challenger. $\mathcal{B}$ can run this protocol because it received sk_S from $\mathcal{A}$ in a previous step. The protocol ends with $\hat{\sigma}^*$
- $\mathcal{B}$ sends σ^*_{red} to the challenger.

Our adversary $\mathcal{B}$ perfectly simulates $\mathsf{redForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. If $\mathsf{isAuth}_{\mathsf{CL}}(tx^*_{\mathsf{red}}, \sigma^*_{\mathsf{red}}, epk) = 1$, this implies that $\mathsf{Vf}(tx^*_{\mathsf{red}}, \sigma^*_{\mathsf{red}}, pk_{S,R}) = 1$, which will also satisfied in the two party adaptor security game. Furthermore, while tx_{red} will be in memory of 2-aEUF-CMA Security, tx^*_{red} will not be in memory. If the adversary is able to send a forgery before the T, expires, it will win $\mathsf{redForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$. This implies that an adversary winning $\mathsf{redForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ can be used to break 2-aEUF-CMA Security. However, this contradicts the assumption that the two party adaptor signature scheme provides 2-aEUF-CMA Security and this concludes the proof of Theorem 5. ∎

Theorem 6 (CL redeemability)**.** *Assume that the two party adaptor signature scheme satisfies two-party pre-signature adaptability. Then, our protocol offers CL redeemability according to Definition 6.*

Proof: We show $\mathsf{ExpRedeem}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$, expanded with the interactions of the PTLC construction in Figure 14

PTLC: $\mathsf{ExpRedeem}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda)$

$$
\begin{aligned}
&(\mathbb{C}, w, T, \mathsf{st}_0) \leftarrow \mathcal{A}(1^\lambda)\\
&(pk_R, sk_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(\lambda)\\
&(\ddot{pk}_R, \ddot{sk}_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(\lambda)\\
&(pk_S, sk_S, pk_S^{\mathsf{ref}}, \mathsf{st}_a) \leftarrow \mathcal{A}(pk_R, \mathsf{st}_0)\\
&pk_{S,R} \leftarrow \mathsf{KAgg}(pk_S, pk_R)\\
&epk := pk_{S,R} \vee (pk_S^{\mathsf{ref}} \wedge T)\\
&tx_{\mathsf{red}} \leftarrow \mathsf{ctTx}_{\mathsf{red}}(pk_{S,R}, \ddot{pk}_R)\\
&(tx_{\mathsf{ref}}, \mathsf{st}_b) \leftarrow \mathcal{A}(tx_{\mathsf{red}}, \mathsf{st}_a)\\
&(\hat{\sigma}, \mathsf{st}_1) \leftarrow \Pi_{\mathsf{pSig}} \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_b), \\ R(sk_R, tx_{\mathsf{red}}, \mathbb{C}) \end{matrix} \right\rangle\\
&\textbf{if } \mathsf{pVf}(tx_{\mathsf{red}}, \mathbb{C}, \hat{\sigma}) = 0 \textbf{ abort}\\
&\sigma_{\mathsf{red}} \leftarrow \mathsf{Adapt}(\hat{\sigma}, w)\\
&b_0 := \Pi_{DS}.\mathsf{Vf}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, pk_{S,R}) = 0\\
&b_1 := (\mathbb{C}, w) \in \mathcal{R}\\
&\textbf{return } b_0 \wedge b_1
\end{aligned}
$$

Fig. 14. PTLC: Experiment for $\mathsf{ExpRedeem}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$

Assume by contradiction that there exists a PPT adversary

$\mathcal{A}$ such that $\Pr[\mathsf{ExpRedeem}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break Two-Party Adaptability (see Appendix C) with the following steps:

- $\mathcal{B}$ receives $(\mathbb{C}, w, T)$ from $\mathcal{A}$.
- $\mathcal{B}$ runs $(pk_R, sk_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$, $(\ddot{pk}_R, \ddot{sk}_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$ and shares pk_R with $\mathcal{A}$.
- $\mathcal{A}$ replies with pk_S, sk_S and pk_S^{ref}.
- $\mathcal{B}$ runs KAgg in order to obtain $pk_{S,R}$, which is used to also create epk and tx_{red}. Sends tx_{red} to $\mathcal{A}$ and receives tx_{ref}.
- $\mathcal{B}$ runs Π_{pSig} with $\mathcal{A}$, until the protocol ends, outputting $\hat{\sigma}$.
- $\mathcal{B}$ checks if $\mathsf{pVf}(tx_{\mathsf{red}}, \mathbb{C}, \hat{\sigma})$ verifies. If it verifies, $\mathcal{B}$ sends $\mathbb{C}$, w, pk_S, pk_R, tx_{red} and $\hat{\sigma}$ to the challenger.

Our adversary $\mathcal{B}$ perfectly simulates $\mathsf{ExpRedeem}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if adversary wins $\mathsf{ExpRedeem}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$, this implies that $\mathcal{B}$ failed to obtain a valid signature when running $\mathsf{Adapt}(\hat{\sigma}, w)$ with a valid witness. Therefore, the challenger will fail as well. However, this contradicts the assumption that the two party adaptor signature scheme provides Two-Party Adaptability and this concludes the proof of Theorem 6. ■

Theorem 7 (CL extractability)**.** *Assume that the two party adaptor signature scheme satisfies two-party witness extractability. Then, our protocol offers CL extractability according to Definition 6.*

Proof: We show $\mathsf{ExpExtract}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$, expanded with the interactions of the PTLC construction in Figure 15

PTLC: $\mathsf{ExpExtract}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda)$

$(\mathbb{C}, T, \mathsf{st}_0) \leftarrow \mathcal{A}(1^\lambda)$
$(pk_S, sk_S) \leftarrow \mathsf{KGen}(\lambda)$
$(pk_S^{\mathsf{ref}}, sk_S^{\mathsf{ref}}) \leftarrow \mathsf{KGen}(\lambda)$
$(\ddot{pk}_S, \ddot{sk}_S) \leftarrow \mathsf{KGen}(\lambda)$
$(pk_R, sk_R, \mathsf{st}_a) \leftarrow \mathcal{A}(pk_S, pk_S^{\mathsf{ref}})$
$pk_{S,R} \leftarrow \mathsf{KAgg}(pk_S, pk_R)$
$epk := pk_{S,R} \vee (pk_S^{\mathsf{ref}} \wedge T)$
$tx_{\mathsf{ref}} \leftarrow \mathsf{ctTx}_{\mathsf{ref}}(pk_{S,R}, \ddot{pk}_S)$
$(tx_{\mathsf{red}}, \mathsf{st}_b) \leftarrow \mathcal{A}(tx_{\mathsf{ref}}, \mathsf{st}_a)$
$(\hat{\sigma}, \mathsf{st}_1) \leftarrow \Pi_{\mathsf{pSig}} \left\langle \begin{matrix} S(sk_S, tx_{\mathsf{red}}, \mathbb{C}), \\ \mathcal{A}(\mathsf{st}_b) \end{matrix} \right\rangle$
if $\mathsf{pVf}(tx_{\mathsf{red}}, \mathbb{C}, \hat{\sigma}) = 0$ **abort**
$\sigma_{\mathsf{red}} \leftarrow \mathcal{A}(\mathsf{st}_1)$
$w' \leftarrow \mathsf{Extract}(\sigma_{\mathsf{red}}, \hat{\sigma}, \mathbb{C})$
$b_0 := \mathsf{Vf}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, pk_{S,R})$
$b_1 := (\mathbb{C}, w') \notin \mathcal{R}$
return $b_0 \wedge b_1$

Fig. 15. PTLC: Experiment for $\mathsf{ExpExtract}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$

Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{ExpExtract}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break Two-Party Witness Extractability (see Appendix C) with the following steps:

- $\mathcal{B}$ receives pk_S from the challenger.
- $\mathcal{B}$ receives $(\mathbb{C}, T)$ from $\mathcal{A}$.
- $\mathcal{B}$ runs $(pk_S^{\mathsf{ref}}, sk_S^{\mathsf{ref}}) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$, $(\ddot{pk}_S, \ddot{sk}_S) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$ and shares $pk_S, pk_S^{\mathsf{ref}}$ with $\mathcal{A}$.
- $\mathcal{A}$ replies with pk_R, sk_R.
- $\mathcal{B}$ runs KAgg in order to obtain $pk_{S,R}$, which is used to also create epk and tx_{ref}. tx_{ref} is sent to $\mathcal{A}$ to obtain tx_{red}. $\mathcal{B}$ sends pk_R, sk_R, tx_{red},$\mathbb{C}$ to challenger in two stages. First, the keys, then the transaction and $\mathbb{C}$.
- Challenger starts Π_{pSig} protocol, using tx_{red} and $\mathbb{C}$. $\mathcal{B}$ simply relays the messages between challenger and $\mathcal{A}$, until the protocol ends, outputting $\hat{\sigma}$.
- $\mathcal{A}$ sends (σ_{red}) to $\mathcal{B}$, which forwards it to the challenger.

Our adversary $\mathcal{B}$ perfectly simulates $\mathsf{ExpExtract}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. If condition b_0 is satisfied in $\mathsf{ExpExtract}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ this means that $\mathsf{Vf}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, pk_{S,R}) = 1$, which will also be satisfied in the adaptor Two-Party Witness Extractability. Finally, b_1 is equivalent to the second winning condition for Two-Party Witness Extractability. This implies that an adversary that wins $\mathsf{ExpExtract}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ can be used to break Two-Party Witness Extractability. Note that $\mathcal{A}$ does not have oracle access, and that tx_{red} is the message used to generate the presignature with the challenger, ensuring that it will not be in the challenger's memory. However, this contradicts the assumption that the two party adaptor signature scheme provides Two-Party Witness Extractability and this concludes the proof of Theorem 7. ■

I. Security proofs for emulating a CL with a BL

Correctness. We first analyze the correctness of the construction.

Theorem 17 (CL correctness)**.** *Assume that the aggregatable digital signature scheme, the 2 party adaptor signature scheme and the Verifiable Timed Commitment schemes are correct. Then, the CL emulation from a BL correct CL according to Definition 5.*

Proof: We need to show for the redeem path that

$$\Pr\left[\begin{matrix}\mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, epk) = 1 \\ (\mathbb{C}, w') \in \mathcal{R}\end{matrix}\right] = 1$$

and for the refund path that

$$\Pr\left[\mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}, epk) = 1\right] = 1$$

CL emulation is an extension of PTLC with a refund branch that is not trusted on the ledger. Therefore, prove correctness for the *redeem* branch ($\mathsf{Auth}_{\mathsf{red}}$ and GWit) is equivalent to prove it for PTLC (see Theorem 16).

Refund refers to prove that $\mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}, epk) = 1$. By definition of $\mathsf{isAuth}_{\mathsf{CL}}$ for the refund path, we have that $\mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}, epk) = \Pi_{DS}.\mathsf{Vf}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}, pk_{S,R})$. By the definition of $\mathsf{Auth}_{\mathsf{ref}}$, we have that $\sigma_{\mathsf{red}} \leftarrow \mathsf{Adapt}(\hat{\sigma}_{\mathsf{ref}}, w^{\mathsf{ref}})$ and $w^{\mathsf{ref}} \leftarrow \mathsf{forceOpen}(P)$. By definition of ctEAcc, $P \leftarrow \Pi_{VTD}.\mathsf{Commit}(T, w^{\mathsf{ref}})$, while $\hat{\sigma}_{\mathsf{ref}}$ is the output of Π_{pSig} on public statement $\mathbb{C}^{\mathsf{ref}}$ and keys pk_S and pk_R. Therefore, since we assume that the two party aggregatable digital signature and Verifiable Timed Commitment are correct, $\mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}, epk) = 1$. And this concludes the proof of Theorem 17 ■

Security. We now analyze the security properties of our construction.

Theorem 8 (CL redeem unforgeability)**.** *Assume that the two party adaptor signature scheme satisfies 2-aEUF-CMA security and the verifiable timed dlog guarantees privacy. Then, our protocol offers CL redeem unforgeability according to Definition 6.*

Proof: We consider the following game hops:

Game $\mathsf{redForge}^{G_0}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$: This game, formally defined in Figure 16 with the blue line, corresponds to the original game for CL redeem unforgeability. The game is expanded with the interactions described in our implementation.

Game $\mathsf{redForge}^{G_1}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$: This game, formally defined in Figure 16, works exactly as G_0 but with the red line instead of the blue line. The challenger, instead of using Commit, uses simulator $\mathcal{S}$ in order to generate P and π.

Game $\mathsf{redForge}^{G_2}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$: This game, formally defined in Figure 16, works exactly as G_1 with the exception the violet line. The challenger, aborts if $tx^*_{\mathsf{ref}} = tx_{\mathsf{ref}}$.

CL emulation: $\mathsf{redForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda)$

$(T, \mathsf{st}_0) \leftarrow \mathcal{A}(1^\lambda)$
$(\mathbb{C}, w) \leftarrow \mathsf{createR}(1^\lambda)$
$\mathsf{st}_a \leftarrow \mathcal{A}(\mathbb{C}, \mathsf{st}_0)$
$(pk_R, sk_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$
$(\ddot{pk}_R, \ddot{sk}_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$
$(pk_S, sk_S, \mathsf{st}_b) \leftarrow \mathcal{A}(pk_R, \mathsf{st}_a)$
$pk_{S,R} \leftarrow \mathsf{KAgg}(pk_S, pk_R)$
$epk := pk_{S,R}$
$tx_{\mathsf{red}} \leftarrow \mathsf{ctTx}_{\mathsf{red}}(pk_{S,R}, \ddot{pk}_R)$
$(tx_{\mathsf{ref}}, \mathsf{st}_c) \leftarrow \mathcal{A}(tx_{\mathsf{red}}, \mathsf{st}_b)$
$(\hat{\sigma}_{\mathsf{red}}, \mathsf{st}_d) \leftarrow \Pi_{\mathsf{pSig}}\left\langle \begin{array}{c}\mathcal{A}(\mathsf{st}_c),\\ R(sk_R, tx_{\mathsf{red}}, \mathbb{C})\end{array}\right\rangle$
if $\mathsf{pVf}(tx_{\mathsf{red}}, \mathbb{C}, \hat{\sigma}_{\mathsf{red}}) = 0$ **abort**
$(\mathbb{C}^{\mathsf{ref}}, w^{\mathsf{ref}}) \leftarrow \mathsf{createR}(1^\lambda)$
$(P, \pi) \leftarrow \mathsf{Commit}(w^{\mathsf{ref}}, T)$ $(P, \pi) \leftarrow \mathcal{S}(\mathbb{C}^{\mathsf{ref}}, T)$
$\mathsf{st}_e \leftarrow \mathcal{A}(P, \pi, \mathbb{C}^{\mathsf{ref}}, \mathsf{st}_d)$
$(\hat{\sigma}_{\mathsf{ref}}, \mathsf{st}_1) \leftarrow \Pi_{\mathsf{pSig}}\left\langle \begin{array}{c}\mathcal{A}(\mathsf{st}_e),\\ R(sk_R, tx_{\mathsf{ref}}, \mathbb{C}^{\mathsf{ref}})\end{array}\right\rangle$
if $\mathsf{pVf}(tx_{\mathsf{red}}, \mathbb{C}^{\mathsf{ref}}, \hat{\sigma}_{\mathsf{ref}}) = 0$ **abort**
$(tx^*_{\mathsf{red}}, \sigma^*_{\mathsf{red}}) \leftarrow \mathcal{A}(\mathsf{st}_1)$
if $tx^*_{\mathsf{red}} = tx_{\mathsf{ref}}$ **abort**
$\tau \leftarrow \mathtt{readTime}(\cdot)$
$b_0 := \mathsf{Vf}(tx^*_{\mathsf{red}}, \sigma^*_{\mathsf{red}}, pk_{S,R})$
$b_1 := \tau < T$
$b_2 := tx^*_{\mathsf{red}} \neq tx_{\mathsf{red}}$
return $b_0 \wedge b_1$

Fig. 16. CL emulation: Experiments $\mathsf{redForge}^{G_0}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ (ignoring the violet and red lines, and including the blue line); $\mathsf{redForge}^{G_1}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ (including red line, and ignoring the violet and blue lines); and $\mathsf{redForge}^{G_2}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ (including the violet and red lines, and ignoring the blue line).

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

$$\left| \begin{array}{l} \Pr[\mathsf{redForge}^{G_0}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda) = 1] \\ - \Pr[\mathsf{redForge}^{G_1}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda) = 1] \end{array} \right| > \mathsf{negl}(\lambda)$$

Assume that the Verifiable Timed Commitment provides Timed Privacy. Then $\Pr[\mathsf{Bad}_1] \leq \mathsf{negl}(\lambda)$.

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

$$\Pr\left[b = b^* \middle| \begin{array}{c} b \leftarrow_\$ \{0,1\} \\ \mathsf{redForge}^{G_b}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda) \\ b^* \leftarrow \mathcal{A}() \end{array} \right] > \frac{1}{2} + \mathsf{negl}(\lambda)$$

We can construct adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break VTC Timed Privacy (see Appendix D) with the following steps:

- $\mathcal{B}$ receives $\mathbb{C}, P, \pi$ from the challenger.
- $\mathcal{B}$ receives T from $\mathcal{A}$.
- $\mathcal{B}$ runs $(\mathbb{C}', w') \leftarrow \mathsf{createR}(1^\lambda)$ and shares $\mathbb{C}'$ with $\mathcal{A}$.
- $\mathcal{B}$ runs $(pk_R, sk_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$, $(\ddot{pk}_R, \ddot{sk}_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$ and shares pk_R with $\mathcal{A}$.
- $\mathcal{A}$ returns (pk_S, sk_S) to $\mathcal{B}$.
- $\mathcal{B}$ runs KAgg in order to obtain $pk_{S,R}$, which is used to also create epk and tx_{red}. Sends tx_{red} to $\mathcal{A}$ to obtain tx_{ref}.
- $\mathcal{B}$ engage in Π_{pSig} with $\mathcal{A}$ using $tx_{\mathsf{red}}, \mathbb{C}'$ and obtains $\hat{\sigma}_{\mathsf{red}}$.
- $\mathcal{B}$ sends $\mathbb{C}, P, \pi$ to $\mathcal{A}$.
- $\mathcal{B}$ engage in Π_{pSig} with $\mathcal{A}$ using $tx_{\mathsf{ref}}, \mathbb{C}$ and obtains $\hat{\sigma}_{\mathsf{ref}}$.
- $\mathcal{A}$ sends $(tx^*_{\mathsf{ref}}, \sigma^*_{\mathsf{ref}})$ to $\mathcal{B}$.
- $\mathcal{A}$ sends b^* to $\mathcal{B}$, which in turn forwards it to the challenger.

Our adversary $\mathcal{B}$ perfectly simulates $\mathsf{redForge}^{G_0}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ and $\mathsf{redForge}^{G_1}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. If the adversary can distinguish the two games with provability higher than $1/2 + \mathsf{negl}(\lambda)$ this implies that it can be used to break VTC Timed Privacy. However, this contradicts with the assumption that the Verifiable Timed Commitment guarantees Privacy. Thus, $\Pr[\mathsf{Bad}_1] \leq \mathsf{negl}(\lambda)$ and this claim has been proven. ∎

Claim 2. *Let* Bad_2 *be the event that the challenger aborts because* $tx^*_{\mathsf{red}} = tx_{\mathsf{ref}}$. *Assume that the two party adaptor signature provides 2-aEUF-CMA Security (see Appendix C). Then* $\Pr[\mathsf{Bad}_2] \leq \mathsf{negl}(\lambda)$.

Proof: Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{Bad}_1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break 2-aEUF-CMA Security (see Appendix C) with the following steps:

- $\mathcal{B}$ receives pk_R from the challenger.
- $\mathcal{B}$ receives T from $\mathcal{A}$.
- $\mathcal{B}$ runs $(\mathbb{C}, w) \leftarrow \mathsf{createR}(1^\lambda)$ and shares $\mathbb{C}$ with $\mathcal{A}$.
- $\mathcal{B}$ runs $(pk_R, sk_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$, $(\ddot{pk}_R, \ddot{sk}_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$ and shares pk_R with $\mathcal{A}$.
- $\mathcal{A}$ returns (pk_S, sk_S) to $\mathcal{B}$, who forwards (pk_S, sk_S) to the challenger.
- $\mathcal{B}$ runs KAgg in order to obtain $pk_{S,R}$, which is used to also create epk and tx_{red}. Sends tx_{red} to $\mathcal{A}$ to obtain tx_{ref}.
- $\mathcal{B}$ calls $\mathcal{O}\Pi_{\mathsf{pSig}}$ of the challenger to run Π_{pSig} with $\mathcal{A}$ and obtain $\hat{\sigma}_{\mathsf{red}}$ using $\mathbb{C}$.
- $\mathcal{B}$ sends tx_{ref} as m^* to the challenger and receives $\mathbb{C}^{\mathsf{ref}}$.
- $\mathcal{B}$ runs $(P, \pi) \leftarrow \mathcal{S}(\mathbb{C}^{\mathsf{ref}}, T)$ and sends $\mathbb{C}^{\mathsf{ref}}, P, \pi$ to $\mathcal{A}$.

- Challenger starts Π_{pSig} protocol, using tx_{ref} and $\mathbb{C}^{\mathsf{ref}}$. $\mathcal{B}$ simply relays the messages between challenger and $\mathcal{A}$, until the protocol ends, outputting $\hat{\sigma}$.
- $\mathcal{A}$ sends $(tx^*_{\mathsf{red}}, \sigma^*_{\mathsf{red}})$ to $\mathcal{B}$.
- $\mathcal{B}$ sends σ^*_{red} as σ^* to the challenger.

Our adversary $\mathcal{B}$ perfectly simulates $\mathsf{redForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. If $\mathsf{isAuth}_{\mathsf{CL}}(tx^*_{\mathsf{red}}, \sigma^*_{\mathsf{red}}, epk) = 1$, this implies that $\mathsf{Vf}(tx^*_{\mathsf{red}}, \sigma^*_{\mathsf{red}}, pk_{S,R}) = 1$, which will also satisfied in the two party adaptor security game. Furthermore, while tx_{red} will be in memory of 2-aEUF-CMA Security, $tx^*_{\mathsf{red}} := tx_{\mathsf{ref}}$ will not be in memory. If the adversary is able to send a forgery before the T, expires, it will win $\mathsf{redForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$. This implies that an adversary winning $\mathsf{redForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ can be used to break 2-aEUF-CMA Security. However, this contradicts the assumption that the two party adaptor signature scheme provides 2-aEUF-CMA Security. Thus, $\Pr[\mathsf{Bad}_2] \leq \mathsf{negl}(\lambda)$ and this claim has been proven. ■

Claim 3. *Assume that the two party adaptor signature provides 2-aEUF-CMA Security (see Appendix C). Then* $\Pr[\mathsf{redForge}^{G_2}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof: Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{redForge}^{G_2}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break 2-aEUF-CMA Security (see Appendix C) with the following steps:

- $\mathcal{B}$ receives pk_R from the challenger.
- $\mathcal{B}$ receives T from $\mathcal{A}$.
- $\mathcal{B}$ runs $(\mathbb{C}, w) \leftarrow \mathsf{createR}(1^\lambda)$ and shares $\mathbb{C}$ with $\mathcal{A}$.
- $\mathcal{B}$ runs $(pk_R, sk_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$, $(\ddot{pk}_R, \ddot{sk}_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$ and shares pk_R with $\mathcal{A}$.
- $\mathcal{A}$ returns (pk_S, sk_S) to $\mathcal{B}$, who forwards (pk_S, sk_S) to the challenger.
- $\mathcal{B}$ runs KAgg in order to obtain $pk_{S,R}$, which is used to also create epk and tx_{red}. Sends tx_{red} to $\mathcal{A}$ to obtain tx_{ref}.
- $\mathcal{B}$ calls $\mathcal{O}\Pi_{\mathsf{pSig}}$ of the challenger to run Π_{pSig} with $\mathcal{A}$ and obtain $\hat{\sigma}_{\mathsf{red}}$ using $\mathbb{C}$.
- $\mathcal{B}$ runs $(\mathbb{C}^{\mathsf{ref}}, w^{\mathsf{ref}}) \leftarrow \mathsf{createR}(1^\lambda)$ and $(P, \pi) \leftarrow \mathcal{S}(\mathbb{C}^{\mathsf{ref}}, T)$ and sends $\mathbb{C}^{\mathsf{ref}}, P, \pi$ to $\mathcal{A}$.
- $\mathcal{B}$ calls $\mathcal{O}\Pi_{\mathsf{pSig}}$ of the challenger to run Π_{pSig} with $\mathcal{A}$ and obtain $\hat{\sigma}_{\mathsf{ref}}$ using $\mathbb{C}^{\mathsf{ref}}$.
- $\mathcal{A}$ sends $(tx^*_{\mathsf{red}}, \sigma^*_{\mathsf{red}})$ to $\mathcal{B}$.
- $\mathcal{B}$ sends tx^*_{red} to the challenger, and both engage in Π_{pSig} protocol, using a freshly created $\mathbb{C}'$ by the challenger. $\mathcal{B}$ can run this protocol because it received sk_S from $\mathcal{A}$ in a previous step. The protocol ends with $\hat{\sigma}^*_{\mathsf{red}}$.
- $\mathcal{B}$ sends σ^*_{red} to the challenger.

Our adversary $\mathcal{B}$ perfectly simulates $\mathsf{redForge}^{G_2}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. If $\mathsf{isAuth}_{\mathsf{CL}}(tx^*_{\mathsf{red}}, \sigma^*_{\mathsf{red}}, epk) = 1$, this implies that $\mathsf{Vf}(tx^*_{\mathsf{red}}, \sigma^*_{\mathsf{red}}, pk_{S,R}) = 1$, which will also satisfied in the two party adaptor security game. Furthermore, while $tx_{\mathsf{red}}, tx_{\mathsf{ref}}$ will be in memory of 2-aEUF-CMA Security, tx^*_{red} will not be in memory. If the adversary is able to send a forgery before the T, expires, it will win $\mathsf{redForge}^{G_2}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$. This implies that an adversary winning $\mathsf{redForge}^{G_2}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ can be used to break 2-aEUF-CMA Security. However, this contradicts the assumption that the two party adaptor signature scheme provides 2-aEUF-CMA Security and this claim has been proven. ■

This concludes the proof of Theorem 8. ■

Theorem 9 (CL refund unforgeability)**.** *Assume that the two party adaptor signature scheme satisfies two-party pre-signature adaptability. Then, our protocol offers CL redeemability according to Definition 6.*

Proof: $\mathsf{refForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$, formally defined in Figure 17, corresponds to the original game for CL refund unforgeability. The game is expanded with the interactions described in our implementation.

CL emulation: $\mathsf{refForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda)$
$(\mathbb{C}, T, \mathsf{st}_0) \leftarrow \mathcal{A}(1^\lambda)$
$(pk_S, sk_S) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$
$(\ddot{pk}_S, \ddot{sk}_S) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$
$(pk_R, sk_R, \mathsf{st}_a) \leftarrow \mathcal{A}(pk_S, \mathsf{st}_0)$
$pk_{S,R} \leftarrow \mathsf{KAgg}(pk_S, pk_R)$
$epk := pk_{S,R}$
$tx_{\mathsf{ref}} \leftarrow \mathsf{ctTx}_{\mathsf{ref}}(pk_{S,R}, \ddot{pk}_S)$
$(tx_{\mathsf{red}}, \mathsf{st}_b) \leftarrow \mathcal{A}(tx_{\mathsf{ref}}, \mathsf{st}_a)$
$(\hat{\sigma}_{\mathsf{red}}, \mathsf{st}_c) \leftarrow \Pi_{\mathsf{pSig}} \left\langle \begin{matrix} R(sk_S, tx_{\mathsf{red}}, \mathbb{C}), \\ \mathcal{A}(\mathsf{st}_b) \end{matrix} \right\rangle$
if $\mathsf{pVf}(tx_{\mathsf{red}}, \mathbb{C}, \hat{\sigma}_{\mathsf{red}}) = 0$ **abort**
$(P, \pi, \mathbb{C}^{\mathsf{ref}}, \mathsf{st}_d) \leftarrow \mathcal{A}(\mathsf{st}_c)$
if $\mathsf{Vf}(\pi, P, \mathbb{C}^{\mathsf{ref}}, T) = 0$ **abort**
$(\hat{\sigma}_{\mathsf{ref}}, \mathsf{st}_1) \leftarrow \Pi_{\mathsf{pSig}} \left\langle \begin{matrix} S(sk_S, tx_{\mathsf{ref}}, \mathbb{C}^{\mathsf{ref}}), \\ \mathcal{A}(\mathsf{st}_d) \end{matrix} \right\rangle$
if $\mathsf{pVf}(tx_{\mathsf{ref}}, \mathbb{C}^{\mathsf{ref}}, \hat{\sigma}_{\mathsf{ref}}) = 0$ **abort**
$(tx^*_{\mathsf{ref}}, \sigma^*_{\mathsf{ref}}) \leftarrow \mathcal{A}(\mathsf{st}_1)$
$b_0 := tx^*_{\mathsf{ref}} \notin \{tx_{\mathsf{red}}, tx_{\mathsf{ref}}\}$
$b_1 := \mathsf{Vf}(tx^*_{\mathsf{ref}}, \sigma^*_{\mathsf{ref}}, pk_{S,R})$
return $b_0 \wedge b_1$

Fig. 17. CL emulation: Experiment $\mathsf{refForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$.

Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{refForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break 2-aEUF-CMA Security (see Appendix C) with the following steps:

- $\mathcal{B}$ receives pk_S from the challenger.
- $\mathcal{B}$ receives $\mathbb{C}, T$ from $\mathcal{A}$.
- $\mathcal{B}$ runs $(\ddot{pk}_S, \ddot{sk}_S) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$ and shares pk_S with $\mathcal{A}$.
- $\mathcal{A}$ returns (pk_R, sk_R) to $\mathcal{B}$, who forwards (pk_R, sk_R) to the challenger.
- $\mathcal{B}$ runs KAgg in order to obtain $pk_{S,R}$, which is used to also create epk and tx_{ref}. Sends tx_{ref} to $\mathcal{A}$ to obtain tx_{red}.
- $\mathcal{B}$ calls $\mathcal{O}\Pi_{\mathsf{pSig}}$ of the challenger to run Π_{pSig} with $\mathcal{A}$ and obtain $\hat{\sigma}_{\mathsf{red}}$ using $\mathbb{C}$.
- $\mathcal{A}$ sends $P, \pi, \mathbb{C}^{\mathsf{ref}}$.
- $\mathcal{B}$ calls $\mathcal{O}\Pi_{\mathsf{pSig}}$ of the challenger to run Π_{pSig} with $\mathcal{A}$ and obtain $\hat{\sigma}_{\mathsf{ref}}$ using $\mathbb{C}^{\mathsf{ref}}$.
- $\mathcal{A}$ sends $(tx^*_{\mathsf{ref}}, \sigma^*_{\mathsf{ref}})$ to $\mathcal{B}$.
- $\mathcal{B}$ sends tx^*_{ref} to the challenger, and both engage in Π_{pSig} protocol, using a freshly created $\mathbb{C}'$ by the challenger. $\mathcal{B}$

can run this protocol because it received sk_S from $\mathcal{A}$ in a previous step. The protocol ends with $\hat{\sigma}^*_{\mathsf{ref}}$.

- $\mathcal{B}$ sends σ^*_{ref} to the challenger.

Our adversary $\mathcal{B}$ perfectly simulates $\mathsf{refForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. If $\mathsf{isAuth}_{\mathsf{CL}}(tx^*_{\mathsf{ref}}, \sigma^*_{\mathsf{ref}}, epk) = 1$, this implies that $\mathsf{Vf}(tx^*_{\mathsf{ref}}, \sigma^*_{\mathsf{ref}}, pk_{S,R}) = 1$, which will also satisfied in the two party adaptor security game. Furthermore, while $tx_{\mathsf{red}}, tx_{\mathsf{ref}}$ will be in memory of 2-aEUF-CMA Security, tx^*_{ref} will not be in memory. If the adversary is able to send a forgery before the T, expires, it will win $\mathsf{refForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$. This implies that an adversary winning $\mathsf{refForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ can be used to break 2-aEUF-CMA Security. However, this contradicts the assumption that the two party adaptor signature scheme provides 2-aEUF-CMA Security and this concludes the proof of Theorem 9. ∎

Theorem 10 (CL redeemability)**.** *Assume that the two party adaptor signature scheme satisfies two-party pre-signature adaptability. Then, our protocol offers CL redeemability according to Definition 6.*

Proof: $\mathsf{ExpRedeem}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$, formally defined in Figure 18, corresponds to the original game for CL redeemability. The game is expanded with the interactions described in our implementation.

CL emulation: $\mathsf{ExpRedeem}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda)$

$(\mathbb{C}, w, T, \mathsf{st}_0) \leftarrow \mathcal{A}(1^\lambda)$
$(pk_R, sk_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$
$(\ddot{pk}_R, \ddot{sk}_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$
$(pk_S, sk_S, \mathsf{st}_a) \leftarrow \mathcal{A}(pk_R, \mathsf{st}_0)$
$pk_{S,R} \leftarrow \mathsf{KAgg}(pk_S, pk_R)$
$epk := pk_{S,R}$
$tx_{\mathsf{red}} \leftarrow \mathsf{ctTx}_{\mathsf{red}}(pk_{S,R}, \ddot{pk}_R)$
$(tx_{\mathsf{ref}}, \mathsf{st}_b) \leftarrow \mathcal{A}(tx_{\mathsf{red}}, \mathsf{st}_a)$
$(\hat{\sigma}_{\mathsf{red}}, \mathsf{st}_c) \leftarrow \Pi_{\mathsf{pSig}} \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_b), \\ R(sk_R, tx_{\mathsf{red}}, \mathbb{C}) \end{matrix} \right\rangle$
if $\mathsf{pVf}(tx_{\mathsf{red}}, \mathbb{C}, \hat{\sigma}_{\mathsf{red}}) = 0$ **abort**
$(\mathbb{C}^{\mathsf{ref}}, w^{\mathsf{ref}}) \leftarrow \mathsf{createR}(1^\lambda)$
$(P, \pi) \leftarrow \mathsf{Commit}(T, w^{\mathsf{ref}})$
$\mathsf{st}_d \leftarrow \mathcal{A}(P, \pi, \mathbb{C}^{\mathsf{ref}}, \mathsf{st}_c)$
$(\hat{\sigma}_{\mathsf{ref}}, \mathsf{st}_1) \leftarrow \Pi_{\mathsf{pSig}} \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_d), \\ R(sk_R, tx_{\mathsf{ref}}, \mathbb{C}^{\mathsf{ref}}) \end{matrix} \right\rangle$
if $\mathsf{pVf}(tx_{\mathsf{red}}, \mathbb{C}^{\mathsf{ref}}, \hat{\sigma}_{\mathsf{ref}}) = 0$ **abort**
$\sigma_{\mathsf{red}} \leftarrow \mathsf{Adapt}(\hat{\sigma}_{\mathsf{red}}, w)$
$b_0 := \mathsf{Vf}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, pk_{S,R}) = 0$
$b_1 := (\mathbb{C}, w) \in \mathcal{R}$
return $b_0 \wedge b_1$

Fig. 18. CL emulation: Experiment $\mathsf{ExpRedeem}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$.

Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{ExpRedeem}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break Two-Party Adaptability (see Appendix C) with the following steps:

- $\mathcal{B}$ receives $(\mathbb{C}, w, T)$ from $\mathcal{A}$.
- $\mathcal{B}$ runs $(pk_R, sk_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$, $(\ddot{pk}_R, \ddot{sk}_R) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$ and shares pk_R with $\mathcal{A}$.
- $\mathcal{A}$ replies with pk_S and sk_S.
- $\mathcal{B}$ runs KAgg in order to obtain $pk_{S,R}$, which is used to also create epk and tx_{red}. Sends tx_{red} to $\mathcal{A}$ and receives tx_{ref}.
- $\mathcal{B}$ runs Π_{pSig} with $\mathcal{A}$, using tx_{red} and $\mathbb{C}$, until the protocol ends, outputting $\hat{\sigma}_{\mathsf{red}}$.
- $\mathcal{B}$ runs $(\mathbb{C}^{\mathsf{ref}}, w^{\mathsf{ref}}) \leftarrow \mathsf{createR}(1^\lambda)$, $(P, \pi) \leftarrow \mathsf{Commit}(T, w^{\mathsf{ref}})$ and shares P, π and $\mathbb{C}^{\mathsf{ref}}$ with $\mathcal{A}$.
- $\mathcal{B}$ runs Π_{pSig} with $\mathcal{A}$, using tx_{ref} and $\mathbb{C}^{\mathsf{ref}}$, until the protocol ends, outputting $\hat{\sigma}_{\mathsf{ref}}$.
- $\mathcal{B}$ checks if $\mathsf{pVf}(tx_{\mathsf{red}}, \mathbb{C}, \hat{\sigma}_{\mathsf{red}})$ verifies. If it verifies, $\mathcal{B}$ sends $\mathbb{C}$, w, pk_S, pk_R, tx_{red} and $\hat{\sigma}_{\mathsf{red}}$ to the challenger.

Our adversary $\mathcal{B}$ perfectly simulates $\mathsf{ExpRedeem}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if adversary wins $\mathsf{ExpRedeem}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$, this implies that $\mathcal{B}$ failed to obtain a valid signature when running $\mathsf{Adapt}(\hat{\sigma}_{\mathsf{red}}, w)$ with a valid witness. Therefore, the challenger will fail as well. However, this contradicts the assumption that the two party adaptor signature scheme provides Two-Party Adaptability and this concludes the proof of Theorem 10. ∎

Theorem 11 (CL refundability)**.** *Assume that verifiable timed dlog is sound and that the two party adaptor signature scheme satisfies two-party pre-signature adaptability. Then, our protocol offers CL refundability according to Definition 6.*

Proof: We consider the following game hops:

Game $\mathsf{ExpRefund}^{G_0}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$: This game, formally defined in Figure 19 without the grey line, corresponds to the original game for CL refundability. The game is expanded with the interactions described in our implementation.

Game $\mathsf{ExpRefund}^{G_1}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$: This game, formally defined in Figure 19, works exactly as G_0 with the exception highlighted in the grey line. The challenger, aborts if $(\mathbb{C}^{\mathsf{ref}}, w^{\mathsf{ref}}) \notin \mathcal{R}$.

Claim 4. *Let* Bad_1 *be the event that the challenger aborts in because* $(\mathbb{C}^{\mathsf{ref}}, w^{\mathsf{ref}}) \notin \mathcal{R}$. *Assume that the Verifiable Timed Commitment provides Soundness (see Appendix C). Then* $\Pr[\mathsf{Bad}_1] \leq \mathsf{negl}(\lambda)$.

Proof: Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{Bad}_1] > \mathsf{negl}(\lambda)$. We can construct an adversary bdv that uses $\mathcal{A}$ to break VTC Soundness (see Appendix C) with the following steps.

- $\mathcal{B}$ receives T from $\mathcal{A}$.
- $\mathcal{B}$ runs $(\mathbb{C}, w) \leftarrow \mathsf{createR}(1^\lambda)$ and shares $\mathbb{C}$ with $\mathcal{A}$.
- $\mathcal{B}$ runs $(pk_S, sk_S) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$, $(\ddot{pk}_S, \ddot{sk}_S) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$ and shares pk_S with $\mathcal{A}$.
- $\mathcal{A}$ returns (pk_R, sk_R) to $\mathcal{B}$, which runs KAgg in order to obtain $pk_{S,R}$, which is used to also create epk and tx_{red}.
- $\mathcal{B}$ sends tx_{red} to $\mathcal{A}$ to obtain tx_{ref}.
- $\mathcal{B}$ engages in Π_{pSig} with $\mathcal{A}$ and obtains $\hat{\sigma}_{\mathsf{red}}$ using $tx_{\mathsf{red}}, \mathbb{C}$.
- $\mathcal{A}$ sends $\mathbb{C}^{\mathsf{ref}}, P, \pi$ to $\mathcal{B}$.
- $\mathcal{B}$ engages in Π_{pSig} with $\mathcal{A}$ and obtains $\hat{\sigma}_{\mathsf{ref}}$ using $tx_{\mathsf{ref}}, \mathbb{C}^{\mathsf{ref}}$.
- $\mathcal{B}$ runs $w^{\mathsf{ref}} \leftarrow \mathsf{forceOpen}(P)$.
- $\mathcal{B}$ sends $(\mathbb{C}^{\mathsf{ref}}, P, \pi, T)$ to the challenger.

CL emulation: $\mathsf{ExpRefund}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda)$

$(T, \mathsf{st}_0) \leftarrow \mathcal{A}(1^\lambda)$
$(\mathbb{C}, w) \leftarrow \mathsf{createR}(1^\lambda)$
$\mathsf{st}_a \leftarrow \mathcal{A}(\mathbb{C}, \mathsf{st}_0)$
$(pk_S, sk_S) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$
$(\ddot{pk}_S, \ddot{sk}_S) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$
$(pk_R, sk_R, \mathsf{st}_b) \leftarrow \mathcal{A}(pk_S, \mathsf{st}_a)$
$pk_{S,R} \leftarrow \mathsf{KAgg}(pk_S, pk_R)$
$epk := pk_{S,R}$
$tx_{\mathsf{ref}} \leftarrow \mathsf{ctTx}_{\mathsf{ref}}(pk_{S,R}, \ddot{pk}_S)$
$(tx_{\mathsf{red}}, \mathsf{st}_c) \leftarrow \mathcal{A}(tx_{\mathsf{ref}}, \mathsf{st}_b)$
$(\hat{\sigma}_{\mathsf{red}}, \mathsf{st}_d) \leftarrow \Pi_{\mathsf{pSig}} \left\langle \begin{matrix} R(sk_S, tx_{\mathsf{red}}, \mathbb{C}), \\ \mathcal{A}(\mathsf{st}_c) \end{matrix} \right\rangle$
if $\mathsf{pVf}(tx_{\mathsf{red}}, \mathbb{C}, \hat{\sigma}_{\mathsf{red}}) = 0$ **abort**
$(P, \pi, \mathbb{C}^{\mathsf{ref}}, \mathsf{st}_e) \leftarrow \mathcal{A}(\mathsf{st}_d)$
if $\mathsf{Vf}(\pi, P, \mathbb{C}^{\mathsf{ref}}, T) = 0$ **abort**
$(\hat{\sigma}_{\mathsf{ref}}, \mathsf{st}_1) \leftarrow \Pi_{\mathsf{pSig}} \left\langle \begin{matrix} S(sk_S, tx_{\mathsf{ref}}, \mathbb{C}^{\mathsf{ref}}), \\ \mathcal{A}(\mathsf{st}_e), \end{matrix} \right\rangle$
if $\mathsf{pVf}(tx_{\mathsf{ref}}, \mathbb{C}^{\mathsf{ref}}, \hat{\sigma}_{\mathsf{ref}}) = 0$ **abort**
$w^{\mathsf{ref}} \leftarrow \mathsf{forceOpen}(P)$
if $(\mathbb{C}^{\mathsf{ref}}, w^{\mathsf{ref}}) \notin \mathcal{R}$ **abort**
$\sigma_{\mathsf{ref}} \leftarrow \mathsf{Adapt}(\hat{\sigma}_{\mathsf{ref}}, w^{\mathsf{ref}})$
return $\mathsf{Vf}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}, pk_{S,R}) = 0$

Fig. 19. CL emulation: Experiments $\mathsf{ExpRefund}^{G_0}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ (ignoring the line highlighted in grey) and $\mathsf{ExpRefund}^{G_1}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ (including line highlighted in grey).

Our adversary $\mathcal{B}$ perfectly simulates $\mathsf{ExpRefund}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if adversary wins $\mathsf{ExpRefund}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$, this implies that $\mathsf{Vf}(\pi, P, \mathbb{C}^{\mathsf{ref}}, T) = 1$ which also satisfies VTC Soundness. Moreover, $\mathsf{forceOpen}(P)$ outputs w^{ref} such that $(\mathbb{C}^{\mathsf{ref}}, w^{\mathsf{ref}}) \notin \mathcal{R}$ ($\Pr[\mathsf{Bad}_1] > \mathsf{negl}(\lambda)$ assumption). This implies that an adversary winning $\mathsf{ExpRefund}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ can be used to break VTC Soundness. However, this contradicts the assumption that the Verifiable Timed Commitment provides Soundness and this claim has been proven. ■

Claim 5. *Assume that the two party adaptor signature provides Two-Party Pre-Signature Adaptability (see Appendix C). Then* $\Pr[\mathsf{ExpRefund}^{G_1}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof: Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{ExpRefund}^{G_1}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adersary $\mathcal{B}$ that uses $\mathcal{A}$ to break Two-Party Pre-Signature Adaptability (see Appendix C) with the following steps:

- $\mathcal{B}$ receives T from $\mathcal{A}$.
- $\mathcal{B}$ runs $(\mathbb{C}, w) \leftarrow \mathsf{createR}(1^\lambda)$ and shares $\mathbb{C}$ with $\mathcal{A}$.
- $\mathcal{B}$ runs $(pk_S, sk_S) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$, $(\ddot{pk}_S, \ddot{sk}_S) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$ and shares pk_S with $\mathcal{A}$.
- $\mathcal{A}$ returns (pk_R, sk_R) to $\mathcal{B}$, which runs KAgg in order to obtain $pk_{S,R}$, which is used to also create epk and tx_{red}.
- $\mathcal{B}$ sends tx_{red} to $\mathcal{A}$ to obtain tx_{ref}.
- $\mathcal{B}$ engages in Π_{pSig} with $\mathcal{A}$ and obtains $\hat{\sigma}_{\mathsf{red}}$ using $tx_{\mathsf{red}}, \mathbb{C}$.
- $\mathcal{A}$ sends $\mathbb{C}^{\mathsf{ref}}, P, \pi$ to $\mathcal{B}$.
- $\mathcal{B}$ engages in Π_{pSig} with $\mathcal{A}$ and obtains $\hat{\sigma}_{\mathsf{ref}}$ using $tx_{\mathsf{ref}}, \mathbb{C}^{\mathsf{ref}}$.
- $\mathcal{B}$ runs $w^{\mathsf{ref}} \leftarrow \mathsf{forceOpen}(P)$.
- $\mathcal{B}$ runs $\sigma_{\mathsf{ref}} \leftarrow \mathsf{Adapt}(\hat{\sigma}_{\mathsf{ref}}, w^{\mathsf{ref}})$
- $\mathcal{B}$ sends $(tx_{\mathsf{ref}}, \mathbb{C}^{\mathsf{ref}}, w^{\mathsf{ref}}, pk_S, pk_R, \hat{\sigma}_{\mathsf{ref}}, \sigma_{\mathsf{ref}})$ to the challenger.

Our adversary $\mathcal{B}$ perfectly simulates $\mathsf{ExpRefund}^{G_1}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if adversary wins $\mathsf{ExpRefund}_{\Pi_{CL},\mathcal{R},\mathcal{A}} - G_1$, this implies that $\mathcal{B}$ failed to obtain a valid signature when running $\mathsf{Adapt}(\hat{\sigma}_{\mathsf{red}}, w)$ with a valid witness. Therefore, the challenger will fail as well. However, this contradicts the assumption that the two party adaptor signature scheme provides Two-Party Adaptability and this claim has been proven. ■

This concludes the proof of Theorem 11. ■

Theorem 12 (CL extractability)**.** *Assume that the two party adaptor signature scheme satisfies two-party witness extractability. Then, our protocol offers CL extractability according to Definition 6.*

Proof: $\mathsf{ExpExtract}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$, formally defined in Figure 20, corresponds to the original game for CL redeem unforgeability. The game is expanded with the interactions described in our implementation.

CL emulation: $\mathsf{ExpExtract}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda)$

$(\mathbb{C}, T, \mathsf{st}_0) \leftarrow \mathcal{A}(1^\lambda)$
$(pk_S, sk_S) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$
$(\ddot{pk}_S, \ddot{sk}_S) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$
$(pk_R, sk_R, \mathsf{st}_a) \leftarrow \mathcal{A}(pk_S, \mathsf{st}_0)$
$pk_{S,R} \leftarrow \mathsf{KAgg}(pk_S, pk_R)$
$epk := pk_{S,R}$
$tx_{\mathsf{ref}} \leftarrow \mathsf{ctTx}_{\mathsf{ref}}(pk_{S,R}, \ddot{pk}_S)$
$(tx_{\mathsf{red}}, \mathsf{st}_b) \leftarrow \mathcal{A}(tx_{\mathsf{ref}}, \mathsf{st}_a)$
$(\hat{\sigma}_{\mathsf{red}}, \mathsf{st}_c) \leftarrow \Pi_{\mathsf{pSig}} \left\langle \begin{matrix} R(sk_S, tx_{\mathsf{red}}, \mathbb{C}), \\ \mathcal{A}(\mathsf{st}_b) \end{matrix} \right\rangle$
if $\mathsf{pVf}(tx_{\mathsf{red}}, \mathbb{C}, \hat{\sigma}_{\mathsf{red}}) = 0$ **abort**
$(P, \pi, \mathbb{C}^{\mathsf{ref}}, \mathsf{st}_d) \leftarrow \mathcal{A}(\mathsf{st}_c)$
if $\mathsf{Vf}(\pi, P, \mathbb{C}^{\mathsf{ref}}, T) = 0$ **abort**
$(\hat{\sigma}_{\mathsf{ref}}, \mathsf{st}_1) \leftarrow \Pi_{\mathsf{pSig}} \left\langle \begin{matrix} S(sk_S, tx_{\mathsf{ref}}, \mathbb{C}^{\mathsf{ref}}), \\ \mathcal{A}(\mathsf{st}_d) \end{matrix} \right\rangle$
if $\mathsf{pVf}(tx_{\mathsf{ref}}, \mathbb{C}^{\mathsf{ref}}, \hat{\sigma}_{\mathsf{ref}}) = 0$ **abort**
$\sigma_{\mathsf{red}} \leftarrow \mathcal{A}(\mathsf{st}_1)$
$w \leftarrow \mathsf{Extract}(\sigma_{\mathsf{red}}, \hat{\sigma}_{\mathsf{red}}, \mathbb{C})$
$b_0 := \mathsf{Vf}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, pk_{S,R})$
$b_1 := (\mathbb{C}, w) \notin \mathcal{R}$
return $b_0 \wedge b_1$

Fig. 20. CL emulation: Experiment $\mathsf{ExpExtract}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$.

Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{ExpExtract}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break Two-Party

Witness Extractability (see Appendix C) with the following steps:

- $\mathcal{B}$ receives pk_S from the challenger.
- $\mathcal{B}$ receives $(\mathbb{C}, T)$ from $\mathcal{A}$.
- $\mathcal{B}$ runs $(\ddot{pk}_S, \ddot{sk}_S) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$ and shares pk_S with $\mathcal{A}$.
- $\mathcal{A}$ replies with pk_R, sk_R.
- $\mathcal{B}$ runs KAgg in order to obtain $pk_{S,R}$, which is used to also create epk and tx_{ref}. tx_{ref} is sent to $\mathcal{A}$ to obtain tx_{red}. $\mathcal{B}$ sends pk_R, sk_R, tx_{red},$\mathbb{C}$ to challenger in two stages. First, the keys, then the transaction and $\mathbb{C}$.
- Challenger starts Π_{pSig} protocol, using tx_{red} and $\mathbb{C}$. $\mathcal{B}$ simply relays the messages between challenger and $\mathcal{A}$, until the protocol ends, outputting $\hat{\sigma}$.
- $\mathcal{A}$ sends $P, \pi, \mathbb{C}^{\mathsf{ref}}$.
- $\mathcal{B}$ calls $\mathcal{O}\Pi_{\mathsf{pSig}}$ of the challenger to run Π_{pSig} with $\mathcal{A}$ and obtain $\hat{\sigma}_{\mathsf{ref}}$ using $\mathbb{C}$.
- $\mathcal{A}$ sends σ_{red} to $\mathcal{B}$, which forwards it as σ to the challenger.

Our adversary $\mathcal{B}$ perfectly simulates $\mathsf{ExpExtract}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. If condition b_0 is satisfied in $\mathsf{ExpExtract}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ this means that $\mathsf{Vf}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, pk_{S,R}) = 1$, which will also be satisfied in the adaptor Two-Party Witness Extractability. Finally, b_1 is equivalent to the second winning condition for Two-Party Witness Extractability. This implies that an adversary that wins $\mathsf{ExpExtract}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ can be used to break Two-Party Witness Extractability. Note that $\mathcal{A}$ does not have oracle access, and that tx_{red} is the message used to generate the presignature with the challenger, ensuring that it will not be in the challenger's memory. However, this contradicts the assumption that the two party adaptor signature scheme provides Two-Party Witness Extractability and this concludes the proof of Theorem 12. ■

J. *Witness unforgeability is equivalent to hard relation*

An additional property, called *witness unforgeability* could provide guarantees to an honest sender that the receiver cannot generate a redeem transaction without knowledge of the corresponding witness. We define this property as follows.

Definition 15 (CL Witness Unforgeability)**.** *A CL is said to offer CL Witness Unforgeability 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 $\Pr[\mathsf{wForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$, where $\mathsf{wForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ is defined in Fig. 21.*

If we assume that the hard relation does not hold, it is trivial to see that this property does not hold either. Now, we show that if the hard relation assumption holds, this property must hold as well.

Theorem 18 (CL Witness Unforgeability)**.** *Assume that that the hard relation assumption holds. Then, CL offers CL redeem unforgeability according to Definition 15.*

Proof: Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{wForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break the hard relation, with the following steps:

- The challenger produces they an instance of the hard relation, $(\mathbb{C}, w)$ and shares $\mathbb{C}$ with $\mathcal{B}$.

$\mathsf{wForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}(\lambda)$

$$(T, \mathsf{st}_0) \leftarrow \mathcal{A}(1^\lambda)$$
$$(\mathbb{C}, w) \leftarrow \mathsf{createR}(1^\lambda)$$
$$\left\langle \begin{matrix} (epk, \mathsf{aux}_S, \mathsf{aux}_\mathcal{A}, tx_{\mathsf{red}}, tx_{\mathsf{ref}}), \\ (epk, \mathsf{aux}_\mathcal{A}, tx_{\mathsf{red}}, tx_{\mathsf{ref}}, \mathsf{st}_1) \end{matrix} \right\rangle \leftarrow \mathsf{ctEAcc}\left\langle \begin{matrix} S(\mathbb{C}, T), \\ \mathcal{A}(\mathsf{st}_0) \end{matrix} \right\rangle$$
$$\sigma_{\mathsf{red}} \leftarrow \mathcal{A}(\mathsf{st}_1)$$
$$w' \leftarrow \mathsf{GWit}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, \mathsf{aux}_S)$$
$$b_0 := \mathsf{isAuth}_{\mathsf{CL}}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}, epk)$$
$$b_1 := (\mathbb{C}, w') \in \mathcal{R}$$
$$\textbf{return } b_0 \wedge b_1$$

Fig. 21. Experiment for CL Witness Unforgeability.

- $\mathcal{B}$ receives T and st_0 from $\mathcal{A}$, and follows all the steps as described in Fig. 21.
- $\mathcal{B}$ runs algorithm GWit using aux_S, $tx_{\mathsf{red}}, \sigma_{\mathsf{red}}$ as input to generate w'.
- $\mathcal{B}$ forwards w' to the challenger.

Our adversary $\mathcal{B}$ perfectly simulates $\mathsf{wForge}_{\Pi_{CL},\mathcal{R},\mathcal{A}}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm.

If $\mathsf{isAuth}_{\mathsf{CL}}$ returns 1 for $(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}, epk)$ and $(\mathbb{C}, w') \in \mathcal{R}$, this means that $(\mathbb{C}, w') \in \mathcal{R}$ will also hold for the challenger. However, this contradicts the assumption that the hard relation assumption holds, so $\mathcal{A}$ cannot exist and this concludes the proof of Theorem 18. ■