CBDC-Cash: How to Fund and Defund CBDC Wallets

Diego Castejon-Molina
diego.castejon@imdea.org
IMDEA Software Institute
Universidad Politécnica de Madrid
Madrid, Spain

Dimitrios Vasilopoulos
dimitrios.vasilopoulos@imdea.org
IMDEA Software Institute
Madrid, Spain

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

ABSTRACT

The interest shown by central banks in deploying Central Bank Digital Currency (CBDC) has spurred a blooming number of conceptually different proposals from central banks and academia. Yet, they share the common, transversal goal of providing citizens with an additional digital monetary instrument. Citizens, equipped with CBDC wallets, should have access to CBDC fund and defund operations that allow the distribution of CBDC from the central bank to citizens with the intermediation of commercial banks. Despite their key role in the CBDC deployment as acknowledged, e.g., by the European Central Bank, operations fund and defund have not been formally studied yet. In this state of affairs, this work strives to cryptographically define the problem of fund and defund of CBDC wallets as well as the security and privacy notions of interest.

We consider a setting with three parties (citizen, commercial bank and central bank) and three ledgers: the CBDC ledger, the retail ledger (where citizens have their accounts with their commercial banks) and the wholesale ledger (where commercial banks have their accounts with the central bank). We follow a modular approach, initially defining the functionality of two types of ledgers: Basic Ledger (BL), which supports basic transactions, and Conditional Payment Ledger (CP), which additionally supports conditional transactions. We then use BL and CP to define the CBDC-Cash Environment (CCE) primitive, which captures the core functionality of operations fund and defund. We require that CCE satisfies balance security: either operation fund/defund is successful, or no honest party loses their funds. CCE also satisfies that fund/defund cannot be used to breach the privacy of the CBDC ledger. Finally, we provide two efficient and secure constructions for CCE to cover both CP and BL types of CBDC ledger. Our performance evaluation shows that our constructions impose small computation and communication overhead to the underlying ledgers.

The modular design of CCE allows for the incorporation in our CCE constructions of any CBDC ledger proposal that can be proven a secure instance of CP or BL, enabling thereby a seamless method to provide CBDC fund and defund operations.

1 INTRODUCTION

Recently, several central banks have shown interest in creating a digital version of their physical currency, called *Central Bank Digital Currency* (CBDC) [3, 6, 7, 9–13, 16, 17, 21, 24, 27, 36–38, 46, 50, 52, 55–58, 65]. Spurred from this interest, recent academic works have contributed further proposals for CBDC [41, 46, 64, 68]. The different motivation behind each of the aforementioned works has resulted in a heterogenous set of CBDC proposals. For example, KSI-cash [36] and Hamilton [46] build upon a custom made blockchain technology tailored for this purpose; Bakong [50] relies on Hyperledger Iroha; Itcoin [13, 65] is based on Bitcoin's payment channel network; TIPS+ [65] is based on an existing payment service of the Eurosystem [34]; while Platypus [68] and the proposal published by Swiss National Bank (SNB) [21] are based on eCash.

A key functionality transversal to all previous proposals but not formally explored is the distribution of CBDC from central banks to citizens with the intermediation of commercial banks, analogously to how citizens can obtain cash. Following the terminology of the European Central Bank (ECB), this functionality involves two operations: *fund* and *defund* citizen CBDC wallets [33]. For operation fund, analogously to how a citizen can obtain cash in exchange for a payment to her commercial bank, our understanding is that a citizen should be able to fund her CBDC wallet by paying to her commercial bank in exchange for the corresponding CBDC units. Operation defund is then defined in the reverse manner, i.e., the citizen should be able to return CBDC units in exchange for the corresponding increase at her commercial bank account.

Given the heterogeneity in current CBDC proposals, it is inconceivable to design protocols for operations fund and defund tailored to the design choices of each proposal. Instead, in this work we strive to place fund and defund as well as their security and privacy notions of interest on a firm cryptographic foundation. We note that this goal is in line with recent requirements from ECB for the digital euro: "*The set of funding and defunding functionalities to be offered by supervised intermediaries would need to ensure a common baseline and user experience, irrespective of the supervised intermediary that provides them with the digital euro*", where here the term supervised intermediary refers to financial institutions supervised by the central bank, such as commercial banks.

1.1 Our Approach

The Ledgers. Instead of considering the implementation details of every CBDC ledger proposed so far (and yet to come), as well as the ledgers used in the banking sector, we define the minimal functionalities shared across these ledgers required to support funding and defunding CBDC wallets. A ledger must provide: (1) a list of

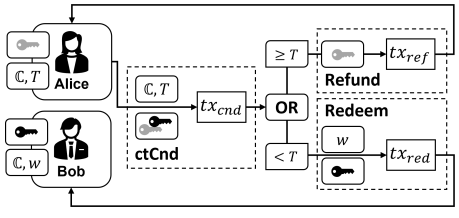

Figure 1: CP transactions: Alice and Bob create a conditional payment transaction locking Alice's coins with condition $\mathbb{C}$ and timeout T. Before T expires, Bob can redeem with witness w. After T expires, if Bob did not redeem, Alice refunds.

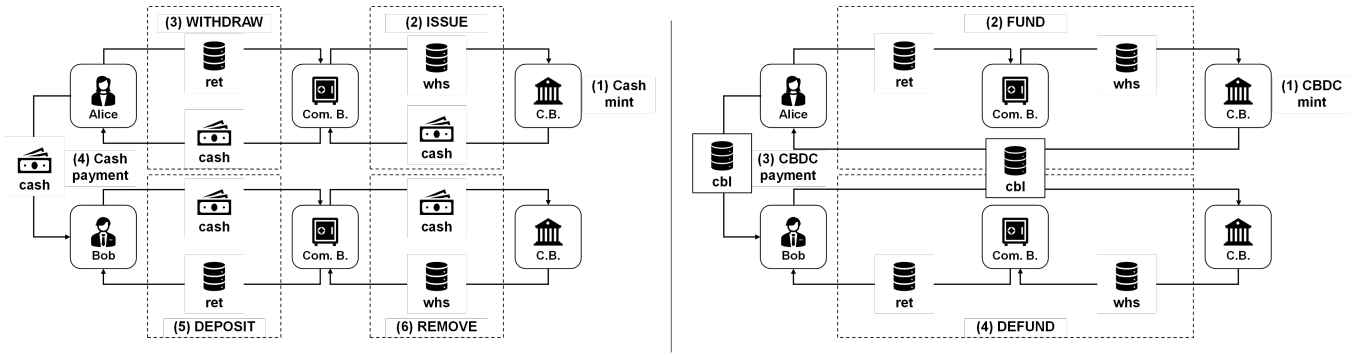

Figure 2: Lifecycle of cash (left) and lifecycle of CBDC (right). The distribution of both cash and CBDC is intermediated by supervised entities (i.e., commercial banks). Arrows indicate the direction of payment flows. The same banknote is passed on to different parties (left figure). The CBDC ledger (cbl) is duplicated for readability, but there is only one cbl ledger (right figure).

ordered transactions; (2) a double-spending prevention mechanism; and (3) a transaction validation mechanism.

The transaction validation mechanism heavily relies on the transaction format supported by the ledger. Hence, we coarsely group existing ledgers into two types. First, *basic ledgers* (BL) that support basic transactions transferring funds from a sender account to a receiver account if the transaction is correctly authorized by the sender. Ledgers in [20–22, 36, 68] are examples of BL. Second, *conditional payment ledgers* (CP) that support the additional functionality illustrated in Figure 1. Alice (the sender) and Bob (the receiver) can jointly create a *conditional payment transaction* (tx_{cnd}) that transfers Alice's coins into an escrow account, which is shared by both parties and annotated with a condition $\mathbb{C}$ and a timeout T. Thenceforth, Bob can claim the escrowed coins using the solution w to condition $\mathbb{C}$ in order to create a *redeem transaction* (tx_{red}), before timeout T expires. Alternatively, Alice can create a *refund transaction* (tx_{ref}) to reclaim the coins after timeout T expires. Note that it is important that the ledger's double-spending prevention mechanism ensures that after tx_{cnd} is published, only one of transaction tx_{red} and tx_{ref} is successfully submitted to the ledger. Ledgers in [13, 15, 44, 46, 49, 65] are examples of CP.

We define the security and privacy properties of interest for the two ledger types considered in this work. In the case of BL, the notion of security we want is *transaction unforgeability*, i.e., our adversary should not be able to steal funds from an honest party's account submitting transactions to the ledger (it is not enough that she forges a valid authorization for a transaction if the ledger rejects it). As regards to CP, the link between CP transactions (i.e. tx_{red} or tx_{ref} can only spend from a specific escrow account, created with a specific tx_{cnd}), mandates more elaborated security properties defined with respect to each party's role. For instance, if we focus on redeem transaction type, tx_{red}, (c.f. Figure 1), Alice (i.e., the sender of tx_{cnd}) requires an additional unforgeability property, which ensures her that Bob (i.e., the rightful receiver of tx_{red}) cannot successfully authorize and submit tx_{red} on the ledger without the knowledge of the solution w to condition $\mathbb{C}$. Continuing with the same example, Bob (i.e., the receiver of tx_{red}) requires an additional security property, which ensures him that he can always successfully authorize and submit tx_{red} on the ledger before the timeout expires, provided that he knows the solution w to condition $\mathbb{C}$. Similarly, we define security tailored to other transaction types.

Concerning privacy, we require that an observer of ledger transactions (e.g., the central bank) does not infer information about who pays what to whom. As stated by ECB: "*the Eurosystem would not be able to infer how many digital euro any individual end user held nor to infer end user's payment patterns*", where here the Eurosystem is the central bank of the euro-area. We characterize this requirement with the notion of *transaction indistinguishability*, which we borrow from existing CBDC proposals [21, 68]. Here, the adversary succeeds if it manages to distinguish between two transactions between accounts not under her control.

We use the definition of the two ledger types as a building block to provide a modular design for operations fund/defund.

Understanding Operations Fund/Defund from the Lifecycle of Cash. The lifecycle of cash [8, 25, 29, 42, 60], describes how *cash* is distributed from the central bank to citizens. This process uses two ledgers: (1) the *retail ledger* (ret), that is maintained by a commercial bank and manages transactions between the commercial bank and citizens having an account with it; and (2) the *wholesale ledger* (whs), that is maintained by the central bank and manages transactions between the central bank itself and the various commercial banks within its jurisdiction. Figure 2 (left) depicts the involved steps: (1) the central bank *mints* new banknotes; (2) the central bank *issues* cash exchanging the new banknotes with a commercial bank for a payment in whs; (3) citizens can *withdraw* cash from an automatic teller machine (ATM), having their account in ret debited by their commercial bank; (4) citizens can use cash to perform *cash payments* between each other; (5) cash holders may wish to *deposit* cash back into their bank account: citizen's account in ret is credited and the commercial bank receives banknotes; and (6) the central bank can *remove* cash from circulation exchanging banknotes from a commercial bank for a payment on whs.

The aforementioned process for cash distribution has limitations. To name a few: (1) central banks know the volume of cash issued to commercial banks, but not the volume used by citizens [30] and (2) the distribution of cash requires a large infrastructure of central bank and commercial bank branches [31]. More accurate statistics of units in circulation may assist central banks in the design of

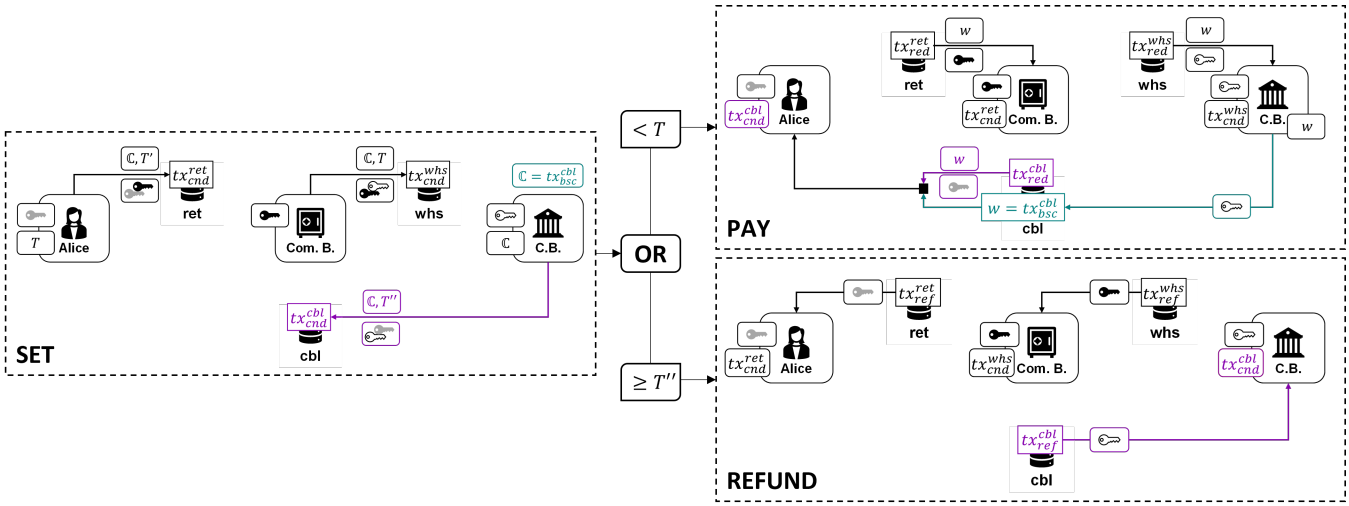

Figure 3: Operation fund modelled with CCE. Violet is for CP and teal is for BL whereas black elements denote common functionalities. Parties engage in Set **creating conditional payments. If ledger time is $< T$, the central bank can complete the operation revealing w or submitting $tx^{\text{cbl}}_{\text{bsc}}$. Otherwise, parties get refunded.**

monetary policies, while a smaller infrastructure for central bank money distribution reduces the costs of the system.

In this state of affairs, a digital version of cash, namely CBDC, could provide a way to overcome the aforementioned limitations. Two high-level possibilities could be contemplated for the deployment of CBDC: (1) faithfully mimic the lifecycle of cash, replacing cash payments with CBDC transactions; or (2) remove the intermediation of commercial banks and allow citizens to receive CBDC directly from the central bank. Central banks are not aligned with the second option and have expressed on several occasions that CBDC should not rule out intermediaries [3, 7, 9, 17, 21, 24, 33, 36, 50, 52, 55, 65] because: (1) central banks lack the infrastructure, experience and capabilities to deal directly with citizens, and (2) the disintermediation could have negative economic effects (e.g. on financial stability). Moreover, ECB has recently stated that "*Supervised intermediaries would also perform funding and defunding-related tasks*".

Therefore, inspired from the lifecycle of cash, we design how a CBDC wallet can be funded and defunded as depicted in Figure 2 (right). The digital nature of CBDC ledger (cbl) allows the combination of *issue* and *withdraw* in a single operation (the *fund* operation), as well as *deposit* and *remove* in another operation (the *defund* operation). We combine these operations rather than just replicating the lifecycle of cash for two reasons. First, for efficiency since this reduces the number of transactions required for operations fund/defund (c.f. Figure 2), without removing the intermediation of commercial banks. Second, to ensure that the central bank has more accurate statistics about CBDC units in circulation. The CBDC lifecycle then works as follows: (1) the central bank *mints* new CBDC units in cbl; (2) the central bank, the commercial bank and the citizen engage in *fund*. This operation ends with citizen's account with commercial bank in ret being debited, and her CBDC wallet in cbl credited; (3) citizens with funded CBDC wallets can perform *CBDC payments*; (4) the central bank, the commercial bank and the citizen engage in *defund*. This operation ends with the account of the citizen with commercial bank in ret being credited, and her account in cbl debited.

In summary, similar to methods available in the literature for payment synchronization across ledgers [4, 23, 40, 45, 47, 48, 59, 62, 69], we can see operations fund and defund as a cyclic swap where each party wishes to receive a payment from a second party before paying to the third one. More specifically, in operation fund: (1) the central bank delivers CBDC units to the citizen in cbl only if it receives a payment from citizen's commercial bank in whs; (2) in turn, the commercial bank pays the central bank only if it gets paid by the citizen in ret; and (3) the latter pays the commercial bank only if it receives CBDC units in cbl. Conversely, in operation defund: (1) the citizen pays CBDC units to the central bank in cbl only if it receives a payment from its commercial bank in ret; (2) which in turn pays the citizen only if it gets paid by the central bank on whs; and (3) the latter pays the commercial bank only if it receives CBDC units in cbl. It becomes thereby apparent that fund and defund are the same operation reversing the roles of sender and receiver. This observation allows us to formally characterize the core functionality for both operations as a cryptographic primitive that involves three parties and three ledgers. We call it *CBDC-Cash Environment* and describe it next.

The CBDC-Cash Environment (CCE) Functionality. Intuitively, CCE captures the synchronization of payments across three ledgers: the retail ledger (ret), the wholesale ledger (whs) and the CBDC ledger (cbl). We model both ret and whs as CP. This reflects the fact that available implementations of both ledgers provide functionality captured by the definition of CP. For instance, available ret ledgers are associated with credit/debit cards that allow for authorization holds [67]. An example of whs is TARGET2 [35], operated by ECB, that supports *Payment versus Payment* (PvP) operations [32]. PvP links the successful execution of a transaction on a wholesale or

retail ledger to a payment happening on another ledger. Finally, since we do not want to tie our definition of CCE to any CBDC proposal, we consider that cbl could be either of type CP or BL.

Following with our modular design, CCE is thereby defined over three ledgers Π_{CP_0}, Π_{CP_1} and cbl and comprises three parties, which interact over these three ledgers: (1) debtor D is the party handing over CBDC, (2) intermediary I is the party facilitating the exchange, and (3) creditor C is the party receiving CBDC. CCE defines two protocols, namely, Set and Pay. Set is a three-party protocol where D, I and C jointly create two conditional transactions tx^0_{cnd}, tx^1_{cnd} on ledgers Π_{CP_0} and Π_{CP_1}, respectively, and an additional transaction $tx^{\mathsf{cbl}}_{\mathsf{cnd}}$ on ledger cbl, if the latter is of type CP. Pay is also a three-party protocol where D, I and C jointly create two redeem transactions tx^0_{red}, tx^1_{red} on ledgers Π_{CP_0} and Π_{CP_1}, respectively, and an additional transaction on ledger cbl that is either (1) a redeem transaction $tx^{\mathsf{cbl}}_{\mathsf{red}}$ if cbl is of type CP; or (2) a basic transaction $tx^{\mathsf{cbl}}_{\mathsf{bsc}}$ if cbl is a BL.

We have additionally defined the security and privacy notions of interest. As regards to security, we define *balance security*, meaning that an honest party does not lose coins, even when the other two parties collude. We get inspired for this security notion by Requirement 14 of the report on digital euro [24]: Instead of assuming that both central and commercial banks are trusted entities that collaborate with each other, CCE guarantees balance security even in the event that a commercial bank, which is the victim of an attack, attempts to steal funds in fund/defund operations. Concerning privacy, we require that the privacy notion of cbl is not breached when used as one of the ledgers for CCE.

Finally, in this work we provide concrete constructions for CCE, which we describe next.

Our Constructions of CCE. We provide two constructions for CCE corresponding to the cases that cbl is of type BL and CP, respectively. We also formally prove that both constructions achieve security according to CCE definition. In the construction where ledger cbl is of type CP (c.f. violet flow in Figure 3), all three ledgers support conditional payment transactions. Hence, in protocol Set we can use the same condition $\mathbb{C}$ in order to coordinate the conditional payment transactions required by the cyclic swap. Then, all parties redeem if debtor D reveals the solution w to condition $\mathbb{C}$ and protocol Pay is invoked before timeout T expires. Otherwise, all parties can refund their locked coins after timeout T'' expires.

In the construction where ledger cbl is of type BL (c.f. teal flow in Figure 3), not all three ledgers support conditional payment transactions. As a consequence, there exist no obvious way to coordinate payments across ledgers Π_{CP_0}, Π_{CP_1} and cbl, making the execution of the cyclic swap challenging. To overcome this hurdle, we leverage the *Payment versus Payment* (PvP) functionality available in existing retail and wholesale ledgers. Hence, we build our construction upon two conditional payment ledgers Π_{CP_0} and Π_{CP_1} that allow for conditional payment transaction which can be redeemed if and only if a specific transaction $tx^{\mathsf{cbl}}_{\mathsf{bsc}}$ is accepted in cbl. The blueprint of our construction is as follows: in protocol Set we use a specific transaction $tx^{\mathsf{cbl}}_{\mathsf{bsc}}$ (not yet submitted to cbl) as the condition $\mathbb{C}^*$ for transactions tx^0_{cnd}, tx^1_{cnd} on ledgers Π_{CP_0} and Π_{CP_1}. These transactions can be redeemed if in protocol Pay debtor D submits $tx^{\mathsf{cbl}}_{\mathsf{bsc}}$ to cbl before timeout T expires. Otherwise, creditor C and intermediary I can refund their locked coins after timeout T'' expires. Thus, PvP is a fundamental property as it allows that our two constructions for CCE follow the same blueprint regardless of the CBDC ledger type.

Fund/Defund a CBDC Wallet. Leveraging CCE, one could implement operations fund and defund assigning the roles of debtor and creditor to the appropriate parties as well as setting ledgers Π_{CP_0}, Π_{CP_1} as ret or whs, accordingly. More specifically, in operation fund, the central bank is debtor D, the commercial bank is intermediary I and the citizen is creditor C, while ledgers Π_{CP_0}, Π_{CP_1} correspond to ret and whs, respectively, as illustrated in Figure 3. At the beginning, creditor C selects an appropriate timeout T and debtor D creates a pair of condition $\mathbb{C}$ and its corresponding solution w. The three parties invoke protocol Set with input timeout T and condition $\mathbb{C}$, which outputs two conditional transactions tx^0_{cnd}, tx^1_{cnd} on ledgers Π_{CP_0} and Π_{CP_1}, respectively, and an additional transaction $tx^{\mathsf{cbl}}_{\mathsf{cnd}}$ on ledger cbl, if the latter is of type CP. Thereafter, they can complete the operation invoking protocol Pay on input w and the conditional transactions created by protocol Set before timeout T expires. If the timeout expires (or protocol Pay did not terminate successfully), each party can, on its own, execute the refund operation available at the underlying CP ledger. Conversely, in operation defund, the citizen is debtor D, the commercial bank is intermediary I and the central bank is creditor C, while ledgers Π_{CP_0}, Π_{CP_1} correspond to whs and ret, respectively. Henceforth, protocols Set and Pay are invoked in the same manner as in operation fund.

Overall, our CBDC-cash environment, contributes to the intermediated distribution of CBDC from central banks to citizens in two ways. On the one hand, it captures the core functionality required for operations fund and defund. Hence, any construction that satisfies the security and privacy notions of CCE can be used to fund and defund CBDC wallets. On the other hand, the modular nature of our constructions allows for the integration of any combination of wholesale, retail and CBDC ledgers. The requisite, in this case, is that they are all secure instantiations of CP or BL accordingly.

1.2 Our Contributions

Our contributions can be summarized as follows:

(1) New Cryptographic Primitives for Ledgers. We present two new cryptographic primitives: *basic ledger* (BL) and *conditional payment ledger* (CP). We additionally define the security and transaction indistinguishability notions. To showcase the utility of these new primitives, we give a generic BL construction that relies on an unforgeable against chosen message attacks (EUF-CMA) digital signature for transaction authorization and show it is a secure BL. We then use an *eCash* based ledger as an example of such ledger and we further show it is a secure and privacy-preserving BL. Finally, we give a CP construction based on a EUF-CMA digital signature and hash-time lock contracts (HTLC) for transaction authorization and show that it is a secure CP instantiation.

(2) New Cryptographic Primitive for Fund/Defund. We present a new cryptographic primitive, namely, *CBDC-cash environment* (CCE), that captures the core functionality required for operations fund and defund. We require that CBDC-cash environment satisfies *balance security* for the three participants, i.e., either all three payments are executed at the respective ledgers as expected, or honest

participants can get their coins back. Moreover, we define the privacy notion of interest in terms of transaction indistinguishability, i.e., fund/defund cannot be used to breach CBDC ledger's privacy.
(3) Practically Efficient Constructions. We present two practical and efficient constructions for CCE: In the first construction, we assume that the CBDC ledger is an instance of CP, whereas in the second construction, we assume that the CBDC ledger is an instance of BL. The second construction relies on the *Payment versus Payment* (PvP) functionality available in existing retail and wholesale ledgers (e.g., TARGET2). We have formally defined this property, which could be a contribution of independent interest for future works. We have formally proven that both constructions are secure. Finally, we give a proof-of-concept implementation of the two constructions and our evaluation shows that the computation and communication overhead over the underlying ledgers is small. Thus, our proposal's scalability is only limited by the slowest ledger.

2 OUR LEDGER DEFINITION

Common Functionality. A ledger maintains a list of ordered transactions (TX_L) and it has a mechanism to validate transactions and prevent double spending. In general, a ledger accepts a new transaction if (1) the transaction is authorized by the owner of the transferred funds, (2) sender's account has sufficient funds, and (3) the same transaction is not already in the ledger. We refer to these three conditions as the ledger predicates `IsValid`, `IsFunded` and `IsUnique`, respectively. The mechanism to validate transactions substantiates the predicate `IsValid` and the mechanism to prevent double spending substantiates the predicates `IsFunded` and `IsUnique`. Additionally, a ledger has an internal mechanism to keep track of time (e.g., blockheight in Bitcoin) and users have access to a function `readTime`$(\cdot)$ that returns the ledger's time in a standard format (e.g., Unix time).

We next define the functionality and properties particular to both ledger types we consider in this work.

2.1 Basic Ledger (BL)

Basic Accounts and Transactions. In a ledger of type BL, coins are maintained in *simple accounts*. A simple account is represented by a public key pk and the corresponding private key sk is used to spend the coins. Coins are transferred between simple accounts using *transactions*. A *basic* transaction tx_{bsc} is a tuple (tx_{id}, pk_S, pk_R) that comprises a unique identifier $tx_{\text{bsc}}[tx_{id}]$, sender's account $tx_{\text{bsc}}[pk_S]$ and receiver's account $tx_{\text{bsc}}[pk_R]$. [1]
Notation. We use the following syntactic sugars: (1) isSender$(tx, pk) := (tx[pk_S] = pk)$ and (2) isRcvr$(tx, pk) := (tx[pk_R] = pk)$.

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

Definition 1 (Basic Ledger (BL)). *Basic ledger comprises the four PPT algorithms* (ctAcc, ctTx, subTx, ckTx) *defined below:*

- $(pk, sk) \leftarrow \mathsf{ctAcc}(1^\lambda)$: *the account creation algorithm takes as input the security parameter λ and outputs a key pair (pk, sk).*
- $(tx_{\text{bsc}}, \sigma_{\text{bsc}}) \leftarrow \mathsf{ctTx}(sk_S, pk_R)$: *the transaction creation algorithm takes as inputs sender's secret key sk_S and receiver's account pk_R. It outputs a transaction tx_{bsc} that transfers coins from pk_S to pk_R and its authorization σ_{bsc}.*
- $1/0 \leftarrow \mathsf{subTx}(tx_{\text{bsc}}, \sigma_{\text{bsc}})$: *the transaction submission algorithm takes as input a transaction-authorization pair $(tx_{\text{bsc}}, \sigma_{\text{bsc}})$. It outputs 1 (for accept) and adds the transaction to TX_L if the ledger predicates are fulfilled, or 0 (for reject) otherwise.*
- $1/0 \leftarrow \mathsf{ckTx}(tx_{\text{bsc}})$: *the check transaction inclusion algorithm takes as input a transaction tx_{bsc} and outputs 1 (for included) if the transaction is on the ledger or 0 (for not included) otherwise.*

Correctness. We use the aforementioned predicates `IsFunded`, `IsUnique` and `IsValid` to abstract away the details on how each ledger ensures these three functionalities.

Definition 2 (BL Correctness). *A BL 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 $(tx_{\text{bsc}}, \sigma_{\text{bsc}}) \leftarrow \mathsf{ctTx}(sk_S, pk_R)$, it holds that:*

$$\Pr\left[\ \mathsf{ckTx}(tx_{\text{bsc}}) = 1 \ \middle|\ \begin{array}{l}\mathtt{IsFunded}(tx_{\text{bsc}}, \sigma_{\text{bsc}})\\ \mathtt{IsUnique}(tx_{\text{bsc}})\\ \mathtt{IsValid}(tx_{\text{bsc}}, \sigma_{\text{bsc}})\\ \mathsf{subTx}(tx_{\text{bsc}}, \sigma_{\text{bsc}}) = 1\end{array}\right] = 1$$

Security. The notion of security for BL is *BL unforgeability* and ensures that the adversary cannot successfully authorize and submit transactions transferring coins from an honest user's account.

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* bscForge *is defined in Figure 4.*

Constructions. In Appendix C we present a construction for BL that relies on any existentially unforgeable digital signature scheme for transaction authorization. Such a generic construction aims to serve as blueprint to capture the CBDC ledgers existing in the literature that rely on different instances of digital signature scheme (e.g. KSIcash [36], Bakong [50], Hamilton [46], TIPS+ [65], Platypus [68] and the SNB proposal [21]). As an illustrative example, we also show how to construct a BL from eCash using the above blueprint.

2.2 Conditional Payment Ledger (CP)

CP is an extension of BL that allows a sender to perform a payment to a receiver, conditioned on the knowledge of the solution to some hard problem and a timeout based on ledger-dependent time.
Hard Relations. 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 $\mathcal{L}_\mathcal{R} := \{\mathbb{C} \in \mathcal{D}_S \mid \exists w \in \mathcal{D}_w$ s.t. $(\mathbb{C}, w) \in \mathcal{R}\}$. We say that $\mathcal{R}$ is a hard relation if the following holds: (1) There exists a PPT algorithm $\mathsf{createR}(1^\lambda)$ that computes $(\mathbb{C}, w) \in \mathcal{R}$; (2) the relation is decidable in polynomial time; and (3) 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.
Escrow Accounts. An *escrow account* models co-ownership of coins for a pre-determined amount of time T. This guarantees that a malicious sender does not create multiple conditional payments to different receivers using the same funds and is a standard procedure in both banking sector ledgers and blockchain-based cryptocurrencies (e.g., authorization holds in credit/debit cards [67] or 2-out-of-2 multisig addresses[53]). An escrow account is represented by

[1] In practice, a transaction also contains the amount of units that are going to be transferred and might require some ledger-dependent data. We omit explicit mention of these fields for a cleaner formulation.

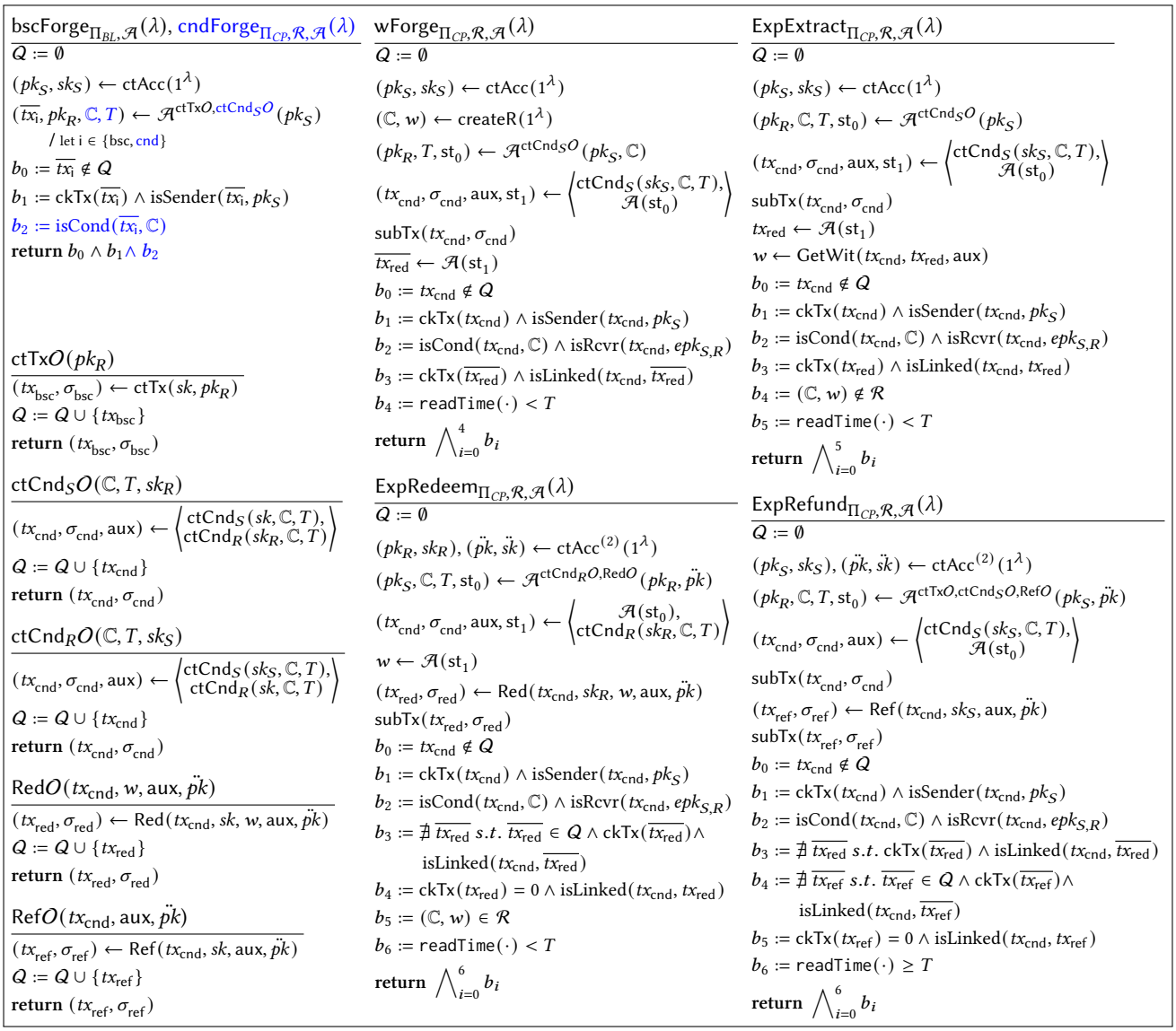

Figure 4: Experiments for BL unforgeability, CP unforgeability (additions are in blue), CP witness unforgeability, CP redeemability, CP extractability and CP refundability. In all experiments, adversary $\mathcal{A}$ interacts with the challenger over the ledger (Π_{BL} or Π_{CP}): All transactions forwarded to the challenger by $\mathcal{A}$ are successfully submitted to the ledger.

an escrow public key $epk_{S,R}$ that is computed by the algorithm ctEAcc(pk_S, pk_R). The inner workings of ctEAcc is implementation dependent. Coins in an escrow account can be either (1) redeemed by the receiver before timeout T expires; or (2) refunded back to the sender after timeout T expires.

CP Transactions. In addition to basic transactions tx_{bsc}, CP defines the following transactions: (1) *conditional payment transactions* $tx_{\mathsf{cnd}} := (tx_{id}, pk_S, epk_{S,R}, \mathbb{C}, T)$, which transfer coins from sender's account pk_S to an escrow account $epk_{S,R}$; (2) *redeem transactions* tx_{red}, which transfer coins from an escrow account $epk_{S,R}$ to a simple account $\ddot{pk}$, using sk_R and the solution w to condition $\mathbb{C}$ before timeout T expires; and (3) *refund transactions* tx_{ref}, which transfer coins from an escrow account $epk_{S,R}$ to a simple account $\ddot{pk}$, using sk_S after timeout T expires. We assume the existence of a function $\mathtt{createT}(\cdot)$ that, based on the internal ledger clock, outputs an appropriate timeout T for tx_{cnd}.

Notation. We use the following syntactic sugars (in addition to those in Section 2.1): (1) $\mathsf{isLinked}(tx_{\mathsf{cnd}}, tx_{\mathsf{red}}/tx_{\mathsf{ref}}) := (tx_{\mathsf{cnd}}[R] = tx_{\mathsf{red}}/tx_{\mathsf{ref}}[S])$ and (2) $\mathsf{isCond}(tx_{\mathsf{cnd}}, C) := (tx_{\mathsf{cnd}}[E] = (\cdot, C, \cdot))$.

Definition 4 (Conditional Payment Ledger (CP)). *A conditional payment ledger defined w.r.t. a BL and a hard relation $\mathcal{R}$, extends the BL with three PPT algorithms (*Red, Ref, GetWit*) and an interactive protocol* ctCnd *defined below:*

- $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux}) \leftarrow \left\langle \begin{matrix} \mathsf{ctCnd}_S(sk_S, \mathbb{C}, T), \\ \mathsf{ctCnd}_R(sk_R, \mathbb{C}, T) \end{matrix} \right\rangle$: *the conditional transaction creation protocol is executed by a sender with inputs sender's private key sk_S, payment condition $\mathbb{C}$ and payment timeout T; and a receiver with inputs receiver's private key sk_R, $\mathbb{C}$ and T. It outputs for both parties a conditional payment transaction tx_{cnd} that transfers coins from pk_S to an escrow account $epk_{S,R}$, its authorization σ_{cnd} and auxiliary information* aux.
- $(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}) \leftarrow \mathsf{Red}(tx_{\mathsf{cnd}}, sk_R, w, \mathsf{aux}, \ddot{pk})$: *the redeem algorithm takes as input a conditional payment transaction tx_{cnd}, receiver's secret key sk_R, a witness w, the auxiliary information* aux *and a receiving public key $\ddot{pk}$. It outputs a basic transaction tx_{red} (called redeem transaction) that transfers coins from the escrow account $epk_{S,R}$ to $\ddot{pk}$ and its authorization σ_{red}.*
- $(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}) \leftarrow \mathsf{Ref}(tx_{\mathsf{cnd}}, sk_S, \mathsf{aux}, \ddot{pk})$: *the refund algorithm takes as inputs a conditional payment transaction tx_{cnd}, sender's secret key sk_S, the auxiliary information* aux *and a receiving public key $\ddot{pk}$. It outputs a basic transaction tx_{ref} (called refund transaction) that transfers coins from the escrow account $epk_{S,R}$ to $\ddot{pk}$ and its authorization σ_{ref}.*
- $\{w, \perp\} \leftarrow \mathsf{GetWit}(tx_{\mathsf{cnd}}, tx_{\mathsf{red}}, \mathsf{aux})$: *the witness extraction algorithm takes as inputs a conditional transaction tx_{cnd} with payment condition $\mathbb{C}$, a redeem transaction tx_{red} s.t. $isLinked(tx_{\mathsf{cnd}}, tx_{\mathsf{red}}) = 1$ and the auxiliary information* aux*. It outputs a witness w s.t. $(\mathbb{C}, w) \in \mathcal{R}$ or $\perp$.*

Correctness. A CP is *correct* if: (1) a correctly generated conditional payment transaction tx_{cnd}, that satisfies the ledger predicates, is always accepted into the ledger; and then either (a) a redeem transaction tx_{red}, that satisfies the ledger predicates while generated by algorithm Red on input tx_{cnd} with payment condition $\mathbb{C}$ and the corresponding witness w, is always accepted into the ledger when $\mathtt{readTime}(\cdot) < T$; or (b) a refund transaction tx_{ref}, that is generated by algorithm Ref on input tx_{cnd} and satisfies the ledger predicates, is always accepted into the ledger when $\mathtt{readTime}(\cdot) \geq T$. We give the formal definition for CP correctness in Appendix A.

Security. In the following we consider the security goals of CP, which we formally describe in Figure 4.

Similarly to BL unforgeability, *CP unforgeability* ensures that the adversary cannot successfully authorize and submit conditional payment transactions transferring coins from an honest user's account. The adversary has access to a signing oracle for protocol ctCnd with key sk_S, while not being allowed to query the oracle on transaction $\overline{tx_{\mathsf{cnd}}}$ which outputs as forgery on the ledger.

Likewise, *CP witness unforgeability* ensures that given a conditional payment transaction (on the ledger), a malicious receiver (i.e., the adversary) that does not hold witness w cannot successfully authorize and submit redeem transactions transferring coins from the escrow account $epk_{S,R}$. The adversary has access to a signing oracle for protocol ctCnd with key sk_S, while not being allowed to query the oracle on transaction tx_{cnd}.

CP redeemability ensures that given a conditional payment transaction (on the ledger) with payment condition $\mathbb{C}$ and the corresponding witness w, an honest receiver can always successfully authorize and submit a redeem transaction transferring coins from the escrow account $epk_{S,R}$. Hence, a malicious sender (i.e., the adversary) cannot force the protocol ctCnd to output a conditional payment transaction tx_{cnd} that cannot be redeemed. The adversary has access to signing oracles for protocol ctCnd and algorithm Red with key sk_R, while not being allowed to query the oracles on transaction tx_{cnd} nor on transaction $\overline{tx_{\mathsf{red}}}$ (transferring coins from the escrow account $epk_{S,R}$) which later submits to the ledger.

CP extractability ensures that a pair of transactions $(tx_{\mathsf{cnd}}, tx_{\mathsf{red}})$ —s.t. both transactions are on the ledger, tx_{cnd} has payment condition $\mathbb{C}$ and tx_{red} transfers coins from the escrow account $epk_{S,R}$– can be used to extract a witness w for $\mathbb{C}$. Hence, a malicious receiver (i.e., the adversary) cannot redeem a conditional payment transaction without revealing a witness for $\mathbb{C}$. The adversary has access to a signing oracle for protocol ctCnd with key sk_S, while not being allowed to query the oracle on transaction tx_{cnd}.

CP refundability ensures that given a conditional payment transaction (on the ledger) that has not been redeemed, an honest sender can always successfully authorize and submit a refund transaction transferring coins from the escrow account $epk_{S,R}$. Hence, a malicious receiver (i.e., the adversary) cannot force the protocol ctCnd to output a conditional payment transaction tx_{cnd} that cannot be refunded. The adversary has access to signing oracles for protocol ctCnd and algorithms ctTx and Ref with key sk_S, while not being allowed to query the oracles on transaction tx_{cnd} nor on transaction $\overline{tx_{\mathsf{ref}}}$ (transferring coins from the escrow account $epk_{S,R}$) which later submits to the ledger. In addition, the adversary is not allowed to submit to the ledger a redeem transaction transferring coins from the escrow account $epk_{S,R}$.

In Appendix A, we present two additional unforgeability properties of CP for transactions redeem and refund, namely, CP redeem unforgeability and CP refund unforgeability. We show that CP redeemability and CP refundability imply CP redeem unforgeability and CP refund unforgeability, respectively. Moreover, we show that both CP redeem unforgeability and CP refund unforgeability imply BL unforgeability. Finally, we say that a CP is secure if all previous security properties hold. We formally define it in Definition 5.

Definition 5 (CP security). *Let $G :=$ {bscForge, cndForge, wForge, ExpRedeem, ExpExtract, ExpRefund} be the games defined in Figure 4. A CP 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* $\Pr[G_i(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Constructions. In Appendix D, we extend our generic construction for BL in Appendix C and provide a construction for CP that builds on hash-time lock contracts (HTLC). We also prove that it is a secure CP. This generic construction aims to serve as blueprint to capture CBDC ledgers that support conditional payments (e.g. itCoin [65], Iberpay's Smart Money [44]).

2.3 Privacy in BL

We adapt the notion of *transaction indistinguishability* as defined in [21, 68] to our definition of BL. More specifically, we (1) take into account transaction details which could leak information and (2) introduce the notion of *masked public accounts* (e.g., a blinded public key in *eCash* or a commitment in Platypus). We assume the existence of algorithm $\mathsf{maskAcc}(pk)$ which on input an account pk outputs the masked account $\widetilde{pk}$. In a nutshell, the adversary is provided with three masked accounts and is required to fund two of them. Thenceforth, one of the accounts is chosen (u.a.r.) and a

$\mathsf{bscIND}_{\Pi_{BL},\mathcal{A}}(\lambda)$

$\mathcal{Q} := \emptyset$

$(pk_j, sk_j) \leftarrow \mathsf{ctAcc}^{(3)}(1^\lambda)$ / for $j \in \{0,1,2\}$

$\widetilde{pk_j} \leftarrow \mathsf{maskAcc}^{(3)}(pk_j)$ / for $j \in \{0,1,2\}$

$(tx^0_{\mathsf{bsc}}, tx^1_{\mathsf{bsc}}, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{ctAcc}\mathcal{O},\mathsf{ctTx}\mathcal{O}}(\widetilde{pk_0}, \widetilde{pk_1}, \widetilde{pk_2})$

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

$(tx^*_{\mathsf{bsc}}, \sigma^*_{\mathsf{bsc}}) \leftarrow \mathsf{ctTx}(sk_c, \widetilde{pk_2})$

$\mathsf{subTx}(tx^*_{\mathsf{bsc}}, \sigma^*_{\mathsf{bsc}})$

$c^* \leftarrow \mathcal{A}(\mathsf{st}_0, tx^*_{\mathsf{bsc}})$

$b_0 := (c \stackrel{?}{=} c^*)$

$b_1 := \mathsf{ckTx}(tx^0_{\mathsf{bsc}}) \wedge \mathsf{isRcvr}(tx^0_{\mathsf{bsc}}, \widetilde{pk_0})$

$b_2 := \mathsf{ckTx}(tx^1_{\mathsf{bsc}}) \wedge \mathsf{isRcvr}(tx^1_{\mathsf{bsc}}, \widetilde{pk_1})$

$b_3 := \nexists\, \overline{tx_{\mathsf{bsc}}}^i \; s.t. \; \overline{tx_{\mathsf{bsc}}}^i \in \mathcal{Q} \wedge \mathsf{isSender}(\overline{tx_{\mathsf{bsc}}}^i, \widetilde{pk_i})$
/ let $i \in \{0,1\}$

return $\bigwedge_{i=0}^{3} b_i$

Figure 5: Experiment for BL transaction indistinguishability.

basic transaction transferring coins from the selected account to the third account is submitted to the ledger. Finally, the adversary is required to identify the funded account that was used as the sender in the transaction. The adversary can generate arbitrary accounts under its control and has access to oracle $\mathsf{ctAcc}\mathcal{O}$ that enables it to initialize additional honest users in the system. Moreover, it has access to a signing oracle that enables it to create arbitrary transactions for honest users' accounts, while not being allowed to query the oracle on transactions transferring coins from either of the two funded accounts.

Note that we model privacy only for BL. In Section 5, we discuss the challenges presented in defining the privacy notion for CP. Moreover, in Appendix C we prove that our construction of BL from *eCash* satisfies BL transaction indistinguishability.

Definition 6 (Transaction Indistinguishability). *A BL or CP ledger is said to offer BL transaction indistinguishability 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 $\Pr[\mathsf{bscIND}_{\Pi_{BL},\mathcal{A}}(\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$ where* bscIND *is defined in Figure 5.*

2.4 Payment Versus Payment (PvP) in CP

If a CP provides PvP functionality against a BL, this means that it is possible to set a conditional payment transaction tx_{cnd} in CP such that it can be redeemed if and only if a specific transaction tx_{bsc} is accepted in BL. We can characterize as hard relation (c.f. Section 2.2) the successful submission of a transaction to BL as follows: We define $\mathcal{D}_S$ as all possible transactions transferring coins from a given sender's account pk_S. $\mathcal{D}_w$ is defined as all transactions transferring coins from the same sender's account pk_S that are in the ledger TX_L. $\mathbb{C}^*$ is a specific transaction tx_{bsc} transferring coins from that sender's account pk_S to a given receiver's account pk_R, while w^* is that same transaction tx_{bsc} included in TX_L. This is a hard relation since: (1) there exists a PPT algorithm to create a $(\mathbb{C}^*, w^*)$ pair: $\mathbb{C}^*$ is created with ctTx and w^* by calling subTx, (2) the relation is decidable in polynomial time, since it only requires to verify that

$\mathsf{ExpPvP}_{\Pi_{CP},\Pi_{BL},\mathcal{A}}(\lambda)$

$\mathcal{Q} := \emptyset$

$(pk^{\mathsf{CP}}_S, sk^{\mathsf{CP}}_S) \leftarrow \mathsf{CP.ctAcc}(1^\lambda)$

$(pk^{\mathsf{BL}}_R, sk^{\mathsf{BL}}_R) \leftarrow \mathsf{BL.ctAcc}(1^\lambda)$

$(pk^{\mathsf{CP}}_R, pk^{\mathsf{BL}}_S, \mathbb{C}^* := tx^{\mathsf{BL}}_{\mathsf{bsc}}, T, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{ctCnd}_S\mathcal{O}}(pk^{\mathsf{CP}}_S, pk^{\mathsf{BL}}_R)$

$(tx^{\mathsf{CP}}_{\mathsf{cnd}}, \sigma^{\mathsf{CP}}_{\mathsf{cnd}}, \mathsf{aux}^{\mathsf{CP}}, \mathsf{st}_1) \leftarrow \left\langle \begin{matrix} \mathsf{ctCnd}_S(sk^{\mathsf{CP}}_C, \mathbb{C}^*, T), \\ \mathcal{A}(\mathsf{st}_0) \end{matrix} \right\rangle$

$\mathsf{subTx}(tx^{\mathsf{CP}}_{\mathsf{cnd}}, \sigma^{\mathsf{CP}}_{\mathsf{cnd}})$

$\overline{tx_{\mathsf{red}}}^{\mathsf{CP}} \leftarrow \mathcal{A}(\mathsf{st}_1)$

$b_0 := tx^{\mathsf{CP}}_{\mathsf{cnd}} \notin \mathcal{Q}$

$b_1 := \mathsf{ckTx}(tx^{\mathsf{CP}}_{\mathsf{cnd}}) \wedge \mathsf{isSender}(tx^{\mathsf{CP}}_{\mathsf{cnd}}, pk^{\mathsf{CP}}_S)$

$b_2 := \mathsf{isCond}(tx^{\mathsf{CP}}_{\mathsf{cnd}}, \mathbb{C}^*) \wedge \mathsf{isRcvr}(tx^{\mathsf{CP}}_{\mathsf{cnd}}, epk^{\mathsf{CP}}_{S,R})$

$b_3 := \mathsf{ckTx}(\overline{tx_{\mathsf{red}}}^{\mathsf{CP}}) \wedge \mathsf{isLinked}(tx^{\mathsf{CP}}_{\mathsf{cnd}}, \overline{tx_{\mathsf{red}}}^{\mathsf{CP}})$

$b_4 := \mathsf{isSender}(tx^{\mathsf{BL}}_{\mathsf{bsc}}, pk^{\mathsf{BL}}_S) \wedge \mathsf{isRcvr}(tx^{\mathsf{BL}}_{\mathsf{bsc}}, pk^{\mathsf{BL}}_R)$

$b_5 := \mathsf{ckTx}(tx^{\mathsf{BL}}_{\mathsf{bsc}}) = 0$

$b_6 := \mathtt{readTime}(\cdot) < T$

return $\bigwedge_{i=0}^{6} b_i$

Figure 6: Experiment for CP PvP. $\mathbb{C}^*$ denotes a transaction $tx^{\mathsf{BL}}_{\mathsf{bsc}}$ not submitted to ledger BL**.**

$\mathbb{C}^*$ and w^* are the same transaction and that $w^* \in TX_L$, and (3) for all PPT adversaries, producing a valid w^* knowing $\mathbb{C}^*$ is negligible, since this would break BL unforgeability.

The PVP notion implies CP witness unforgeability (see App A.3).

Definition 7 (CP PvP). *A CP offers CP PvP against a BL if for all $\lambda \in \mathbb{N}$, it holds that $\Pr[\mathsf{ExpPvP}_{\Pi_{CP},\Pi_{BL},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$, where* ExpPvP *is defined in Figure 6.*

3 CBDC-CASH ENVIRONMENT (CCE)

In this section, we present CBDC-cash environment (CCE), a cryptographic scheme that formalizes the core functionality for operations fund and defund.

CCE comprises three parties: (1) debtor D is the party handing over CBDC, (2) intermediary I is the party facilitating the exchange, and (3) creditor C is the party receiving CBDC. These three parties interact over the ledgers Π_{CP_0}, Π_{CP_1} and cbl. We assume w.l.o.g. the following: (1) the existence of a common reference time across ledgers such that function `readTime`$(\cdot)$ return the same value for all ledgers. (2) a parameter δ, which represents the time required by a user to propagate information between two ledgers; and (3) that all CP rely on the same hard relation $\mathcal{R}$. In Section 5 we discuss how to relax this last assumption.

Protocol outputs vary depending on whether ledger cbl is a CP or a BL, hence we use colors violet (for CP) and teal (for BL) to highlight the differences.

Definition 8 (CBDC-Cash Environment (CCE)). *CBDC-cash environment is a tuple of interactive protocols* $\Pi_{CCE} := (\mathsf{Set}, \mathsf{Pay})$ *executed by creditor C, intermediary I and debtor D, w.r.t. two conditional payment ledgers* $\{\Pi_{CP_0}, \Pi_{CP_1}\}$ *and a CBDC ledger* cbl.

- $\left\langle \begin{matrix} (tx^{\mathsf{cbl}}_{\mathsf{cnd}}/\perp, \mathsf{aux}_C), \\ (tx^{0}_{\mathsf{cnd}}, \mathsf{aux}_I), \\ (tx^{1}_{\mathsf{cnd}}, \mathsf{aux}_D) \end{matrix} \right\rangle \leftarrow \left\langle \begin{matrix} \mathsf{Set}_C(sk^0_C, sk^{\mathsf{cbl}}_C, T), \\ \mathsf{Set}_I(sk^1_I, sk^0_I), \\ \mathsf{Set}_D(sk^{\mathsf{cbl}}_D, sk^1_D, \mathbb{C}) \end{matrix} \right\rangle$: *Protocol* Set *is executed by (1) creditor C, with inputs creditor's private keys* sk^0_C, sk^{cbl}_C *and timeout* T*; (2) intermediary I, with inputs intermediary's private keys* sk^1_I, sk^0_I*; and (3) debtor D, with inputs debtor's private keys* sk^{cbl}_D, sk^1_D *and condition* $\mathbb{C}$*. It outputs: (1) on ledger* Π_{CP_0} *a conditional transaction* tx^0_{cnd} *transferring coins from* pk_C *to an escrow account* $epk_{C,I}$*; (2) on ledger* Π_{CP_1} *a conditional transaction* tx^1_{cnd} *transferring coins from* pk_I *to an escrow account* $epk_{I,D}$*; and (3) on ledger* cbl *(if* cbl *is a CP) a conditional transaction* $tx^{\mathsf{cbl}}_{\mathsf{cnd}}$ *transferring coins from* pk_D *to an escrow account* $epk_{D,C}$*, otherwise (if* cbl *is a BL) no transaction is submitted to ledger* cbl*. It also outputs auxiliary information* aux_C, aux_I *and* aux_D *for creditor C, intermediary I and debtor D, respectively.*
- $\left\langle \begin{matrix} (tx^{\mathsf{cbl}}_{\mathsf{red}}/tx^{\mathsf{cbl}}_{\mathsf{bsc}}), \\ (tx^{0}_{\mathsf{red}}), \\ (tx^{1}_{\mathsf{red}}) \end{matrix} \right\rangle \leftarrow \left\langle \begin{matrix} \mathsf{Pay}_C(sk^{\mathsf{cbl}}_C, tx^{\mathsf{cbl}}_{\mathsf{cnd}}, \mathsf{aux}_C, \ddot{pk}^{\mathsf{cbl}}_C), \\ \mathsf{Pay}_I(sk^0_I, tx^0_{\mathsf{cnd}}, \mathsf{aux}_I, \ddot{pk}^0_I), \\ \mathsf{Pay}_D(sk^1_D, tx^1_{\mathsf{cnd}}, w, \mathsf{aux}_D, \ddot{pk}^1_D) \end{matrix} \right\rangle$: *Protocol* Pay *is executed by (1) creditor C, with inputs creditor's private key* sk^{cbl}_C*, conditional transaction* $tx^{\mathsf{cbl}}_{\mathsf{cnd}}$ *(if* cbl *is a CP) and auxiliary information* aux_C*; (2) intermediary I, with inputs intermediary's private key* sk^0_I*, conditional transaction* tx^0_{cnd} *and auxiliary information* aux_I*; and (3) debtor D, with inputs debtor's private key* sk^1_D*, conditional transaction* tx^1_{cnd}*, witness* w *and auxiliary information* aux_D*. It outputs: (1) on ledger* Π_{CP_0} *a redeem transaction* tx^0_{red} *transferring coins from the escrow account* $epk_{C,I}$ *to a receiving account* $\ddot{pk}^0_I$*; (2) on ledger* Π_{CP_1} *a redeem transaction* tx^1_{red} *transferring coins from the escrow account* $epk_{I,D}$ *to a receiving account* $\ddot{pk}^1_D$*; and (3) on ledger* cbl *(if* cbl *is a CP) a redeem transaction* $tx^{\mathsf{cbl}}_{\mathsf{red}}$ *transferring coins from the escrow account* $epk_{D,C}$ *to a receiving account* $\ddot{pk}^{\mathsf{cbl}}_C$*, otherwise (if* cbl *is a BL) a basic transaction* $tx^{\mathsf{cbl}}_{\mathsf{bsc}}$ *transferring coins from* pk_D *to a receiving account* $\ddot{pk}^{\mathsf{cbl}}_C$.

Correctness. CCE is *correct* if: (1) after invoking protocol Set, ledgers Π_{CP_0}, Π_{CP_1}, cbl include conditional payment transactions tx^0_{cnd}, tx^1_{cnd}, $tx^{\mathsf{cbl}}_{\mathsf{cnd}}$, respectively, with the same payment condition $\mathbb{C}$; and given the conditional payment transactions obtained in Set, either (a) after invoking Pay (before timeout T expires) all three ledgers Π_{CP_0}, Π_{CP_1} and cbl include transaction tx^0_{red}, tx^1_{red} $tx^{\mathsf{cbl}}_{\mathsf{red}}/tx^{\mathsf{cbl}}_{\mathsf{bsc}}$, respectively; or (b) after invoking algorithm Ref (after timeout $T + 2\delta$ expires) ledgers Π_{CP_0}, Π_{CP_1}, cbl include refund transactions tx^0_{ref}, tx^1_{ref}, $tx^{\mathsf{cbl}}_{\mathsf{ref}}$.

We formally define CCE correctness in Appendix B.

Security. Our security notion is *CCE balance security*. Intuitively, this property aims to ensure that an honest party does not lose coins, even when the other two parties collude.

Balance security of creditor ensures that given a conditional payment transaction tx^0_{cnd} (on ledger Π_{CP_0}) transferring coins from creditor C to intermediary I with payment condition $\mathbb{C}$ and the corresponding witness w, the adversary cannot successfully authorize and submit a redeem transaction tx^0_{red} without outputting a transaction $tx^{\mathsf{cbl}}_{\mathsf{red}}/tx^{\mathsf{cbl}}_{\mathsf{bsc}}$ (on ledger cbl) transferring coins from debtor D to creditor C. The adversary has access to oracles $\mathsf{Set}_C O$, $\mathsf{Pay}_C O$ and $\mathsf{Ref}_C O$ with keys $(sk^0_C, sk^{\mathsf{cbl}}_C)$, sk^{cbl}_C and sk^0_C, respectively, that provide transcript of honest executions of each protocol.

$BSC_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda)$

$Q := \emptyset$;

$(pk^0_C, sk^0_C), (\ddot{pk}^0_C, \ddot{sk}^0_C) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}^{(2)}(1^\lambda)$

$(pk^{\mathsf{cbl}}_C, sk^{\mathsf{cbl}}_C), (\ddot{pk}^{\mathsf{cbl}}_C, \ddot{sk}^{\mathsf{cbl}}_C) \leftarrow \mathsf{cbl}.\mathsf{ctAcc}^{(2)}(1^\lambda)$

$T \leftarrow \mathtt{createT}(\cdot)$

$(pk^0_I, pk^{\mathsf{cbl}}_D, \mathbb{C}, \mathsf{st}_0)$

$\quad \leftarrow \mathcal{A}^{\mathsf{Set}_C O, \mathsf{Pay}_C O, \mathsf{Ref}_C O}(pk^0_C, pk^{\mathsf{cbl}}_C, \ddot{pk}^0_C, \ddot{pk}^{\mathsf{cbl}}_C, T)$

$\left\langle \begin{matrix} (tx^{\mathsf{cbl}}_{\mathsf{cnd}}/\perp, \mathsf{aux}_C), \\ (tx^0_{\mathsf{cnd}}, \mathsf{aux}_I, \mathsf{st}_1) \end{matrix} \right\rangle \leftarrow \left\langle \begin{matrix} \mathsf{Set}_C(sk^0_C, sk^{\mathsf{cbl}}_C, T), \\ \mathcal{A}(\mathsf{st}_0) \end{matrix} \right\rangle$

$\tau := \mathtt{readTime}(\cdot)$

if $\tau < T$

$\quad \left\langle \begin{matrix} (tx^{\mathsf{cbl}}_{\mathsf{red}}/tx^{\mathsf{cbl}}_{\mathsf{bsc}}), \\ (tx^0_{\mathsf{red}}) \end{matrix} \right\rangle \leftarrow \left\langle \begin{matrix} \mathsf{Pay}_C(sk^{\mathsf{cbl}}_C, tx^{\mathsf{cbl}}_{\mathsf{cnd}}, \mathsf{aux}_C, \ddot{pk}^{\mathsf{cbl}}_C), \\ \mathcal{A}(\mathsf{st}_1) \end{matrix} \right\rangle$

if $\tau \geq T + \delta$

$\quad (tx^0_{\mathsf{ref}}, \sigma^0_{\mathsf{ref}}) \leftarrow \Pi_{CP_0}.\mathsf{Ref}(tx^0_{\mathsf{cnd}}, sk^0_C, \mathsf{aux}, \ddot{pk}^0_C)$

$\quad \Pi_{CP_0}.\mathsf{subTx}(tx^0_{\mathsf{ref}}, \sigma^0_{\mathsf{ref}})$

$a_0 := tx^0_{\mathsf{cnd}}, tx^{\mathsf{cbl}}_{\mathsf{cnd}}, tx^{\mathsf{cbl}}_{\mathsf{bsc}} \notin Q$

$a_1 := \Pi_{CP_0}.\mathsf{ckTx}(tx^0_{\mathsf{cnd}}) \wedge \mathsf{isSender}(tx^0_{\mathsf{cnd}}, pk^0_C)$

$a_2 := \mathsf{isCond}(tx^0_{\mathsf{cnd}}, \mathbb{C}) \wedge \mathsf{isRcvr}(tx^0_{\mathsf{cnd}}, epk^0_{C,I})$

$a_3 := \mathsf{cbl}.\mathsf{ckTx}(tx^{\mathsf{cbl}}_{\mathsf{cnd}}) \wedge \mathsf{isSender}(tx^{\mathsf{cbl}}_{\mathsf{cnd}}, pk^{\mathsf{cbl}}_D)$

$a_4 := \mathsf{isCond}(tx^{\mathsf{cbl}}_{\mathsf{cnd}}, \mathbb{C}) \wedge \mathsf{isRcvr}(tx^{\mathsf{cbl}}_{\mathsf{cnd}}, epk^{\mathsf{cbl}}_{D,C})$

$a_5 := \mathsf{isSender}(tx^{\mathsf{cbl}}_{\mathsf{bsc}}, pk^{\mathsf{cbl}}_D) \wedge \mathsf{isRcvr}(tx^{\mathsf{cbl}}_{\mathsf{bsc}}, pk^{\mathsf{cbl}}_C)$

$b_0 := \Pi_{CP_0}.\mathsf{ckTx}(tx^0_{\mathsf{red}}) \wedge \mathsf{isLinked}(tx^0_{\mathsf{cnd}}, tx^0_{\mathsf{red}})$

$b_1 := \nexists\, \overline{tx_{\mathsf{red}}}^{\mathsf{cbl}}\ s.t.\ \overline{tx_{\mathsf{red}}}^{\mathsf{cbl}} \in Q \wedge \mathsf{ckTx}(\overline{tx_{\mathsf{red}}}^{\mathsf{cbl}}) \wedge$

$\quad \mathsf{isLinked}(tx^{\mathsf{cbl}}_{\mathsf{cnd}}, \overline{tx_{\mathsf{red}}}^{\mathsf{cbl}})$

$b_2 := \mathsf{cbl}.\mathsf{ckTx}(tx^{\mathsf{cbl}}_{\mathsf{red}}) = 0 \wedge \mathsf{isLinked}(tx^{\mathsf{cbl}}_{\mathsf{cnd}}, tx^{\mathsf{cbl}}_{\mathsf{red}})$

$b_3 := \mathsf{cbl}.\mathsf{ckTx}(tx^{\mathsf{cbl}}_{\mathsf{bsc}}) = 0$

$b_4 := \mathtt{readTime}(\cdot) < T$

$c_0 := \nexists\, \overline{tx_{\mathsf{red}}}^{0}\ s.t.\ \mathsf{ckTx}(\overline{tx_{\mathsf{red}}}^{0}) \wedge \mathsf{isLinked}(tx^0_{\mathsf{cnd}}, \overline{tx_{\mathsf{red}}}^{0})$

$c_1 := \nexists\, \overline{tx_{\mathsf{ref}}}^{0}\ s.t.\ \overline{tx_{\mathsf{ref}}}^{0} \in Q \wedge \mathsf{ckTx}(\overline{tx_{\mathsf{ref}}}^{0}) \wedge$

$\quad \mathsf{isLinked}(tx^0_{\mathsf{cnd}}, \overline{tx_{\mathsf{ref}}}^{0})$

$c_2 := \Pi_{CP_0}.\mathsf{ckTx}(tx^0_{\mathsf{ref}}) = 0 \wedge \mathsf{isLinked}(tx^0_{\mathsf{cnd}}, tx^0_{\mathsf{ref}})$

$c_3 := \mathtt{readTime}(\cdot) \geq T + \delta$

return $\bigwedge_{i=0}^{5} a_i \wedge (\bigwedge_{i=0}^{4} b_i \vee \bigwedge_{i=0}^{3} c_i)$

Figure 7: Experiment for balance security of creditor C. Winning conditions and algorithm inputs/outputs that apply only when cbl **is a CP or a BL are denoted with colors violet and teal. Winning conditions** a_i, b_i **and** c_i **correspond to the outcomes of protocols** Set, Pay **and algorithm** Ref.

Balance security of intermediary ensures that given a conditional payment transaction tx^1_{cnd} (on ledger Π_{CP_1}) transferring coins from intermediary I to debtor D with payment condition $\mathbb{C}$ and the corresponding witness w, the adversary cannot successfully authorize and submit a redeem transaction tx^1_{red} without outputting a transaction tx^0_{red} (on ledger Π_{CP_0}) transferring coins from creditor C to intermediary I. The adversary has access to oracles $\mathsf{Set}_I\mathcal{O}$, $\mathsf{Pay}_I\mathcal{O}$ and $\mathsf{Ref}_I\mathcal{O}$ with keys (sk^1_I, sk^0_I), sk^0_I and sk^1_I, respectively, that provide transcripts of honest executions of each protocol.

Balance security of debtor ensures that given a conditional payment transaction tx^1_{cnd} (on ledger Π_{CP_1}) transferring coins from intermediary I to debtor D with payment condition $\mathbb{C}$ and the corresponding witness w, D can always successfully authorize and submit a redeem transaction tx^1_{red} by submitting a transaction $tx^{\mathsf{cbl}}_{\mathsf{red}}/tx^{\mathsf{cbl}}_{\mathsf{bsc}}$ (on ledger cbl) transferring coins from D to creditor C. Hence, the adversary cannot force the protocol Set to output a conditional payment transaction tx^1_{cnd} that cannot be redeemed invoking protocol Pay. The adversary has access to oracles $\mathsf{Set}_D\mathcal{O}$, $\mathsf{Pay}_D\mathcal{O}$ and $\mathsf{Ref}_D\mathcal{O}$ with keys $(sk^{\mathsf{cbl}}_D, sk^1_D)$, sk^1_D and sk^{cbl}_D, respectively, that provide transcripts of honest executions of each protocol.

The corollary of CCE balance security is atomicity, i.e., given a set of conditional payment transactions outputted by protocol Set, either all transactions are redeemed or none. Due to space limitation, we present here the game of creditor balance security (c.f. Figure 7) and we refer the reader to Appendix B for the games of intermediary and debtor balance security (c.f. Figure 11).

Definition 9 (CCE Balance Security)**.** *Let $G := \{BSC, BSI, BSD\}$ be the games defined in Figure 7 and Figure 11. A CCE 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* $\Pr[G_i(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Privacy. Finally, we extend the notion of *transaction indistinguishability* (c.f. Definition 6) for ledger cbl such that the adversary can now leverage CCE in order to fund masked accounts. Furthermore, it has access to an additional oracle that provides transcripts of executions of protocols Set and the corresponding Pay. Naturally, the adversary is not allowed to query the oracles on any transaction transferring coins from either of the two accounts used by the challenger in the challenge phase of the game. In Appendix B, we formally define CCE transaction indistinguishability and show that it implies BL transaction indistinguishability, when ledger cbl is a BL. This shows that a construction achieving our notion of transaction indistinguishability for CCE does not break the privacy required by the underlying cbl ledger.

4 OUR CONSTRUCTIONS

We describe here a construction of CBDC-cash environment for the case that cbl is a BL and show the case cbl is a CP in Appendix F. We also provide proof of concept implementations for the two cases and evaluate the execution time and communication overhead.

Common System Assumptions. Parties interact with ledgers over authenticated and encrypted channels. Creditor C and debtor D communicate with intermediary I in the same manner, with I relaying messages between C and D.

4.1 BL-CBDC

Here we present a concrete instantiation of CBDC-cash environment w.r.t. two conditional payment ledgers Π_{CP_0}, Π_{CP_1} that provide CP PvP functionality against ledger cbl (which is a BL). As described in Section 2.4, this means that it is possible to set conditional payment transactions tx^0_{cnd}, tx^1_{cnd} on ledgers Π_{CP_0} and Π_{CP_1}, respectively, such that they can be redeemed if and only if a specific transaction $tx^{\mathsf{cbl}}_{\mathsf{bsc}}$ is accepted in ledger cbl. We denote transaction $tx^{\mathsf{cbl}}_{\mathsf{bsc}}$ as $\mathbb{C}^*$ when it is not on ledger cbl and as w^*, otherwise.

Overview. We illustrate protocols Set and Pay in Figure 8. Protocol Set is executed by (1) creditor C, with inputs private keys sk^0_C, sk^{cbl}_C and timeout T; (2) intermediary I, with inputs private keys sk^1_I; and (3) debtor D, with inputs private keys sk^{cbl}_D, sk^1_D and a transaction $\mathbb{C}^*$ transferring coins from debtor D to creditor C that is not yet submitted to cbl. The Set protocol works as follows: (1) C and D forward to the other parties timeout T and transaction $\mathbb{C}^*$, respectively. (2) If transaction $\mathbb{C}^*$ is well-formed, C engages with I in protocol $\Pi_{CP_0}.\mathsf{ctCnd}$ which results in a conditional payment transaction tx^0_{cnd} (on ledger Π_{CP_0}) transferring coins from C to I with payment condition $\mathbb{C}^*$ and timeout $T' := T + \delta$. (3) If transaction tx^0_{cnd} is well-formed, I engages with D in protocol $\Pi_{CP_1}.\mathsf{ctCnd}$ which results in a conditional payment transaction tx^1_{cnd} (on ledger Π_{CP_1}) transferring coins from I to D with payment condition $\mathbb{C}^*$ and timeout $T' := T + \delta$. (4) Finally, D checks that tx^1_{cnd} is well-formed and the protocol terminates outputting auxiliary information aux_C, aux_I and aux_D for C, I and D, respectively.

Note that timeouts at each conditional payment transaction have been staggered by the factor δ so that each participant has enough time to redeem the payment at each ledger after D triggers the Pay protocol. In a bit more detail, the protocol Pay is executed by (1) creditor C, with inputs private key sk^{cbl}_C and auxiliary information aux_C; (2) intermediary I, with inputs private key sk^0_I, conditional transaction tx^0_{cnd} and auxiliary information aux_I; and (3) debtor D, with inputs private key sk^1_D, conditional transaction tx^1_{cnd}, witness w^* (i.e., transaction $tx^{\mathsf{cbl}}_{\mathsf{bsc}}$ is on ledger cbl), and auxiliary information aux_D, The protocol Pay works as follows: (1) All parties parse their respective auxiliary information. (2) Thereafter, I and D verify that transaction $tx^{\mathsf{cbl}}_{\mathsf{bsc}}$ is on ledger cbl, and subsequently they redeem transactions tx^0_{cnd} and tx^1_{cnd} invoking algorithms $\Pi_{CP_0}.\mathsf{Red}$ and $\Pi_{CP_1}.\mathsf{Red}$, respectively.

If the transaction $tx^{\mathsf{cbl}}_{\mathsf{bsc}}$ is not successfully submitted to ledger cbl, the creditor C and the intermediary I can recover their locked coins invoking algorithms $\Pi_{CP_0}.\mathsf{Red}$ and $\Pi_{CP_1}.\mathsf{Red}$, respectively.

Security Analysis. We formally show that our construction satisfies CCE correctness in Appendix E.1. Furthermore, we state below in Theorem 1 our formal security claim and defer the complete proof to Appendix E.2.

Theorem 1 (BL-CBDC Balance Security). *Let ledgers Π_{CP_0} and Π_{CP_1} provide CP PvP against ledger* cbl *and satisfy CP unforgeability, CP redeemability, CP refundability. Let ledger* cbl *satisfy BL unforgeability. Then, our construction for CCE depicted in Figure 8 is secure according to Definition 9.*

$\mathbf{C}(sk_C^0, sk_C^{\mathsf{cbl}}, T)$	$\mathbf{I}(sk_I^1, sk_I^0)$	$\mathbf{D}(\mathbb{C}^*, sk_D^{\mathsf{cbl}}, sk_D^1)$
$\mathsf{Send}(T)$	$\mathsf{Route}(T)$	$\mathsf{Receive}(T)$
$\mathsf{Receive}(\mathbb{C}^*)$	$\mathsf{Route}(\mathbb{C}^*)$	$\mathsf{Send}(\mathbb{C}^*)$
$T' := T + \delta$	$T' := T + \delta$	
$b_0 := \mathsf{isSender}(\mathbb{C}^*, pk_D^{\mathsf{cbl}}) \wedge$ $\mathsf{isRcvr}(\mathbb{C}^*, pk_C^{\mathsf{cbl}})$		
if $\neg b_0$ **abort**		**if** $\neg b_0$ **abort**
$(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0)$ $\leftarrow \Pi_{CP_0}.\mathsf{ctCnd}_S(sk_C^0, \mathbb{C}^*, T')$	$(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0)$ $\leftarrow \Pi_{CP_0}.\mathsf{ctCnd}_R(sk_I^0, \mathbb{C}^*, T')$	
$\Pi_{CP_0}.\mathsf{subTx}(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0)$		
	$b_1 := \Pi_{CP_0}.\mathsf{ckTx}(tx_{\mathsf{cnd}}^0) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^0, pk_C^0)$	
	$b_2 := \mathsf{isCond}(tx_{\mathsf{cnd}}^0, \mathbb{C}^*) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^0, epk_{C,I}^0)$	
	if $\neg(b_1 \wedge b_2)$ **abort**	
	$(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1)$ $\leftarrow \Pi_{CP_1}.\mathsf{ctCnd}_S(sk_I^1, \mathbb{C}^*, T)$	$(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1)$ $\leftarrow \Pi_{CP_1}.\mathsf{ctCnd}_R(sk_D^1, \mathbb{C}^*, T)$
	$\Pi_{CP_1}.\mathsf{subTx}(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1)$	
		$b_3 := \Pi_{CP_1}.\mathsf{ckTx}(tx_{\mathsf{cnd}}^1) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^1, pk_I^1)$
		$b_4 := \mathsf{isCond}(tx_{\mathsf{cnd}}^1, \mathbb{C}^*) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^1, epk_{I,D}^1)$
		if $\neg(b_3 \wedge b_4)$ **abort**
$\mathsf{aux}_C := (\mathsf{aux}^0, tx_{\mathsf{cnd}}^0, \mathbb{C}^*)$	$\mathsf{aux}_I := (\mathsf{aux}^0, \mathsf{aux}^1, tx_{\mathsf{cnd}}^1, \mathbb{C}^*)$	$\mathsf{aux}_D := (\mathsf{aux}^1, \mathbb{C}^*)$
return (aux_C)	**return** $(tx_{\mathsf{cnd}}^0, \mathsf{aux}_I)$	**return** $(tx_{\mathsf{cnd}}^1, \mathsf{aux}_D)$

$\mathbf{C}(sk_C^{\mathsf{cbl}}, \mathsf{aux}_C, \perp)$	$\mathbf{I}(sk_I^0, tx_{\mathsf{cnd}}^0, \mathsf{aux}_I, \ddot{pk}_I^0)$	$\mathbf{D}(sk_D^1, tx_{\mathsf{cnd}}^1, w^*, \mathsf{aux}_D, \ddot{pk}_D^1)$
$(\mathsf{aux}^0, tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathbb{C}^*) \leftarrow \mathsf{aux}_C$	$(\mathsf{aux}^0, \mathsf{aux}^1, tx_{\mathsf{cnd}}^1, \mathbb{C}^*) \leftarrow \mathsf{aux}_I$	$(\mathsf{aux}^1, \mathbb{C}^*) \leftarrow \mathsf{aux}_D$
if $\mathsf{cbl.ckTx}(w^* := tx_{\mathsf{bsc}}^{\mathsf{cbl}}) = 0$ **abort**	**if** $\mathsf{cbl.ckTx}(w^* := tx_{\mathsf{bsc}}^{\mathsf{cbl}}) = 0$ **abort**	**if** $\mathsf{cbl.ckTx}(w^* := tx_{\mathsf{bsc}}^{\mathsf{cbl}}) = 0$ **abort**
	$(tx_{\mathsf{red}}^0, \sigma_{\mathsf{red}}^0) \leftarrow \Pi_{CP_0}.\mathsf{Red}(tx_{\mathsf{cnd}}^0, sk_I^0, w^*, \mathsf{aux}^0, \ddot{pk}_I^0)$	$(tx_{\mathsf{red}}^1, \sigma_{\mathsf{red}}^1) \leftarrow \Pi_{CP_1}.\mathsf{Red}(tx_{\mathsf{cnd}}^1, sk_D^1, w^*, \mathsf{aux}^1, \ddot{pk}_I^1)$
	$\Pi_{CP_0}.\mathsf{subTx}(tx_{\mathsf{red}}^0, \sigma_{\mathsf{red}}^0)$	$\Pi_{CP_1}.\mathsf{subTx}(tx_{\mathsf{red}}^1, \sigma_{\mathsf{red}}^1)$
	$b_0 := \Pi_{CP_0}.\mathsf{ckTx}(tx_{\mathsf{red}}^0) \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^0, tx_{\mathsf{red}}^0)$	$b_1 := \Pi_{CP_1}.\mathsf{ckTx}(tx_{\mathsf{red}}^1) \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^1, tx_{\mathsf{red}}^1)$
	if $\neg b_0$ **abort**	**if** $\neg b_1$ **abort**
return $(tx_{\mathsf{bsc}}^{\mathsf{cbl}})$	**return** (tx_{red}^0)	**return** (tx_{red}^1)

Figure 8: Our construction for Set **(top) and** Pay **(bottom) for BL-CBDC. Transaction $tx_{\mathsf{bsc}}^{\mathsf{cbl}}$, linked to CP PvP, is denoted as $\mathbb{C}^*$ when it is not on ledger** cbl **and as w^*, otherwise.**

4.2 Performance Analysis

Code. We implemented a proof of concept [2] of the constructions described in Section 4.1 and Appendix F. Our implementation is in python and relies on the library *cryptography* [5] for key generation, signature generation and verification; and on *flask* [54] and *sqlite* [43] for the servers that run the BL and CP ledgers. In our analysis, we make a distinction between (1) *account operations*: the operations that are executed by the users (creditor, intermediary and debtor) on their accounts (i.e., ctAcc, ctTx, ctCnd, Red and Ref); and (2) *ledger operations*: the operations executed on the ledger server itself (i.e., subTx and ckTx). For experimentation purposes, we added function fundAcc to pre-fund user accounts.

Testbed. We run four servers, one of type BL and three of type CP. Two servers of the latter type are used to simulate ret and whs ledgers, while the remaining BL and CP servers are used to simulate the two possible versions of cbl. We run our experiments using VirtualBox 7.0.4 to run a virtual machine with 3 processors and 8192Mb of memory with the O.S. Ubuntu 20.04.5 LTS. The host machine is a *Intel(R) Core(TM) i7-8565 CPU @1.80GHz 1.99GHz, with 16GB of RAM.*

Evaluation. We first evaluate the total execution time for both Set and Pay protocols for both variants CP-CBDC and BL-CBDC. For each variant, we execute one after the other the Set and Pay protocols and record the total execution time since Set starts until Pay ends. We repeat this experiment one thousand times to extract the mean and standard deviation times as result. In particular, we observe that the CP-CBDC variant takes 310 ± 50ms whereas BL-CBDC variant takes 240 ± 30ms. We note that the latter execution

time could be further reduced since the Pay protocol allows for the concurrent execution of Red in ret and whs. Concurrent execution is possible since C, I and D could learn w (i.e., the fact that $tx^{\mathsf{cbl}}_{\mathsf{bsc}}$ is published in cbl) as soon as the Pay protocol is initiated (see Figure 8).

Second, we dissect the running time required for the account operations only to account for the overhead required by the CBDC users (e.g., the application executed at the citizens phone to handle her CBDC wallet). The CP-CBDC variant takes 7 ± 1ms whereas the BL-CBDC variant takes 1.8 ± 0.2ms. This shows that the majority of the execution time is taken by the ledger server operations, which are dependent on the concrete implementation of each ledger.

Finally, we evaluate the communication complexity. We observe that in our implementation the weight of a transaction in BL is 168 bytes and in CP is 272 bytes. The weight of the T is 8 bytes and $\mathbb{C}$ weights 32 bytes in CP-CBDC and 8 bytes in BL-CBDC. We have calculated the communication complexity of each variant based on the number of times that ckTx and subTx are called. We top this value with the initial messages that share T and $\mathbb{C}$ in the beginning of Set . The CP-CBDC requires 11 transactions in CP, amounting for a total of 3032 bytes. The BL-CBDC requires 3 transactions in BL and 6 in CP, totalling 2152bytes.

In summary, we make the following observations. First, the BL-CBDC variant is more efficient in both computation and communication complexity. Nevertheless, both variants are practical when executed in commodity hardware. Second, we note that the computation and communication overhead that our protocols impose to the ledger servers is small: up to 310ms for both variants and up to 40 bytes to agree on the timeouts and conditions. Furthermore, the computation overhead imposed to user devices is small, up to 7ms for both variants. We thereby infer that the bottleneck of the scalability for CCE is the slowest ledger. Fortunately, the whs and ret ledgers process very large number of transactions daily [26, 28] and available proposals for cbl do set scalability as a goal (e.g. [46, 68]).

5 DISCUSSION

Privacy and Accountability. Despite CBDC ledgers providing transaction indistinguishability, authorities can prevent the financing of terrorism (FT) or money laundering (ML) reusing existing infrastructure for FT and ML prevention. The ret ledgers are operated by institutions that are mandated to contribute with the authorities to prevent ML and combating FT [39]. As such, funding or defunding a CBDC wallet could be subject to inspection, just like it happens with cash. A commercial bank might demand information to complete large or unusual CBDC funds or defunds.

The information required by the commercial bank could be provided, for example, with documents detailing how CBDC is used (e.g. purchase contract of a good), just like it happens with cash. If the CBDC ledger supports transaction indistinguishability, citizens could leverage zero-knowledge proof systems (e.g., as in [64, 68]) to prove the required statements regarding their balances and transaction amounts and demonstrate the legitimate origin/use of CBDC. We thereby consider accountability an orthogonal problem.

Emulating a CP. Our construction in Figure 8 for CCE where the CBDC ledger is of BL type fundamentally relies on the CP PvP functionality. If such functionality is not available, one could think of a cryptographic protocol built on top of a BL to support a functionality similar to CP. In particular, one could leverage multiparty signatures to allow that a simple account is controlled by more than one key. Moreover, when locking funds in such a shared account, both parties can use adaptor signatures [4] to tie the authorization of a redeem transaction to a condition $\mathbb{C}$ and verifiable timed signatures [61] to postpone the release of the authorization for the refund transaction. This blueprint to emulate the CP functionality on top of a BL ledger has been already explored by Thyagarajan et al [63]. The thereby emulated CP, allows for a construction for CCE in the same manner as for any other CP-CBDC.

Schemes similar to CP. The knowledgeable reader might have observed that CP bears strong resemblance to adaptor signatures [4]. While adaptor signatures allow for conditioning the creation of a digital signature for a given message on the knowledge of a cryptographic secret, our CP scheme additionally captures the requirements for the message itself to be included to the ledger, such as the transaction format and the appropriate use of escrow accounts. In fact, one could consider adaptor signatures as a building block for a CP construction, as discussed in the previous point about how to emulate a CP. The definition of CP also resembles the functionality provided by HTLC contracts [53] or the banking sector, e.g. authorization holds of credit cards [67]. With our modularity goal in mind, we have defined CP such that encompasses the core functionality of a ledger that can perform conditional payments, without having to commit to a specific use-case. In fact, we show in Appendix D how to construct a CP using HTLC as building block and prove that it is a secure construction for CP.

Different $\mathbb{C}$ in CCE. Our construction for CCE rely on the fact that the three involved ledgers support the same hard relation. If this is not the case, debtor D who knows the pair $(\mathbb{C}, w)$ could create a pair $(\mathbb{C}', w)$ for a second hard relation and use a zero-knowledge proof to convince other participants that both $\mathbb{C}$ and $\mathbb{C}'$ have the same witness. For instance, Chase et al. [18] propose a zero-knowledge proof of discrete logarithm equality across groups. This would allow to use a different group, where the discrete logarithm problem is hard, at each of the ledgers involved in operations fund and defund.

Privacy in CP. We grounded our privacy definition for BL in existing privacy definitions of CBDC proposals. A natural extension to the transaction indistinguishability property of BL is to consider the extra functionality for CP. For example, one could add oracle access to the algorithms Red and Ref in the bscIND game. However, this is not sufficient, since the game would only guarantee privacy of basic transactions. As such, one could further modify bscIND to cover all four transaction types. However, this would enable an adversary to distinguish transactions of different types, therefore breaking the transaction indistinguishability notion. In summary, we find the definition of privacy for CP an interesting future work.

6 CONCLUSIONS

In this work, we investigate how to fund and defund CBDC wallets as the process of distributing CBDC units to citizens from central banks using commercial banks as intermediaries. In particular, we present two primitives that capture the functionality of, coarsely speaking, two types of ledgers existing today: BL and CP. Using

them as building blocks, we introduce a new cryptographic primitive, the CBDC-cash environment (CCE), which captures the core functionality of fund and defund. We give two constructions for CCE supporting CBDC ledgers of type BL and CP, respectively. The BL construction relies on a property of ledgers of the banking sector called *payment versus payment*, which we formally define. Finally, we give a proof-of-concept implementation of the two constructions proposed in this work and our evaluation shows that the computation and communication overhead over the underlying ledgers is small, even with commodity hardware.

Acknowledgements. We acknowledge that the work presented in this paper has been partially funded by Madrid regional government as part of the program S2018/TCS-4339 (BLOQUES-CM) co-funded by EIE Funds of the European Union; by grant IJC2020-043391-I/MCIN/AEI/10.13039/501100011033 and European Union NextGenerationEU/PRTR; and by PRODIGY Project (TED2021-132464B-I00) funded by MCIN/AEI/10.13039/501100011033/ and the European Union NextGenerationEU/PRTR.

REFERENCES

[1] Masayuki Abe and Tatsuaki Okamoto. 2000. Provably secure partially blind signatures. In *Annual International Cryptology Conference*. Springer, 271–286.

[2] Anonymous. [n. d.]. Source Code Implementation of this Project. https://anonymous.4open.science/r/cbdc-code-71DC.

[3] Raphael Auer and Rainer Boehme. 2021. *Central bank digital currency: the quest for minimally invasive technology*. BIS Working Papers 948. Bank for International Settlements. https://ideas.repec.org/p/bis/biswps/948.html

[4] Lukas Aumayr, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostáková, Matteo Maffei, Pedro Moreno-Sanchez, and Siavash Riahi. 2021. Generalized Channels from Limited Blockchain Scripts and Adaptor Signatures. In *ASIACRYPT 2021* (Singapore, Singapore). Springer-Verlag, Berlin, Heidelberg, 635–664. https://doi.org/10.1007/978-3-030-92075-3_22

[5] The Python Cryptographic Authority and individual contributors. [n. d.]. Cryptography. https://cryptography.io/en/latest/

[6] Bank of Canada. 2020. Contingency Planning for a Central Bank Digital Currency. Bank of Canada webpage. https://www.bankofcanada.ca/2020/02/contingency-planning-central-bank-digital-currency/ Accessed on 10.05.2022.

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

[8] Bank of England. 2022. Note circulation scheme. Bank of England webpage. https://www.bankofengland.co.uk/banknotes/note-circulation-scheme Accessed on 26.04.2022.

[9] Bank of International Settlements. 2020. Central bank digital currencies: foundational principles and core features. BIS other publications. https://www.bis.org/publ/othp33.pdf Accessed on 26.04.2022.

[10] Bank of International Settlements. 2021. Central bank digital currencies: financial stability implications. BIS other publications. https://www.bis.org/publ/othp42_fin_stab.pdf Accessed on 26.04.2022.

[11] Bank of International Settlements. 2021. Central bank digital currencies: System design and interoperability. BIS other publications. https://www.bis.org/publ/othp42_system_design.pdf Accessed on 26.04.2022.

[12] Bank of International Settlements. 2021. Central bank digital currencies: user needs and adoption. BIS other publications. https://www.bis.org/publ/othp42_user_needs.pdf Accessed on 26.04.2022.

[13] Marco Benedetti, Francesco De Sclavis, Marco Favorito, Giuseppe Galano, Sara Giammusso, Antonio Muci, and Matteo Nardelli. 2022. A PoW-less Bitcoin with Certified Byzantine Consensus. https://doi.org/10.48550/ARXIV.2207.06870

[14] Ahto Buldas, Dirk Draheim, Mike Gault, Risto Laanoja, Takehiko Nagumo, Märt Saarepera, Syed Attique Shah, Joosep Simm, Jamie Steiner, Tanel Tammet, et al. 2022. An ultra-scalable blockchain platform for universal asset tokenization: design and implementation. *IEEE Access* 10 (2022), 77284–77322.

[15] Vitalik Buterin et al. 2014. A next-generation smart contract and decentralized application platform. *white paper* 3, 37 (2014), 2–1.

[16] Central Bank of Nigeria. 2021. Design Paper for the eNaira. eNaira webpage. https://enaira.gov.ng/download/eNaira_Design_Paper.pdf Accessed on 26.04.2022.

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

[18] Melissa Chase, Michele Orrù, Trevor Perrin, and Greg Zaverucha. 2022. Proofs of discrete logarithm equality across groups. Cryptology ePrint Archive, Paper 2022/1593. https://eprint.iacr.org/2022/1593 https://eprint.iacr.org/2022/1593.

[19] David Chaum. 1982. Blind Signatures for Untraceable Payments. In *Advances in Cryptology: Proceedings of CRYPTO '82*. Plenum, 199–203.

[20] David Chaum, Amos Fiat, and Moni Naor. 1988. Untraceable Electronic Cash. In *Advances in Cryptology - CRYPTO (Lecture Notes in Computer Science, Vol. 403)*. Springer, 319–327. https://doi.org/10.1007/0-387-34799-2_25

[21] David Chaum, Christian Grothoff, and Thomas Moser. 2021. How to issue a central bank digital currency. Swiss National Bank Working Papers. https://www.snb.ch/n/mmr/reference/working_paper_2021_03/source/working_paper_2021_03.n.pdf Accessed on 26.04.2022.

[22] Florian Dold. 2019. *The GNU Taler system: practical and provably secure electronic payments. (Le système GNU Taler: Paiements électroniques pratiques et sécurisés)*. Ph. D. Dissertation. University of Rennes 1, France. https://tel.archives-ouvertes.fr/tel-02138082

[23] Andreas Erwig, Sebastian Faust, Kristina Hostáková, Monosij Maitra, and Siavash Riahi. 2021. Two-party adaptor signatures from identification schemes. In *IACR International Conference on Public-Key Cryptography*. Springer, 451–480.

[24] European Central Bank. 2020. A Digital Euro. Digital Euro. https://www.ecb.europa.eu/pub/pdf/other/Report_on_a_digital_euro~4d7268b458.en.pdf Accessed on 26.04.2022.

[25] European Central Bank. 2020. Issuance and circulation. European Central Bank main webpage. https://www.ecb.europa.eu/euro/intro/issuance/html/index.en.html Accessed on 26.04.2022.

[26] European Central Bank. 2020. Study on the payment attitudes of consumers in the euro area (SPACE). European Central Bank surveys. https://www.ecb.europa.eu/pub/pdf/other/ecb.spacereport202012~bb2038bbb6.en.pdf?05ce2c97d994fbcf1c93213ca04347dd Accessed on 13.05.2022.

[27] European Central Bank. 2021. Digital euro experimentation scope and key learnings. Digital Euro. https://www.ecb.europa.eu/pub/pdf/other/ecb.digitaleuroscopekeylearnings202107~564d89045e.en.pdf Accessed on 26.04.2022.

[28] European Central Bank. 2021. Payments statistics: 2020. European Central Bank press release. https://www.ecb.europa.eu/press/pr/stats/paysec/html/ecb.pis2020~5d0ea9dfa5.en.html Accessed on 13.05.2022.

[29] European Central Bank. 2021. What is seigniorage? European Central Bank main webpage. https://www.ecb.europa.eu/ecb/educational/explainers/tell-me/html/seigniorage.en.html Accessed on 26.04.2022.

[30] European Central Bank. 2022. Banknotes and coins circulation. European Central Bank main webpage. https://www.ecb.europa.eu/stats/policy_and_exchange_rates/banknotes+coins/circulation/html/index.en.html Accessed on 22.12.2022.

[31] European Central Bank. 2022. Cash logistics. European Central Bank main webpage. https://www.ecb.europa.eu/stats/policy_and_exchange_rates/banknotes+coins/cash_logistics/html/index.en.html Accessed on 22.12.2022.

[32] European Central Bank. 2022. Information Guide for TARGET2 users. European Central Bank main webpage. https://www.ecb.europa.eu/paym/target/target2/profuse/nov_2021/shared/pdf/Information_Guide_for_TARGET2_users_version_15.1.pdf Accessed on 10.10.2022.

[33] European Central Bank. 2022. Progress on the investigation phase of a digital euro – second report. Digital Euro. https://www.ecb.europa.eu/paym/digital_euro/investigation/governance/shared/files/ecb.degov221221_Progress.en.pdf Accessed on 30.12.2022.

[34] European Central Bank. 2022. What is TARGET Instant Payment Settlement (TIPS)? European Central Bank main webpage. https://www.ecb.europa.eu/paym/target/tips/html/index.en.html Accessed on 26.04.2022.

[35] European Central Bank. 2022. What is TARGET2? European Central Bank main webpage. https://www.ecb.europa.eu/paym/target/target2/html/index.en.html Accessed on 10.10.2022.

[36] 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. 2021. Work stream 3: A New Solution Blockchain and eID. Eesti Pank Varia. https://haldus.eestipank.ee/sites/default/files/2021-07/Work%20stream%203%20-%20A%20New%20Solution%20-%20Blockchain%20and%20eID_1.pdf Accessed on 26.04.2022.

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

[38] European Central Bank and Bank of Japan. 2019. Synchronized cross-border payments. STELLA project. https://www.ecb.europa.eu/paym/intro/publications/pdf/ecb.miptopical190604.en.pdf Accessed on 26.04.2022.

[39] European ParliamentT and Council of European Union. 2015. DIRECTIVE (EU) 2015/849. EUR-Lex. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=

CELEX%3A02015L0849-20210630 Accessed on 12.05.2022.
[40] Matthew Green and Ian Miers. 2017. Bolt: Anonymous Payment Channels for Decentralized Currencies. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 473–489. https://doi.org/10.1145/3133956.3134093
[41] Jonas Gross, Johannes Sedlmeir, Matthias Babel, Alexander Bechtel, and Benjamin Schellinger. 2021. Designing a central bank digital currency with support for cash-like privacy.
[42] Randall Hanegraaf, Nicole Jonker, S. Mandley, and Jelle Miedema. 2018. Life Cycle Assessment of Cash Payments. De nederlandsche Bank Working Paper. https://www.dnb.nl/publicaties/publicatieoverzicht/publicaties-onderzoek/working-papers-2018/life-cycle-assessment-of-cash-payments/ Accessed on 26.04.2022.
[43] D. Richard Hipp. [n. d.]. SQLite. https://www.sqlite.org/index.html
[44] Iberpay. 2021. SMART MONEY Initiative: Preparations for the possible launch of a digital euro or bank digital money by the Spanish financial sector. Iberpay webpage. https://www.iberpay.es/media/21555/iberpay-report-on-smart-money-initiative-by-spanish-banking-sector.pdf Accessed on 24.11.2022.
[45] Zilin Liu, Anjia Yang, Jian Weng, Tao Li, Huang Zeng, and Xiaojian Liang. 2022. GMHL: Generalized Multi-Hop Locks for Privacy-Preserving Payment Channel Networks. Cryptology ePrint Archive, Report 2022/115. https://ia.cr/2022/115.
[46] James Lovejoy, Cory Fields, Madars Virza, Tyler Frederick, David Urness, Kevin Karwaski, Anders Brownworth, and Neha Narula. 2022. A High Performance Payment Processing System Designed for Central Bank Digital Currencies. Federal Reserve of Boston and Digital Currency Initiative. https://dam-prod.media.mit.edu/x/2022/02/04/Hamilton-Whitepaper-2022.pdf Accessed on 26.04.2022.
[47] Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, Matteo Maffei, and Srivatsan Ravi. 2017. Concurrency and privacy with payment-channel networks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 455–471.
[48] Giulio Malavolta, Pedro A. Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. 2019. Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability. *Proceedings 2019 Network and Distributed System Security Symposium* (2019).
[49] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review* (2008), 21260.
[50] National Bank of Cambodia. 2020. Project Bakong: Next Generation Payment System. Bakong Project webpage. https://bakong.nbc.org.kh/download/NBC_BAKONG_White_Paper.pdf Accessed on 26.04.2022.
[51] Shen Noether, Adam Mackenzie, et al. 2016. Ring confidential transactions. *Ledger* 1 (2016), 1–18.
[52] People's Bank of China. 2021. Progress of Research and Development of E-CNY in China. People's Bank of China Press Release. http://www.pbc.gov.cn/en/3688110/3688172/4157443/4293696/2021071614584691871.pdf Accessed on 26.04.2022.
[53] Joseph Poon and Thaddeus Dryja. 2016. The bitcoin lightning network: Scalable off-chain instant payments. bitconlightning.com. https://www.bitcoinlightning.com/wp-content/uploads/2018/03/lightning-network-paper.pdf Accessed on 06.05.2022.
[54] Armin Ronacher. [n. d.]. Flask web development, one drop at a time. https://flask.palletsprojects.com/en/2.2.x/
[55] Sveriges Riksbank. 2017. E-krona project, report 1. E-krona reports. https://www.riksbank.se/globalassets/media/rapporter/e-krona/2017/rapport_ekrona_uppdaterad_170920_eng.pdf Accessed on 26.04.2022.
[56] Sveriges Riksbank. 2018. E-krona project, report 2. E-krona reports. https://www.riksbank.se/globalassets/media/rapporter/e-krona/2018/the-riksbanks-e-krona-project-report-2.pdf Accessed on 26.04.2022.
[57] Sveriges Riksbank. 2021. E-krona pilot phase 1. E-krona reports. https://www.riksbank.se/globalassets/media/rapporter/e-krona/2021/e-krona-pilot-phase-1.pdf Accessed on 26.04.2022.
[58] Sveriges Riksbank. 2022. E-krona pilot phase 2. E-krona reports. https://www.riksbank.se/globalassets/media/rapporter/e-krona/2022/e-krona-pilot-phase-2.pdf Accessed on 26.04.2022.
[59] Erkan Tairi, Pedro Moreno-Sanchez, and Matteo Maffei. 2021. A 2 l: Anonymous atomic locks for scalability in payment channel hubs. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1834–1851.
[60] The Bank of Japan. 2022. Outline of the Issuance and Circulation of Banknotes and Coins. The Bank of Japan webpage. https://www.boj.or.jp/en/note_tfjgs/note/outline/index.htm/ Accessed on 26.04.2022.
[61] Sri Aravinda Krishnan Thyagarajan, Adithya Bhat, Giulio Malavolta, Nico Döttling, Aniket Kate, and Dominique Schröder. 2020. Verifiable timed signatures made practical. In *CCS*. 1733–1750.
[62] Sri Aravinda Krishnan Thyagarajan and Giulio Malavolta. 2021. Lockable Signatures for Blockchains: Scriptless Scripts for All Signatures. *2021 IEEE Symposium on Security and Privacy (SP)* (2021), 937–954.
[63] Sri Aravinda Krishnan Thyagarajan, Giulio Malavolta, and Pedro Moreno-Sanchez. 2022. Universal Atomic Swaps: Secure Exchange of Coins Across All Blockchains. In *IEEE Symposium on Security and Privacy, SP*. IEEE, 1299–1316. https://doi.org/10.1109/SP46214.2022.9833731
[64] Alin Tomescu, Adithya Bhat, Benny Applebaum, Ittai Abraham, Guy Gueta, Benny Pinkas, and Avishay Yanai. 2022. UTT: Decentralized Ecash with Accountable Privacy. Cryptology ePrint Archive, Paper 2022/452. https://eprint.iacr.org/2022/452 https://eprint.iacr.org/2022/452.
[65] Emanuele Urbinati, Alessia Belsito, Daniele Cani, Angela Caporrini, Marco Capotosto, Simone Folino, Giuseppe Galano, Gabriele Goretti, Giancarloand Marcelli, Pietro Tiberi, and Alessia Vita. 2021. A digital euro: a contribution to the discussion on technical design choices. Banca d'Italia. https://www.bancaditalia.it/pubblicazioni/mercati-infrastrutture-e-sistemi-di-pagamento/questioni-istituzionali/2021-010/N.10-MISP.pdf?language_id=1 Accessed on 26.04.2022.
[66] Nicolas Van Saberhagen. 2013. CryptoNote v 2.0. (2013).
[67] VISA. 2020. Authorization and Reversal Processing Requirements for Merchant. VISA webpage. https://usa.visa.com/content/dam/VCOM/global/support-legal/documents/best-practices-authorization-and-reversal-processing.pdf Accessed on 10.10.2022.
[68] Karl Wüst, Kari Kostiainen, and Srdjan Capkun. 2021. Platypus: A Central Bank Digital Currency with Unlinkable Transactions and Privacy Preserving Regulation. Cryptology ePrint Archive, Report 2021/1443. https://ia.cr/2021/1443.
[69] Alexei Zamyatin, Mustafa Al-Bassam, Dionysis Zindros, Eleftherios Kokoris-Kogias, Pedro Moreno-Sanchez, Aggelos Kiayias, and William J Knottenbelt. 2021. Sok: Communication across distributed ledgers. In *International Conference on Financial Cryptography and Data Security*. Springer, 3–36.

A CP ADDITIONAL MATERIAL

In this section, we define correctness of a CP and we present two additional security properties, namely, *CP redeem unforgeability* and *CP refund unforgeability*. Moreover, we prove that CP PvP implies CP witness unforgeability.

A.1 CP Correctness

We define correctness[2] of a CP in Definition 10. For this, we use the three predicates IsFunded, IsUnique and IsValid discussed in Section 2.1.

Definition 10 (CP correctness)**.** *A CP 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* $(\mathbb{C}, w) \in \mathcal{R}$*, all* T*, all the following conditions are satisfied:*

(1) ***Commit:*** *For all* $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux}) \leftarrow \left\langle \begin{matrix} \mathsf{ctCnd}_S(sk_S, \mathbb{C}, T), \\ \mathsf{ctCnd}_R(sk_R, \mathbb{C}, T) \end{matrix} \right\rangle$*, it holds that:*

$$\Pr\left[\ \mathsf{ckTx}(tx_{\mathsf{cnd}}) = 1 \ \middle|\ \begin{matrix} \mathtt{IsFunded}(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}) \\ \mathtt{IsUnique}(tx_{\mathsf{cnd}}) \\ \mathtt{IsValid}(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}) \\ \mathsf{subTx}(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}) = 1 \end{matrix}\ \right] = 1$$

(2) ***Redeem:*** *If timeout* T *has not expired, in addition to step 1, for all* $(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}) \leftarrow \mathsf{Red}(tx_{\mathsf{cnd}}, sk_R, w, \mathsf{aux}, \ddot{pk})$*, all* $w' \leftarrow \mathsf{GetWit}(tx_{\mathsf{cnd}}, tx_{\mathsf{red}}, \mathsf{aux})$*, it holds that*

$$\Pr\left[\ \begin{matrix} \mathsf{ckTx}(tx_{\mathsf{red}}) = 1 \\ (\mathbb{C}, w') \in \mathcal{R} \end{matrix} \ \middle|\ \begin{matrix} \mathtt{IsFunded}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}) \\ \mathtt{IsUnique}(tx_{\mathsf{red}}) \\ \mathtt{IsValid}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}) \\ \mathsf{subTx}(tx_{\mathsf{red}}, \sigma_{\mathsf{red}}) = 1 \end{matrix}\ \right] = 1$$

(3) ***Refund:*** *Let*

$$b := \Pr\left[\ \mathsf{ckTx}(tx_{\mathsf{ref}}) = 1 \ \middle|\ \begin{matrix} \mathtt{IsFunded}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}) \\ \mathtt{IsUnique}(tx_{\mathsf{ref}}) \\ \mathtt{IsValid}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}) \\ \mathsf{subTx}(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}) = 1 \end{matrix}\ \right]$$

If timeout T *has expired, in addition to step 1, for all* $(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}) \leftarrow \mathsf{Ref}(tx_{\mathsf{cnd}}, sk_S, \mathsf{aux}, \ddot{pk})$*, it holds that:*

(a) $b = 0$*, if* $\exists\ tx_{\mathsf{red}}$ *s.t.* $\mathsf{ckTx}(tx_{\mathsf{red}}) \wedge \mathit{isLinked}(tx_{\mathsf{cnd}}, tx_{\mathsf{red}})$.

(b) $b = 1$*, if* $\nexists\ tx_{\mathsf{red}}$ *s.t.* $\mathsf{ckTx}(tx_{\mathsf{red}}) \wedge \mathit{isLinked}(tx_{\mathsf{cnd}}, tx_{\mathsf{red}})$.

A.2 CP Redeem Unforgeability and CP Refund Unforgeability

In this section, we present two additional security properties of Conditional payment ledger, namely, *CP redeem unforgeability* and *CP refund unforgeability*.

In a nutshell, *CP redeem unforgeability* guarantees that given a conditional payment transaction (on the ledger) with payment condition $\mathbb{C}$ and its corresponding witness w, the adversary cannot successfully authorize and submit redeem transactions transferring coins from the escrow account $epk_{S,R}$. The adversary has access to signing oracles for protocol ctCnd and algorithm Red with key sk_R, while not being allowed to query the oracles on transaction tx_{cnd} nor on transaction $\overline{tx_{\mathsf{red}}}$ which outputs as forgery on the ledger.

Similarly, *CP refund unforgeability* guarantees that given a conditional transaction (on the ledger) that has not been redeemed, the adversary cannot successfully authorize and submit refund transactions transferring coins from the escrow account $epk_{S,R}$. The adversary has access to signing oracles for protocol ctCnd and algorithms ctTx and Ref with key sk_S, while not being allowed to query the oracles on transaction tx_{cnd} nor on transaction $\overline{tx_{\mathsf{ref}}}$ which outputs as forgery on the ledger.

We show that CP redeemability implies CP redeem unforgeability which in turn implies BL unforgeability. In addition, we show that CP refundability implies CP refund unforgeability which in turn implies BL unforgeability.

Security Reductions. For all the security reductions in this work, an adversary $\mathcal{B}$ makes use of another adversary $\mathcal{A}$ in a black-box manner. The only exception being the information (i.e. transactions) that $\mathcal{A}$ submits to the ledger(s) she has access to. We assume that $\mathcal{B}$ can obtain this information for the ledgers accessible to both $\mathcal{B}$ and $\mathcal{A}$.

Theorem 2. *If a Conditional payment ledger satisfies CP redeemability, then it also satisfies CP redeem unforgeability.*

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{redForge}_{\Pi_{CP},\mathcal{R},\mathcal{A}}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win ExpRedeem. $\mathcal{B}$ interacts with both $\mathcal{A}$ and the challenger over the ledger Π_{CP}.

- Challenger provides $\mathcal{B}$ with public keys $(pk_R, \ddot{pk})$.
- $\mathcal{B}$ forwards pk_R to $\mathcal{A}$, which returns $(pk_S, \mathbb{C}, w, T)$. Thereafter, $\mathcal{B}$ forwards $(pk_S, \mathbb{C}, T)$ to the challenger.
- $\mathcal{B}$ engages with the challenger and $\mathcal{A}$ on protocol ctCnd acting as a relay. This results in $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux})$.
- Henceforth, $\mathcal{A}$ outputs the forgery $\overline{tx_{\mathsf{red}}}$ on Π_{CP}.
- $\mathcal{B}$ forwards w to the challenger.
- Finally, the challenger uses w to output $(tx_{\mathsf{red}}, \sigma_{\mathsf{red}})$ and submits it to Π_{CP}.

If $\mathcal{A}$ makes a query to $\mathsf{ctCnd}_R O$, $\mathcal{B}$ does not have the private key sk_R, hence it forwards the query to $\mathsf{ctCnd}_R O$ and relays the answer. Similarly, if $\mathcal{A}$ makes a query to $\mathsf{Red}O$, $\mathcal{B}$ does not have the private key sk_R, hence it forwards the query to $\mathsf{Red}O$ and relays the answer. Note that Q is synchronized in both games.

Our adversary $\mathcal{B}$ perfectly simulates redForge to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ is successful in wining redForge with non-negligible probability, this implies that (1) $tx_{\mathsf{cnd}}, \overline{tx_{\mathsf{red}}} \notin Q$ (condition b_0); (2) $tx_{\mathsf{cnd}} \in \Pi_{CP}.TX_L$ and $\mathsf{isSender}(tx_{\mathsf{cnd}}, pk_S) = 1$ (condition b_1); (3) $\mathsf{isCond}(tx_{\mathsf{cnd}}, \mathbb{C}) = 1$ and $\mathsf{isRcvr}(tx_{\mathsf{cnd}}, epk_{S,R}) = 1$ (condition b_2); (4) $\overline{tx_{\mathsf{red}}} \in \Pi_{CP}.TX_L$ and $\mathsf{isLinked}(tx_{\mathsf{cnd}}, \overline{tx_{\mathsf{red}}}) = 1$ (condition b_3); (5) $(\mathbb{C}, w) \in \mathcal{R}$ (condition b_4); (6) timeout T has not expired (condition b_5). , It is easy to see that these are equivalent to conditions b_0, b_1, b_2, b_4, b_5 and b_6 of game ExpRedeem. Moreover, since Q is synchronized in both games, $\overline{tx_{\mathsf{red}}} \notin Q$ and $\overline{tx_{\mathsf{red}}} \in \Pi_{CP}.TX_L$, tx_{red} cannot satisfy the IsFunded predicate, hence condition b_3 of game ExpRedeem holds. However, this result contradicts the assumption that Π_{CP} satisfies CP redeemability. Therefore, such $\mathcal{A}$ cannot exist, thus this theorem has been proven. □

[2]Note that this correctness definition illustrates two clearly differentiated stages: before timeout T expires, only redeem transactions are accepted, after timeout T expires, only refund transactions are accepted. It could be possible that in some ledgers, during this second stage both redeem and refund transaction could be accepted, however, we abstract from those details by assuming that a rational party will run Red in the stage that guarantees a probability of success equal to one.

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

$Q := \emptyset$

$(pk_R, sk_R) \leftarrow \mathsf{ctAcc}(1^\lambda)$

$(pk_S, \mathbb{C}, w, T, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{ctCnd}_R O, \mathsf{Red} O}(pk_R)$

$(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux}, \mathsf{st}_1) \leftarrow \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_0), \\ \mathsf{ctCnd}_R(sk_R, \mathbb{C}, T) \end{matrix} \right\rangle$

$\overline{tx_{\mathsf{red}}} \leftarrow \mathcal{A}(\mathsf{st}_1)$

$b_0 := tx_{\mathsf{cnd}}, \overline{tx_{\mathsf{red}}} \notin Q$

$b_1 := \mathsf{ckTx}(tx_{\mathsf{cnd}}) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}, pk_S)$

$b_2 := \mathsf{isCond}(tx_{\mathsf{cnd}}, \mathbb{C}) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}, epk_{S,R})$

$b_3 := \mathsf{ckTx}(\overline{tx_{\mathsf{red}}}) \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}, \overline{tx_{\mathsf{red}}})$

$b_4 := (\mathbb{C}, w,) \in \mathcal{R}$

$b_5 := \mathtt{readTime}(\cdot) < T$

return $\bigwedge_{i=0}^{5} b_i$

Figure 9: Experiments for CP redeem unforgeability.

.

Theorem 3. *If a Conditional payment ledger satisfies CP redeem unforgeability, then it also satisfies BL unforgeability.*

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{bscForge}_{\Pi_{BL},\mathcal{A}}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win redForge. $\mathcal{B}$ interacts with both $\mathcal{A}$ and the challenger over the ledger Π_{CP}.

- Challenger provides $\mathcal{B}$ with receiver public key pk_R.
- $\mathcal{B}$ generates (pk_S, sk_S), $\mathbb{C}$ and T and forwards $(pk_S, \mathbb{C}, T)$ to the challenger.
- $\mathcal{B}$ engages with the challenger on protocol ctCnd that outputs the tuple $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux})$.
- Thereafter $\mathcal{B}$ reads the escrow public key $epk_{S,R}$ from tx_{cnd} and forwards it to $\mathcal{A}$, which returns $(\overline{tx_{\mathsf{bsc}}}, \overline{pk_R})$. $\overline{tx_{\mathsf{bsc}}}$ being a forgery on Π_{CP}.
- Finally, $\mathcal{B}$ forwards $\overline{tx_{\mathsf{red}}} := \overline{tx_{\mathsf{bsc}}}$ to the challenger.

If $\mathcal{A}$ makes a query to $\mathsf{ctTx}O$, $\mathcal{B}$ computes the tuple $(tx_{\mathsf{bsc}}, \sigma_{\mathsf{bsc}})$ and forwards it to $\mathcal{A}$. Note that Q is synchronized in both games.

Our adversary $\mathcal{B}$ perfectly simulates bscForge to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ is successful in wining bscForge with non-negligible probability, this implies that (1) $tx_{\mathsf{bsc}} \notin Q$ (condition b_0); (2) $tx_{\mathsf{bsc}} \in \Pi_{CP}.TX_L$ and $\mathsf{isSender}(tx_{\mathsf{bsc}}, epk_{S,R}) = 1$ (condition b_1); , It is easy to see that these are equivalent to conditions b_0 and b_3 of game redForge. Moreover, conditions b_1 and b_2 hold due to the correct execution of protocol ctCnd between $\mathcal{B}$ and the challenger. Furthermore, we know that Q is synchronized in both games. However, this result contradicts the assumption that Π_{CP} satisfies CP redeem unforgeability. Therefore, such $\mathcal{A}$ cannot exist, thus this theorem has been proven. □

Theorem 4. *If a Conditional payment ledger satisfies CP refundability, then it also satisfies CP refund unforgeability.*

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{refForge}_{\Pi_{CP},\mathcal{A}}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win ExpRefund. $\mathcal{B}$ interacts with both $\mathcal{A}$ and the challenger over the ledger Π_{CP}.

- Challenger provides $\mathcal{B}$ with public keys $(pk_S, \ddot{pk})$.
- $\mathcal{B}$ forwards pk_S to $\mathcal{A}$, which returns $(pk_R, \mathbb{C}, T)$. Thereafter, $\mathcal{B}$ forwards $(pk_R, \mathbb{C}, T)$ to the challenger.
- $\mathcal{B}$ engages with the challenger and $\mathcal{A}$ on protocol ctCnd acting as a relay. This results in $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux})$. Thereafter, the challenger submits $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}})$ to Π_{CP}.
- Henceforth, $\mathcal{A}$ outputs the forgery $\overline{tx_{\mathsf{ref}}}$ on Π_{CP}.
- Finally, the challenger generates a refund transaction tx_{ref} from tx_{cnd} and submits $(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}})$ to Π_{CP}.

If $\mathcal{A}$ makes a query to $\mathsf{ctTx}O$, $\mathcal{B}$ does not have the private key sk_S, hence it forwards the query to $\mathsf{ctTx}O$ and relays the answer. Similarly, if $\mathcal{A}$ makes a query to $\mathsf{ctCnd}_S O$, $\mathcal{B}$ does not have the private key sk_S, hence it forwards the query to $\mathsf{ctCnd}_S O$ and relays the answer. Finally, if $\mathcal{A}$ makes a query to $\mathsf{Ref}O$, $\mathcal{B}$ does not have the private key sk_S, hence it forwards the query to $\mathsf{Ref}O$ and relays the answer. Note that Q is synchronized in both games.

Our adversary $\mathcal{B}$ perfectly simulates refForge to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ is successful in wining refForge with non-negligible probability, this implies that (1) $tx_{\mathsf{cnd}}, \overline{tx_{\mathsf{ref}}} \notin Q$ (condition b_0); (2) $tx_{\mathsf{cnd}} \in \Pi_{CP}.TX_L$ and $\mathsf{isSender}(tx_{\mathsf{cnd}}, pk_S) = 1$ (condition b_1); (3) $\mathsf{isCond}(tx_{\mathsf{cnd}}, \mathbb{C}) = 1$ and $\mathsf{isRcvr}(tx_{\mathsf{cnd}}, epk_{S,R}) = 1$ (condition b_2); (4) $\overline{tx_{\mathsf{ref}}} \notin \Pi_{CP}.TX_L$ and $\mathsf{isLinked}(tx_{\mathsf{cnd}}, \overline{tx_{\mathsf{ref}}}) = 1$ (condition b_3); (5) timeout T has expired (condition b_4). , It is easy to see that these are equivalent to conditions b_0, b_1, b_2, b_4 and b_6 of game ExpRefund. Moreover, conditions b_1 and b_2 hold due to the correct execution of protocol ctCnd between $\mathcal{B}$ and the challenger. Furthermore, we know that Q is synchronized in both games. However, this result contradicts the assumption that Π_{CP} satisfies CP refundability. Therefore, such $\mathcal{A}$ cannot exist, thus this theorem has been proven. □

Theorem 5. *If a Conditional payment ledger satisfies CP refund unforgeability, then it also satisfies BL unforgeability.*

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{bscForge}_{\Pi_{BL},\mathcal{A}}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win refForge. $\mathcal{B}$ interacts with both $\mathcal{A}$ and the challenger over the ledger Π_{CP}.

- Challenger provides $\mathcal{B}$ with receiver public key pk_S.
- $\mathcal{B}$ generates (pk_R, sk_R), $\mathbb{C}$ and T and forwards $(pk_S, \mathbb{C}, T)$ to the challenger.
- $\mathcal{B}$ engages with the challenger on protocol ctCnd that outputs the tuple $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux})$. Thereafter, the challenger submits $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}})$ to Π_{CP}.
- Henceforth, $\mathcal{B}$ reads the escrow public key $epk_{S,R}$ from tx_{cnd} and forwards it to $\mathcal{A}$, which returns $(\overline{tx_{\mathsf{bsc}}}, \overline{pk_R})$. $\overline{tx_{\mathsf{bsc}}}$ being a forgery on Π_{CP}.
- Finally, $\mathcal{B}$ forwards $\overline{tx_{\mathsf{red}}} := \overline{tx_{\mathsf{bsc}}}$ to the challenger.

If $\mathcal{A}$ makes a query to $\mathsf{ctTx}O$, $\mathcal{B}$ does not have the private key sk_S, hence it forwards the query to $\mathsf{ctTx}O$ and relays the answer. Note that Q is synchronized in both games.

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

$Q := \emptyset$

$(pk_S, sk_S) \leftarrow \mathsf{ctAcc}(1^\lambda)$

$(pk_R, \mathbb{C}, T, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{ctTx}O,\mathsf{ctCnd}_S O,\mathsf{Ref}O}(pk_S)$

$(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux}, \mathsf{st}_1) \leftarrow \left\langle \begin{matrix} \mathsf{ctCnd}_S(sk_S, \mathbb{C}, T), \\ \mathcal{A}(\mathsf{st}_0) \end{matrix} \right\rangle$

$\mathsf{subTx}(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}})$

$\overline{tx_{\mathsf{ref}}} \leftarrow \mathcal{A}(\mathsf{st}_1)$

$b_0 := tx_{\mathsf{cnd}}, \overline{tx_{\mathsf{ref}}} \notin Q$

$b_1 := \mathsf{ckTx}(tx_{\mathsf{cnd}}) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}, pk_S)$

$b_2 := \mathsf{isCond}(tx_{\mathsf{cnd}}, \mathbb{C}) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}, epk_{S,R})$

$b_3 := \mathsf{ckTx}(\overline{tx_{\mathsf{ref}}}) \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}, \overline{tx_{\mathsf{ref}}})$

$b_4 := \mathtt{readTime}(\cdot) \geq T$

return $\bigwedge_{i=0}^{4} b_i$

Figure 10: Experiments for CP refund unforgeability.

Our adversary $\mathcal{B}$ perfectly simulates bscForge to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ is successful in wining bscForge with non-negligible probability, this implies that (1) $tx_{\mathsf{bsc}} \notin Q$ (condition b_0); (2) $tx_{\mathsf{bsc}} \in \Pi_{CP}.TX_L$ and $\mathsf{isSender}(tx_{\mathsf{bsc}}, epk_{S,R}) = 1$ (condition b_1); , It is easy to see that these are equivalent to conditions b_0 and b_3 of game redForge. Moreover, conditions b_1 and b_2 hold due to the correct execution of protocol ctCnd between $\mathcal{B}$ and the challenger. Furthermore, we know that Q is synchronized in both games. However, this result contradicts the assumption that Π_{CP} satisfies CP refund unforgeability. Therefore, such $\mathcal{A}$ cannot exist, thus this theorem has been proven. □

A.3 CP PvP implies CP Witness Unforgeability

Theorem 6. *If a Conditional payment ledger satisfies CP PvP against a Basic ledger, then it also satisfies CP witness unforgeability.*

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{wForge}_{\Pi_{CP},\mathcal{R},\mathcal{A}}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win ExpPvP. $\mathcal{B}$ interacts with $\mathcal{A}$ over the ledger CP and the challenger over the ledgers CP and BL.

- Challenger provides $\mathcal{B}$ with public keys $(pk_S^{\mathsf{CP}}, pk_R^{\mathsf{BL}})$.
- $\mathcal{B}$ sets $pk_S := pk_S^{\mathsf{CP}}$, generates $(pk_S^{\mathsf{BL}}, sk_S^{\mathsf{BL}})$ and $\mathbb{C} := tx_{\mathsf{bsc}}^{\mathsf{BL}}$ and forwards $(pk_S, \mathbb{C})$ to $\mathcal{A}$, which returns (pk_R, T).
- $\mathcal{B}$ sets $pk_R^{\mathsf{CP}} := pk_R$ and forwards $(pk_R^{\mathsf{CP}}, pk_S^{\mathsf{BL}}, tx_{\mathsf{bsc}}^{\mathsf{BL}}, T)$ to the challenger.
- $\mathcal{B}$ engages with the challenger and $\mathcal{A}$ on protocol ctCnd acting as a relay. This results in $(tx_{\mathsf{cnd}}^{\mathsf{CP}}, \sigma_{\mathsf{cnd}}^{\mathsf{CP}}, \mathsf{aux}^{\mathsf{CP}}) := (tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux})$. Thereafter, the challenger submits $(tx_{\mathsf{cnd}}^{\mathsf{CP}}, \sigma_{\mathsf{cnd}}^{\mathsf{CP}})$ to CP.
- $\mathcal{A}$ outputs the forgery $\overline{tx_{\mathsf{red}}}$ on CP.
- Finally, $\mathcal{B}$ forwards $\overline{tx_{\mathsf{red}}}^{\mathsf{CP}} := \overline{tx_{\mathsf{red}}}$ to the challenger.

If $\mathcal{A}$ makes an O_{ctCnd_S} query, $\mathcal{B}$ does not have the private key pk_S^{CP}, hence it forwards the query to O_{ctCnd_S} and relays the answer. Note that Q is synchronized in both games.

Our adversary $\mathcal{B}$ perfectly simulates wForge to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ is successful in wining wForge with non-negligible probability, this implies that (1) $tx_{\mathsf{cnd}} \notin Q$ (condition b_0); (2) $tx_{\mathsf{cnd}} \in \Pi_{CP}.TX_L$ and $\mathsf{isSender}(tx_{\mathsf{cnd}}, pk_S) = 1$ (condition b_1); (3) $\mathsf{isCond}(tx_{\mathsf{cnd}}, \mathbb{C}) = 1 \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}, epk_{S,R}) = 1$ (condition b_2); (4) $\mathsf{ckTx}(\overline{tx_{\mathsf{red}}}) = 1 \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}, \overline{tx_{\mathsf{red}}}) = 1$ (condition b_3). , It is easy to see that these are equivalent to conditions b_0, b_2, b_2 and b_3 of game ExpPvP. Moreover, $tx_{\mathsf{bsc}}^{\mathsf{BL}}$ is generated calling $\mathsf{ctTx}(sk_S^{\mathsf{BL}}, pk_R^{\mathsf{BL}})$, hence condition b_4 of game ExpPvP holds. Furthermore, $\mathcal{A}$ neither has the private key sk_S^{BL} nor it interacts with ledger BL, hence condition b_5 of game ExpPvP holds. Finally, we know that Q is synchronized in both games. However, this result contradicts the assumption that Π_{CP} satisfies CP PvP against BL. Therefore, such $\mathcal{A}$ cannot exist, thus this theorem has been proven. □

B CCE ADDITIONAL MATERIAL

In this section, we define correctness for CCE, CCE transaction indistinguishability and we present the games of intermediary and debtor balance security.

B.1 CCE Correctness

Hereby, we define correctness for CCE

Definition 11 (CCE correctness)**.** *A CCE is said to be* correct *if for all* $\lambda \in \mathbb{N}$, *all* $(pk_C^{\mathsf{cbl}}, sk_C^{\mathsf{cbl}}) \leftarrow \mathsf{cbl.ctAcc}(1^\lambda)$, *all* $(pk_D^{\mathsf{cbl}}, sk_D^{\mathsf{cbl}}) \leftarrow \mathsf{cbl.ctAcc}(1^\lambda)$, *all* $(pk_D^0, sk_D^0) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$, *all* $(pk_I^0, sk_I^0) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$, *all* $(pk_I^1, sk_I^1) \leftarrow \Pi_{CP_1}.\mathsf{ctAcc}(1^\lambda)$, *all* $(pk_C^1, sk_C^1) \leftarrow \Pi_{CP_1}.\mathsf{ctAcc}(1^\lambda)$, *all* $(\mathbb{C}, w) \in \mathcal{R}$, *all the following conditions are satisfied:*

(1) **Set-Up:** *For all* $\left\langle \begin{matrix} (tx_{\mathsf{cnd}}^{\mathsf{cbl}}/\perp, \mathsf{aux}_C), \\ (tx_{\mathsf{cnd}}^0, \mathsf{aux}_I), \\ (tx_{\mathsf{cnd}}^1, \mathsf{aux}_D) \end{matrix} \right\rangle \leftarrow \left\langle \begin{matrix} \mathsf{Set}_C(sk_C^0, sk_C^{\mathsf{cbl}}, T), \\ \mathsf{Set}_I(sk_I^1, sk_I^0), \\ \mathsf{Set}_D(sk_D^{\mathsf{cbl}}, sk_D^1, \mathbb{C}) \end{matrix} \right\rangle$, *it holds that:*

$$\Pr\begin{bmatrix} \mathsf{ckTx}(tx_{\mathsf{cnd}}^0) = 1 \\ \mathsf{ckTx}(tx_{\mathsf{cnd}}^1) = 1 \\ \mathsf{ckTx}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}) = 1 \end{bmatrix} = 1$$

(2) **Pay:** *If timeout* T *has not expired, in addition to step 1, for all* $(\ddot{pk}_j^i, \ddot{sk}_j^i) \leftarrow \mathsf{ctAcc}(1^\lambda)$, *for every* $(i, j) \in \{(0, I), (1, D), (\mathsf{cbl}, C)\}$, *for all* $\left\langle \begin{matrix} (tx_{\mathsf{red}}^{\mathsf{cbl}}/tx_{\mathsf{bsc}}^{\mathsf{cbl}}), \\ (tx_{\mathsf{red}}^0), \\ (tx_{\mathsf{red}}^1) \end{matrix} \right\rangle \leftarrow \left\langle \begin{matrix} \mathsf{Pay}_C(sk_C^{\mathsf{cbl}}, tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathsf{aux}_C, \ddot{pk}_C^{\mathsf{cbl}}), \\ \mathsf{Pay}_I(sk_I^0, tx_{\mathsf{cnd}}^0, \mathsf{aux}_I, \ddot{pk}_I^0), \\ \mathsf{Pay}_D(sk_D^1, tx_{\mathsf{cnd}}^1, w, \mathsf{aux}_D, \ddot{pk}_D^1) \end{matrix} \right\rangle$, *it holds that:*

$$\Pr\begin{bmatrix} \mathsf{ckTx}(tx_{\mathsf{red}}^0) = 1 \\ \mathsf{ckTx}(tx_{\mathsf{red}}^1) = 1 \\ \mathsf{ckTx}(tx_{\mathsf{red}}^{\mathsf{cbl}}/tx_{\mathsf{bsc}}^{\mathsf{cbl}}) = 1 \end{bmatrix} = 1$$

(3) **Refund:** *Let*

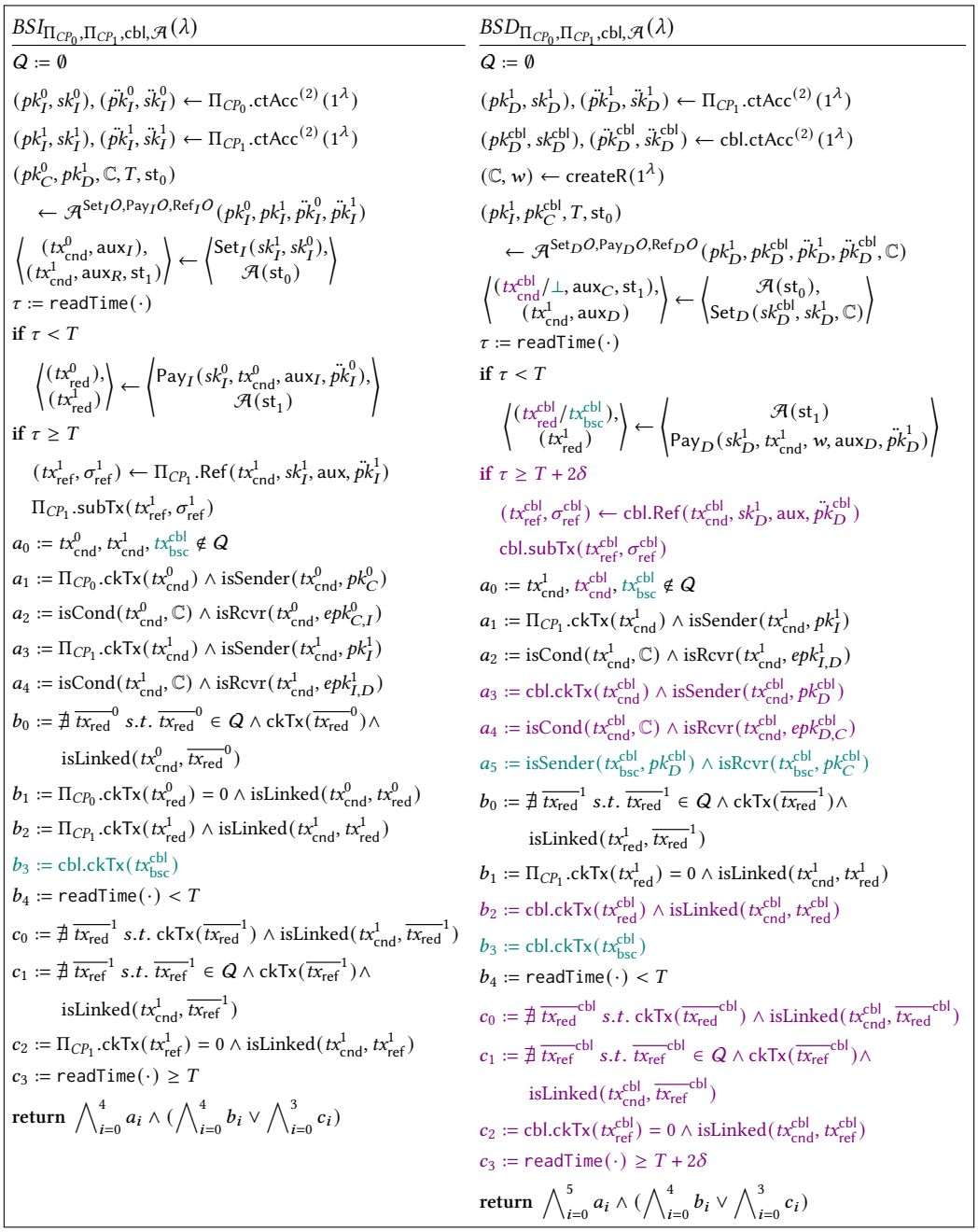

Figure 11: Experiments for balance security of intermediary and debtor C. Winning conditions and algorithm inputs/outputs that apply only when cbl is a CP or a BL are denoted with colors violet and teal, respectively. Winning conditions a_i, b_i and c_i correspond to the outcomes of protocols Set, Pay and algorithm Ref, respectively.

$$b := \Pr\left[\mathrm{ckTx}(tx_{\mathsf{ref}}^i) = 1 \;\middle|\; \begin{matrix}\mathtt{IsFunded}(tx_{\mathsf{ref}}[S])\\ \mathtt{IsUnique}(tx_{\mathsf{ref}})\\ \mathtt{IsValid}(tx_{\mathsf{ref}}^i, \sigma_{\mathsf{ref}}^i)\\ \mathrm{subTx}(tx_{\mathsf{ref}}^i, \sigma_{\mathsf{ref}}^i) = 1\end{matrix} \right]$$

If timeout $T+2\delta$ has expired, in addition to step 1, for all $(\ddot{pk}_j^i, \ddot{sk}_j^i) \leftarrow \mathrm{ctAcc}(1^\lambda)$, for all $(tx_{\mathsf{ref}}^i, \sigma_{\mathsf{ref}}^i) \leftarrow \mathrm{Ref}(tx_{\mathsf{cnd}}^i, \sigma_{\mathsf{cnd}}^i, sk_j, \mathrm{aux}, \ddot{pk}_j^i)$, for every $(i,j) \in \{(0,C),(1,I),(\mathsf{cbl},D)\}$, it holds that:
(a) $b = 0$, if $\exists\ tx_{\mathsf{red}}^i$ s.t. $\mathrm{ckTx}(tx_{\mathsf{red}}^i) \wedge isLinked(tx_{\mathsf{cnd}}^i, tx_{\mathsf{red}}^i)$.
(b) $b = 1$, if $\nexists\ tx_{\mathsf{red}}^i$ s.t. $\mathrm{ckTx}(tx_{\mathsf{red}}^i) \wedge isLinked(tx_{\mathsf{cnd}}^i, tx_{\mathsf{red}}^i)$.

B.2 CCE Security

In Figure 11, we present the games of intermediary and debtor balance security.

B.3 CCE Privacy

CCE basic transaction indistinguishability ensures privacy towards a third party (i.e., the adversary) observing transactions submitted to cbl. The adversary is provided with three masked accounts and is required to fund two of them. Thenceforth, one of the accounts is chosen (u.a.r.) and a basic transaction transferring coins from the selected account to the third account is submitted to the ledger. Finally, the adversary is required to identify the funded account that was used as the sender in the transaction. The adversary can generate arbitrary accounts under its control and has access to oracle ctAcc$\mathcal{O}$ that enables it to initialize additional honest users in the system. Moreover, it has access to oracles ctTx$\mathcal{O}$ that enables it to create arbitrary transactions for honest users' accounts; and ccePay$\mathcal{O}$ that provides transcripts of executions of protocols Set and Pay. Finally, the adversary is not allowed to query the oracle on transactions transferring coins from either of the two funded accounts.

Theorem 7. *If ledger* cbl *is a BL that satisfies BL transaction indistinguishability, then CCE satisfies CCE basic transaction indistinguishability.*

Proof. Assume by contradiction that there exists a PPT $\mathcal{A}$ such that $\Pr[\mathsf{CCE}-\mathsf{bscIND}_{\mathsf{cbl},\mathcal{A}}(\lambda)=1] > \frac{1}{2}+\mathsf{negl}(\lambda)$. Then we can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win bscIND. $\mathcal{B}$ interacts with both $\mathcal{A}$ and the challenger over the ledger cbl.

- Challenger provides $\mathcal{B}$ with masked public accounts $(\widetilde{pk_0}, \widetilde{pk_1}, \widetilde{pk_2})$.
- $\mathcal{B}$ sets $\widetilde{pk_0}^{\mathsf{cbl}} := \widetilde{pk_0}$, $\widetilde{pk_1}^{\mathsf{cbl}} := \widetilde{pk_1}$, $\widetilde{pk_2}^{\mathsf{cbl}} := \widetilde{pk_2}$ and forwards $(\widetilde{pk_0}^{\mathsf{cbl}}, \widetilde{pk_1}^{\mathsf{cbl}}, \widetilde{pk_2}^{\mathsf{cbl}})$ to $\mathcal{A}$, which returns $[tx_{\mathsf{bsc}}^{\mathsf{cbl}}]_0$ and $[tx_{\mathsf{bsc}}^{\mathsf{cbl}}]_1$.
- Thereafter, $\mathcal{B}$ forwards $(tx_{\mathsf{bsc}}^0 := tx_{\mathsf{bsc}}^{\mathsf{cbl}^0}, tx_{\mathsf{bsc}}^1 := tx_{\mathsf{bsc}}^{\mathsf{cbl}^1})$ to the challenger.
- The challenger samples a random bit $c \in \{0,1\}$, creates a transaction tx_{bsc}^* transferring coins from $\widetilde{pk_c}$ to $\widetilde{pk_2}$. Henceforth, the challenger submits tx_{bsc}^* to ledger cbl and forwards it to $\mathcal{B}$.
- $\mathcal{B}$ forwards $tx_{\mathsf{bsc}}^{\mathsf{cbl}^*} := tx_{\mathsf{bsc}}^*$ to $\mathcal{A}$, which replies with a guess bit c^*. Finally, $\mathcal{B}$ forwards c^* to the challenger.

If $\mathcal{A}$ makes a query to ctAcc$\mathcal{O}$ or ctTx$\mathcal{O}$, $\mathcal{B}$ forwards it to oracles ctAcc$\mathcal{O}$ and ctTx$\mathcal{O}$, respectively, and relays the answer. If $\mathcal{A}$ makes a query to ccePay$\mathcal{O}$, $\mathcal{B}$ follows all the steps of protocols Set and Pay per the protocol description forwarding any call to algorithm cbl.ctTx to oracle ctTx$\mathcal{O}$ and forwards the transcript to $\mathcal{A}$. Note that $\mathcal{Q}$ is synchronized in both games.

Our adversary $\mathcal{B}$ perfectly simulates CCE − bscIND to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ is successful in winning CCE − bscIND with non-negligible probability, this implies that (1) $c = c^*$ (condition b_0). (2) $\mathsf{cbl.ckTx}(tx_{\mathsf{bsc}}^{\mathsf{cbl}^0}) = 1$ and $\mathsf{isRcvr}(tx_{\mathsf{bsc}}^{\mathsf{cbl}^0}, \widetilde{pk_0}^{\mathsf{cbl}}) = 1$ (condition b_1); (3) $\mathsf{cbl.ckTx}(tx_{\mathsf{bsc}}^{\mathsf{cbl}^1}) = 1$ and $\mathsf{isRcvr}(tx_{\mathsf{bsc}}^{\mathsf{cbl}^1}, \widetilde{pk_1}^{\mathsf{cbl}}) = 1$ (condition b_2); (4) $\mathcal{A}$ did not use oracle to generate any transaction transferring coins either from $\widetilde{pk_0}^{\mathsf{cbl}}$ or $\widetilde{pk_1}^{\mathsf{cbl}}$ (condition b_3). It is easy to see that these are equivalent to conditions b_0, b_1, b_2, b_3 of game bscIND. Furthermore, we know that $\mathcal{Q}$ is synchronized in both games. However, this result contradicts the assumption that cbl satisfies BL transaction indistinguishability. Therefore, such adversary $\mathcal{A}$ cannot exist, thus this theorem has been proven. □

$\mathsf{CCE}-\mathsf{bscIND}_{\mathsf{cbl},\mathcal{A}}(\lambda)$

$\mathcal{Q} := \emptyset$

$(pk_j^{\mathsf{cbl}}, sk_j^{\mathsf{cbl}}) \leftarrow \mathsf{cbl.ctAcc}^{(3)}(1^\lambda)$ / for $j \in \{0,1,2\}$

$\widetilde{pk_j}^{\mathsf{cbl}} \leftarrow \mathsf{maskAcc}^{(3)}(pk_j)$ / for $j \in \{0,1,2\}$

$(tx_{\mathsf{bsc}}^{\mathsf{cbl}^0}, tx_{\mathsf{bsc}}^{\mathsf{cbl}^1}, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{ctAcc}\mathcal{O},\mathsf{ctTx}\mathcal{O},\mathsf{ccePay}\mathcal{O},}(\widetilde{pk_0}^{\mathsf{cbl}}, \widetilde{pk_1}^{\mathsf{cbl}}, \widetilde{pk_2}^{\mathsf{cbl}})$

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

$(tx_{\mathsf{bsc}}^{\mathsf{cbl}^*}, \sigma_{\mathsf{bsc}}^{\mathsf{cbl}^*}) \leftarrow \mathsf{cbl.ctTx}(sk_c^{\mathsf{cbl}}, \widetilde{pk_2}^{\mathsf{cbl}})$

$\mathsf{cbl.subTx}(tx_{\mathsf{bsc}}^{\mathsf{cbl}^*}, \sigma_{\mathsf{bsc}}^{\mathsf{cbl}^*})$

$c^* \leftarrow \mathcal{A}(\mathsf{st}_0, tx_{\mathsf{bsc}}^{\mathsf{cbl}^*})$

$b_0 := (c \stackrel{?}{=} c^*)$

$b_1 := \mathsf{cbl.ckTx}(tx_{\mathsf{bsc}}^{\mathsf{cbl}^0}) \wedge \mathsf{isRcvr}(tx_{\mathsf{bsc}}^{\mathsf{cbl}^0}, \widetilde{pk_0}^{\mathsf{cbl}})$

$b_2 := \mathsf{cbl.ckTx}(tx_{\mathsf{bsc}}^{\mathsf{cbl}^1}) \wedge \mathsf{isRcvr}(tx_{\mathsf{bsc}}^{\mathsf{cbl}^1}, \widetilde{pk_1}^{\mathsf{cbl}})$

$b_3 := \nexists\, \overline{tx_{\mathsf{bsc}}}^{\mathsf{cbl}^i}\ s.t.\ \overline{tx_{\mathsf{bsc}}}^{\mathsf{cbl}^i} \notin \mathcal{Q} \wedge \mathsf{isSender}(\overline{tx_{\mathsf{bsc}}}^{\mathsf{cbl}^i}, \widetilde{pk_i}^{\mathsf{cbl}})$

/ let $i \in \{0,1\}$.

return $\bigwedge_{i=0}^{3} b_i$

$\mathsf{ccePay}\mathcal{O}(\mathbb{C}, w, T, sk_C^0, sk_C^{\mathsf{cbl}}, \widetilde{pk_C}^{\mathsf{cbl}})$

$(pk_D^{\mathsf{cbl}}, sk_D^{\mathsf{cbl}}) \leftarrow \mathsf{cbl.ctAcc}(1^\lambda)$

$(pk_I^0, sk_I^0), (\ddot{pk}_I^0, \ddot{sk}_I^0) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}^{(2)}(1^\lambda)$

$(pk_I^1, sk_I^1), (pk_D^1, sk_D^1), (\ddot{pk}_D^1, \ddot{sk}_D^1) \leftarrow \Pi_{CP_1}.\mathsf{ctAcc}^{(3)}(1^\lambda)$

$$\left\langle \begin{matrix}(\bot, \mathsf{aux}_C),\\ (tx_{\mathsf{cnd}}^0, \mathsf{aux}_I),\\ (tx_{\mathsf{cnd}}^1, \mathsf{aux}_D)\end{matrix} \right\rangle \leftarrow \left\langle \begin{matrix}\mathsf{Set}_C(sk_C^0, sk_C^{\mathsf{cbl}}, T),\\ \mathsf{Set}_I(sk_I^1, sk_I^0),\\ \mathsf{Set}_D(sk_D^{\mathsf{cbl}}, sk_D^1, \mathbb{C})\end{matrix} \right\rangle$$

$$\left\langle \begin{matrix}(tx_{\mathsf{bsc}}^{\mathsf{cbl}}),\\ (tx_{\mathsf{red}}^0),\\ (tx_{\mathsf{red}}^1)\end{matrix} \right\rangle \leftarrow \left\langle \begin{matrix}\mathsf{Pay}_C(sk_C^{\mathsf{cbl}}, \bot, \mathsf{aux}_C, \widetilde{pk_C}^{\mathsf{cbl}}),\\ \mathsf{Pay}_I(sk_I^0, tx_{\mathsf{cnd}}^0, \mathsf{aux}_I, \ddot{pk}_I^0),\\ \mathsf{Pay}_D(sk_D^1, tx_{\mathsf{cnd}}^1, w, \mathsf{aux}_D, \ddot{pk}_D^1)\end{matrix} \right\rangle$$

$\mathcal{Q} := \mathcal{Q} \cup \{tx_{\mathsf{bsc}}^{\mathsf{cbl}}\}$

return $(tx_{\mathsf{bsc}}^{\mathsf{cbl}}, tx_{\mathsf{cnd}}^0, tx_{\mathsf{red}}^0, tx_{\mathsf{cnd}}^1, tx_{\mathsf{red}}^1)$

Figure 12: Experiment for CCE transaction indistinguishability

C BL CONSTRUCTIONS

The simplest form of ledger, namely BL, comprises an ordered list of transactions, a mechanism to prevent double spending and a mechanism to authorize transactions. Instantiating these building blocks, we can model several ledgers as BL such as e-cash [19, 20] (together with its latest iterations GNU-Taler [22], UTT [64] and Platypus [68]), Monero [51, 66] and KSIcash [14]).

In this section, we first provide an instantiation of BL w.r.t. a digital signature scheme and prove that it satisfies BL unforgeability.

As an illustrative example, we modify the generic instantiation to capture how IsUnique and IsFunded are implemented in *eCash* and prove that it offers BL transaction indistinguishability.

C.1 Our Generic Construction

Building Blocks. Our generic construction relies on a digital signature scheme and a double-spending mechanism as described next. We require a digital signature scheme $\Pi_{DS} := (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$, where: (1) KGen gets as input 1^λ and outputs a key pair (pk, sk); (2) Sig gets as input sk and a message m and outputs a signature σ; and (3) 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 13.

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

Figure 13: Experiment for *EUF-CMA*.

Moreover, we assume the existence of a mechanism to prevent double spending. Predicates IsFunded and IsUnique enforce this mechanism.

Our Construction. We use the aforementioned building blocks as shown in Figure 14. In particular, transactions are authorized by a signature created with the transaction sender's signing key. Accordingly, the transaction validation (i.e., IsValid predicate) checks whether the signature accompanying a transaction successfully verifies with respect to the transaction sender's public key.

Correctness. We now analyze the correctness of our construction.

Theorem 8 (BL correctness). *Assume that the digital signature scheme is correct and* IsFunded *and* IsUnique *hold. Then, the generic instantiation is a correct BL according to Definition 2.*

Proof. The ledger accounts are created by executing the ctAcc algorithm, which simply calls Π_{DS}.KGen to create the key pairs. This function is executed twice, in order to generate (pk_S, sk_S) and (pk_R, sk_R).

Then, the sender uses his private key (sk_S) and receiver's public key (pk_R) to run algorithm ctTx and obtain a transaction and authorization pair $(tx_{\text{bsc}}, \sigma_{\text{bsc}})$.

We assume that for tx_{bsc} both IsFunded and IsUnique hold. Then, IsValid checks that the signature provided is valid. This check holds for correctness of the digital signature scheme. Since all three predicates are valid, then subTx will add the transaction to the ledger and return 1. Therefore, the transaction is in the ledger, so ckTx will return 1 when queried about transaction tx_{bsc}. □

Security. We now analyze the security of our construction.

Key generation: Algorithm $\underline{\mathsf{ctAcc}(1^\lambda)}$ does the following:
- Compute $(pk, sk) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$
- Return (pk, sk)

Create transaction: Algorithm $\underline{\mathsf{ctTx}(sk_S, pk_R)}$ does the following:
- Compute pk_S from sk_S
- Set $tx_{\text{bsc}} := (tx_{id}, pk_S, pk_R)$
- Compute $\sigma \leftarrow \Pi_{DS}.\mathsf{Sig}(sk_S, tx_{\text{bsc}})$
- Set $\sigma_{\text{bsc}} := \sigma$
- Return $(tx_{\text{bsc}}, \sigma_{\text{bsc}})$

Submit transaction: Algorithm $\underline{\mathsf{subTx}(tx_{\text{bsc}}, \sigma_{\text{bsc}})}$ does the following:
- Set $a := \mathtt{IsFunded}(tx_{\text{bsc}}, \sigma_{\text{bsc}})$
- Set $b := \mathtt{IsUnique}(tx_{\text{bsc}})$
- Set $c := \mathtt{IsValid}(tx_{\text{bsc}}, \sigma_{\text{bsc}})$
- **if** $a \wedge b \wedge c$
  - Add $(tx_{\text{bsc}}, \sigma_{\text{bsc}})$ to TX_L
  - Return 1
- **else** Return 0

Check transaction: Algorithm $\underline{\mathsf{ckTx}(tx_{\text{bsc}})}$ does the following:
- **if** $tx_{\text{bsc}} \in TX_L$
  - Return 1
- **else** Return 0

IsValid: Predicate that takes as input $(tx_{\text{bsc}}, \sigma_{\text{bsc}})$ and does the following:
- Parse $\sigma := \sigma_{\text{bsc}}$
- Return $\Pi_{DS}.\mathsf{Vf}(tx_{\text{bsc}}, \sigma, tx_{\text{bsc}}[S])$

Figure 14: Instantiation of BL assuming a digital signature scheme and a double spending prevention mechanism.

Theorem 9 (BL unforgeability). *Assume that the digital signature scheme is unforgeable. Then, the generic instantiation 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 as pk_S to $\mathcal{A}$, which replies with a basic transaction, tx_{bsc} and the receiver's public key pk_R.
- $\mathcal{B}$ checks in the ledger for σ_{bsc} and renames the pair $(tx_{\text{bsc}}, \sigma_{\text{bsc}}$ as $((m, \sigma)))$ and forwards this to the challenger.

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

Our adversary B perfectly simulates bscForge to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. If the transaction is in the ledger, this implies that IsValid holds and that the authorization

for the transaction is valid. Therefore, the pair forwarded to the challenger will allow to forge a signature. 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 9. □

C.2 BL Construction from eCash

In *eCash* based ledgers [19, 20, 22, 68], users obtain tokens that are blindly signed by a bank, which is the operator of the ledger. In order to make a payment, users must present an unblinded signature and a token. If an unsigned token is provided, this token is not funded. Only those tokens signed by the bank are funded. The tokens that are spent are added to a list, to prevent other users from spending the same token again.

Using our nomenclature, tokens are the user accounts. The nature of *eCash* requires that these accounts are only used one time. In eCash, `IsFunded` and `IsUnique` are instantiated by a *blind signature scheme* and a list of spent accounts respectively.

Blind Signature Scheme [19]. $\Pi_{BS} := (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$. We have split the description of the signing protocol Sig between a user and a signer into three algorithms Blind, BSig and UnBlind as follows:

- $(B, f) \leftarrow \mathsf{Blind}(1^\lambda, m)$: A PPT algorithm that on input the security parameter λ and the message m outputs a blinded version of the message B and unblinding parameters f.
- $\sigma_{bs} \leftarrow \mathsf{BSig}(B, sk)$: A PPT algorithm that on input the blinded message B and the signing key sk outputs a blinded signature σ_{bs}.
- $\sigma \leftarrow \mathsf{UnBlind}(\sigma_{bs}, f)$: A DPT algorithm that on input the blinded signature σ_{bs} and the unblinding parameters f outputs a signature σ.

We require that Π_{BS} is correct and secure under the standard notion of existential unforgeability under chosen message attack (EUF-CMA), as well as blind. We refer the reader to Definition 12 for a complete description of the blind property for our description of the Sig protocol.

Definition 12 (Blindness [1])**.** *A blind signature scheme* $\Pi_{BS} := (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$*, with* Sig *described as three algorithms* (Blind, BSig, UnBlind)*, is said to offer* blindness *if for all* $\lambda \in \mathbb{N}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all PPT adversaries* $\mathcal{A}$*, the following holds,* $\Pr[\mathsf{Blindness}(\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$ *where* Blindness *is defined in Figure 15.*

Our Construction. In order to integrate the double spending prevention mechanism of *eCash* in the generic construction presented in Appendix C.1, the instantiation presented in Figure 14 is modified and becomes Figure 16. To do so, we require a helper function called getTx that given a secret key and blinding factor, returns a transaction in which the masked public key of that secret key was a receiver. The ledger operator holds a key pair, $(pk_{ledger}, sk_{ledger})$ that uses to blindly sign transactions according to Π_{BS}. The public key of the ledger, pk_{ledger} is known by all users. Each user has a memory (e.g. $\mathsf{memory_i}$) to keep track of parameter f, used to mask accounts and unblind blinded signatures. To create a transaction, a user obtains the masked key of the receiver, $\widetilde{pk_R}$ and his own private key sk_S. First, the user is going to generate the transaction tuple, tx_{bsc} as in Figure 14. However, then he is going obtain the transaction in which his account received funds. This transaction

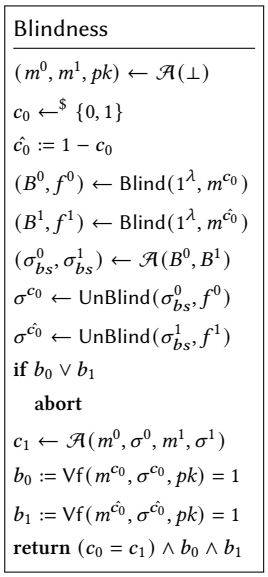

Figure 15: Experiment for *Blindness*.

has a blinded signature, σ_{bs} on his masked key. Using f, he can then unblind the signature to obtain the signature of the ledger on his account, which allows him to spend funds. He then signs the transaction. The authorization of the transaction, σ_{bsc} is a tuple that contains both the signature of sk_S over tx_{bsc} (σ) and the signature of the ledger over his account pk_S (σ_{us}). If the transaction is submitted to the ledger, together wit the authorization, it is validated following the predicates: `IsValid` works as in Figure 14, `IsFunded` verifies if the signature of the ledger (σ_{us}) on $tx_{\text{bsc}}[S]$ is valid and `IsUnique` checks that the sender is not in the transaction list, TX_L.

Correctness. Correctness works as in Appendix C.1, with the additional implementation details of `IsUnique` and `IsFunded`. In particular, `IsFunded` requires that $\widetilde{pk_S}$ was a receiver in a previous transaction that was included in the ledger.

Security. The security definition of BL assumes that `IsFunded` and `IsUnique`, which substantiate the double-spending prevention mechanism, cannot be used to break the BL unforgeability property. Therefore, our assumption is the same as in Appendix C.1. Since the transaction authorization is here also instantiated with a EUF-CMA secure digital signature scheme, BL unforgeability holds, as we proved in Appendix C.1.

Privacy. We now discuss the privacy of eCash as an instantiation of BL.

Theorem 10 (BL eCash transaction indistinguishability). *Assume that the blind signature scheme is blind. Then, the eCash instantiation offers BL transaction indistinguishability according to Definition 6.*

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

- $\mathcal{B}$ generates three key pairs, (pk^0, sk^0), (pk^1, sk^1) and (pk^2, sk^2).

Key generation: Algorithm $\underline{\mathsf{ctAcc}(1^\lambda)}$ does the following:
Compute $(pk, sk) \leftarrow \Pi_{DS}.\mathsf{KGen}(1^\lambda)$.
Return (pk, sk).
Mask key: Algorithm $\underline{\mathsf{maskAcc}(pk)}$ does the following:
Compute $(\widetilde{pk_i}, f) \leftarrow \mathsf{Blind}(1^\lambda, pk_i)$
Add f to $\mathsf{memory}_\mathsf{i}$
Return $\widetilde{pk_i}$
Create transaction: Algorithm $\underline{\mathsf{ctTx}(sk_S, \widetilde{pk_R})}$ does the following:
Compute pk_S from sk_S
Set $tx_{\text{bsc}} := (tx_{id}, pk_S, \widetilde{pk_R})$
Read f from memory_S
Compute $(tx'_{\text{bsc}}, \sigma'_{\text{bsc}}, \sigma_{bs}) \leftarrow \mathsf{getTx}(sk_S, f)$
Compute $\sigma_{us} \leftarrow \mathsf{UnBlind}(\sigma_{bs}, f)$
Compute $\sigma \leftarrow \Pi_{DS}.\mathsf{Sig}(sk_S, tx_{\text{bsc}})$
Set $\sigma_{\text{bsc}} := (\sigma, \sigma_{us})$
Return $(tx_{\text{bsc}}, \sigma_{\text{bsc}})$
Submit transaction: Algorithm $\underline{\mathsf{subTx}(tx_{\text{bsc}}, \sigma_{\text{bsc}})}$ does the following:
Set $a := \mathtt{IsFunded}(tx_{\text{bsc}}, \sigma_{\text{bsc}})$
Set $b := \mathtt{IsUnique}(tx_{\text{bsc}})$
Set $c := \mathtt{IsValid}(tx_{\text{bsc}}, \sigma_{\text{bsc}})$
if $a \wedge b \wedge c$
Compute $\sigma_{bs} \leftarrow \mathsf{BSig}(tx_{\text{bsc}}[R], sk_{ledger})$
Add $(tx_{\text{bsc}}, \sigma_{\text{bsc}}, \sigma_b)$ to TX_L
Return 1
else Return 0
Check transaction: Algorithm $\underline{\mathsf{ckTx}(tx_{\text{bsc}})}$ does the following:
if $tx_{\text{bsc}} \in TX_L$
Return 1
else Return 0
IsFunded: Predicate that takes as input $(tx_{\text{bsc}}, \sigma_{\text{bsc}})$ and does the following:
Parse $\sigma_{\text{bsc}} := (\sigma, \sigma_{us})$
Return $\Pi_{BS}.\mathsf{Vf}(tx_{\text{bsc}}[S], \sigma_{us}, pk_{ledger})$
IsUnique: Predicate that takes as input (tx_{bsc}) and does the following:
if $tx_{\text{bsc}}[S] \notin TX_L$
Return 1
else Return 0
IsValid: Predicate that takes as input $(tx_{\text{bsc}}, \sigma_{\text{bsc}})$ and does the following:
Parse $\sigma_{\text{bsc}} := (\sigma, \sigma_{us})$
Return $\Pi_{DS}.\mathsf{Vf}(tx_{\text{bsc}}, \sigma, tx_{\text{bsc}}[S])$

Figure 16: E-cash as an instantiation of BL.

- $\mathcal{B}$ renames pk^0 as m^0 and pk^1 as m^1. It also obtains the public key of the ledger, which is publicly known by all participants and sends all three items to the challenger.
- The challenger will provide $\mathcal{B}$ with B^0 and B^1. $\mathcal{B}$ masks pk^2 and sends all three items to $\mathcal{A}$.
- $\mathcal{A}$ replies with two transactions, tx^0_{bsc} and tx^1_{bsc}. The first one uses B^0 as receiver, while the second one uses B^1.
- $\mathcal{B}$ calls ckTx for both transactions. If they are in the ledger, obtains the blinded signature for each of them. Then, it sends both blinded signatures to the challenger.
- Since the transactions were in the ledger, the blinded signatures are valid, and the challenger will reply to $\mathcal{B}$, sending the original messages, m^0 and m^1, accompanied with the unblinded signature for each.
- $\mathcal{B}$ flips a bit c and creates a transaction using pk^c. Sends this transaction and its authorization to $\mathcal{A}$, which replies with a bit guess.
- $\mathcal{B}$ forwards the bit to the challenger if c was zero, otherwise, it sends the inverse of the bit received.

It is easy to see that the conditions for winning bscIND are equivalent to those of the blindness game. The first two conditions of bscIND imply that both transactions are on the ledger, which guarantees that they will have a valid blind signature. Therefore, if both of these transaction hold, the unblinded signatures in the blindness game will also verify. The last condition is that the bit is guessed correctly. $\mathcal{B}$ does not know how to distinguish between B^0 and B^1. However, with the response of $\mathcal{A}$, it will learn if pk^c is related to B^0 or B^1. If the bit c was zero and the bit received was zero, then pk^c is related to B^0 and the bit selected by the challenger was also zero. If the bit c was zero and the bit received is one, then the bit selected by the challenger was also one. Conversely, if the bit c was one and the bit received was zero, then pk^c is related to B^0 and the bit selected by the challenger was one. If the bit c was one and the bit received is one, then the bit selected by the challenger was zero.

However, this result contradicts the assumption that the blind signature scheme provides blindness. Therefore, such adversary $\mathcal{A}$ cannot exist and this concludes the proof of Theorem 10. □

D CP CONSTRUCTIONS

Bitcoin and Ethereum are examples of ledgers with scripting capabilities that allow to accept transactions based on additional requirements, like ledger dependent time and a cryptographic condition. A prime example of this the Hash TimeLock Contracts [53]. An HTLC allows to create a transaction in which a sender and a receiver agree to lock funds in an escrow address, under a $\mathbb{C}$ (a hash value) and a timelock. In order to spend the funds from this escrow account, there are two possible paths: (1) 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 (2) after the timelock, a signature of transaction valid under the sender's secret key.

It is easy to see that the functionality of the HTLC matches the definition of CP (Definition 4). In this section we show how the HTLC contract blueprint (i.e., assuming any type of hard relation, not only the hash function) to build a CP and show that it is a correct and secure CP, according to Definition 10 and Definition 5.

Create conditional transaction: Protocol ctCnd is executed by sender S and receiver R and does the following:

$S(sk_S, \mathbb{C}, T)$	$R(sk_R, \mathbb{C}, T)$
Compute pk_S from sk_S	Compute pk_R from sk_R
Receive(pk_R)	Send(pk_R)
Compute $epk_{S,R} \leftarrow$ ctEAcc(pk_S, pk_R)	
Set $E := (epk_{S,R}, \mathbb{C}, T)$	
Set $tx_{\mathsf{cnd}} := (tx_{id}, pk_S, E)$	
Compute $\sigma \leftarrow \Pi_{DS}.\mathsf{Sig}(sk_S, tx_{\mathsf{cnd}})$	
Set $\sigma_{\mathsf{cnd}} := \sigma$	
Set aux $:= \perp$	
Send($tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}$, aux)	Receive($tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}$, aux)
Return $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}$, aux)	Return $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}$, aux)

Redeem: Algorithm Red($tx_{\mathsf{cnd}}, sk_R, w$, aux, $\ddot{pk}$) is run by receiver R and does the following:

Set $tx_{\mathsf{red}} := (tx_{id}, tx_{\mathsf{cnd}}[E], \ddot{pk})$
Compute $\sigma \leftarrow \Pi_{DS}.\mathsf{Sig}(sk_R, tx_{\mathsf{red}})$
Set $\sigma_{\mathsf{red}} := (\sigma, w)$
Return $(tx_{\mathsf{red}}, \sigma_{\mathsf{red}})$

Refund: Algorithm Ref(tx_{cnd}, sk_S, aux, $\ddot{pk}$) is run by sender S and does the following:

Set $tx_{\mathsf{ref}} := (tx_{id}, tx_{\mathsf{cnd}}[E], \ddot{pk})$
Compute $\sigma \leftarrow \Pi_{DS}.\mathsf{Sig}(sk_S, tx_{\mathsf{ref}})$
Set $\sigma_{\mathsf{ref}} := (\sigma)$
Return $(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}})$

Get witness: Algorithm GetWit($tx_{\mathsf{cnd}}, tx_{\mathsf{red}}$, aux) does the following:

Read $(tx_{\mathsf{red}}, \sigma_{\mathsf{red}})$ from TX_L
Parse $(\sigma, w) := \sigma_{\mathsf{red}}$
Return w

Figure 17: Implementation of HTLC's CP functionalities.

Building Blocks. Our construction relies on a digital signature scheme $\Pi_{DS} := (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$. We assume that Π_{DS} is correct and secure under the standard notion of existential unforgeability under chosen message attack (EUF-CMA). We further assume the existence of a mechanism to prevent double spending. Predicates `IsFunded` and `IsUnique` enforce this mechanism.

Our Construction. In Appendix C we proved that a ledger based on a digital signature algorithm and a double spending prevention mechanism is a secure BL. Given that, we focus the presentation of our construction here only on the specific functions that CP adds on top of BL. This is presented in Figure 17.

D.1 Correctness and Security Analysis

We first analyze the correctness of our construction.

Theorem 11 (CP correctness). *Assume that the digital signature scheme and the double spending prevention mechanism are correct. Then, the generic HTLC instantiation is a correct CP according to Definition 10.*

Proof. The ledger accounts are created by executing the ctAcc algorithm, which simply calls Π_{DS}.KGen to create the key pairs. This function is executed twice, in order to generate (pk_S, sk_S) and (pk_R, sk_R).

Conditional Transaction. The sender and the receiver engage in the ctCnd protocol in order to generate a conditional transaction, tx_{cnd}, based on a timeout T and a $\mathbb{C}$. This requires the sender to generate a signature on the conditional transaction, which is submitted to the ledger.

We assume that tx_{cnd} satisfies the requirements of the double spending prevention mechanism (which imply that both `IsFunded` and `IsUnique` hold). Then, `IsValid` checks that the signature provided is valid. This check holds because the digital signature scheme is correct. Since all three predicates are valid, then subTx will add the transaction to the ledger and return 1. Therefore, the transaction is in the ledger, so ckTx will not return 0 when queried about transaction tx_{cnd}.

Now, from here onwards there are two possible paths to follow.

Redeem. If `readTime`$(\cdot) < T$, then the receiver can generate a redeem transaction, tx_{red}, using the conditional transaction, tx_{cnd}, his secret key sk_R, the witness, w, and the account in which he desires to receive the funds $\ddot{pk}$. The receiver the generates a signature on the redeem transaction. The w is also added as part of the authorization. Since no transaction has been accepted yet that spends from tx_{cnd}, `IsUnique` and `IsFunded` hold. Then, `IsValid` checks that the signature provided is corresponds to the receiver's private key (since `readTime`$(\cdot) < T$). This check holds because the digital signature scheme is correct. Additionally, since `readTime`$(\cdot) < T$, the ledger also checks that the witness w corresponds to the $\mathbb{C}$ used in tx_{cnd}. Since all three predicates are valid, if the w corresponds to the $\mathbb{C}$ for the chosen hard relation $\mathcal{R}$, then subTx will add the transaction to the ledger and return 1. Therefore, the transaction is in the ledger, so ckTx will not return 0 when queried about transaction tx_{red}.

Refund. Alternatively, if `readTime`$(\cdot) \geq T$ and no redeem transaction has been accepted in the ledger, then the sender can claim back his funds in a refund transaction, tx_{ref}. To do so, he uses the conditional transaction, tx_{cnd}, his secret key, sk_S and the account in which he desires to get the money back $\ddot{pk}$. The sender the generates a signature on the refund transaction. Since no transaction has been accepted yet that spends from tx_{cnd}, `IsUnique` and `IsFunded` hold. Then, `IsValid` checks that the signature provided is corresponds to the sender's private key (since `readTime`$(\cdot) \geq T$). This check holds because the digital signature scheme is correct. Since all three predicates are valid, then subTx will add the transaction to the ledger and return 1. Therefore, the transaction is in the ledger, so ckTx will not return 0 when queried about transaction tx_{ref}.

And this concludes the proof of Theorem 11 □

We now analyze the security properties of our construction, starting from CP unforgeability.

Theorem 12 (CP unforgeability). *Assume that the digital signature scheme is unforgeable. Then, our protocol offers CP unforgeability according to Definition 5.*

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

- The challenger produces they key pair (pk, sk) and shares pk with $\mathcal{B}$.
- $\mathcal{B}$ forwards pk as pk_S to $\mathcal{A}$, which replies with a conditional transaction, tx_{bsc}, the receiver's public key pk_R, a public statement $\mathbb{C}$ and a timeout T.
- $\mathcal{B}$ checks in the ledger for σ_{cnd} and renames the pair $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}$ as $((m, \sigma)))$ and forwards this to the challenger.

The $\mathsf{ctTx}\mathcal{O}$ and $\mathsf{ctCnd}_S\mathcal{O}$ oracles of cndForge require to call the $\mathsf{Sig}\mathcal{O}$ oracle of the signature unforgeability game in order to generate valid signatures. This guarantees that all messages queried in cndForge are also queried in the signature unforgeability game.

Our adversary B perfectly simulates cndForge to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. If the transaction is in the ledger, this implies that `IsValid` holds and that the authorization for the transaction is valid. Therefore, the pair forwarded to the challenger will allow to forge a signature. 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 12. □

Theorem 13 (CP witness unforgeability). *Assume that* $(\mathbb{C}, w) \in \mathcal{R}$. *Then, our protocol offers CP witness unforgeability according to Definition 5.*

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

- $\mathcal{B}$ receives $\mathbb{C}$ from the challenger.
- $\mathcal{B}$ executes ctAcc to generate (pk_S, sk_S). Sends $\mathbb{C}$ and pk_S to $\mathcal{A}$.
- $\mathcal{A}$ sends pk_R and T to $\mathcal{B}$.
- $\mathcal{B}$ and $\mathcal{A}$ engage in the ctCnd protocol as described in Figure 17, resulting in $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux})$. $\mathcal{B}$ follows the S role.
- $\mathcal{B}$ submits the transaction to the ledger. Since $\mathcal{B}$ controls pk_S and his goal is to emulate the challenger of wForge, he would have selected a pk_S such that `IsFunded` and `IsUnique` hold. Similarly, since $\mathcal{B}$ knows sk_S, he can generate a valid authorization and `IsValid` will hold.
- $\mathcal{A}$ sends $\mathcal{B}$ a redeem transaction $\overline{tx_{\mathsf{red}}}$.
- $\mathcal{B}$ searches in TX_L for $\overline{tx_{\mathsf{red}}}$ and obtains σ_{red}.
- $\mathcal{B}$ extracts w from σ_{red} and sends it to the challenger.

The $\mathsf{ctCnd}_S\mathcal{O}$ can be executed by $\mathcal{B}$, since it holds pk_S.

Our adversary $\mathcal{B}$ perfectly simulates wForge to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Due to Figure 17, a redeem transaction is only accepted if the w that accompanies it corresponds to $\mathbb{C}$ for the selected $\mathcal{R}$. Therefore, since $\overline{tx_{\mathsf{red}}}$ is in the ledger, this means that its authorization must have a valid w, so the forgery sent to the challenger will hold and $\mathcal{B}$ would have used $\mathcal{A}$ to break the hard relation. However, this contradicts the assumptions of the hard relation (given $\mathbb{C}$ it is hard to compute w), so $\mathcal{A}$ cannot exist and this concludes the proof of Theorem 13. □

Theorem 14 (CP redeemability). *Assume that that the digital signature scheme is unforgeable. Then, the generic instantiation offers CP redeemability according to Definition 5.*

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

- $\mathcal{B}$ receives from the challenger a public key, which renames as pk_R. He also generates the account to receive the funds of the redeem transaction, $\ddot{pk}$.
- $\mathcal{B}$ shares both pk_R and $\ddot{pk}$ with $\mathcal{A}$.
- $\mathcal{A}$ responds to $\mathcal{B}$ with her public key, pk_S, a public statement, $\mathbb{C}$ and a timeout T.
- $\mathcal{B}$ and $\mathcal{A}$ engage in the ctCnd protocol as described in Figure 17, resulting in $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux})$. $\mathcal{B}$ follows the R role.
- $\mathcal{A}$ now sends $\mathcal{B}$ a w for the instance of the hard relation.
- $\mathcal{B}$ executes Red by performing a call to the signature oracle of the challenger to obtain $tx_{\mathsf{red}}, \sigma_{\mathsf{red}}$ and submits the pair to the ledger.

If $\mathcal{A}$ makes a $\mathsf{ctCnd}_R\mathcal{O}$ query, $\mathcal{B}$ can follow the protocol without problem, since he knows pk_R and can forward it to $\mathcal{A}$. If $\mathcal{A}$ makes a $\mathsf{Red}\mathcal{O}$ oracle query, then $\mathcal{B}$ makes a query to the signature oracle of the unforgeability game. This ensures that the query memory of both oracles is synchronized.

Our adversary $\mathcal{B}$ perfectly simulates ExpRedeem to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $(tx_{\mathsf{red}}, \sigma_{\mathsf{red}})$ is not accepted in the ledger, this means that at least one of `IsValid`, `IsUnique` and `IsFunded` fail. The first one, `IsValid` must hold, since $\mathcal{B}$ is simply forwarding a signature from the challenger. Therefore, either `IsUnique` or `IsFunded` (or both) fail. This means that $\mathcal{A}$ somehow has managed to spend the funds before with another redeem transaction signed using pk_R, since $\mathtt{readTime}(\cdot) < T$. This means that $\mathcal{A}$ sent a transaction to the ledger with a forged authorization. $\mathcal{B}$ only has to search the ledger for a transaction that spends for the escrow account and its authorization and share it with the challenger to provide a valid forgery. However, this contradicts the assumption that the digital signature scheme is unforgeable and, therefore, $\mathcal{A}$ cannot exists. And this concludes the proof of Theorem 14. □

Theorem 15 (CP refundability). *Assume that that the digital signature scheme is unforgeable. Then, the generic instantiation offers CP refundability according to Definition 5.*

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

- $\mathcal{B}$ receives from the challenger a public key, which renames as pk_S. He also generates the account to receive the funds of the redeem transaction, $\ddot{pk}$.
- $\mathcal{B}$ shares both pk_S and $\ddot{pk}$ with $\mathcal{A}$.
- $\mathcal{A}$ responds to $\mathcal{B}$ with her public key, pk_R, a public statement, $\mathbb{C}$ and a timeout T.
- $\mathcal{B}$ and $\mathcal{A}$ engage in the ctCnd protocol as described in Figure 17, resulting in $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux})$. $\mathcal{B}$ follows the S role. He queries

the signature oracle of the challenger when required to provide a signature on pk_S
- $\mathcal{B}$ submits $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}})$ to the ledger.
- $\mathcal{B}$ executes Ref by performing a call to the signature oracle of the challenger to obtain $tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}}$ and submits the pair to the ledger.

If $\mathcal{A}$ makes a $\mathsf{ctCnd}_S O$ query, $\mathcal{B}$ can follow the protocol until Sig is required, when he forwards this as a signature query. If $\mathcal{A}$ makes a Ref$O$ oracle query, then $\mathcal{B}$ makes a query to the signature oracle of the unforgeability game. This ensures that the query memory of the oracles is synchronized.

Our adversary $\mathcal{B}$ perfectly simulates ExpRefund to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}})$ is not accepted in the ledger, this means that at least one of IsValid, IsUnique and IsFunded fail. The first one, IsValid must hold, since $\mathcal{B}$ is simply forwarding a signature from the challenger. Therefore, either IsUnique or IsFunded (or both) fail. This means that $\mathcal{A}$ somehow has managed to spend the funds with another refund transaction signed with pk_S, since $\mathtt{readTime}(\cdot) \geq T$ and one of the conditions of $\mathsf{ExpRefund}_{\Pi_{CP},\mathcal{R},\mathcal{A}}$ is that no redeem transaction is in the ledger. This means that $\mathcal{A}$ sent a transaction to the ledger with a forged authorization. $\mathcal{B}$ only has to search the ledger for a transaction that spends for the escrow account and its authorization and share it with the challenger to provide a valid forgery. However, this contradicts the assumption that the digital signature scheme is unforgeable and, therefore, $\mathcal{A}$ cannot exists. And this concludes the proof of Theorem 15. □

Theorem 16 (CP extractability). *Assume that* $(\mathbb{C}, w) \in \mathcal{R}$. *Then, the generic instantiation offers CP extractability according to Definition 5.*

Proof. Game ExpExtract requires the sender and $\mathcal{A}$ to engage in the ctCnd protocol to generate a pair $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}})$ that is accepted by the ledger. Then, $\mathcal{A}$ has to successfully send a redeem transaction, tx_{red}, to the ledger. In order for the ledger to accept the tx_{red} transaction, in addition to the three predicates, the ledger checks if $\mathtt{readTime}(\cdot) < T$ and $(\mathbb{C}, w) \in \mathcal{R}$. Since w is added as part of the σ_{red}, it is straightforward for the ledger to check. If the transaction is in the ledger, this means that the ledger is convinced that $(\mathbb{C}, w) \in \mathcal{R}$. Now, if the challenger checks σ_{red} to obtain the w, it cannot happen that $(\mathbb{C}, w) \notin \mathcal{R}$. Therefore, $\mathcal{A}$ cannot exist and this concludes the proof of Theorem 16. □

E CORRECTNESS AND SECURITY OF CCE BL-CBDC IMPLEMENTATION

E.1 Proof of Correctness

Theorem 17 (CCE correctness). *Assume that both CP and BL are correct. Assume that* IsFunded *holds for the accounts of C, I and D. Assume* IsUnique *and* IsValid *hold for transactions created according to the protocol. Then, the construction for CCE with* cbl *of type BL depicted in Figure 8 is correct according to Definition 11.*

Proof. CCE starts with all parties generating their accounts in Π_{CP_0}, Π_{CP_1} and cbl. The following pairs are generated: $(pk_C^{\mathsf{cbl}}, sk_C^{\mathsf{cbl}})$, (pk_C^0, sk_C^0), (pk_I^0, sk_I^0), (pk_I^1, sk_I^1), (pk_D^1, sk_D^1), $(pk_D^{\mathsf{cbl}}, sk_D^{\mathsf{cbl}})$.

The protocol has two stages. The first one is Set. The second stage could either be Pay or refund. We first analyze our implementation for the Set protocol.

Set-Up. Before starting Set, the D runs cbl.ctTx in order to generate $tx_{\mathsf{bsc}}^{\mathsf{cbl}}, \sigma_{\mathsf{bsc}}^{\mathsf{cbl}}$. However, the transaction $\mathbb{C}^* := tx_{\mathsf{bsc}}^{\mathsf{cbl}}$ is not submitted to the ledger.

The implementation for Set starts with C sending T to I and D, and D replying to all with $tx_{\mathsf{bsc}}^{\mathsf{cbl}}$. Then, C and I engage in a Π_{CP_0}.ctCnd protocol using $\mathbb{C}^*$ as condition and $T + \delta$, together with their keys (sk_C^0) and (sk_I^0), in ledger Π_{CP_0}. The protocol produces $(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0)$. Now, C calls $\Pi_{CP_0}.\mathsf{subTx}(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0)$. The intermediary checks if tx_{cnd}^0 is in the ledger, if C is the sender of that transaction, if the $\mathbb{C}^*$ was used as condition in the transaction and if the receiver is the shared account. Since we assume that (sk_C^0) is funded, IsFunded holds. Similarly, we assume that for $tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0$, IsUnique and IsValid hold. Additionally, the Π_{CP_0}.ctCnd protocol used $\mathbb{C}^*$ as input, as well as (sk_C^0) and (sk_I^0). Therefore, since CP is correct, $\Pi_{CP_0}.\mathsf{subTx}(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0) = 1$. This implies that tx_{cnd}^0 is in the ledger, C is the sender of tx_{cnd}^0, $\mathbb{C}^*$ was used in tx_{cnd}^0 and the receiver is the shared account.

Finally, D engages with I to produce $(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1)$. It is easy to see that this is correct for the same reasons as for the generation of $(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0)$. Therefore, D obtains that tx_{cnd}^1 is in the ledger, I is the sender of tx_{cnd}^1, $\mathbb{C}^*$ was used in tx_{cnd}^1 and the receiver is the shared account since Π_{CP_1} is correct.

Therefore, we have shown that for our implementation of the Set protocol, it holds that:

$$\Pr\begin{bmatrix}\mathsf{ckTx}(tx_{\mathsf{cnd}}^0) = 1\\ \mathsf{ckTx}(tx_{\mathsf{cnd}}^1) = 1\end{bmatrix} = 1$$

Pay. Now, if Set has been successful, two branches could happen. Either the conditional transactions are redeemed or they are refunded. If timeout T has not expired, C, I and D can initiate the Pay protocol to redeem the locked funds.

Before initiating Pay, D has to submit $(tx_{\mathsf{bsc}}^{\mathsf{cbl}}, \sigma_{\mathsf{bsc}}^{\mathsf{cbl}})$ to cbl. Since we assume that the account of the D is funded, IsFunded holds. Similarly, we assume that IsValid and IsUnique hold as well. Therefore, since cbl is a correct BL, it holds that $\mathsf{cbl}.\mathsf{subTx}(tx_{\mathsf{bsc}}^{\mathsf{cbl}}, \sigma_{\mathsf{bsc}}^{\mathsf{cbl}}) = 1$, so the transaction is in the ledger.

The implementation of Pay starts by each party obtaining the information stored in the aux that they obtained from the correct execution of Set . We have already proven that $w^* := tx_{\mathsf{bsc}}^{\mathsf{cbl}}$ is on cbl because of BL correctness. Since $tx_{\mathsf{bsc}}^{\mathsf{cbl}}$ is in cbl, both C and I learn the w of the hard relation.

Therefore, they can run Π_{CP_0}.Red and Π_{CP_1}.Red, using w^* as witness, and obtain $(tx_{\mathsf{red}}^0, \sigma_{\mathsf{red}}^0)$ for I and $(tx_{\mathsf{red}}^1, \sigma_{\mathsf{red}}^1)$ for C, respectively. Now, when they submit these transactions to the ledger, IsFunded will hold, since there has not been accepted any redeem transaction in the ledger that spends either from tx_{cnd}^0 or tx_{cnd}^1. Similarly, we assume that IsValid and IsUnique hold. Since Π_{CP_0} and Π_{CP_1} are correct, then tx_{red}^0 and tx_{red}^1 are successfully submitted to the ledger, so $\mathsf{ckTx}(tx_{\mathsf{red}}^0) = 1$ and $\mathsf{ckTx}(tx_{\mathsf{red}}^1) = 1$. Additionally, since I and C take tx_{cnd}^0 and tx_{cnd}^1 as input for Π_{CP_0}.Red and

Π_{CP_1}.Red, tx^0_{red} is linked to tx^0_{cnd}, and tx^1_{red} is linked to tx^1_{cnd}. Therefore, the Pay protocol will not abort. As such,

$$\Pr\begin{bmatrix}\mathsf{ckTx}(tx^0_{\mathsf{red}})=1\\ \mathsf{ckTx}(tx^1_{\mathsf{red}})=1\\ \mathsf{ckTx}(tx^{\mathsf{cbl}}_{\mathsf{bsc}})=1\end{bmatrix}=1$$

and the Pay implementation has been proven correct.

Refund. Finally, if Set has happened and $\tau \geq T + 2\delta$, there are two possible options: Either refund happens or not.

If Pay has happened, then we have a transaction that is spending from the shared accounts used in $tx_{\mathsf{cnd}}0$ and $tx_{\mathsf{cnd}}1$. Therefore, if any of the parties attempts to submit a refund transaction tx^i_{ref} (for $i \in \{0,1\}$), this transaction will be rejected since IsFunded does not hold: the funds of the shared account have been depleted during the Pay execution, which we have proven correct. Therefore, since CP is correct, it must hold that we have $b = 0$ if Pay takes place.

However, if Pay has not happened, none of the two ledgers have a transaction that spends from the shared accounts of $tx_{\mathsf{cnd}}0$ and $tx_{\mathsf{cnd}}1$. Therefore, if any of the parties attempts to submit a refund transaction tx^i_{ref} (for $i \in \{0,1\}$), IsFunded will hold for this transaction: the funds of the shared account have not been spent yet since Pay has not happened. Since our assumption is that IsUnique and IsValid hold, since CP is correct, it must hold that $b = 1$ if Pay does not take place.

And this concludes the proof of Theorem 17 □

E.2 Proof of Security

Theorem 1 (BL-CBDC Balance Security). *Let ledgers Π_{CP_0} and Π_{CP_1} provide CP PvP against ledger* cbl *and satisfy CP unforgeability, CP redeemability, CP refundability. Let ledger* cbl *satisfy BL unforgeability. Then, our construction for CCE depicted in Figure 8 is secure according to Definition 9.*

Proof. In Figure 18 we show the oracles for our CCE construction when cbl is of type BL. As described in Definition 9, in order to show that our protocol achieves balance security for the withdraw implementation, we need to show that:

$$\Pr[BSC^{BL}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda)=1] \leq \mathsf{negl}(\lambda)$$

$$\Pr[BSI^{BL}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda)=1] \leq \mathsf{negl}(\lambda)$$

$$\Pr[BSD^{BL}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda)=1] \leq \mathsf{negl}(\lambda)$$

Lemma 1 (Balance security for C)**.** *For all PPT adversaries $\mathcal{A}$, it holds that* $\Pr[BSC^{BL}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda)=1] \leq \mathsf{negl}(\lambda)$.

Proof. The game BSC^{BL} contains two mutually exclusive paths that the adversary $\mathcal{A}$ could exploit. We will prove Lemma 1 by performing reductions on the two possible paths since:

$$\begin{aligned}&\Pr[BSC^{BL}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda)] =\\ &\quad\Pr[BSC^{BL}-G_0^{\tau<T}(\lambda)=1 \vee BSC^{BL}-G_0^{\tau\geq T+\delta}(\lambda)=1] \leq\\ &\quad\Pr[BSC^{BL}-G_0^{\tau<T}(\lambda)=1]+\Pr[BSC^{BL}-G_0^{\tau\geq T+\delta}(\lambda)=1]\end{aligned}$$

In the path where $\tau < T$, we consider the following game:

Game $BSC^{BL} - G_0^{\tau<T}$: This game, formally defined in Figure 19, corresponds to the original game for Balance security for C restricted to the path where $\tau < T$. The game is expanded with the interactions described in our implementation.

On the path where $\tau \geq T + \delta$, we only consider the following game:

Game $BSC^{BL} - G_0^{\tau\geq T+\delta}$: This game, formally defined in Figure 20, corresponds to the original game for Balance security for C restricted to the path where $\tau \geq T + \delta$. The game is expanded with the interactions described in our implementation.

Claim 1. *Let ledger Π_{CP_0} have CP PvP against* cbl*. Then,* $\Pr[BSC^{BL} - G_0^{\tau<T}(\lambda)=1] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[BSC^{BL} - G_0^{\tau<T}(\lambda)=1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win ExpPvP. $\mathcal{B}$ interacts with both $\mathcal{A}$ and the challenger over the ledgers Π_{CP_0} and cbl.

- Challenger provides $\mathcal{B}$ with public keys $(pk^{\mathsf{CP}}_S, pk^{\mathsf{BL}}_R)$. $\mathcal{B}$ sets $pk^0_C := pk^{\mathsf{CP}}_S$ and $pk^{\mathsf{cbl}}_C := pk^{\mathsf{BL}}_R$ and generates T.
- $\mathcal{B}$ forwards $(pk^0_C, pk^{\mathsf{cbl}}_C, T)$ to $\mathcal{A}$, witch returns $(pk^0_I, pk^{\mathsf{cbl}}_D, \mathbb{C}^* := tx^{\mathsf{cbl}}_{\mathsf{bsc}})$. By assumption $\mathcal{A}$ wins the game $BSC^{BL} - G_0^{\tau<T}$, hence condition a_3 is fulfilled and $\mathcal{B}$ does not abort.
- $\mathcal{B}$ sets $pk^{\mathsf{CP}}_R := pk^0_I$, $pk^{\mathsf{BL}}_S := pk^{\mathsf{cbl}}_D$ and sets $tx^{\mathsf{BL}}_{\mathsf{bsc}} := tx^{\mathsf{cbl}}_{\mathsf{bsc}}$ as $\mathbb{C}^*$ and forwards $(pk^{\mathsf{CP}}_R, pk^{\mathsf{BL}}_S, \mathbb{C}^*, T)$ to the challenger.
- $\mathcal{B}$ engages with the challenger and $\mathcal{A}$ on protocol Π_{CP_0}.ctCnd acting as a relay. This results in $(tx^{\mathsf{CP}}_{\mathsf{cnd}}, \sigma^{\mathsf{CP}}_{\mathsf{cnd}}, \mathsf{aux}^{\mathsf{CP}}) := (tx^0_{\mathsf{cnd}}, \sigma^0_{\mathsf{cnd}}, \mathsf{aux}^0)$. Thereafter, the challenger submits $(tx^{\mathsf{CP}}_{\mathsf{cnd}}, \sigma^{\mathsf{CP}}_{\mathsf{cnd}})$ to Π_{CP_0}.
- $\mathcal{A}$ produces the forgery $\overline{tx_{\mathsf{red}}}^0$ on Π_{CP_0} and $\mathcal{B}$ forwards $\overline{tx_{\mathsf{red}}}^{\mathsf{CP}} := \overline{tx_{\mathsf{red}}}^0$ to the challenger.

If $\mathcal{A}$ makes a query to $\mathsf{Set}_C O$, $\mathcal{B}$ follows all the steps of protocol Set as described in Figure 26. However, $\mathcal{B}$ does not have the private key sk^0_C, so when Set requires this input, $\mathcal{B}$ queries $\mathsf{ctCnd}_S O$ for the output of $\Pi_{CP_0}.\mathsf{ctCnd}_S(sk^0_C, \mathbb{C}, T')$ and relays the answer. Note that Q is synchronized in both games. If $\mathcal{A}$ makes a query to $\mathsf{Pay}_C O$, $\mathcal{B}$ follows all the steps of protocol Set, as described in Figure 27.

Our adversary $\mathcal{B}$ perfectly simulates $BSC^{BL} - G_0^{\tau<T}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ can win $BSC^{BL} - G_0^{\tau<T}$ with no negligible probability, this implies that (1) $tx^0_{\mathsf{cnd}}, tx^{\mathsf{cbl}}_{\mathsf{bsc}} \notin Q$ (condition a_0); (2) $tx^0_{\mathsf{cnd}} \in \Pi_{CP_0}.TX_L$ and $\mathsf{isSender}(tx^0_{\mathsf{cnd}}, pk^0_C) = 1$ (condition a_1); (3) $\mathsf{isCond}(tx^0_{\mathsf{cnd}}, \mathbb{C}^*) = 1$ and $\mathsf{isRcvr}(tx^0_{\mathsf{cnd}}, epk^0_{C,I}) = 1$ (condition a_2); (4) $\overline{tx_{\mathsf{red}}}^0 \notin \Pi_{CP_0}.TX_L$ and $\mathsf{isLinked}(tx^0_{\mathsf{cnd}}, \overline{tx_{\mathsf{red}}}^0) = 1$ (condition b_0); (5) $\mathsf{isSender}(tx^{\mathsf{cbl}}_{\mathsf{bsc}}, pk^{\mathsf{cbl}}_D) = 1$ and $\mathsf{isRcvr}(tx^{\mathsf{cbl}}_{\mathsf{bsc}}, pk^{\mathsf{cbl}}_C) = 1$ (condition a_3). (6) $tx^{\mathsf{cbl}}_{\mathsf{bsc}} \notin \mathsf{cbl}.TX_L$ (condition b_1); It is easy to see that these are equivalent to conditions b_0, b_1, b_2, b_3, b_4 and b_5 of game ExpPvP. Furthermore, we know that Q is synchronized in both games. However, this result contradicts the assumption that Π_{CP_0} satisfies CP PvP against cbl. Therefore, such adversary $\mathcal{A}$ cannot exist, thus this claim has been proven. □

So far, we have obtained that $\Pr[BSC^{BL} - G_0^{\tau<T}(\lambda)=1] \leq \mathsf{negl}(\lambda)$. This means that for the path where $\tau < T$, $\mathcal{A}$ can win with negligible probability.

$\underline{\mathsf{Set}_C O(sk_I^0, \mathbb{C}^*, T)}$

$T' := T + \delta$

$(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0) \leftarrow \left\langle \begin{matrix} \mathsf{ctCnd}_S(sk_C^0, \mathbb{C}^*, T'), \\ \mathsf{ctCnd}_R(sk_I^0, \mathbb{C}^*, T') \end{matrix} \right\rangle$

$\mathsf{aux}_C := (\mathsf{aux}^0, tx_{\mathsf{cnd}}^0, \mathbb{C}^*)$

$Q := Q \cup \{\mathsf{aux}_C\}$

return $(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}_C)$

$\underline{O_{\mathsf{Ref}_C}(tx_{\mathsf{cnd}}^0, \mathsf{aux}_C, \ddot{pk}_C^0)}$

$\mathsf{aux}_C := (\mathsf{aux}^0, tx_{\mathsf{cnd}}^0, \mathbb{C}^*)$

$(tx_{\mathsf{ref}}^0, \sigma_{\mathsf{ref}}^0)$

$\quad \leftarrow \Pi_{CP_0}.\mathsf{Ref}(tx_{\mathsf{cnd}}^0, sk_C^0, \mathsf{aux}^0, \ddot{pk}_C^0)$

$Q := Q \cup \{tx_{\mathsf{ref}}^0\}$

return $(tx_{\mathsf{ref}}^0, \sigma_{\mathsf{ref}}^0)$

$\underline{\mathsf{Set}_I O(sk_C^0, sk_D^1, \mathbb{C}^*, T)}$

$T' := T + \delta$

$(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0) \leftarrow \left\langle \begin{matrix} \mathsf{ctCnd}_S(sk_C^0, \mathbb{C}^*, T'), \\ \mathsf{ctCnd}_R(sk_I^0, \mathbb{C}^*, T') \end{matrix} \right\rangle$

$(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1) \leftarrow \left\langle \begin{matrix} \mathsf{ctCnd}_S(sk_I^1, \mathbb{C}^*, T), \\ \mathsf{ctCnd}_R(sk_D^1, \mathbb{C}^*, T) \end{matrix} \right\rangle$

$\mathsf{aux}_I := (\mathsf{aux}^0, \mathsf{aux}^1, tx_{\mathsf{cnd}}^1, \mathbb{C}^*)$

$Q := Q \cup \{(tx_{\mathsf{cnd}}^0, \mathsf{aux}_I)\}$

return $(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}_I)$

$\underline{\mathsf{Pay}_I O(tx_{\mathsf{cnd}}^0, \mathsf{aux}_I, w^*, \ddot{pk}_I^0)}$

$\mathsf{aux}_I := (\mathsf{aux}^0, \mathsf{aux}^1, tx_{\mathsf{cnd}}^1, \mathbb{C}^*)$

$(tx_{\mathsf{red}}^0, \sigma_{\mathsf{red}}^0)$

$\quad \leftarrow \Pi_{CP_0}.\mathsf{Red}(tx_{\mathsf{cnd}}^0, sk_I^0, w^*, \mathsf{aux}^0, \ddot{pk}_I^0)$

$Q := Q \cup \{tx_{\mathsf{red}}^0\}$

return $(tx_{\mathsf{red}}^0, \sigma_{\mathsf{red}}^0)$

$\underline{\mathsf{Ref}_I O(tx_{\mathsf{cnd}}^1, \mathsf{aux}_I, \ddot{pk}_I^1)}$

$\mathsf{aux}_I := (\mathsf{aux}^0, \mathsf{aux}^1, tx_{\mathsf{cnd}}^1, \mathbb{C}^*)$

$(tx_{\mathsf{ref}}^1, \sigma_{\mathsf{ref}}^1)$

$\quad \leftarrow \Pi_{CP_1}.\mathsf{Ref}(tx_{\mathsf{cnd}}^1, sk_I^1, \mathsf{aux}^1, \ddot{pk}_I^1)$

$Q := Q \cup \{tx_{\mathsf{ref}}^1\}$

return $(tx_{\mathsf{ref}}^1, \sigma_{\mathsf{ref}}^1)$

$\underline{\mathsf{Set}_D O(sk_I^1, \mathbb{C}^*, T)}$

$(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1) \leftarrow \left\langle \begin{matrix} \mathsf{ctCnd}_S(sk_I^1, \mathbb{C}^*, T), \\ \mathsf{ctCnd}_R(sk_D^1, \mathbb{C}^*, T) \end{matrix} \right\rangle$

$\mathsf{aux}_D := (\mathsf{aux}^1, \mathbb{C}^*)$

$Q := Q \cup \{(tx_{\mathsf{cnd}}^1, \mathsf{aux}_D)\}$

return $(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}_D)$

$\underline{\mathsf{Pay}_D O(tx_{\mathsf{cnd}}^1, \mathsf{aux}_D, w^*, \ddot{pk}_D^1)}$

$\mathsf{aux}_D := (\mathsf{aux}^1, \mathbb{C}^*)$

$(tx_{\mathsf{red}}^1, \sigma_{\mathsf{red}}^1)$

$\quad \leftarrow \Pi_{CP_1}.\mathsf{Red}(tx_{\mathsf{cnd}}^1, sk_D^1, w^*, \mathsf{aux}^1, \ddot{pk}_D^1)$

$Q := Q \cup \{tx_{\mathsf{red}}^1\}$

return $(tx_{\mathsf{red}}^1, \sigma_{\mathsf{red}}^1)$

Figure 18: CCE oracles when cbl **is a BL**

We now explore the other path.

Claim 2. *Let ledger Π_{CP} provide CP refundability. Then* $\Pr[BSC^{BL} - G_0^{\tau \geq T+\delta}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that there exists a PPT $\mathcal{A}$ such that $\Pr[BSC^{BL} - G_0^{\tau \geq T+\delta}(\lambda) = 1] > \mathsf{negl}(\lambda)$. Then we can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win ExpRefund. $\mathcal{B}$ interacts with $\mathcal{A}$ over the ledgers Π_{CP_0} and cbl and with the challenger over the ledger Π_{CP_0}.

- Challenger provides $\mathcal{B}$ with public keys $(pk_S^{\mathsf{CP}}, pk_R^{\mathsf{BL}})$. $\mathcal{B}$ sets $pk_C^0 := pk_S^{\mathsf{CP}}$ and $pk_C^{\mathsf{cbl}} := pk_R^{\mathsf{BL}}$ and generates T.
- $\mathcal{B}$ forwards $(pk_C^0, pk_C^{\mathsf{cbl}}, T)$ to $\mathcal{A}$, witch returns $(pk_I^0, pk_D^{\mathsf{cbl}}, \mathbb{C}^* := tx_{\mathsf{bsc}}^{\mathsf{cbl}})$. By assumption $\mathcal{A}$ wins the game $BSC^{BL} - G_0^{\tau < T}$, hence condition a_3 is fulfilled and $\mathcal{B}$ does not abort.
- $\mathcal{B}$ sets $pk_R^{\mathsf{CP}} := pk_I^0$, $pk_S^{\mathsf{BL}} := pk_D^{\mathsf{cbl}}$ and sets $tx_{\mathsf{bsc}}^{\mathsf{BL}} := tx_{\mathsf{bsc}}^{\mathsf{cbl}}$ as $\mathbb{C}^*$, and forwards $(pk_R^{\mathsf{CP}}, pk_S^{\mathsf{BL}}, \mathbb{C}^*, T)$ to the challenger.
- $\mathcal{B}$ engages with the challenger and $\mathcal{A}$ on protocol $\Pi_{CP_0}.\mathsf{ctCnd}$ acting as a relay. This results in $(tx_{\mathsf{cnd}}^{\mathsf{CP}}, \sigma_{\mathsf{cnd}}^{\mathsf{CP}}, \mathsf{aux}^{\mathsf{CP}}) := (tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0)$. Thereafter, the challenger submits $(tx_{\mathsf{cnd}}^{\mathsf{CP}}, \sigma_{\mathsf{cnd}}^{\mathsf{CP}})$ to Π_{CP_0}.
- Finally, the challenger generates a refund transaction tx_{ref} from tx_{cnd} and submits $(tx_{\mathsf{red}}, \sigma_{\mathsf{red}})$ to Π_{CP_0}.

If $\mathcal{A}$ makes a query to $\mathsf{Set}_C O$, $\mathcal{B}$ follows all the steps of protocol Set as described in Figure 26. However, $\mathcal{B}$ does not have the private key sk_C^0, so when Set requires this input, $\mathcal{B}$ queries $\mathsf{ctCnd}_S O$ for the output of $\Pi_{CP_0}.\mathsf{ctCnd}_S(sk_C^0, \mathbb{C}, T')$ and relays the answer. If $\mathcal{A}$ makes a query to $\mathsf{Ref}_C O$, $\mathcal{B}$ does not have the private key sk_C^0, hence it queries $\mathsf{Ref} O$ and relays the answer. Note that Q is synchronized in both games. If $\mathcal{A}$ makes a query to $\mathsf{Pay}_C O$, $\mathcal{B}$ follows all the steps of protocol Set, as described in Figure 27.

Our adversary $\mathcal{B}$ perfectly simulates $BSC^{BL} - G_0 G_0^{\tau \geq T+\delta}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ is successful in wining $BSC^{BL} - G_0 G_0^{\tau \geq T+\delta}$ with non-negligible probability, this implies that (1) $tx_{\mathsf{cnd}}^0, tx_{\mathsf{bsc}}^{\mathsf{cbl}} \notin Q$ (condition a_0); (2) $tx_{\mathsf{cnd}}^0 \in \Pi_{CP_0}.TX_L$ and $\mathsf{isSender}(tx_{\mathsf{cnd}}^0, pk_C^0) = 1$ (condition a_1); (3) $\mathsf{isCond}(tx_{\mathsf{cnd}}^0, tx_{\mathsf{bsc}}^{\mathsf{cbl}}) = 1$ and $\mathsf{isRcvr}(tx_{\mathsf{cnd}}^0, epk_{C,I}^0) = 1$ (condition a_2); (4) $\mathcal{A}$ did not put $\overline{tx_{\mathsf{red}}}^0$ on Π_{CP_0} (condition c_0); (5) $\mathcal{A}$ did not use $\mathsf{Ref}_C O$ to put $\overline{tx_{\mathsf{ref}}}^0$ on Π_{CP_0} (condition c_1); (6) $tx_{\mathsf{ref}}^0 \notin \Pi_{CP_0}.TX_L$ and $\mathsf{isLinked}(tx_{\mathsf{cnd}}^0, tx_{\mathsf{ref}}^0) = 1$ (condition c_2); (7) timeout $T + \delta$ has expired (condition c_3). It is easy to see that these are equivalent to conditions b_0, b_1, b_2, b_3, b_4, b_5 and b_6 of game ExpRefund. Furthermore, we know that the two Q are synchronized. However, this result contradicts the assumption that Π_{CP_0} satisfies CP refundability. Therefore, such $\mathcal{A}$ cannot exist and this claim has been proven. □

$BSC^{BL}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}} - G_0^{\tau<T}(\lambda)$

$\mathcal{Q} := \emptyset$

$(pk^0_C, sk^0_C) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$

$(pk^{\mathsf{cbl}}_C, sk^{\mathsf{cbl}}_C) \leftarrow \mathsf{cbl}.\mathsf{ctAcc}(1^\lambda)$

$T \leftarrow \mathtt{createT}(\cdot)$

$T' := T + \delta$

$(pk^0_I, pk^{\mathsf{cbl}}_D, \mathbb{C}^* := tx^{\mathsf{cbl}}_{\mathsf{bsc}}, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathrm{Set}_C O, \mathrm{Pay}_C O}(pk^0_C, pk^{\mathsf{cbl}}_C, T)$

if $\neg a_3$ **abort**

$(tx^0_{\mathsf{cnd}}, \sigma^0_{\mathsf{cnd}}, \mathsf{aux}^0, \mathsf{st}_1) \leftarrow \left\langle \begin{matrix} \mathsf{ctCnd}_S(sk^0_C, \mathbb{C}^*, T'), \\ \mathcal{A}(\mathsf{st}_0) \end{matrix} \right\rangle$

$\mathsf{subTx}(tx^0_{\mathsf{cnd}}, \sigma^0_{\mathsf{cnd}})$

$\overline{tx_{\mathsf{red}}}^0 \leftarrow \mathcal{A}(\mathsf{st}_1)$

$a_0 := tx^0_{\mathsf{cnd}}, tx^{\mathsf{cbl}}_{\mathsf{bsc}} \notin \mathcal{Q}$

$a_1 := \Pi_{CP_0}.\mathsf{ckTx}(tx^0_{\mathsf{cnd}}) \wedge \mathsf{isSender}(tx^0_{\mathsf{cnd}}, pk^0_C)$

$a_2 := \mathsf{isCond}(tx^0_{\mathsf{cnd}}, \mathbb{C}^*) \wedge \mathsf{isRcvr}(tx^0_{\mathsf{cnd}}, epk^0_{C,I})$

$a_3 := \mathsf{isSender}(tx^{\mathsf{cbl}}_{\mathsf{bsc}}, pk^{\mathsf{cbl}}_D) \wedge \mathsf{isRcvr}(tx^{\mathsf{cbl}}_{\mathsf{bsc}}, pk^{\mathsf{cbl}}_C)$

$b_0 := \Pi_{CP_0}.\mathsf{ckTx}(\overline{tx_{\mathsf{red}}}^0) \wedge \mathsf{isLinked}(tx^0_{\mathsf{cnd}}, \overline{tx_{\mathsf{red}}}^0)$

$b_1 := \mathsf{cbl}.\mathsf{ckTx}(tx^{\mathsf{cbl}}_{\mathsf{bsc}}) = 0$

$b_2 := \mathtt{readTime}(\cdot) < T$

return $\bigwedge_{i=0}^{3} a_i \wedge \bigwedge_{i=0}^{2} b_i$

Figure 19: Experiment for $BSC^{BL} - G_0^{\tau<T}(\lambda)$.

Therefore, we reach to:

$$\begin{aligned}
&\Pr[BSC^{BL}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda)] \\
&\quad = \Pr[BSC^{BL} - G_0^{\tau<T}(\lambda) = 1 \vee BSC^{BL} - G_0^{\tau\geq T+\delta}(\lambda) = 1] \\
&\quad \leq \Pr[BSC^{BL} - G_0^{\tau<T}(\lambda) = 1] + \Pr[BSC^{BL} - G_0^{\tau\geq T+\delta}(\lambda) = 1] \\
&\quad \leq \mathsf{negl}(\lambda).
\end{aligned}$$

This concludes the proof of Lemma 1. □

Lemma 2. *(Balance security for I) For all PPT adversaries $\mathcal{A}$, it holds that* $\Pr[BSI^{BL}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$

Proof. The game BSI^{BL} contains two mutually exclusive paths that the adversary $\mathcal{A}$ could exploit. We will prove Lemma 2 by performing reductions on the two possible paths since:

$$\begin{aligned}
&\Pr[BSI^{BL}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda) = 1] = \\
&\qquad \Pr[BSI^{BL} - G_0^{\tau<T}(\lambda) = 1 \vee BSI^{BL} - G_0^{\tau\geq T}(\lambda) = 1] \leq \\
&\qquad \Pr[BSI^{BL} - G_0^{\tau<T}(\lambda) = 2] + \Pr[BSI^{BL} - G_0^{\tau\geq T}(\lambda) = 1]
\end{aligned}$$

In the path whare $\tau < T$, we perform the following game hops:

Game $BSI^{BL} - G_0^{\tau<T}$: This game, formally defined in Figure 21, corresponds to the original game for Balance security for I restricted to the path where $\tau < T$. The game is expanded with the interactions described in our implementation.

Game $BSI^{BL} - G_1^{\tau<T}$: This game, formally defined in Figure 22, works exactly as G_0 with the following exception (highlighted in the gray lines). At the start of the protocol, $\mathcal{A}$ produces a forgery

$BSC^{BL}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}} - G_0^{\tau\geq T+\delta}(\lambda)$

$\mathcal{Q} := \emptyset$

$(pk^0_C, sk^0_C) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$

$(pk^{\mathsf{cbl}}_C, sk^{\mathsf{cbl}}_C) \leftarrow \mathsf{cbl}.\mathsf{ctAcc}(1^\lambda)$

$T \leftarrow \mathtt{createT}(\cdot)$

$T' := T + \delta$

$(pk^0_I, pk^{\mathsf{cbl}}_D, \mathbb{C}^* := tx^{\mathsf{cbl}}_{\mathsf{bsc}}, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathrm{Set}_C O, \mathrm{Pay}_C O}(pk^0_C, pk^{\mathsf{cbl}}_C, T)$

if $\neg a_3$ **abort**

$(tx^0_{\mathsf{cnd}}, \sigma^0_{\mathsf{cnd}}, \mathsf{aux}^0) \leftarrow \left\langle \begin{matrix} \mathsf{ctCnd}_S(sk^0_C, \mathbb{C}^*, T'), \\ \mathcal{A}(\mathsf{st}_0) \end{matrix} \right\rangle$

$\mathsf{subTx}(tx^0_{\mathsf{cnd}}, \sigma^0_{\mathsf{cnd}})$

$(tx^0_{\mathsf{ref}}, \sigma^0_{\mathsf{ref}}) \leftarrow \Pi_{CP_0}.\mathsf{Ref}(tx^0_{\mathsf{cnd}}, sk^0_C, \mathsf{aux}^0, \ddot{pk}^0_C)$

$\Pi_{CP_0}.\mathsf{subTx}(tx^0_{\mathsf{ref}}, \sigma^0_{\mathsf{ref}})$

$a_0 := tx^0_{\mathsf{cnd}}, tx^{\mathsf{cbl}}_{\mathsf{bsc}} \notin \mathcal{Q}$

$a_1 := \Pi_{CP_0}.\mathsf{ckTx}(tx^0_{\mathsf{cnd}}) \wedge \mathsf{isSender}(tx^0_{\mathsf{cnd}}, pk^0_C)$

$a_2 := \mathsf{isCond}(tx^0_{\mathsf{cnd}}, tx^{\mathsf{cbl}}_{\mathsf{bsc}}) \wedge \mathsf{isRcvr}(tx^0_{\mathsf{cnd}}, epk^0_{C,I})$

$a_3 := \mathsf{isSender}(tx^{\mathsf{cbl}}_{\mathsf{bsc}}, pk^{\mathsf{cbl}}_D) \wedge \mathsf{isRcvr}(tx^{\mathsf{cbl}}_{\mathsf{bsc}}, pk^{\mathsf{cbl}}_C)$

$c_0 := \nexists\, \overline{tx_{\mathsf{red}}}^0\ s.t.\ \mathsf{ckTx}(\overline{tx_{\mathsf{red}}}^0) \wedge \mathsf{isLinked}(tx^0_{\mathsf{cnd}}, \overline{tx_{\mathsf{red}}}^0)$

$c_1 := \nexists\, \overline{tx_{\mathsf{ref}}}^0\ s.t.\ \overline{tx_{\mathsf{ref}}}^0 \in \mathcal{Q} \wedge \mathsf{ckTx}(\overline{tx_{\mathsf{ref}}}^0) \wedge$
$\quad \mathsf{isLinked}(tx^0_{\mathsf{cnd}}, \overline{tx_{\mathsf{ref}}}^0)$

$c_2 := \Pi_{CP_0}.\mathsf{ckTx}(tx^0_{\mathsf{ref}}) = 0 \wedge \mathsf{isLinked}(tx^0_{\mathsf{cnd}}, tx^0_{\mathsf{ref}})$

$c_3 := \mathtt{readTime}(\cdot) \geq T + \delta$

return $\bigwedge_{i=0}^{3} a_i \wedge \bigwedge_{i=0}^{3} c_i$

Figure 20: Experiment for $BSC^{BL} - G_0^{\Pi_{CP_0}.\tau\geq T+\delta}(\lambda)$.

$\overline{tx_{\mathsf{cnd}}}^1$ impersonating C. This allows $\mathcal{A}$ to lock C's funds on Π_{CP_1}, without locking its own funds on Π_{CP_0}. If this forgery is valid, G_1 aborts.

On the path where $\tau \geq T$, we only consider the following game:

Game $BSI^{BL} - G_0^{\tau\geq T}$: This game, formally defined in Figure 23, corresponds to the original game for Balance security for I, restricted to the path where $\tau \geq T$. The game is expanded with the interactions described in our implementation.

Claim 3. *Let* Bad_1 *be the event that* $BSI^{BL} - G_1^{\tau<T}$ *aborts on gray lines of Figure 22. Assume that ledger* Π_{CP_1} *provides CP unforgeability, 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 $\mathcal{B}$ that uses $\mathcal{A}$ to win cndForge. $\mathcal{B}$ interacts with $\mathcal{A}$ over the ledgers Π_{CP_0} and Π_{CP_1} and with the challenger over the ledger Π_{CP_1}.

- Challenger provides $\mathcal{B}$ with a sender public key pk_S. $\mathcal{B}$ sets $pk^1_I := pk_S$ and generates (pk^0_I, sk^0_I) and $(\ddot{pk}^0_C, \ddot{sk}^0_C)$.
- $\mathcal{B}$ forwards $(pk^0_I, pk^1_I, \ddot{sk}^0_C)$ to $\mathcal{A}$, which returns $(pk^0_C, pk^1_D, w^* := tx^{\mathsf{cbl}}_{\mathsf{bsc}}, T)$ and $\overline{tx_{\mathsf{cnd}}}^1$, the latter being a forgery on Π_{CP_1}.

$BSI^{BL}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}} - G_0^{\tau<T}(\lambda)$

$\mathcal{Q} := \emptyset$

$(pk_I^0, sk_I^0) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$

$(\ddot{pk}_I^0, \ddot{sk}_I^0) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$

$(pk_I^1, sk_I^1) \leftarrow \Pi_{CP_1}.\mathsf{ctAcc}(1^\lambda)$

$(pk_C^0, pk_D^1, \mathbb{C}^* := tx_{\mathsf{bsc}}^{\mathsf{cbl}}, T, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{Set}_I O, \mathsf{Pay}_I O}(pk_I^0, pk_I^1, \ddot{pk}_I^0)$

$T' := T + \delta$

$(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0, \mathsf{st}_1) \leftarrow \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_0), \\ \Pi_{CP_0}.\mathsf{ctCnd}_R(sk_I^0, \mathbb{C}^*, T') \end{matrix} \right\rangle$

if $\neg(a_1 \wedge a_2)$ **abort**

$(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1, \mathsf{st}_2) \leftarrow \left\langle \begin{matrix} \Pi_{CP_1}.\mathsf{ctCnd}_S(sk_I^1, \mathbb{C}^*, T), \\ \mathcal{A}(\mathsf{st}_1) \end{matrix} \right\rangle$

$\Pi_{CP_1}.\mathsf{subTx}(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1)$

$tx_{\mathsf{red}}1 \leftarrow \mathcal{A}(\mathsf{st}_2)$

if $\neg b_3$ **abort**

$(tx_{\mathsf{red}}^0, \sigma_{\mathsf{red}}^0) \leftarrow \Pi_{CP_0}.\mathsf{Red}(tx_{\mathsf{cnd}}^0, sk_I^0, w^*, \mathsf{aux}^0, \ddot{pk}_I^0)$

$\Pi_{CP_0}.\mathsf{subTx}(tx_{\mathsf{red}}^0, \sigma_{\mathsf{red}}^0)$

$a_0 := tx_{\mathsf{cnd}}^0, tx_{\mathsf{cnd}}^1, tx_{\mathsf{bsc}}^{\mathsf{cbl}} \notin \mathcal{Q}$

$a_1 := \Pi_{CP_0}.\mathsf{ckTx}(tx_{\mathsf{cnd}}^0) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^0, pk_C^0)$

$a_2 := \mathsf{isCond}(tx_{\mathsf{cnd}}^0, \mathbb{C}^*) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^0, epk_{C,I}^0)$

$a_3 := \Pi_{CP_1}.\mathsf{ckTx}(tx_{\mathsf{cnd}}^1) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^1, pk_I^1)$

$a_4 := \mathsf{isCond}(tx_{\mathsf{cnd}}^1, \mathbb{C}^*) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^1, epk_{I,D}^1)$

$b_0 := \nexists\, \overline{tx_{\mathsf{red}}}^0 \; s.t. \; \overline{tx_{\mathsf{red}}}^0 \in \mathcal{Q} \wedge \mathsf{ckTx}(\overline{tx_{\mathsf{red}}}^0) \wedge$
$\quad \mathsf{isLinked}(tx_{\mathsf{cnd}}^0, \overline{tx_{\mathsf{red}}}^0)$

$b_1 := \Pi_{CP_0}.\mathsf{ckTx}(tx_{\mathsf{red}}^0) = 0 \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^0, tx_{\mathsf{red}}^0)$

$b_2 := \Pi_{CP_1}.\mathsf{ckTx}(tx_{\mathsf{red}}^1) \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^1, tx_{\mathsf{red}}^1)$

$b_3 := \mathsf{cbl}.\mathsf{ckTx}(tx_{\mathsf{bsc}}^{\mathsf{cbl}})$

$b_4 := \mathtt{readTime}(\cdot) < T$

return $\bigwedge_{i=0}^{4} a_i \wedge \bigwedge_{i=0}^{4} b_i$

Figure 21: Experiment for $BSI^{BL} - G_0^{\tau<T}(\lambda)$.

- $\mathcal{B}$ forwards $(tx_{\mathrm{i}} := \overline{tx_{\mathsf{cnd}}}^1, pk_S := pk_D^1, \mathbb{C} := \mathbb{C}^*, T)$ to challenger.

If $\mathcal{A}$ makes a query to $\mathsf{Set}_I O$, $\mathcal{B}$ follows all the steps of protocol Set as described in Figure 26. However, $\mathcal{B}$ does not have the private key sk_I^1, so when Set requires this input, $\mathcal{B}$ queries $\mathsf{ctCnd}_S O$ for the output of $\Pi_{CP_1}.\mathsf{ctCnd}_S(sk_I^1, \mathbb{C}, T)$ and relays the answer. Note that $\mathcal{Q}$ is synchronized in both games. If $\mathcal{A}$ makes a query to $\mathsf{Pay}_I O$, $\mathcal{B}$ follows all the steps of protocol Set, as described in Figure 27.

Our adversary $\mathcal{B}$ perfectly simulates $BSI^{BL} - G_1^{\tau<T}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ can force $\mathcal{B}$ to abort on the gray lines of Figure 22, it implies that (1) $\overline{tx_{\mathsf{cnd}}}^1 \notin \mathcal{Q}$ (condition $\hat{c}_0$); (2) $\overline{tx_{\mathsf{cnd}}}^1 \in \Pi_{CP_1}.TX_L$ and $\mathsf{isSender}(\overline{tx_{\mathsf{cnd}}}^1, pk_C^1) = 1$ (condition $\hat{c}_1$); (3) $\mathsf{isCond}(\overline{tx_{\mathsf{cnd}}}^1, \mathbb{C}^*) = 1$ (condition $\hat{c}_2$). It is easy to see that these are equivalent to conditions b_0, b_1 and b_2 of game cndForge. Furthermore, we know that $\mathcal{Q}$ is synchronized in both

$BSI^{BL}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}} - G_1^{\tau<T}(\lambda)$

$\mathcal{Q} := \emptyset$

$(pk_I^0, sk_I^0) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$

$(\ddot{pk}_I^0, \ddot{sk}_I^0) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$

$(pk_I^1, sk_I^1) \leftarrow \Pi_{CP_1}.\mathsf{ctAcc}(1^\lambda)$

$(pk_C^0, pk_D^1, \mathbb{C}^* := tx_{\mathsf{bsc}}^{\mathsf{cbl}}, T, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{Set}_I O, \mathsf{Pay}_I O}(pk_I^0, pk_I^1, \ddot{pk}_I^0)$

$T' := T + \delta$

$\overline{tx_{\mathsf{cnd}}}^1 \leftarrow \mathcal{A}(\mathsf{st}_0)$

$\hat{c}_0 := \overline{tx_{\mathsf{cnd}}}^1 \notin \mathcal{Q}$

$\hat{c}_1 := \Pi_{CP_1}.\mathsf{ckTx}(\overline{tx_{\mathsf{cnd}}}^1) \wedge \mathsf{isSender}(\overline{tx_{\mathsf{cnd}}}^1, pk_I^1)$

$\hat{c}_2 := \mathsf{isCond}(\overline{tx_{\mathsf{cnd}}}^1, \mathbb{C}^*)$

if $\hat{c}_0 \wedge \hat{c}_1 \wedge \hat{c}_2$ **abort**

$(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0, \mathsf{st}_1) \leftarrow \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_0), \\ \Pi_{CP_0}.\mathsf{ctCnd}_R(sk_I^0, \mathbb{C}^*, T') \end{matrix} \right\rangle$

if $\neg(a_1 \wedge a_2)$ **abort**

$(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1, \mathsf{st}_2) \leftarrow \left\langle \begin{matrix} \Pi_{CP_1}.\mathsf{ctCnd}_S(sk_I^1, \mathbb{C}^*, T), \\ \mathcal{A}(\mathsf{st}_1) \end{matrix} \right\rangle$

$\Pi_{CP_1}.\mathsf{subTx}(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1)$

$tx_{\mathsf{red}}1 \leftarrow \mathcal{A}(\mathsf{st}_2)$

if $\neg b_3$ **abort**

$(tx_{\mathsf{red}}^0, \sigma_{\mathsf{red}}^0) \leftarrow \Pi_{CP_0}.\mathsf{Red}(tx_{\mathsf{cnd}}^0, sk_I^0, w^*, \mathsf{aux}^0, \ddot{pk}_I^0)$

$\Pi_{CP_0}.\mathsf{subTx}(tx_{\mathsf{red}}^0, \sigma_{\mathsf{red}}^0)$

$a_0 := tx_{\mathsf{cnd}}^0, tx_{\mathsf{cnd}}^1, tx_{\mathsf{bsc}}^{\mathsf{cbl}} \notin \mathcal{Q}$

$a_1 := \Pi_{CP_0}.\mathsf{ckTx}(tx_{\mathsf{cnd}}^0) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^0, pk_C^0)$

$a_2 := \mathsf{isCond}(tx_{\mathsf{cnd}}^0, \mathbb{C}^*) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^0, epk_{C,I}^0)$

$a_3 := \Pi_{CP_1}.\mathsf{ckTx}(tx_{\mathsf{cnd}}^1) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^1, pk_I^1)$

$a_4 := \mathsf{isCond}(tx_{\mathsf{cnd}}^1, \mathbb{C}^*) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^1, epk_{I,D}^1)$

$b_0 := \nexists\, \overline{tx_{\mathsf{red}}}^0 \; s.t. \; \overline{tx_{\mathsf{red}}}^0 \in \mathcal{Q} \wedge \mathsf{ckTx}(\overline{tx_{\mathsf{red}}}^0) \wedge$
$\quad \mathsf{isLinked}(tx_{\mathsf{cnd}}^0, \overline{tx_{\mathsf{red}}}^0)$

$b_1 := \Pi_{CP_0}.\mathsf{ckTx}(tx_{\mathsf{red}}^0) = 0 \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^0, tx_{\mathsf{red}}^0)$

$b_2 := \Pi_{CP_1}.\mathsf{ckTx}(tx_{\mathsf{red}}^1) \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^1, tx_{\mathsf{red}}^1)$

$b_3 := \mathsf{cbl}.\mathsf{ckTx}(tx_{\mathsf{bsc}}^{\mathsf{cbl}})$

$b_4 := \mathtt{readTime}(\cdot) < T$

return $\bigwedge_{i=0}^{4} a_i \wedge \bigwedge_{i=0}^{4} b_i$

Figure 22: Experiment for $BSI^{BL} - G_1^{\tau<T}(\lambda)$.

games. However, this result contradicts the assumption that Π_{CP_1} satisfies CP unforgeability. Therefore, such adversary $\mathcal{A}$ cannot exist, thus this claim has been proven. □

Since games $BSI^{BL} - G_0^{\tau<T}$ and $BSI^{BL} - G_1^{\tau<T}$ are equivalent except for event Bad_1 occuring, it holds that

$$\Pr[BSI^{BL} - G_0^{\tau<T}(\lambda) = 1] \leq \Pr[G_1^{\tau<T}(\lambda) = 1] + \mathsf{negl}(\lambda)$$

Claim 4. *Let ledger Π_{CP_0} provide CP redeemability. Then* $\Pr[BSI^{BL} - G_1^{\tau<T}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[BSI^{BL} - G_1^{\tau<T}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win ExpRedeem. $\mathcal{B}$ interacts with $\mathcal{A}$ over the ledgers Π_{CP_0} and Π_{CP_1} and with the challenger over the ledger Π_{CP_0}.

- Challenger provides $\mathcal{B}$ with public keys $(pk_R, \ddot{pk})$. $\mathcal{B}$ sets $pk_I^0 := pk_R$, $\ddot{pk}_I^0 := \ddot{pk}$ and generates (pk_I^1, sk_I^1).
- $\mathcal{B}$ forwards $(pk_I^0, pk_I^1, \ddot{sk}_I^0)$ to $\mathcal{A}$, which returns $(pk_C^0, pk_D^1, \mathbb{C}^* := tx_{\mathsf{bsc}}^{\mathsf{cbl}}, T)$ and $\overline{tx_{\mathsf{cnd}}}^1$, the latter being a forgery on Π_{CP_1}. Due to $\Pr[\mathsf{Bad}_1] \leq \mathsf{negl}(\lambda)$, this forgery does not make the game to abort.
- $\mathcal{B}$ sets $pk_S := pk_C^0$, $\mathbb{C} := \mathbb{C}^*$, $T' := T + \delta$ and forwards $(pk_S, \mathbb{C}, T')$ to the challenger.
- $\mathcal{B}$ engages with the challenger and $\mathcal{A}$ on protocol $\Pi_{CP_0}.\mathsf{ctCnd}$ acting as a relay. This results in $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux}) := (tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0)$. By assumption $\mathcal{A}$ wins the game $BSI^{BL} - G_1^{\tau<T}$, hence conditions a_1 and a_2 are fulfilled and $\mathcal{B}$ does not abort.
- $\mathcal{B}$ engages with $\mathcal{A}$ on protocol $\Pi_{CP_1}.\mathsf{ctCnd}$ that outputs the tuple $(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1)$. Henceforth, $\mathcal{B}$ submits the pair $(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1)$ to Π_{CP_1}.
- $\mathcal{A}$ outputs tx_{red}^1. By assumption $\mathcal{A}$ wins the game $BSI^{BL} - G_1^{\tau<T}$, hence condition b_3 is fulfilled and $\mathcal{B}$ does not abort. Henceforth, $\mathcal{B}$ forwards $w := w^*$ to the challenger.
- Finally, the challenger uses witness w to output $(tx_{\mathsf{red}}, \sigma_{\mathsf{red}})$ and submits it to Π_{CP_0}.

If $\mathcal{A}$ makes a query to $\mathsf{Set}_C\mathcal{O}$, $\mathcal{B}$ follows all the steps of protocol Set as described in Figure 26. However, $\mathcal{B}$ does not have the private key sk_I^0, so when Set requires this input, $\mathcal{B}$ queries $\mathsf{ctCnd}_R\mathcal{O}$ for the output of $\Pi_{CP_0}.\mathsf{ctCnd}_R(sk_I^0, \mathbb{C}, T')$ and relays the answer. Similarly, when $\mathcal{A}$ makes a query to $\mathsf{Pay}_C\mathcal{O}$, $\mathcal{B}$ follows all the steps of protocol Pay as described in Figure 27. Nevertheless, $\mathcal{B}$ does not have the private key sk_I^0, hence it queries $\mathsf{Red}\mathcal{O}$ for the output of $\Pi_{CP_0}.\mathsf{Red}(tx_{\mathsf{cnd}}^0, sk_I^0, w^*, \mathsf{aux}^0, \ddot{pk}_I^0)$ and relays the answer. Note that $\mathcal{Q}$ is synchronized in both games.

Our adversary $\mathcal{B}$ perfectly simulates $BSI^{BL} - G_1^{\tau<T}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ can win $BSI^{BL} - G_1^{\tau<T}$ with non-negligible probability, this implies that (1) $tx_{\mathsf{cnd}}^0 \notin \mathcal{Q}$ (condition a_0); (2) $tx_{\mathsf{cnd}}^0 \in \Pi_{CP_0}.TX_L$ and $\mathsf{isSender}(tx_{\mathsf{cnd}}^0, pk_C^0) = 1$ (condition a_1); (3) $\mathsf{isCond}(tx_{\mathsf{cnd}}^0, \mathbb{C}^*) = 1$ and $\mathsf{isRcvr}(tx_{\mathsf{cnd}}^0, epk_{C,I}^0) = 1$ (condition a_2); (4) (5) $\mathcal{A}$ did not use $\mathsf{Pay}_I\mathcal{O}$ to put $\overline{tx_{\mathsf{red}}}^0$ on Π_{CP_0} (condition b_0); (6) $tx_{\mathsf{red}}^0 \notin \Pi_{CP_0}.TX_L$ and $\mathsf{isLinked}(tx_{\mathsf{cnd}}^0, tx_{\mathsf{red}}^0) = 1$ (condition b_1); (7) $(\mathbb{C}^*, w^*) \in \mathcal{R}$ (CP PvP); (8) timeout T has not expired (condition b_3). It is easy to see that these are equivalent to conditions $b_0, b_1, b_2, b_3, b_4, b_5$ and b_6 of game ExpRedeem. Furthermore, we know that $\mathcal{Q}$ is synchronized in both games. However, this result contradicts the assumption that Π_{CP} satisfies CP redeemability. Therefore, such $\mathcal{A}$ cannot exist, thus this claim has been proven. □

So far, we have obtained that

$$\Pr[BSI^{BL} - G_0^{\tau<T}(\lambda) = 1] \leq \Pr[BSI^{BL} - G_1^{\tau<T}(\lambda) = 1] + \Pr[\mathsf{Bad}_1] \leq \Pr[BSI^{BL} - G_1^{\tau<T}(\lambda) = 1] + \Pr[\mathsf{Bad}_2] + \Pr[\mathsf{Bad}_1] \leq \mathsf{negl}(\lambda).$$

This means that for the path where $\tau < T$, $\mathcal{A}$ can win with negligible probability.

We now explore the other path.

Claim 5. *Let ledger Π_{CP_1} provide CP refundability. Then* $\Pr[BSI^{BL} - G_0^{\tau\geq T}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[BSI^{BL} - G_0^{\tau\geq T}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win ExpRefund. $\mathcal{B}$ interacts with $\mathcal{A}$ over the ledgers Π_{CP_0} and Π_{CP_1} and with the challenger over the ledger Π_{CP_1}.

- Challenger provides $\mathcal{B}$ with public keys $(pk_S, \ddot{pk})$. $\mathcal{B}$ sets $pk_I^1 := pk_S$, $\ddot{pk}_I^1 := \ddot{pk}$ and generates (pk_I^0, sk_I^0).
- $\mathcal{B}$ forwards $(pk_I^0, pk_I^1, \ddot{pk}_I^1)$ to $\mathcal{A}$, which returns $(pk_C^0, pk_D^1, \mathbb{C}^* := tx_{\mathsf{bsc}}^{\mathsf{cbl}}, T)$.
- $\mathcal{B}$ sets $pk_R := pk_D^1$, $\mathbb{C} := \mathbb{C}^*$ and forwards $(pk_R, \mathbb{C}, T)$ to the challenger.
- $\mathcal{B}$ engages with $\mathcal{A}$ on protocol $\Pi_{CP_0}.\mathsf{ctCnd}$ that outputs the tuple $(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0)$. By assumption $\mathcal{A}$ wins the game $BSI^{BL} - G_0^{\tau\geq T}$, hence conditions a_1 and a_2 are fulfilled and $\mathcal{B}$ does not abort.
- $\mathcal{B}$ engages with the challenger and $\mathcal{A}$ on protocol $\Pi_{CP_1}.\mathsf{ctCnd}$ acting as a relay. This results in $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux}) := (tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1)$. Thereafter, the challenger submits $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}})$ to Π_{CP_1}.
- Finally, the challenger generates a refund transaction tx_{ref} from tx_{cnd} and submits $(tx_{\mathsf{red}}, \sigma_{\mathsf{red}})$ to Π_{CP_1}.

If $\mathcal{A}$ makes a query to $\mathsf{Set}_I\mathcal{O}$, $\mathcal{B}$ follows all the steps of protocol Set as described in Figure 26. However, $\mathcal{B}$ does not have the private key sk_I^1, so when Set requires this input, $\mathcal{B}$ queries $\mathsf{ctCnd}_S\mathcal{O}$ for the output of $\Pi_{CP_1}.\mathsf{ctCnd}_S(sk_I^1, \mathbb{C}, T)$ and relays the answer. If $\mathcal{A}$ makes a query to $\mathsf{Ref}_I\mathcal{O}$, $\mathcal{B}$ does not have the private key sk_I^1, hence it queries $\mathsf{Ref}\mathcal{O}$ and relays the answer. Note that $\mathcal{Q}$ is synchronized in both games.

Our adversary $\mathcal{B}$ perfectly simulates $BSI^{BL} - G_0^{\tau\geq T}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ is successful in wining $BSI^{BL} - G_0^{\tau\geq T}$ with non-negligible probability, this implies that (1) $tx_{\mathsf{cnd}}^1 \notin \mathcal{Q}$ (condition a_0); (2) $tx_{\mathsf{cnd}}^1 \in \Pi_{CP_1}.TX_L$ and $\mathsf{isSender}(tx_{\mathsf{cnd}}^1, pk_I^1) = 1$ (condition a_3); (3) $\mathsf{isCond}(tx_{\mathsf{cnd}}^1, \mathbb{C}^*) = 1$ and $\mathsf{isRcvr}(tx_{\mathsf{cnd}}^1, epk_{I,D}^1) = 1$ (condition a_4); (4) $\mathcal{A}$ did not put $\overline{tx_{\mathsf{ref}}}^1$ on Π_{CP_1} (condition c_0); (5) $\mathcal{A}$ did not use $\mathsf{Ref}_I\mathcal{O}$ to put $\overline{tx_{\mathsf{ref}}}^1$ on Π_{CP_1} (condition c_1); (6) $tx_{\mathsf{ref}}^1 \notin \Pi_{CP_1}.TX_L$ and $\mathsf{isLinked}(tx_{\mathsf{cnd}}^1, tx_{\mathsf{ref}}^1) = 1$ (condition c_2); (7) timeout T has expired (condition c_3). , It is easy to see that these are equivalent to conditions $b_0, b_1, b_2, b_3, b_4, b_5$ and b_6 of game ExpRefund. Furthermore, we know that $\mathcal{Q}$ is synchronized in both games. However, this result contradicts the assumption that Π_{CP_1} satisfies CP refundability. Therefore, such $\mathcal{A}$ cannot exist, thus this claim has been proven. □

$BSI^{BL}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}} - G_0^{\tau \geq T}(\lambda)$

$\mathcal{Q} := \emptyset$

$(pk_I^0, sk_I^0) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$

$(\ddot{pk}_I^0, \ddot{sk}_I^0) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$

$(pk_I^1, sk_I^1) \leftarrow \Pi_{CP_1}.\mathsf{ctAcc}(1^\lambda)$

$(pk_C^0, pk_D^1, \mathbb{C}^* := tx_{\mathsf{bsc}}^{\mathsf{cbl}}, T, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{Set}_I O, \mathsf{Pay}_I O}(pk_I^0, pk_I^1, \ddot{pk}_I^0)$

$T' := T + \delta$

$(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0, \mathsf{st}_1) \leftarrow \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_0), \\ \Pi_{CP_0}.\mathsf{ctCnd}_R(sk_I^0, \mathbb{C}^*, T') \end{matrix} \right\rangle$

if $\neg(a_1 \wedge a_2)$ **abort**

$(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1) \leftarrow \left\langle \begin{matrix} \Pi_{CP_1}.\mathsf{ctCnd}_S(sk_I^1, \mathbb{C}^*, T), \\ \mathcal{A}(\mathsf{st}_1) \end{matrix} \right\rangle$

$\Pi_{CP_1}.\mathsf{subTx}(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1)$

$(tx_{\mathsf{ref}}^1, \sigma_{\mathsf{ref}}^1) \leftarrow \Pi_{CP_1}.\mathsf{Ref}(tx_{\mathsf{cnd}}^1, sk_I^1, \mathsf{aux}^1, \ddot{pk}_I^1)$

$\Pi_{CP_1}.\mathsf{subTx}(tx_{\mathsf{ref}}^1, \sigma_{\mathsf{ref}}^1)$

$a_0 := tx_{\mathsf{cnd}}^0, tx_{\mathsf{cnd}}^1, tx_{\mathsf{bsc}}^{\mathsf{cbl}} \notin \mathcal{Q}$

$a_1 := \Pi_{CP_0}.\mathsf{ckTx}(tx_{\mathsf{cnd}}^0) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^0, pk_C^0)$

$a_2 := \mathsf{isCond}(tx_{\mathsf{cnd}}^0, \mathbb{C}^*) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^0, epk_{C,I}^0)$

$a_3 := \Pi_{CP_1}.\mathsf{ckTx}(tx_{\mathsf{cnd}}^1) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^1, pk_I^1)$

$a_4 := \mathsf{isCond}(tx_{\mathsf{cnd}}^1, \mathbb{C}^*) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^1, epk_{I,D}^1)$

$c_0 := \nexists\ \overline{tx_{\mathsf{red}}}^1\ s.t.\ \mathsf{ckTx}(\overline{tx_{\mathsf{red}}}^1) \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^1, \overline{tx_{\mathsf{red}}}^1)$

$c_1 := \nexists\ \overline{tx_{\mathsf{ref}}}^1\ s.t.\ \overline{tx_{\mathsf{ref}}}^1 \in \mathcal{Q} \wedge \mathsf{ckTx}(\overline{tx_{\mathsf{ref}}}^1) \wedge$
$\mathsf{isLinked}(tx_{\mathsf{cnd}}^1, \overline{tx_{\mathsf{ref}}}^1)$

$c_2 := \Pi_{CP_1}.\mathsf{ckTx}(tx_{\mathsf{ref}}^1) = 0 \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^1, tx_{\mathsf{ref}}^1)$

$c_3 := \mathtt{readTime}(\cdot) \geq T$

return $\bigwedge_{i=0}^{4} a_i \wedge \bigwedge_{i=0}^{3} c_i$

Figure 23: Experiment for $BSI^{BL} - G_0^{\tau \geq T}(\lambda)$.

$BSD^{BL}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}} - G_0^{\tau < T}(\lambda)$

$\mathcal{Q} := \emptyset$

$(pk_D^1, sk_D^1) \leftarrow \Pi_{CP_1}.\mathsf{ctAcc}(1^\lambda)$

$(\ddot{pk}_D^1, \ddot{sk}_D^1) \leftarrow \Pi_{CP_1}.\mathsf{ctAcc}(1^\lambda)$

$(pk_D^{\mathsf{cbl}}, sk_D^{\mathsf{cbl}}) \leftarrow \mathsf{cbl}.\mathsf{ctAcc}(1^\lambda)$

$(tx_{\mathsf{bsc}}^{\mathsf{cbl}}, \sigma_{\mathsf{bsc}}^{\mathsf{cbl}}) \leftarrow \mathsf{cbl}.\mathsf{ctTx}(sk_D^{\mathsf{cbl}}, pk_C^{\mathsf{cbl}})$

$(pk_I^1, pk_C^{\mathsf{cbl}}, T, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{Set}_D O, \mathsf{Pay}_D O}(pk_D^1, pk_D^{\mathsf{cbl}}, \ddot{pk}_D^1, \mathbb{C}^* := tx_{\mathsf{bsc}}^{\mathsf{cbl}})$

$(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1) \leftarrow \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_0), \\ \mathsf{ctCnd}_R(sk_D^1, \mathbb{C}^*, T) \end{matrix} \right\rangle$

if $\neg(a_1 \wedge a_2)$ **abort**

$\mathsf{subTx}(tx_{\mathsf{bsc}}^{\mathsf{cbl}}, \sigma_{\mathsf{bsc}}^{\mathsf{cbl}})$

$(tx_{\mathsf{red}}^1, \sigma_{\mathsf{red}}^1) \leftarrow \Pi_{CP_1}.\mathsf{Red}(tx_{\mathsf{cnd}}^1, sk_D^1, w^*, \mathsf{aux}^1, \ddot{pk}_D^1)$

$\Pi_{CP_1}.\mathsf{subTx}(tx_{\mathsf{red}}^1, \sigma_{\mathsf{red}}^1)$

$a_0 := tx_{\mathsf{cnd}}^1, tx_{\mathsf{bsc}}^{\mathsf{cbl}} \notin \mathcal{Q}$

$a_1 := \Pi_{CP_1}.\mathsf{ckTx}(tx_{\mathsf{cnd}}^1) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^1, pk_D^1)$

$a_2 := \mathsf{isCond}(tx_{\mathsf{cnd}}^1, \mathbb{C}^*) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^1, epk_{I,debitor}^1)$

$a_3 := \mathsf{isSender}(tx_{\mathsf{bsc}}^{\mathsf{cbl}}, pk_D^{\mathsf{cbl}}) \wedge \mathsf{isRcvr}(tx_{\mathsf{bsc}}^{\mathsf{cbl}}, pk_C^{\mathsf{cbl}})$

$b_0 := \nexists\ \overline{tx_{\mathsf{red}}}^1\ s.t.\ \overline{tx_{\mathsf{red}}}^1 \in \mathcal{Q} \wedge \mathsf{ckTx}(\overline{tx_{\mathsf{red}}}^1) \wedge$
$\mathsf{isLinked}(tx_{\mathsf{red}}^1, \overline{tx_{\mathsf{red}}}^1)$

$b_1 := \Pi_{CP_1}.\mathsf{ckTx}(tx_{\mathsf{red}}^1) = 0 \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^1, tx_{\mathsf{red}}^1)$

$b_2 := \mathsf{ckTx}(tx_{\mathsf{bsc}}^{\mathsf{cbl}})$

$b_3 := \mathtt{readTime}(\cdot) < T$

return $\bigwedge_{i=0}^{3} a_i \wedge \bigwedge_{i=0}^{3} b_i$

Figure 24: Experiment for $BSD^{BL} - G_0^{\tau < T}(\lambda)$.

Therefore, we reach to:

$$\begin{aligned}
&\Pr[BSI_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda)] \\
&\quad = \Pr[BSI^{BL} - G_0^{\tau<T}(\lambda) = 1 \vee BSI^{BL} - G_0^{\tau \geq T}(\lambda) = 1] \leq \\
&\quad \Pr[BSI^{BL} - G_0^{\tau<T}(\lambda) = 1] + \Pr[BSI^{BL} - G_0^{\tau \geq T}(\lambda) = 1] \\
&\quad \leq \mathsf{negl}(\lambda).
\end{aligned}$$

This concludes the proof of Lemma 2. □

Lemma 3 (Balance security for D)**.** *For all PPT adversaries* $\mathcal{A}$*, it holds that* $\Pr[BSD^{BL}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. We consider the following game hops:

Game $BSD^{BL} - G_0^{\tau<T}$: This game, formally defined in Figure 24, corresponds to the original game for Balance security for D restricted the path where $\tau < T$. The game is expanded with the interactions described in our implementation.

Game $BSD^{BL} - G_1^{\tau<T}$: This game, formally defined in Figure 25, works exactly as G_0 with the following exception (highlighted in the grey lines). At the beginning of the protocol, $\mathcal{A}$ produces a forgery a $\overline{tx_{\mathsf{bsc}}}^{\mathsf{cbl}}$ impersonating D. This allows $\mathcal{A}$ to lock D's funds on cbl, without locking its own funds on Π_{CP_1}. If this forgery is valid, G_1 aborts.

Claim 6. *Let* Bad_1 *be the event that* $BSD^{BL} - G_1^{\tau<T}$ *aborts on grey lines of Figure 24. Assume that* cbl *provides BL unforgeability. 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 $\mathcal{B}$ that uses $\mathcal{A}$ to win bscForge. $\mathcal{B}$ interacts with $\mathcal{A}$ over the ledgers Π_{CP_1} and cbl and with the challenger over the ledger cbl.

- Challenger provides $\mathcal{B}$ with a sender public key pk_S.
- $\mathcal{B}$ sets $pk_D^{\mathsf{cbl}} := pk_S$ and generate (pk_D^1, sk_D^1), $(\ddot{pk}_D^1, \ddot{sk}_D^1)$ and $(tx_{\mathsf{bsc}}^{\mathsf{cbl}}, \sigma_{\mathsf{bsc}}^{\mathsf{cbl}})$.
- $\mathcal{B}$ forwards $(pk_D^1, pk_D^{\mathsf{cbl}}, \ddot{pk}_D^1, \mathbb{C}^* := tx_{\mathsf{bsc}}^{\mathsf{cbl}})$ to $\mathcal{A}$, which returns $(pk_I^1, pk_C^{\mathsf{cbl}}, T)$ and $\overline{tx_{\mathsf{bsc}}}^{\mathsf{cbl}}$, the latter being a forgery on cbl.
- $\mathcal{B}$ forwards $(tx_{\mathsf{bsc}} := \overline{tx_{\mathsf{bsc}}}^{\mathsf{cbl}}, pk_S := pk_C^{\mathsf{cbl}})$ to the challenger.

If $\mathcal{A}$ makes a query to $\mathsf{Set}_D O$ or $\mathsf{Pay}_D O$ query, $\mathcal{B}$ holds the private keys required to provide an honest transcript, since he does

not need to use sk_D^{cbl} in any of those for our implementation, as described in Figure 8 and Figure 8.

Our adversary $\mathcal{B}$ perfectly simulates $BSD^{BL} - G_1^{\tau<T}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ can force $\mathcal{B}$ to abort on the gray lines of Figure 24, it implies that (1) $\overline{tx_{\mathsf{bsc}}}^{\mathsf{cbl}} \notin Q$ (condition $\hat{c}_0$); (2) $\overline{tx_{\mathsf{bsc}}}^{\mathsf{cbl}} \in \Pi_{CP_0}.TX_L$ and $\mathsf{isSender}(\overline{tx_{\mathsf{bsc}}}^{\mathsf{cbl}}, pk_D^{\mathsf{cbl}}) = 1$ (condition $\hat{c}_1$). It is easy to see that these are equivalent to conditions b_0 and b_1 of game bscForge. Furthermore, we know that Q is synchronized in both games. However, this result contradicts the assumption that Π_{CP_0} satisfies BL unforgeability. Therefore, such adversary $\mathcal{A}$ cannot exist, thus this claim has been proven. □

Since games $BSD^{BL} - G_0^{\tau<T}$ and $BSD^{BL} - G_1^{\tau<T}$ are equivalent except for event Bad_1 occurring, it holds that

$$\Pr[BSD^{BL} - G_0^{\tau<T}(\lambda) = 1] \leq \Pr[BSD^{BL} - G_1^{\tau<T}(\lambda) = 1] + \mathsf{negl}(\lambda)$$

.

Claim 7. *Let ledger Π_{CP} provide CP redeemability. Then* $\Pr[BSD^{BL} - G_1^{\tau<T}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[BSD^{BL} - G_1^{\tau<T}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win ExpRedeem. $\mathcal{B}$ interacts with $\mathcal{A}$ over the ledgers Π_{CP_1} and cbl and with the challenger over the ledger Π_{CP_1}.

- Challenger provides $\mathcal{B}$ with public keys $(pk_R, \ddot{pk})$. $\mathcal{B}$ sets $pk_D^1 := pk_R$, $\ddot{pk}_D^1 := \ddot{pk}$ and generates $(pk_D^{\mathsf{cbl}}, sk_D^{\mathsf{cbl}})$ and $(tx_{\mathsf{bsc}}^{\mathsf{cbl}}, \sigma_{\mathsf{bsc}}^{\mathsf{cbl}})$.
- $\mathcal{B}$ forwards $(pk_D^1, pk_D^{\mathsf{cbl}}, \ddot{pk}_D^1, \mathbb{C}^* := tx_{\mathsf{bsc}}^{\mathsf{cbl}})$ to $\mathcal{A}$, which returns $(pk_I^1, pk_C^{\mathsf{cbl}}, T)$ and $\overline{tx_{\mathsf{bsc}}}^{\mathsf{cbl}}$, the latter being a forgery on cbl. Due to $\Pr[\mathsf{Bad}_1] \leq \mathsf{negl}(\lambda)$, this forgery does not make the game to abort.
- $\mathcal{B}$ sets $pk_S := pk_I^1$, $\mathbb{C} := \mathbb{C}^*$ and forwards $(pk_S, \mathbb{C}, T)$ to the challenger.
- $\mathcal{B}$ engages with the challenger and $\mathcal{A}$ on protocol $\Pi_{CP_1}.\mathsf{ctCnd}$ acting as a relay. This results in $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux}) := (tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1)$. By assumption $\mathcal{A}$ wins the game $BSD^{BL} - G_1^{\tau<T}$, hence conditions a_1 and a_2 are fulfilled and $\mathcal{B}$ does not abort.
- Finally, $\mathcal{B}$ submits the pair $(tx_{\mathsf{bsc}}^{\mathsf{cbl}}, \sigma_{\mathsf{bsc}}^{\mathsf{cbl}})$ to cbl and forwards $w := tx_{\mathsf{bsc}}^{\mathsf{cbl}}$ to the challenger. The latter uses w to output $(tx_{\mathsf{red}}, \sigma_{\mathsf{red}})$ and submits it to cbl.

If $\mathcal{A}$ makes a query to $\mathsf{Set}_D O$, $\mathcal{B}$ follows all the steps of protocol Set as described in Figure 26. However, $\mathcal{B}$ does not have the private key sk_D^1, so when Set requires this input, $\mathcal{B}$ queries $\mathsf{ctCnd}_R O$ for the output of $\Pi_{CP_1}.\mathsf{ctCnd}_R(sk_D^1, \mathbb{C}, T)$ and relays the answer. Similarly, when $\mathcal{A}$ makes a query to $\mathsf{Pay}_C O$, $\mathcal{B}$ follows all the steps of protocol Pay as described in Figure 27. Nevertheless, $\mathcal{B}$ does not have the private key sk_D^1, hence it queries $\mathsf{Red} O$ for the output of $\Pi_{CP_1}.\mathsf{Red}(tx_{\mathsf{cnd}}^1, sk_D^1, w, \mathsf{aux}^1, \ddot{pk}_D^1)$ and relays the answer. Note that Q is synchronized in both games.

Our adversary $\mathcal{B}$ perfectly simulates $BSD^{BL} - G_1^{\tau<T}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ can win $BSD^{BL} - G_1^{\tau<T}$ with non-negligible probability, this implies that (1) $tx_{\mathsf{cnd}}^1 \notin Q$ (condition a_0); (2) $tx_{\mathsf{cnd}}^1 \in \Pi_{CP_1}.TX_L$ and $\mathsf{isSender}(tx_{\mathsf{cnd}}^1, pk_D^1) = 1$ (condition a_1); (3) $\mathsf{isCond}(tx_{\mathsf{cnd}}^1, \mathbb{C}^*) = 1$ and $\mathsf{isRcvr}(tx_{\mathsf{cnd}}^1, epk_{I,D}^1) = 1$ (condition a_2); (4) $\mathcal{A}$ did not use $\mathsf{Pay}_D O$ to put $\overline{tx_{\mathsf{red}}}^1$ on Π_{CP_1} (condition b_0); (5) $tx_{\mathsf{red}}^1 \notin \Pi_{CP_1}.TX_L$ and $\mathsf{isLinked}(tx_{\mathsf{cnd}}^1, tx_{\mathsf{red}}^1) = 1$ (condition b_1); (6) $(\mathbb{C}^*, w^*) \in \mathcal{R}$ (CP PvP); (7) timeout T has not expired (condition b_3). It is easy to see that these are equivalent to conditions b_0, b_1, b_2, b_3, b_4, b_5 and b_6 of game ExpRedeem. Furthermore, we know that Q is synchronized in both games. However, this result contradicts the assumption that Π_{CP_1} satisfies CP redeemability. Therefore, such $\mathcal{A}$ cannot exist, thus this claim has been proven. □

Therefore, we reach to:

$$\begin{aligned}&\Pr[BSD_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda)]\\ &\quad\leq \Pr[BSD^{BL} - G_0^{\tau<T}(\lambda) = 1] + \Pr[BSD^{BL} - G_1^{\tau<T}(\lambda) = 1]\\ &\quad\leq \mathsf{negl}(\lambda).\end{aligned}$$

This concludes the proof of Lemma 3. □

And this concludes the proof of Theorem 1. □

F DETAILS ON THE CP-CBDC INSTANTIATION FOR CCE

Here we present a concrete instantiation of CBDC-cash environment w.r.t. three conditional payment ledgers, namely, Π_{CP_0}, Π_{CP_1} and cbl. Hence, the three parties can realize fund or defund operations agreeing on a condition $\mathbb{C}$.

Overview. We present a high level overview of protocols Set and Pay which are depicted in Fig. 26 and 27. Protocol Set executed by (1) creditor C, with inputs private keys sk_C^0, sk_C^{cbl} and timeout T; (2) intermediary I, with inputs private keys sk_I^1; and (3) debtor D, with inputs private keys sk_D^{cbl}, sk_D^1 and condition $\mathbb{C}$, does the following: (1) Creditor C and debtor D forward to the other parties timeout T and condition $\mathbb{C}$, respectively. (2) debtor D engages with creditor C in protocol cbl.ctCnd which results in a conditional payment transaction $tx_{\mathsf{cnd}}^{\mathsf{cbl}}$ (on ledger cbl) transferring coins from D to C with payment condition $\mathbb{C}$ and timeout $T'' := T + 2\delta$. (3) If transaction $tx_{\mathsf{cnd}}^{\mathsf{cbl}}$ is well-formed, creditor C engages with intermediary I in protocol $\Pi_{CP_0}.\mathsf{ctCnd}$ which results in a conditional payment transaction tx_{cnd}^0 (on ledger Π_{CP_0}) transferring coins from C to I with payment condition $\mathbb{C}$ and timeout $T' := T + \delta$. (4) If transaction tx_{cnd}^0 is well-formed, intermediary I engages with debtor D in protocol $\Pi_{CP_1}.\mathsf{ctCnd}$ which results in a conditional payment transaction tx_{cnd}^1 (on ledger Π_{CP_1}) transferring coins from I to D with payment condition $\mathbb{C}$ and timeout T. (5) Finally, debtor D checks that tx_{cnd}^1 is well-formed and the protocol terminates outputting auxiliary information aux_C, aux_I and aux_D for creditor C, intermediary I and debtor D, respectively.

Note that timeouts at conditional payment transactions $tx_{\mathsf{cnd}}^{\mathsf{cbl}}$ and tx_{cnd}^0 have been staggered by factors 2δ and δ, respectively, so that each participant has enough time to redeem the payment at each ledger after D triggers the Pay protocol. In a bit more detail, the protocol Pay is executed by (1) creditor C, with inputs private key sk_C^{cbl}, conditional transaction $tx_{\mathsf{cnd}}^{\mathsf{cbl}}$ and auxiliary information aux_C; (2) intermediary I, with inputs private key sk_I^0, conditional transaction tx_{cnd}^0 and auxiliary information aux_I; and (3) debtor

$BSD^{BL}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}} - G_1^{\tau<T}(\lambda)$

$\mathcal{Q} := \emptyset$

$(pk_D^1, sk_D^1) \leftarrow \Pi_{CP_1}.\mathsf{ctAcc}(1^\lambda)$

$(\ddot{pk}_D^1, \ddot{sk}_D^1) \leftarrow \Pi_{CP_1}.\mathsf{ctAcc}(1^\lambda)$

$(pk_D^{\mathsf{cbl}}, sk_D^{\mathsf{cbl}}) \leftarrow \mathsf{cbl.ctAcc}(1^\lambda)$

$(tx_{\mathsf{bsc}}^{\mathsf{cbl}}, \sigma_{\mathsf{bsc}}^{\mathsf{cbl}}) \leftarrow \mathsf{cbl.ctTx}(sk_D^{\mathsf{cbl}}, pk_C^{\mathsf{cbl}})$

$(pk_I^1, pk_C^{\mathsf{cbl}}, T, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{Set}_D O, \mathsf{Pay}_D O}(pk_D^1, pk_D^{\mathsf{cbl}}, \ddot{pk}_D^1, \mathbb{C}^* := tx_{\mathsf{bsc}}^{\mathsf{cbl}})$

$\overline{tx_{\mathsf{bsc}}}^{\mathsf{cbl}} \leftarrow \mathcal{A}(\mathsf{st}_0)$

$\hat{c}_0 \notin \mathcal{Q}$

$\hat{c}_1 := \mathsf{cbl.ckTx}(\overline{tx_{\mathsf{bsc}}}^{\mathsf{cbl}}) \wedge \mathsf{isSender}(\overline{tx_{\mathsf{bsc}}}^{\mathsf{cbl}}, pk_D^{\mathsf{cbl}})$

if $\hat{c}_0 \wedge \hat{c}_1$ **abort**

$(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1) \leftarrow \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_0), \\ \mathsf{ctCnd}_R(sk_D^1, \mathbb{C}^*, T) \end{matrix} \right\rangle$

if $\neg(a_1 \wedge a_2)$ **abort**

$\mathsf{subTx}(tx_{\mathsf{bsc}}^{\mathsf{cbl}}, \sigma_{\mathsf{bsc}}^{\mathsf{cbl}})$

$(tx_{\mathsf{red}}^1, \sigma_{\mathsf{red}}^1) \leftarrow \Pi_{CP_1}.\mathsf{Red}(tx_{\mathsf{cnd}}^1, sk_D^1, w^*, \mathsf{aux}^1, \ddot{pk}_D^1)$

$\Pi_{CP_1}.\mathsf{subTx}(tx_{\mathsf{red}}^1, \sigma_{\mathsf{red}}^1)$

$a_0 := tx_{\mathsf{cnd}}^1, tx_{\mathsf{bsc}}^{\mathsf{cbl}} \notin \mathcal{Q}$

$a_1 := \Pi_{CP_1}.\mathsf{ckTx}(tx_{\mathsf{cnd}}^1) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^1, pk_D^1)$

$a_2 := \mathsf{isCond}(tx_{\mathsf{cnd}}^1, \mathbb{C}^*) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^1, epk_{I,debitor}^1)$

$a_3 := \mathsf{isSender}(tx_{\mathsf{bsc}}^{\mathsf{cbl}}, pk_D^{\mathsf{cbl}}) \wedge \mathsf{isRcvr}(tx_{\mathsf{bsc}}^{\mathsf{cbl}}, pk_C^{\mathsf{cbl}})$

$b_0 := \nexists\ \overline{tx_{\mathsf{red}}}^1\ s.t.\ \overline{tx_{\mathsf{red}}}^1 \in \mathcal{Q} \wedge \mathsf{ckTx}(\overline{tx_{\mathsf{red}}}^1) \wedge$
$\mathsf{isLinked}(tx_{\mathsf{red}}^1, \overline{tx_{\mathsf{red}}}^1)$

$b_1 := \Pi_{CP_1}.\mathsf{ckTx}(tx_{\mathsf{red}}^1) = 0 \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^1, tx_{\mathsf{red}}^1)$

$b_2 := \mathsf{ckTx}(tx_{\mathsf{bsc}}^{\mathsf{cbl}})$

$b_3 := \mathtt{readTime}(\cdot) < T$

return $\bigwedge_{i=0}^{3} a_i \wedge \bigwedge_{i=0}^{3} b_i$

Figure 25: Experiment for $BSD^{BL} - G_1^{\tau<T}(\lambda)$.

D, with inputs private key sk_D^1, conditional transaction tx_{cnd}^1, witness w, and auxiliary information aux_D, does the following: (1) All parties parse their respective auxiliary information. (2) Debtor D redeems transaction tx_{cnd}^1 invoking algorithm $\Pi_{CP_1}.\mathsf{Red}$ with witness w. (3) Intermediary I extracts witness w from $(tx_{\mathsf{cnd}}^1, tx_{\mathsf{red}}^1)$ and then redeems transaction tx_{cnd}^0 invoking algorithm $\Pi_{CP_0}.\mathsf{Red}$. (4) Finally, creditor C extracts witness w from $(tx_{\mathsf{cnd}}^0, tx_{\mathsf{red}}^0)$ and then redeems transaction $tx_{\mathsf{cnd}}^{\mathsf{cbl}}$ invoking algorithm cbl.Red.

In case transaction debtor D did not use witness w to redeem transaction tx_{cnd}^1, all parties can recover their locked coins invoking algorithms $\Pi_{CP_0}.\mathsf{Red}$, $\Pi_{CP_1}.\mathsf{Red}$ and cbl.Red.

F.1 Proof of Correctness

Theorem 18 (CCE Correctness). *Assume that all three CP are correct. Assume that* IsFunded *holds for the accounts of C, I and D. Assume* IsUnique *and* IsValid *hold for all transactions. Then, our construction for CCE with* cbl *of type CP is correct according to Definition 11.*

Proof. CCE starts with all parties generating their accounts in Π_{CP_0}, Π_{CP_1} and cbl. The following pairs are generated: $(pk_C^{\mathsf{cbl}}, sk_C^{\mathsf{cbl}})$, (pk_C^0, sk_C^0), (pk_I^0, sk_I^0), (pk_I^1, sk_I^1), (pk_D^1, sk_D^1), $(pk_D^{\mathsf{cbl}}, sk_D^{\mathsf{cbl}})$. Also, an instance of the hard relation $(\mathbb{C}, w) \in \mathcal{R}$ is selected.

The protocol has two stages. The first one is Set. The second stage could either be Pay or refund. We first analyze our implementation for the Set protocol.

Set-Up. The implementation for Set starts with C sending T to I and D, and D replying to all with $\mathbb{C}$. Then, C and D engage in a cbl.ctCnd protocol using $\mathbb{C}$ and $T + 2\delta$, together with their keys (sk_C^{cbl}) and (sk_D^{cbl}), in ledger cbl. The protocol produces $(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathsf{aux}^{\mathsf{cbl}})$. Now, D calls $\mathsf{cbl.subTx}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}})$. The creditor checks if $tx_{\mathsf{cnd}}^{\mathsf{cbl}}$ is in the ledger, if D is the sender of that transaction, if $\mathbb{C}$ was used in the transaction and if the receiver is the shared account. Since we assume that (sk_D^{cbl}) is funded, IsFunded holds. Similarly, we assume that for $tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}}$, IsUnique and IsValid hold. Additionally, the cbl.ctCnd protocol used $\mathbb{C}$ as input, as well as (sk_C^{cbl}) and (sk_D^{cbl}). Therefore, since CP is correct, $\mathsf{cbl.subTx}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}}) = 1$. This implies that $tx_{\mathsf{cnd}}^{\mathsf{cbl}}$ is in the ledger, the D is the sender of $tx_{\mathsf{cnd}}^{\mathsf{cbl}}$, $\mathbb{C}$ was used in $tx_{\mathsf{cnd}}^{\mathsf{cbl}}$ and the receiver is the shared account.

In the next step, C engages with I to produce $(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0)$. It is easy to see that this is correct for the same reasons as for the generation of $(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathsf{aux}^{\mathsf{cbl}})$. Therefore, I obtains that tx_{cnd}^0 is in the ledger, C is the sender of tx_{cnd}^0, $\mathbb{C}$ was used in tx_{cnd}^0 and the receiver is the shared account since Π_{CP_0} is correct.

Finally, D engages with I to produce $(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1)$. It is easy to see that this is correct for the same reasons as for the generation of $(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathsf{aux}^{\mathsf{cbl}})$ and $(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0)$. Therefore, D obtains that tx_{cnd}^1 is in the ledger, I is the sender of tx_{cnd}^1, $\mathbb{C}$ was used in tx_{cnd}^1 and the receiver is the shared account since Π_{CP_1} is correct.

Therefore, we have shown that for our implementation of the Set protocol, it holds that:

$$\Pr\begin{bmatrix} \mathsf{ckTx}(tx_{\mathsf{cnd}}^0) = 1 \\ \mathsf{ckTx}(tx_{\mathsf{cnd}}^1) = 1 \\ \mathsf{ckTx}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}) = 1 \end{bmatrix} = 1$$

Pay. Now, if Set has been successful, two branches could happen. Either the conditional transactions are redeemed or they are refunded. If timeout T has not expired, C, I and D can initiate the Pay protocol to redeem the locked funds.

The implementation of Pay starts by each party obtaining the information stored in the aux that they obtained from the correct execution of Set. Since D is the only party that has w at the beginning, it starts by running the CP function $\Pi_{CP_1}.\mathsf{Red}$, which produces $(tx_{\mathsf{red}}^1, \sigma_{\mathsf{red}}^1)$. D submits this transaction to Π_{CP_1}. The first thing that I checks is if the tx_{red}^1 is in the ledger and if it is linked to tx_{cnd}^1(i.e. if it is spending the locked funds). Since no other transaction that is spending from tx_{cnd}^1 has been added to the ledger, IsFunded holds for tx_{red}^1. Similarly, we assumed that IsValid holds for transactions that use the private keys of any of the parties. We also assumed that the transactions validate IsUnique as well. Therefore, since

$\mathbf{C}(sk_C^0, sk_C^{\text{cbl}}, T)$	$\mathbf{I}(sk_I^1, sk_I^0)$	$\mathbf{D}(sk_D^{\text{cbl}}, sk_D^1, \mathbb{C})$
$\mathsf{Send}(T)$	$\mathsf{Route}(T)$	$\mathsf{Receive}(T)$
$\mathsf{Receive}(\mathbb{C})$	$\mathsf{Route}(\mathbb{C})$	$\mathsf{Send}(\mathbb{C})$
$T'' := T + 2\delta; T' := T + \delta$	$T' := T + \delta$	$T'' := T + 2\delta$
$(tx_{\text{cnd}}^{\text{cbl}}, \sigma_{\text{cnd}}^{\text{cbl}}, \text{aux}^{\text{cbl}})$		$(tx_{\text{cnd}}^{\text{cbl}}, \sigma_{\text{cnd}}^{\text{cbl}}, \text{aux}^{\text{cbl}})$
$\quad \leftarrow \text{cbl.ctCnd}_R(sk_C^{\text{cbl}}, \mathbb{C}, T'')$		$\quad \leftarrow \text{cbl.ctCnd}_S(sk_D^{\text{cbl}}, \mathbb{C}, T'')$
		$\text{cbl.subTx}(tx_{\text{cnd}}^{\text{cbl}}, \sigma_{\text{cnd}}^{\text{cbl}})$
$b_0 := \text{cbl.ckTx}(tx_{\text{cnd}}^{\text{cbl}}) \wedge \text{isSender}(tx_{\text{cnd}}^{\text{cbl}}, pk_D^{\text{cbl}})$		
$b_1 := \text{isCond}(tx_{\text{cnd}}^{\text{cbl}}, \mathbb{C}) \wedge \text{isRcvr}(tx_{\text{cnd}}^{\text{cbl}}, epk_{D,C}^{\text{cbl}})$		
if $\neg(b_0 \wedge b_1)$ **abort**		
$(tx_{\text{cnd}}^0, \sigma_{\text{cnd}}^0, \text{aux}^0)$	$(tx_{\text{cnd}}^0, \sigma_{\text{cnd}}^0, \text{aux}^0)$	
$\quad \leftarrow \Pi_{CP_0}.\text{ctCnd}_S(sk_C^0, \mathbb{C}, T')$	$\quad \leftarrow \Pi_{CP_0}.\text{ctCnd}_R(sk_I^0, \mathbb{C}, T')$	
$\Pi_{CP_0}.\text{subTx}(tx_{\text{cnd}}^0, \sigma_{\text{cnd}}^0)$		
	$b_2 := \Pi_{CP_0}.\text{ckTx}(tx_{\text{cnd}}^0) \wedge \text{isSender}(tx_{\text{cnd}}^0, pk_C^0)$	
	$b_3 := \text{isCond}(tx_{\text{cnd}}^0, \mathbb{C}) \wedge \text{isRcvr}(tx_{\text{cnd}}^0, epk_{C,I}^0)$	
	if $\neg(b_2 \wedge b_3)$ **abort**	
	$(tx_{\text{cnd}}^1, \sigma_{\text{cnd}}^1, \text{aux}^1)$	$(tx_{\text{cnd}}^1, \sigma_{\text{cnd}}^1, \text{aux}^1)$
	$\quad \leftarrow \Pi_{CP_1}.\text{ctCnd}_S(sk_I^1, \mathbb{C}, T)$	$\quad \leftarrow \Pi_{CP_1}.\text{ctCnd}_R(sk_D^1, \mathbb{C}, T)$
	$\Pi_{CP_1}.\text{subTx}(tx_{\text{cnd}}^1, \sigma_{\text{cnd}}^1)$	
		$b_4 := \Pi_{CP_1}.\text{ckTx}(tx_{\text{cnd}}^1) \wedge \text{isSender}(tx_{\text{cnd}}^1, pk_I^1)$
		$b_5 := \text{isCond}(tx_{\text{cnd}}^1, \mathbb{C}) \wedge \text{isRcvr}(tx_{\text{cnd}}^1, epk_{I,D}^1)$
		if $\neg(b_4 \wedge b_5)$ **abort**
$\text{aux}_C := (\text{aux}^{\text{cbl}}, \text{aux}^0, tx_{\text{cnd}}^0)$	$\text{aux}_I := (\text{aux}^0, \text{aux}^1, tx_{\text{cnd}}^1)$	$\text{aux}_D := (\text{aux}^1, \text{aux}^{\text{cbl}}, tx_{\text{cnd}}^{\text{cbl}})$
return $(tx_{\text{cnd}}^{\text{cbl}}, \text{aux}_C)$	**return** $(tx_{\text{cnd}}^0, \text{aux}_I)$	**return** $(tx_{\text{cnd}}^1, \text{aux}_D)$

Figure 26: Instantiation of Set **for CP-CBDC**

Π_{CP_1} is correct, $\Pi_{CP_1}.\text{subTx}((tx_{\text{red}}^1, \sigma_{\text{red}}^1)) = 1$. Similarly, D is taking tx_{cnd}^1 as input to generate $(tx_{\text{red}}^1, \sigma_{\text{red}}^1)$. This implies that I obtains that tx_{red}^1 is in the ledger and it is linked to tx_{cnd}^1, since CP is correct. Now, I runs $\Pi_{CP_1}.\text{GetWit}$, taking as input tx_{red}^1, tx_{cnd}^1 and aux^1. Since Π_{CP_1} is correct, it holds that $(\mathbb{C}, w) \in \mathcal{R}$ for the w generated by $\Pi_{CP_1}.\text{GetWit}$.

Now, I runs $\Pi_{CP_0}.\text{Red}$, which produces $(tx_{\text{red}}^0, \sigma_{\text{red}}^0)$. I submits this transaction to Π_{CP_0}. The first thing that C checks is if the tx_{red}^0 is in the ledger and if it is linked to tx_{cnd}^0(i.e. if it is spending the locked funds). Since no other transaction that is spending from tx_{cnd}^0 has been added to the ledger, `IsFunded` holds for tx_{red}^0. Similarly, we assumed that `IsValid` holds for transactions that use the private keys of any of the parties. We also assumed that the transactions validate `IsUnique` as well. Therefore, since Π_{CP_0} is correct, $\Pi_{CP_0}.\text{subTx}((tx_{\text{red}}^0, \sigma_{\text{red}}^0)) = 1$. Similarly, I is taking tx_{cnd}^0 as input to generate $(tx_{\text{red}}^0, \sigma_{\text{red}}^0)$. This implies that C obtains that tx_{red}^0 is in the ledger and it is linked to tx_{cnd}^0, since CP is correct. Now, C runs $\Pi_{CP_0}.\text{GetWit}$, taking as input tx_{red}^0, tx_{cnd}^0 and aux^0. Since Π_{CP_0} is correct, it holds that $(\mathbb{C}, w) \in \mathcal{R}$ for the w generated by $\Pi_{CP_0}.\text{GetWit}$.

Refund. Finally, C runs cbl.Red, which produces $(tx_{\text{red}}^{\text{cbl}}, \sigma_{\text{red}}^{\text{cbl}})$. C now submits this transaction to cbl. Since no other transaction that is spending from $tx_{\text{cnd}}^{\text{cbl}}$ has been added to the ledger, `IsFunded` holds for $tx_{\text{red}}^{\text{cbl}}$. Similarly, we assumed that `IsValid` holds for transactions that use the private keys of any of the parties. We also assumed that the transactions validate `IsUnique` as well. Therefore, since cbl is correct, $\text{cbl.subTx}(tx_{\text{red}}^{\text{cbl}}, \sigma_{\text{red}}^{\text{cbl}}) = 1$.

Therefore,

$$\Pr\begin{bmatrix}\text{ckTx}(tx_{\text{red}}^0) = 1\\ \text{ckTx}(tx_{\text{red}}^1) = 1\\ \text{ckTx}(tx_{\text{red}}^{\text{cbl}}) = 1\end{bmatrix} = 1$$

and the Pay implementation has been proven correct.

Finally, if Set has happened and timeout $T + 2\delta$ has expired, there are two possible options: Either refund happens or not.

If Pay has happened, then we have a transaction that is spending from the shared accounts used in $tx_{\text{cnd}}\text{cbl}$, $tx_{\text{cnd}}0$ and $tx_{\text{cnd}}1$. Therefore, if any of the parties attempts to submit a refund transaction tx_{ref}^i (for $i \in \{0, 1, \text{cbl}\}$), this transaction will be rejected since `IsFunded` does not hold: the funds of the shared account have been depleted during the Pay execution, which we have proven correct.

$\mathbf{C}(sk_C^{\mathsf{cbl}}, tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathsf{aux}_C, \ddot{pk}_C^{\mathsf{cbl}})$	$\mathbf{I}(sk_I^0, tx_{\mathsf{cnd}}^0, \mathsf{aux}_I, \ddot{pk}_I^0)$	$\mathbf{D}(sk_D^1, tx_{\mathsf{cnd}}^1, w, \mathsf{aux}_D, \ddot{pk}_D^1)$
$(\mathsf{aux}^{\mathsf{cbl}}, \mathsf{aux}^0, tx_{\mathsf{cnd}}^0) \leftarrow \mathsf{aux}_C$	$(\mathsf{aux}^0, \mathsf{aux}^1, tx_{\mathsf{cnd}}^1) \leftarrow \mathsf{aux}_I$	$(\mathsf{aux}^1, \mathsf{aux}^{\mathsf{cbl}}, tx_{\mathsf{cnd}}^{\mathsf{cbl}}) \leftarrow \mathsf{aux}_D$
		$(tx_{\mathsf{red}}^1, \sigma_{\mathsf{red}}^1)$
		$\quad \leftarrow \Pi_{CP_1}.\mathsf{Red}(tx_{\mathsf{cnd}}^1, sk_D^1, w, \mathsf{aux}^1, \ddot{pk}_D^1)$
		$\Pi_{CP_1}.\mathsf{subTx}(tx_{\mathsf{red}}^1, \sigma_{\mathsf{red}}^1)$
	$b_0 := \Pi_{CP_1}.\mathsf{ckTx}(tx_{\mathsf{red}}^1) \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^1, tx_{\mathsf{red}}^1)$	
	if $\neg b_0$ **abort**	
	$w \leftarrow \Pi_{CP_1}.\mathsf{GetWit}(tx_{\mathsf{cnd}}^1, tx_{\mathsf{red}}^1, \mathsf{aux}^1)$	
	$(tx_{\mathsf{red}}^0, \sigma_{\mathsf{red}}^0)$	
	$\quad \leftarrow \Pi_{CP_0}.\mathsf{Red}(tx_{\mathsf{cnd}}^0, sk_I^0, w, \mathsf{aux}^0, \ddot{pk}_I^0)$	
	$\Pi_{CP_0}.\mathsf{subTx}(tx_{\mathsf{red}}^0, \sigma_{\mathsf{red}}^0)$	
$b_1 := \Pi_{CP_0}.\mathsf{ckTx}(tx_{\mathsf{red}}^0) \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^0, tx_{\mathsf{red}}^0)$		
if $\neg b_1$ **abort**		
$w \leftarrow \Pi_{CP_0}.\mathsf{GetWit}(tx_{\mathsf{cnd}}^0, tx_{\mathsf{red}}^0, \mathsf{aux}^0)$		
$(tx_{\mathsf{red}}^{\mathsf{cbl}}, \sigma_{\mathsf{red}}^{\mathsf{cbl}})$		
$\quad \leftarrow \mathsf{cbl}.\mathsf{Red}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, sk_C^{\mathsf{cbl}}, w, \mathsf{aux}^{\mathsf{cbl}}, \ddot{pk}_C^{\mathsf{cbl}})$		
$\mathsf{cbl}.\mathsf{subTx}(tx_{\mathsf{red}}^{\mathsf{cbl}}, \sigma_{\mathsf{red}}^{\mathsf{cbl}})$		
return $(tx_{\mathsf{red}}^{\mathsf{cbl}})$	**return** (tx_{red}^0)	**return** (tx_{red}^1)

Figure 27: Instantiation of Pay **for CP-CBDC**

Therefore, since CP is correct, it must hold that we have $b = 0$ if Pay takes place.

However, if Pay has not happened, none of the three ledgers have a transaction that spends from the shared accounts of $tx_{\mathsf{cnd}}\mathsf{cbl}$, $tx_{\mathsf{cnd}}0$ and $tx_{\mathsf{cnd}}1$. Therefore, if any of the parties attempts to submit a refund transaction tx_{ref}^i (for $i \in \{0, 1, \mathsf{cbl}\}$), `IsFunded` will hold for this transaction: the funds of the shared account have not been spent yet since Pay has not happened. Since our assumption is that transaction ensure that `IsUnique` and `IsValid` hold, since CP is correct, it must hold that $b = 1$ if Pay does not take place.

And this concludes the proof of Theorem 18 □

F.2 Proof of Security

Theorem 19 (CCE balance security (CP-CBDC)). *Let ledgers Π_{CP_0}, Π_{CP_1} and* cbl*, satisfy CP unforgeability, CP witness unforgeability, CP redeemability, CP extractability and CP refundability. Then, our construction for CCE when* cbl *is of type CP depicted in Fig. 26 and 27 is secure according to Definition 9.*

Proof. In Figure 28 we show the oracles for our CCE construction when cbl is of type CP. As described in Definition 9, in order to show that our protocol achieves balance security, we need to show that:

$$\Pr[BSC^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

$$\Pr[BSI^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

$$\Pr[BSD^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

Lemma 4. *(Balance security for C) For all PPT adversaries $\mathcal{A}$, it holds that* $\Pr[BSC^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. The game BSC^{CP} contains two mutually exclusive paths that the adversary $\mathcal{A}$ could exploit. We will prove Lemma 4 by performing reductions on the two possible paths since:

$$\begin{aligned}&\Pr[BSC^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda) = 1] = \\ &\quad \Pr[BSC^{CP} - G_0^{\tau<T}(\lambda) = 1 \vee BSC^{CP} - G_0^{\tau\geq T+\delta}(\lambda) = 1] \leq \\ &\quad \Pr BSC^{CP} - G_0^{\tau<T}(\lambda) = 1] + \Pr[BSC^{CP} - G_0^{\tau\geq T+\delta}(\lambda) = 1]\end{aligned}$$

In the path where $\tau < T$, we perform the following game hops:

Game $BSC^{CP} - G_0^{\tau<T}$: This game, formally defined in Figure 29, corresponds to the original game for Balance security for C restricted to the path where $\tau < T$. The game is expanded with the interactions described in our implementation.

Game $BSC^{CP} - G_1^{\tau<T}$: This game, formally defined in Figure 30, works exactly as $BSC^{CP} - G_0^{\Pi_{CP_1}.\tau<T}$ with the following exception (highlighted in the gray lines): At the beginning of the protocol, $\mathcal{A}$ produces a forgery $\overline{tx_{\mathsf{cnd}}}^0$ impersonating C. This allows $\mathcal{A}$ to lock C's funds on Π_{CP_0}, without locking its own funds on cbl. If this forgery is valid, $BSC^{CP} - G_1^{\tau<T}$ aborts.

Game $BSC^{CP} - G_2^{\tau<T}$: This game, formally defined in Figure 31, works exactly as $BSC^{CP} - G_1^{\tau<T}$ with the following exception (highlighted in the gray lines): If the value w extracted from the

$\mathsf{Set}_C O(sk_D^{\mathsf{cbl}}, sk_I^0, \mathbb{C}, T)$

$T'' := T + 2\delta;\ T' := T + \delta$

$(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathsf{aux}^{\mathsf{cbl}}) \leftarrow \left\langle \begin{matrix} \mathsf{ctCnd}_S(sk_D^{\mathsf{cbl}}, \mathbb{C}, T''), \\ \mathsf{ctCnd}_R(sk_C^{\mathsf{cbl}}, \mathbb{C}, T'') \end{matrix} \right\rangle$

$(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0) \leftarrow \left\langle \begin{matrix} \mathsf{ctCnd}_S(sk_C^0, \mathbb{C}, T'), \\ \mathsf{ctCnd}_R(sk_I^0, \mathbb{C}, T') \end{matrix} \right\rangle$

$\mathsf{aux}_C := (\mathsf{aux}^{\mathsf{cbl}}, \mathsf{aux}^0, tx_{\mathsf{cnd}}^0)$

$Q := Q \cup \{(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathsf{aux}_C)\}$

return $(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}}, tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}_C)$

$\mathsf{Pay}_C O(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathsf{aux}_C, w, \ddot{pk}_C^{\mathsf{cbl}})$

$\mathsf{aux}_C := (\mathsf{aux}^{\mathsf{cbl}}, \mathsf{aux}^0, tx_{\mathsf{cnd}}^0)$

$(tx_{\mathsf{red}}^{\mathsf{cbl}}, \sigma_{\mathsf{red}}^{\mathsf{cbl}})$

$\quad \leftarrow \Pi_{CP_{\mathsf{cbl}}}.\mathsf{Red}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, sk_C^{\mathsf{cbl}}, w, \mathsf{aux}^{\mathsf{cbl}}, \ddot{pk}_C^{\mathsf{cbl}})$

$Q := Q \cup \{tx_{\mathsf{red}}^0\}$

return $(tx_{\mathsf{red}}^{\mathsf{cbl}}, \sigma_{\mathsf{red}}^{\mathsf{cbl}})$

$O_{\mathsf{Ref}_C}(tx_{\mathsf{cnd}}^0, \mathsf{aux}_C, \ddot{pk}_C^0)$

$\mathsf{aux}_C := (\mathsf{aux}^{\mathsf{cbl}}, \mathsf{aux}^0, tx_{\mathsf{cnd}}^0)$

$(tx_{\mathsf{ref}}^0, \sigma_{\mathsf{ref}}^0)$

$\quad \leftarrow \Pi_{CP_0}.\mathsf{Ref}(tx_{\mathsf{cnd}}^0, sk_C^0, \mathsf{aux}^0, \ddot{pk}_D^0)$

$Q := Q \cup \{tx_{\mathsf{ref}}^0\}$

return $(tx_{\mathsf{ref}}^0, \sigma_{\mathsf{ref}}^0)$

$\mathsf{Set}_I O(sk_C^0, sk_D^1, \mathbb{C}, T)$

$T' := T + \delta$

$(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0) \leftarrow \left\langle \begin{matrix} \mathsf{ctCnd}_S(sk_C^0, \mathbb{C}, T'), \\ \mathsf{ctCnd}_R(sk_I^0, \mathbb{C}, T') \end{matrix} \right\rangle$

$(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1) \leftarrow \left\langle \begin{matrix} \mathsf{ctCnd}_S(sk_I^1, \mathbb{C}, T), \\ \mathsf{ctCnd}_R(sk_D^1, \mathbb{C}, T) \end{matrix} \right\rangle$

$\mathsf{aux}_I := (\mathsf{aux}^0, \mathsf{aux}^1, tx_{\mathsf{cnd}}^1)$

$Q := Q \cup \{(tx_{\mathsf{cnd}}^0, \mathsf{aux}_I)\}$

return $(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}_I)$

$\mathsf{Pay}_I O(tx_{\mathsf{cnd}}^0, \mathsf{aux}_I, tx_{\mathsf{bsc}}^{\mathsf{cbl}}, \ddot{pk}_I^0)$

$\mathsf{aux}_I := (\mathsf{aux}^0, \mathsf{aux}^1, tx_{\mathsf{cnd}}^1)$

$(tx_{\mathsf{red}}^0, \sigma_{\mathsf{red}}^0)$

$\quad \leftarrow \Pi_{CP_0}.\mathsf{Red}(tx_{\mathsf{cnd}}^0, sk_I^0, tx_{\mathsf{bsc}}^{\mathsf{cbl}}, \mathsf{aux}^0, \ddot{pk}_I^0)$

$Q := Q \cup \{tx_{\mathsf{red}}^0\}$

return $(tx_{\mathsf{red}}^0, \sigma_{\mathsf{red}}^0)$

$\mathsf{Ref}_I O(tx_{\mathsf{cnd}}^1, \mathsf{aux}_I, \ddot{pk}_I^1)$

$\mathsf{aux}_I := (\mathsf{aux}^0, \mathsf{aux}^1, tx_{\mathsf{cnd}}^1)$

$(tx_{\mathsf{ref}}^1, \sigma_{\mathsf{ref}}^1)$

$\quad \leftarrow \Pi_{CP_1}.\mathsf{Ref}(tx_{\mathsf{cnd}}^1, sk_I^1, \mathsf{aux}^1, \ddot{pk}_I^1)$

$Q := Q \cup \{tx_{\mathsf{ref}}^1\}$

return $(tx_{\mathsf{ref}}^1, \sigma_{\mathsf{ref}}^1)$

$\mathsf{Set}_D O(sk_C^{\mathsf{cbl}}, sk_I^1, \mathbb{C}, T)$

$T'' := T + 2\delta$

$(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathsf{aux}^{\mathsf{cbl}}) \leftarrow \left\langle \begin{matrix} \mathsf{ctCnd}_S(sk_D^{\mathsf{cbl}}, \mathbb{C}, T''), \\ \mathsf{ctCnd}_R(sk_C^{\mathsf{cbl}}, \mathbb{C}, T'') \end{matrix} \right\rangle$

$(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1) \leftarrow \left\langle \begin{matrix} \mathsf{ctCnd}_S(sk_I^1, \mathbb{C}, T), \\ \mathsf{ctCnd}_R(sk_D^1, \mathbb{C}, T) \end{matrix} \right\rangle$

$\mathsf{aux}_D := (\mathsf{aux}^1, \mathsf{aux}^{\mathsf{cbl}}, tx_{\mathsf{cnd}}^{\mathsf{cbl}})$

$Q := Q \cup \{(tx_{\mathsf{cnd}}^1, \mathsf{aux}_D)\}$

return $(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}}, tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}_D)$

$\mathsf{Pay}_D O(tx_{\mathsf{cnd}}^1, \mathsf{aux}_D, tx_{\mathsf{bsc}}^{\mathsf{cbl}}, \ddot{pk}_D^1)$

$\mathsf{aux}_D := (\mathsf{aux}^1, \mathsf{aux}^{\mathsf{cbl}}, tx_{\mathsf{cnd}}^{\mathsf{cbl}})$

$(tx_{\mathsf{red}}^1, \sigma_{\mathsf{red}}^1)$

$\quad \leftarrow \Pi_{CP_1}.\mathsf{Red}(tx_{\mathsf{cnd}}^1, sk_D^1, tx_{\mathsf{bsc}}^{\mathsf{cbl}}, \mathsf{aux}^1, \ddot{pk}_D^1)$

$Q := Q \cup \{tx_{\mathsf{red}}^1\}$

return $(tx_{\mathsf{red}}^1, \sigma_{\mathsf{red}}^1)$

$\mathsf{Ref}_D O(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathsf{aux}_D, \ddot{pk}_D^{\mathsf{cbl}})$

$\mathsf{aux}_D := (\mathsf{aux}^1, \mathsf{aux}^{\mathsf{cbl}}, tx_{\mathsf{cnd}}^{\mathsf{cbl}})$

$(tx_{\mathsf{ref}}^{\mathsf{cbl}}, \sigma_{\mathsf{ref}}^{\mathsf{cbl}})$

$\quad \leftarrow \mathsf{cbl}.\mathsf{Ref}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, sk_D^{\mathsf{cbl}}, \mathsf{aux}^{\mathsf{cbl}}, \ddot{pk}_D^{\mathsf{cbl}})$

$Q := Q \cup \{tx_{\mathsf{ref}}^{\mathsf{cbl}}\}$

return $(tx_{\mathsf{ref}}^{\mathsf{cbl}}, \sigma_{\mathsf{ref}}^{\mathsf{cbl}})$

Figure 28: CCE oracles when cbl **is a CP**

pair $(tx_{\mathsf{cnd}}^0, tx_{\mathsf{red}}^0)$ is not a valid witness for the public statement $\mathbb{C}$, $BSC^{CP} - G_2^{\tau<T}$ aborts.

On the path where $\tau \geq T + \delta$, we only consider the following game:

Game $BSC^{CP} - G_0^{\tau \geq T+\delta}$: This game, formally defined in Figure 32, corresponds to the original game for Balance security for C restricted to the path where $\tau \geq T + \delta$. The game is expanded with the interactions described in our implementation.

Claim 8. *Let* Bad_1 *be the event that* $BSC^{CP} - G_1^{\tau<T}$ *aborts on gray lines of Figure 30. Assume that ledger* Π_{CP_0} *provides CP unforgeability, 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 $\mathcal{B}$ that uses $\mathcal{A}$ to win cndForge. $\mathcal{B}$ interacts with $\mathcal{A}$ over the ledgers Π_{CP_0} and cbl and with the challenger over the ledger Π_{CP_0}.

- Challenger provides $\mathcal{B}$ with a sender public key pk_S. $\mathcal{B}$ sets $pk_C^0 := pk_S$ and generates $(pk_C^{\mathsf{cbl}}, sk_C^{\mathsf{cbl}})$, $(\ddot{pk}_C^{\mathsf{cbl}}, \ddot{sk}_C^{\mathsf{cbl}})$ and T.
- $\mathcal{B}$ forwards $(pk_C^0, pk_C^{\mathsf{cbl}}, \ddot{pk}_C^{\mathsf{cbl}}, T)$ to $\mathcal{A}$, which returns $(pk_I^0, pk_D^{\mathsf{cbl}}, \mathbb{C})$ and $\overline{tx_{\mathsf{cnd}}}^0$, the latter being a forgery on Π_{CP_0}.
- $\mathcal{B}$ forwards $(tx_{\mathsf{cnd}} := \overline{tx_{\mathsf{cnd}}}^0, pk_R := pk_I^0, \mathbb{C}, T)$ to the challenger.

If $\mathcal{A}$ makes a query to $\mathsf{Set}_C O$, $\mathcal{B}$ follows all the steps of protocol Set as described in Figure 26. However, $\mathcal{B}$ does not have the private key sk_C^0, so when Set requires this input, $\mathcal{B}$ queries $\mathsf{ctCnd}_S O$ for the output of $\Pi_{CP_0}.\mathsf{ctCnd}_S(sk_C^0, \mathbb{C}, T')$ and relays the answer. Note that Q is synchronized in both games. If $\mathcal{A}$ makes a query to $\mathsf{Pay}_C O$ query, $\mathcal{B}$ follows all the steps of protocol Set, as described in Figure 27.

Our adversary $\mathcal{B}$ perfectly simulates $BSC^{CP} - G_1^{\tau<T}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ can force $\mathcal{B}$ to abort on the gray lines of Figure 30, it implies that (1) $\overline{tx_{\mathsf{cnd}}}^0 \notin Q$ (condition $\hat{c}_0$); (2) $\overline{tx_{\mathsf{cnd}}}^0 \in \Pi_{CP_0}.TX_L$ and $\mathsf{isSender}(\overline{tx_{\mathsf{cnd}}}^0, pk_C^0) = 1$ (condition $\hat{c}_1$); (3) $\mathsf{isCond}(\overline{tx_{\mathsf{cnd}}}^0, \mathbb{C}) = 1$ (condition $\hat{c}_2$). It is easy to see that these are equivalent to conditions b_0, b_1 and b_2 of game cndForge. Furthermore, we know that Q is synchronized in both games. However, this result contradicts the assumption that Π_{CP_0} satisfies CP unforgeability. Therefore, such adversary $\mathcal{A}$ cannot exist, thus this claim has been proven. □

Since games $BSC^{CP} - G_0^{\tau<T}$ and $BSC^{CP} - G_1^{\tau<T}$ are equivalent except for event Bad_1 occuring, it holds that

$$\Pr[BSC^{CP} - G_0^{\tau<T}(\lambda) = 1] \leq \Pr[BSC^{CP} - G_1^{\tau<T}(\lambda) = 1] + \mathsf{negl}(\lambda).$$

$BSC^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}} - G_0^{\tau<T}(\lambda)$

$\mathcal{Q} := \emptyset$

$(pk_C^0, sk_C^0) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$

$(pk_C^{\mathsf{cbl}}, sk_C^{\mathsf{cbl}}) \leftarrow \mathsf{cbl.ctAcc}(1^\lambda)$

$(\ddot{pk}_C^{\mathsf{cbl}}, \ddot{sk}_C^{\mathsf{cbl}}) \leftarrow \mathsf{cbl.ctAcc}(1^\lambda)$

$T \leftarrow \mathtt{createT}(\cdot)$

$T' := T + \delta;\ T'' := T + 2\delta$

$(pk_I^0, pk_D^{\mathsf{cbl}}, \mathbb{C}, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{Set}_C O, \mathsf{Pay}_C O}(pk_C^0, pk_C^{\mathsf{cbl}}, \ddot{pk}_C^{\mathsf{cbl}}, T)$

$(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathsf{aux}^{\mathsf{cbl}}, \mathsf{st}_1) \leftarrow \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_0), \\ \mathsf{cbl.ctCnd}_R(sk_C^{\mathsf{cbl}}, \mathbb{C}, T'') \end{matrix} \right\rangle$

if $\neg(a_3 \wedge a_4)$ **abort**

$(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0, \mathsf{st}_2) \leftarrow \left\langle \begin{matrix} \Pi_{CP_0}.\mathsf{ctCnd}_S(sk_C^0, \mathbb{C}, T'), \\ \mathcal{A}(\mathsf{st}_1) \end{matrix} \right\rangle$

$\Pi_{CP_0}.\mathsf{subTx}(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0)$

$tx_{\mathsf{red}}^0 \leftarrow \mathcal{A}(\mathsf{st}_2)$

if $\neg b_0$ **abort**

$w \leftarrow \Pi_{CP_0}.\mathsf{GetWit}(tx_{\mathsf{cnd}}^0, tx_{\mathsf{red}}^0, \mathsf{aux}^0)$

$(tx_{\mathsf{red}}^{\mathsf{cbl}}, \sigma_{\mathsf{red}}^{\mathsf{cbl}}) \leftarrow \mathsf{cbl.Red}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, sk_C^{\mathsf{cbl}}, w, \mathsf{aux}^{\mathsf{cbl}}, \ddot{pk}_C^{\mathsf{cbl}})$

$\mathsf{cbl.subTx}(tx_{\mathsf{red}}^{\mathsf{cbl}}, \sigma_{\mathsf{red}}^{\mathsf{cbl}})$

$a_0 := tx_{\mathsf{cnd}}^0, tx_{\mathsf{cnd}}^{\mathsf{cbl}} \notin \mathcal{Q}$

$a_1 := \Pi_{CP_0}.\mathsf{ckTx}(tx_{\mathsf{cnd}}^0) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^0, pk_C^0)$

$a_2 := \mathsf{isCond}(tx_{\mathsf{cnd}}^0, \mathbb{C}) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^0, epk_{C,I}^0)$

$a_3 := \mathsf{cbl.ckTx}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, pk_D^{\mathsf{cbl}})$

$a_4 := \mathsf{isCond}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathbb{C}) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, epk_{D,C}^{\mathsf{cbl}})$

$b_0 := \Pi_{CP_0}.\mathsf{ckTx}(tx_{\mathsf{red}}^0) \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^0, tx_{\mathsf{red}}^0)$

$b_1 := \nexists\, \overline{tx_{\mathsf{red}}}^{\mathsf{cbl}}\ s.t.\ \overline{tx_{\mathsf{red}}}^{\mathsf{cbl}} \in \mathcal{Q} \wedge \mathsf{ckTx}(\overline{tx_{\mathsf{red}}}^{\mathsf{cbl}}) \wedge$
$\mathsf{isLinked}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \overline{tx_{\mathsf{red}}}^{\mathsf{cbl}})$

$b_2 := \mathsf{cbl.ckTx}(tx_{\mathsf{red}}^{\mathsf{cbl}}) = 0 \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, tx_{\mathsf{red}}^{\mathsf{cbl}})$

$b_3 := \mathtt{readTime}(\cdot) < T$

return $\bigwedge_{i=0}^{4} a_i \wedge \bigwedge_{i=0}^{3} b_i$

Figure 29: Experiment for $BSC^{CP} - G_0^{\tau<T}(\lambda)$.

Claim 9. *Let* Bad_2 *be the event that* $BSC^{CP} - G_2^{\tau<T}$ *aborts on gray lines of Figure 31. Assume that ledger* Π_{CP_0} *provides CP extractability, then* $\Pr[\mathsf{Bad}_2] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{Bad}_2] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win ExpExtract. $\mathcal{B}$ interacts with $\mathcal{A}$ over the ledgers Π_{CP_0} and cbl and with the challenger over the ledger Π_{CP_0}.

- Challenger provides $\mathcal{B}$ with a sender public key pk_S. $\mathcal{B}$ sets $pk_C^0 := pk_S$ and generates $(pk_C^{\mathsf{cbl}}, sk_C^{\mathsf{cbl}})$, $(\ddot{pk}_C^{\mathsf{cbl}}, \ddot{sk}_C^{\mathsf{cbl}})$ and T.

$BSC^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}} - G_1^{\tau<T}(\lambda)$

$\mathcal{Q} := \emptyset$

$(pk_C^0, sk_C^0) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$

$(pk_C^{\mathsf{cbl}}, sk_C^{\mathsf{cbl}}) \leftarrow \mathsf{cbl.ctAcc}(1^\lambda)$

$(\ddot{pk}_C^{\mathsf{cbl}}, \ddot{sk}_C^{\mathsf{cbl}}) \leftarrow \mathsf{cbl.ctAcc}(1^\lambda)$

$T \leftarrow \mathtt{createT}(\cdot)$

$T' := T + \delta;\ T'' := T + 2\delta$

$(pk_I^0, pk_D^{\mathsf{cbl}}, \mathbb{C}, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{Set}_C O, \mathsf{Pay}_C O}(pk_C^0, pk_C^{\mathsf{cbl}}, \ddot{pk}_C^{\mathsf{cbl}}, T)$

$\overline{tx_{\mathsf{cnd}}}^0 \leftarrow \mathcal{A}(\mathsf{st}_0)$

$\hat{c}_0 := \overline{tx_{\mathsf{cnd}}}^0 \notin \mathcal{Q}$

$\hat{c}_1 := \Pi_{CP_0}.\mathsf{ckTx}(\overline{tx_{\mathsf{cnd}}}^0) \wedge \mathsf{isSender}(\overline{tx_{\mathsf{cnd}}}^0, pk_C^0)$

$\hat{c}_2 := \mathsf{isCond}(\overline{tx_{\mathsf{cnd}}}^0, \mathbb{C})$

if $\hat{c}_0 \wedge \hat{c}_1 \wedge \hat{c}_2$ **abort**

$(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathsf{aux}^{\mathsf{cbl}}, \mathsf{st}_1) \leftarrow \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_0), \\ \mathsf{cbl.ctCnd}_R(sk_C^{\mathsf{cbl}}, \mathbb{C}, T'') \end{matrix} \right\rangle$

if $\neg(a_3 \wedge a_4)$ **abort**

$(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0, \mathsf{st}_2) \leftarrow \left\langle \begin{matrix} \Pi_{CP_0}.\mathsf{ctCnd}_S(sk_C^0, \mathbb{C}, T'), \\ \mathcal{A}(\mathsf{st}_1) \end{matrix} \right\rangle$

$\Pi_{CP_0}.\mathsf{subTx}(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0)$

$tx_{\mathsf{red}}^0 \leftarrow \mathcal{A}(\mathsf{st}_2)$

if $\neg b_0$ **abort**

$w \leftarrow \Pi_{CP_0}.\mathsf{GetWit}(tx_{\mathsf{cnd}}^0, tx_{\mathsf{red}}^0, \mathsf{aux}^0)$

$(tx_{\mathsf{red}}^{\mathsf{cbl}}, \sigma_{\mathsf{red}}^{\mathsf{cbl}}) \leftarrow \mathsf{cbl.Red}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, sk_C^{\mathsf{cbl}}, w, \mathsf{aux}^{\mathsf{cbl}}, \ddot{pk}_C^{\mathsf{cbl}})$

$\mathsf{cbl.subTx}(tx_{\mathsf{red}}^{\mathsf{cbl}}, \sigma_{\mathsf{red}}^{\mathsf{cbl}})$

$a_0 := tx_{\mathsf{cnd}}^0, tx_{\mathsf{cnd}}^{\mathsf{cbl}} \notin \mathcal{Q}$

$a_1 := \Pi_{CP_0}.\mathsf{ckTx}(tx_{\mathsf{cnd}}^0) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^0, pk_C^0)$

$a_2 := \mathsf{isCond}(tx_{\mathsf{cnd}}^0, \mathbb{C}) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^0, epk_{C,I}^0)$

$a_3 := \mathsf{cbl.ckTx}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, pk_D^{\mathsf{cbl}})$

$a_4 := \mathsf{isCond}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathbb{C}) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, epk_{D,C}^{\mathsf{cbl}})$

$b_0 := \Pi_{CP_0}.\mathsf{ckTx}(tx_{\mathsf{red}}^0) \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^0, tx_{\mathsf{red}}^0)$

$b_1 := \nexists\, \overline{tx_{\mathsf{red}}}^{\mathsf{cbl}}\ s.t.\ \overline{tx_{\mathsf{red}}}^{\mathsf{cbl}} \in \mathcal{Q} \wedge \mathsf{ckTx}(\overline{tx_{\mathsf{red}}}^{\mathsf{cbl}}) \wedge$
$\mathsf{isLinked}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \overline{tx_{\mathsf{red}}}^{\mathsf{cbl}})$

$b_2 := \mathsf{cbl.ckTx}(tx_{\mathsf{red}}^{\mathsf{cbl}}) = 0 \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, tx_{\mathsf{red}}^{\mathsf{cbl}})$

$b_3 := \mathtt{readTime}(\cdot) < T$

return $\bigwedge_{i=0}^{4} a_i \wedge \bigwedge_{i=0}^{3} b_i$

Figure 30: Experiment for $BSC^{CP} - G_1^{\tau<T}(\lambda)$.

- $\mathcal{B}$ forwards $(pk_C^0, pk_C^{\mathsf{cbl}}, \ddot{pk}_C^{\mathsf{cbl}}, T)$ to $\mathcal{A}$, which returns $(pk_I^0, pk_D^{\mathsf{cbl}}, \mathbb{C})$ and $\overline{tx_{\mathsf{cnd}}}^0$, the latter being a forgery on Π_{CP_0}. Due to $\Pr[\mathsf{Bad}_1] \leq \mathsf{negl}(\lambda)$, this forgery does not make the game to abort.
- $\mathcal{B}$ sets $pk_R := pk_I^0$, $T' := T + \delta$ and forwards $(pk_R, \mathbb{C}, T')$ to the challenger.
- $\mathcal{B}$ engages with $\mathcal{A}$ on protocol cbl.ctCnd that outputs the tuple $(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathsf{aux}^{\mathsf{cbl}})$. By assumption $\mathcal{A}$ wins the game BSC^{CP} –

$G_2^{\tau<T}$, hence conditions a_3 and a_4 are fulfilled and $\mathcal{B}$ does not abort.

- $\mathcal{B}$ engages with the challenger and $\mathcal{A}$ on protocol Π_{CP_0}.ctCnd acting as a relay. This results in $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux}) := (tx^0_{\mathsf{cnd}}, \sigma^0_{\mathsf{cnd}}, \mathsf{aux}^0)$. Thereafter, the challenger submits $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}})$ to Π_{CP_0}.
- $\mathcal{A}$ outputs tx^0_{red}. By assumption $\mathcal{A}$ wins the game $BSC^{CP} - G_2^{\tau<T}$, hence condition b_0 is fulfilled and $\mathcal{B}$ does not abort. Henceforth, $\mathcal{B}$ forwards $tx_{\mathsf{red}} := tx^0_{\mathsf{red}}$ to the challenger.
- Finally, both $\mathcal{B}$ and the challenger extract w from $(tx_{\mathsf{cnd}} := tx^0_{\mathsf{cnd}}, tx_{\mathsf{red}} := tx^0_{\mathsf{red}})$.

If $\mathcal{A}$ makes a query to $\mathsf{Set}_C O$, $\mathcal{B}$ follows all the steps of protocol Set as described in Figure 26. However, $\mathcal{B}$ does not have the private key sk^0_C, so when Set requires this input, $\mathcal{B}$ queries $\mathsf{ctCnd}_S O$ for the output of $\Pi_{CP_0}.\mathsf{ctCnd}_S(sk^0_C, \mathbb{C}, T')$ and relays the answer. Note that Q is synchronized in both games. If $\mathcal{A}$ makes a query to $\mathsf{Pay}_C O$, $\mathcal{B}$ follows all the steps of protocol Set, as described in Figure 27.

Our adversary $\mathcal{B}$ perfectly simulates $BSC^{CP} - G_2^{\tau<T}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ can force $\mathcal{B}$ to abort on the gray lines of Figure 31, this implies that (1) $tx^0_{\mathsf{cnd}} \notin Q$ (condition a_0); (2) $tx^0_{\mathsf{cnd}} \in \Pi_{CP_0}.TX_L$ and $\mathsf{isSender}(tx^0_{\mathsf{cnd}}, pk^0_C) = 1$ (condition a_1); (3) $\mathsf{isCond}(tx^0_{\mathsf{cnd}}, \mathbb{C}) = 1$ and $\mathsf{isRcvr}(tx^0_{\mathsf{cnd}}, epk^0_{C,I}) = 1$ (condition a_2); (4) $tx^0_{\mathsf{red}} \in \Pi_{CP_0}.TX_L$ and $\mathsf{isLinked}(tx^0_{\mathsf{cnd}}, tx^0_{\mathsf{red}}) = 1$ (condition b_0); (5) $(\mathbb{C}, w) \notin \mathcal{R}$ (condition $\hat{c}_3$). It is easy to see that these are equivalent to conditions b_0, b_1, b_2, b_3 and b_4 of game ExpExtract. Furthermore, we know that Q is synchronized in both games. However, this result contradicts the assumption that Π_{CP_0} satisfies CP extractability. Therefore, such adversary $\mathcal{A}$ cannot exist, thus this claim has been proven. □

Since games $BSC^{CP} - G_1^{\tau<T}$ and $BSC^{CP} - G_2^{\tau<T}$ are equivalent except for event Bad_2 occuring, it holds that

$$\Pr[BSC^{CP} - G_1^{\tau<T}(\lambda) = 1] \leq \Pr[BSC^{CP} - G_2^{\tau<T}(\lambda) = 1] + \mathsf{negl}(\lambda).$$

Claim 10. *Let ledger* cbl *provide CP redeemability. Then* $\Pr[BSC^{CP} - G_2^{\tau<T}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[BSC^{CP} - G_2^{\tau<T}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win ExpRedeem. $\mathcal{B}$ interacts with $\mathcal{A}$ over the ledgers Π_{CP_0} and cbl and with the challenger over the ledger cbl.

- Challenger provides $\mathcal{B}$ with public keys $(pk_R, \ddot{pk})$. $\mathcal{B}$ sets $pk^{\mathsf{cbl}}_C := pk_R$, $\ddot{pk}^{\mathsf{cbl}}_C := \ddot{pk}$ and generates (pk^0_C, sk^0_C) and T.
- $\mathcal{B}$ forwards $(pk^0_C, pk^{\mathsf{cbl}}_C, \ddot{pk}^{\mathsf{cbl}}_C, T)$ to $\mathcal{A}$, which returns $(pk^0_I, pk^{\mathsf{cbl}}_D, \mathbb{C})$ and $\overline{tx_{\mathsf{cnd}}}^0$, the latter being a forgery on Π_{CP_0}. Due to $\Pr[\mathsf{Bad}_1] \leq \mathsf{negl}(\lambda)$, this forgery does not make the game to abort.
- $\mathcal{B}$ sets $pk_S := pk^{\mathsf{cbl}}_D$, $T'' := T + 2\delta$ and forwards $(pk_S, \mathbb{C}, T'')$ to challenger.
- $\mathcal{B}$ engages with the challenger and $\mathcal{A}$ on protocol cbl.ctCnd acting as a relay. This results in $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux}) := (tx^{\mathsf{cbl}}_{\mathsf{cnd}}, \sigma^{\mathsf{cbl}}_{\mathsf{cnd}}, \mathsf{aux}^{CL})$. By assumption $\mathcal{A}$ wins the game $BSC^{CP} - G_2^{\tau<T}$, hence conditions a_3 and a_4 are fulfilled and $\mathcal{B}$ does not abort.
- $\mathcal{B}$ engages with $\mathcal{A}$ on protocol Π_{CP_0}.ctCnd that outputs the tuple $(tx^0_{\mathsf{cnd}}, \sigma^0_{\mathsf{cnd}}, \mathsf{aux}^0)$. Henceforth, $\mathcal{B}$ submits the pair $(tx^0_{\mathsf{cnd}}, \sigma^0_{\mathsf{cnd}})$ to Π_{CP_0}.
- $\mathcal{A}$ outputs tx^0_{red}. By assumption $\mathcal{A}$ wins the game $BSC^{CP} - G_2^{\tau<T}$, hence condition b_0 is fulfilled and $\mathcal{B}$ does not abort. Thereafter, $\mathcal{B}$ extracts w from $(tx^0_{\mathsf{cnd}}, tx^0_{\mathsf{cnd}})$. Due to $\Pr[\mathsf{Bad}_2] \leq \mathsf{negl}(\lambda)$ it holds that $(\mathbb{C}, w) \in \mathcal{R}$. Hence, $\mathcal{B}$ forwards w to the challenger.
- Finally, the challenger uses w to output $(tx_{\mathsf{red}}, \sigma_{\mathsf{red}})$ and submits it to cbl.

If $\mathcal{A}$ makes a query to $\mathsf{Set}_C O$, $\mathcal{B}$ follows all the steps of protocol Set as described in Figure 26. However, $\mathcal{B}$ does not have the private key sk^{cbl}_C, so when Set requires this input, $\mathcal{B}$ queries $\mathsf{ctCnd}_R O$ for the output of $\mathsf{cbl.ctCnd}_R(sk^{\mathsf{cbl}}_C, \mathbb{C}, T'')$ and relays the answer. Similarly, when $\mathcal{A}$ makes a query to $\mathsf{Pay}_C O$, $\mathcal{B}$ follows all the steps of protocol Pay as described in Figure 27. Nevertheless, $\mathcal{B}$ does not have the private key sk^{cbl}_C, hence it queries $\mathsf{Red} O$ for the output of $\mathsf{cbl.Red}(tx^{\mathsf{cbl}}_{\mathsf{cnd}}, sk^{\mathsf{cbl}}_C, w, \mathsf{aux}^{\mathsf{cbl}}, \ddot{pk}^{\mathsf{cbl}}_C)$ and relays the answer. Note that Q is synchronized in both games.

Our adversary $\mathcal{B}$ perfectly simulates $BSC^{CP} - G_2^{\tau<T}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ can win $BSC^{CP} - G_2^{\tau<T}$ with non-negligible probability, this implies that (1) $tx^{\mathsf{cbl}}_{\mathsf{cnd}} \notin Q$ (condition a_0); (2) $tx^{\mathsf{cbl}}_{\mathsf{cnd}} \in \Pi_{CP_{\mathsf{cbl}}}.TX_L$ and $\mathsf{isSender}(tx^{\mathsf{cbl}}_{\mathsf{cnd}}, pk^{\mathsf{cbl}}_C) = 1$ (condition a_3); (3) $\mathsf{isCond}(tx^{\mathsf{cbl}}_{\mathsf{cnd}}, \mathbb{C}) = 1$ and $\mathsf{isRcvr}(tx^{\mathsf{cbl}}_{\mathsf{cnd}}, epk^{\mathsf{cbl}}_{D,C}) = 1$ (condition a_4); (4) $\mathcal{A}$ did not use $\mathsf{Pay}_C O$ to put $\overline{tx_{\mathsf{red}}}^{\mathsf{cbl}}$ on cbl (condition b_1); (5) $tx^{\mathsf{cbl}}_{\mathsf{red}} \notin \Pi_{CP_{\mathsf{cbl}}}.TX_L$ and $\mathsf{isLinked}(tx^{\mathsf{cbl}}_{\mathsf{cnd}}, tx^{\mathsf{cbl}}_{\mathsf{red}}) = 1$ (condition b_2); (6) $(\mathbb{C}, w) \in \mathcal{R}$ ($\Pr[\mathsf{Bad}_2] \leq \mathsf{negl}(\lambda)$); (7) T T has not expired (condition b_3). It is easy to see that these are equivalent to conditions b_0, b_1, b_2, b_3, b_4, b_5 and b_6 of game ExpRedeem. Furthermore, we know that Q is synchronized in both games. However, this result contradicts the assumption that Π_{CP} satisfies CP redeemability. Therefore, such adversary $\mathcal{A}$ cannot exist, thus this claim has been proven. □

So far, we have obtained that

$$\begin{aligned} Pr[BSC^{CP} - G_0^{\tau<T}(\lambda) = 1] \leq \\ \Pr[BSC^{CP} - G_1^{\tau<T}(\lambda) = 1] + \Pr[\mathsf{Bad}_1] \leq \\ \Pr[BSC^{CP} - G_2^{\tau<T}(\lambda) = 1] + \Pr[\mathsf{Bad}_2] + \Pr[\mathsf{Bad}_1] \leq \mathsf{negl}(\lambda). \end{aligned}$$

This means that for the path where $\tau < T$, $\mathcal{A}$ can win with negligible probability.

We now explore the other path.

Claim 11. *Let ledger* Π_{CP_0} *provide CP refundability. Then* $\Pr[BSC^{CP} - G_0^{\tau\geq T+\delta}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that there exists a PPT $\mathcal{A}$ such that $\Pr[BSC^{CP} - G_0^{\tau\geq T+\delta}(\lambda) = 1] > \mathsf{negl}(\lambda)$. Then we can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win ExpRefund. $\mathcal{B}$ interacts with $\mathcal{A}$ over the ledgers Π_{CP_0} and cbl and with the challenger over the ledger Π_{CP_0}.

- Challenger provides $\mathcal{B}$ with public keys $(pk_S, \ddot{pk})$. $\mathcal{B}$ sets $pk^0_C := pk_S$, $\ddot{pk}^0_C := \ddot{pk}$ and generates $(pk^{\mathsf{cbl}}_C, sk^{\mathsf{cbl}}_C)$ and T.
- $\mathcal{B}$ forwards $(pk^0_C, pk^{\mathsf{cbl}}_C, \ddot{pk}^0_C, T)$ to $\mathcal{A}$, which returns $(pk^0_I, pk^{\mathsf{cbl}}_D, \mathbb{C})$.
- $\mathcal{B}$ sets $pk_R := pk^0_I$, $T' := T + \delta$ and forwards $(pk_R, \mathbb{C}, T')$ to the challenger.
- $\mathcal{B}$ engages with $\mathcal{A}$ on protocol cbl.ctCnd that outputs the tuple $(tx^{\mathsf{cbl}}_{\mathsf{cnd}}, \sigma^{\mathsf{cbl}}_{\mathsf{cnd}}, \mathsf{aux}^{\mathsf{cbl}})$. By assumption $\mathcal{A}$ wins the game BSC^{CP} –

$BSC^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}} - G_2^{\tau<T}(\lambda)$

$\mathcal{Q} := \emptyset$

$(pk_C^0, sk_C^0) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$

$(pk_C^{\mathsf{cbl}}, sk_C^{\mathsf{cbl}}) \leftarrow \mathsf{cbl.ctAcc}(1^\lambda)$

$(\ddot{pk}_C^{\mathsf{cbl}}, \ddot{sk}_C^{\mathsf{cbl}}) \leftarrow \mathsf{cbl.ctAcc}(1^\lambda)$

$T \leftarrow \mathsf{createT}(\cdot)$

$T' := T + \delta;\ T'' := T + 2\delta$

$(pk_I^0, pk_D^{\mathsf{cbl}}, \mathbb{C}, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{Set}_C\mathcal{O},\mathsf{Pay}_C\mathcal{O}}(pk_C^0, pk_C^{\mathsf{cbl}}, \ddot{pk}_C^{\mathsf{cbl}}, T)$

$\overline{tx_{\mathsf{cnd}}}^0 \leftarrow \mathcal{A}(\mathsf{st}_0)$

$\hat{c}_0 := \overline{tx_{\mathsf{cnd}}}^0 \notin \mathcal{Q}$

$\hat{c}_1 := \Pi_{CP_0}.\mathsf{ckTx}(\overline{tx_{\mathsf{cnd}}}^0) \wedge \mathsf{isSender}(\overline{tx_{\mathsf{cnd}}}^0, pk_C^0)$

$\hat{c}_2 := \mathsf{isCond}(\overline{tx_{\mathsf{cnd}}}^0, \mathbb{C})$

if $\hat{c}_0 \wedge \hat{c}_1 \wedge \hat{c}_2$ **abort**

$(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathsf{aux}^{\mathsf{cbl}}, \mathsf{st}_1) \leftarrow \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_0), \\ \mathsf{cbl.ctCnd}_R(sk_C^{\mathsf{cbl}}, \mathbb{C}, T'') \end{matrix} \right\rangle$

if $\neg(a_3 \wedge a_4)$ **abort**

$(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0, \mathsf{st}_2) \leftarrow \left\langle \begin{matrix} \Pi_{CP_0}.\mathsf{ctCnd}_S(sk_C^0, \mathbb{C}, T'), \\ \mathcal{A}(\mathsf{st}_1) \end{matrix} \right\rangle$

$\Pi_{CP_0}.\mathsf{subTx}(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0)$

$tx_{\mathsf{red}}^0 \leftarrow \mathcal{A}(\mathsf{st}_2)$

if $\neg b_0$ **abort**

$w \leftarrow \Pi_{CP_0}.\mathsf{GetWit}(tx_{\mathsf{cnd}}^0, tx_{\mathsf{red}}^0, \mathsf{aux}^0)$

$\hat{c}_3 := (\mathbb{C}, w) \notin \mathcal{R}$

if $\hat{c}_3$ **abort**

$(tx_{\mathsf{red}}^{\mathsf{cbl}}, \sigma_{\mathsf{red}}^{\mathsf{cbl}}) \leftarrow \mathsf{cbl.Red}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, sk_C^{\mathsf{cbl}}, w, \mathsf{aux}^{\mathsf{cbl}}, \ddot{pk}_C^{\mathsf{cbl}})$

$\mathsf{cbl.subTx}(tx_{\mathsf{red}}^{\mathsf{cbl}}, \sigma_{\mathsf{red}}^{\mathsf{cbl}})$

$a_0 := tx_{\mathsf{cnd}}^0, tx_{\mathsf{cnd}}^{\mathsf{cbl}} \notin \mathcal{Q}$

$a_1 := \Pi_{CP_0}.\mathsf{ckTx}(tx_{\mathsf{cnd}}^0) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^0, pk_C^0)$

$a_2 := \mathsf{isCond}(tx_{\mathsf{cnd}}^0, \mathbb{C}) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^0, epk_{C,I}^0)$

$a_3 := \mathsf{cbl.ckTx}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, pk_D^{\mathsf{cbl}})$

$a_4 := \mathsf{isCond}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathbb{C}) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, epk_{D,C}^{\mathsf{cbl}})$

$b_0 := \Pi_{CP_0}.\mathsf{ckTx}(tx_{\mathsf{red}}^0) \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^0, tx_{\mathsf{red}}^0)$

$b_1 := \nexists\ \overline{tx_{\mathsf{red}}}^{\mathsf{cbl}}\ s.t.\ \overline{tx_{\mathsf{red}}}^{\mathsf{cbl}} \in \mathcal{Q} \wedge \mathsf{ckTx}(\overline{tx_{\mathsf{red}}}^{\mathsf{cbl}}) \wedge$
$\mathsf{isLinked}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \overline{tx_{\mathsf{red}}}^{\mathsf{cbl}})$

$b_2 := \mathsf{cbl.ckTx}(tx_{\mathsf{red}}^{\mathsf{cbl}}) = 0 \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, tx_{\mathsf{red}}^{\mathsf{cbl}})$

$b_3 := \mathsf{readTime}(\cdot) < T$

return $\bigwedge_{i=0}^{4} a_i \wedge \bigwedge_{i=0}^{3} b_i$

Figure 31: Experiment for $BSC^{CP} - G_2^{\tau<T}(\lambda)$.

$G_0^{\tau<T}$, hence conditions a_3 and a_4 are fulfilled and $\mathcal{B}$ does not abort.

- $\mathcal{B}$ engages with challenger and $\mathcal{A}$ on protocol $\Pi_{CP_0}.Promise$ acting as a relay. This results in $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux}) := (tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0)$. Thereafter, the challenger submits $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}})$ to Π_{CP_0}.
- Finally, the challenger generates a refund transaction tx_{ref} from tx_{cnd} and submits $(tx_{\mathsf{red}}, \sigma_{\mathsf{red}})$ to Π_{CP_0}.

If $\mathcal{A}$ makes a query to $\mathsf{Set}_C\mathcal{O}$, $\mathcal{B}$ follows all the steps of protocol Set as described in Figure 26. However, $\mathcal{B}$ does not have the private key sk_C^0, so when Set requires this input, $\mathcal{B}$ queries $\mathsf{ctCnd}_S\mathcal{O}$ for the output of $\Pi_{CP_0}.\mathsf{ctCnd}_S(sk_C^0, \mathbb{C}, T')$ and relays the answer. If $\mathcal{A}$ makes a query to $\mathsf{Ref}_C\mathcal{O}$, $\mathcal{B}$ does not have the private key sk_C^0, hence it queries $\mathsf{Ref}\mathcal{O}$ and relays the answer. Note that $\mathcal{Q}$ is synchronized in both games. If $\mathcal{A}$ makes a query to $\mathsf{Pay}_C\mathcal{O}$, $\mathcal{B}$ follows all the steps of protocol Set, as described in Figure 27.

Our adversary $\mathcal{B}$ perfectly simulates $BSC^{CP} - G_0^{\tau\geq T+\delta}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ is successful in wining $BSC^{CP} - G_0^{\tau\geq T+\delta}$ with non-negligible probability, this implies that (1) $tx_{\mathsf{cnd}}^0 \notin \mathcal{Q}$ (condition a_0); (2) $tx_{\mathsf{cnd}}^0 \in \Pi_{CP_0}.TX_L$ and $\mathsf{isSender}(tx_{\mathsf{cnd}}^0, pk_C^0) = 1$ (condition a_1); (3) $\mathsf{isCond}(tx_{\mathsf{cnd}}^0, \mathbb{C}) = 1$ and $\mathsf{isRcvr}(tx_{\mathsf{cnd}}^0, epk_{C,I}^0) = 1$ (condition a_2); (4) $\mathcal{A}$ did not put $\overline{tx_{\mathsf{red}}}^0$ on Π_{CP_0} (condition c_0); (5) $\mathcal{A}$ did not use $\mathsf{Ref}_C\mathcal{O}$ to put $\overline{tx_{\mathsf{ref}}}^0$ on Π_{CP_0} (condition c_1); (6) $tx_{\mathsf{ref}}^0 \notin \Pi_{CP_0}.TX_L$ and $\mathsf{isLinked}(tx_{\mathsf{cnd}}^0, tx_{\mathsf{ref}}^0) = 1$ (condition c_2); (7) timeout $T + \delta$ has expired (condition c_3). , It is easy to see that these are equivalent to conditions b_0, b_1, b_2, b_3, b_4, b_5 and b_6 of game ExpRefund. Furthermore, we know that the two $\mathcal{Q}$ are synchronized. However, this result contradicts the assumption that Π_{CP_0} satisfies CP refundability. Therefore, such $\mathcal{A}$ cannot exist and this claim has been proven. □

Therefore, we reach to:

$$\begin{aligned}
&\Pr[BSC^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda) = 1] \\
&= \Pr[BSC^{CP} - G_0^{\tau<T}(\lambda) = 1 \vee BSC^{CP} - G_0^{\tau\geq T+\delta}(\lambda) = 1] \\
&\leq \Pr[BSC^{CP} - G_0^{\tau<T}(\lambda) = 1] + \Pr[BSC^{CP} - G_0^{\tau\geq T+\delta}(\lambda) = 1] \\
&\leq \mathsf{negl}(\lambda)
\end{aligned}$$

This concludes the proof of Lemma 4. □

Lemma 5. *(Balance security for I) For all PPT adversaries $\mathcal{A}$, it holds that* $\Pr[BSI^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$

Proof. The game BSI^{CP} contains two mutually exclusive paths that the adversary $\mathcal{A}$ could exploit. We will prove Lemma 5 by performing reductions on the two possible paths since:

$$\begin{aligned}
&\Pr[BSI^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda) = 1)] = \\
&\quad \Pr[BSI^{CP} - G_0^{\tau<T}(\lambda) = 1 \vee BSI^{CP} - G_0^{\tau\geq T}(\lambda) = 1] \leq \\
&\quad \Pr[BSI^{CP} - G_0^{\tau<T}(\lambda) = 1] + \Pr[BSI^{CP} - G_0^{\tau\geq T}(\lambda) = 1]
\end{aligned}$$

In the path whare $\tau < T$, we perform the following game hops:

Game $BSI^{CP} - G_0^{\tau<T}$: This game, formally defined in Figure 33, corresponds to the original game for Balance security for I restricted to the path where $\tau < T$. The game is expanded with the interactions described in our implementation.

Game $BSI^{CP} - G_1^{\tau<T}$: This game, formally defined in Figure 34, works exactly as $BSI^{CP} - G_0^{\tau<T}$ with the following exception (highlighted in the gray lines). At the start of the protocol, $\mathcal{A}$ produces a forgery $\overline{tx_{\mathsf{cnd}}}^1$ impersonating C. This allows $\mathcal{A}$ to lock C's funds

$BSC^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}} - G_0^{\tau \geq T+\delta}(\lambda)$

$\mathcal{Q} := \emptyset$

$(pk_C^0, sk_C^0) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$

$(\ddot{pk}_C^0, \ddot{sk}_C^0) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$

$(pk_C^{\mathsf{cbl}}, sk_C^{\mathsf{cbl}}) \leftarrow \mathsf{cbl}.\mathsf{ctAcc}(1^\lambda)$

$T \leftarrow \mathtt{createT}(\cdot)$

$T' := T + \delta;\ T'' := T + 2\delta$

$(pk_I^0, pk_D^{\mathsf{cbl}}, \mathbb{C}, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{Set}_C O, \mathsf{Ref}_C O}(pk_C^0, pk_C^{\mathsf{cbl}}, \ddot{pk}_C^0, T)$

$(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathsf{aux}^{\mathsf{cbl}}, \mathsf{st}_1) \leftarrow \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_0), \\ \mathsf{cbl}.\mathsf{ctCnd}_R(sk_C^{\mathsf{cbl}}, \mathbb{C}, T'') \end{matrix} \right\rangle$

if $\neg(a_3 \wedge a_4)$ **abort**

$(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0) \leftarrow \left\langle \begin{matrix} \Pi_{CP_0}.\mathsf{ctCnd}_S(sk_C^0, \mathbb{C}, T'), \\ \mathcal{A}(\mathsf{st}_1) \end{matrix} \right\rangle$

$\Pi_{CP_0}.\mathsf{subTx}(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0)$

$(tx_{\mathsf{ref}}^0, \sigma_{\mathsf{ref}}^0) \leftarrow \Pi_{CP_0}.\mathsf{Ref}(tx_{\mathsf{cnd}}^0, sk_C^0, \mathsf{aux}^0, \ddot{pk}_C^0)$

$\Pi_{CP_0}.\mathsf{subTx}(tx_{\mathsf{ref}}^0, \sigma_{\mathsf{ref}}^0)$

$a_0 := tx_{\mathsf{cnd}}^0, tx_{\mathsf{cnd}}^{\mathsf{cbl}} \notin \mathcal{Q}$

$a_1 := \Pi_{CP_0}.\mathsf{ckTx}(tx_{\mathsf{cnd}}^0) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^0, pk_C^0)$

$a_2 := \mathsf{isCond}(tx_{\mathsf{cnd}}^0, \mathbb{C}) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^0, epk_{C,I}^0)$

$a_3 := \mathsf{cbl}.\mathsf{ckTx}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, pk_D^{\mathsf{cbl}})$

$a_4 := \mathsf{isCond}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathbb{C}) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, epk_{D,C}^{\mathsf{cbl}})$

$c_0 := \nexists\ \overline{tx_{\mathsf{red}}}^0\ s.t.\ \mathsf{ckTx}(\overline{tx_{\mathsf{red}}}^0) \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^0, \overline{tx_{\mathsf{red}}}^0)$

$c_1 := \nexists\ \overline{tx_{\mathsf{ref}}}^0\ s.t.\ \overline{tx_{\mathsf{ref}}}^0 \in \mathcal{Q} \wedge \mathsf{ckTx}(\overline{tx_{\mathsf{ref}}}^0) \wedge$
$\quad \mathsf{isLinked}(tx_{\mathsf{cnd}}^0, \overline{tx_{\mathsf{ref}}}^0)$

$c_2 := \Pi_{CP_0}.\mathsf{ckTx}(tx_{\mathsf{ref}}^0) = 0 \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^0, tx_{\mathsf{ref}}^0)$

$c_3 := \mathtt{readTime}(\cdot) \geq T + \delta$

return $\bigwedge_{i=0}^{4} a_i \wedge \bigwedge_{i=0}^{3} c_i$

Figure 32: Experiment for $BSC^{CP} - G_0^{\tau \geq T+\delta}(\lambda)$.

$BSI^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}} - G_0^{\tau < T}(\lambda)$

$\mathcal{Q} := \emptyset$

$(pk_I^0, sk_I^0) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$

$(\ddot{pk}_I^0, \ddot{sk}_I^0) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$

$(pk_I^1, sk_I^1) \leftarrow \Pi_{CP_1}.\mathsf{ctAcc}(1^\lambda)$

$(pk_C^0, pk_D^1, \mathbb{C}, T, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{Set}_I O, \mathsf{Pay}_I O}(pk_I^0, pk_I^1, \ddot{pk}_I^0)$

$T' := T + \delta$

$(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0, \mathsf{st}_1) \leftarrow \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_0), \\ \Pi_{CP_0}.\mathsf{ctCnd}_R(sk_I^0, \mathbb{C}, T') \end{matrix} \right\rangle$

if $\neg(a_1 \wedge a_2)$ **abort**

$(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1, \mathsf{st}_2) \leftarrow \left\langle \begin{matrix} \Pi_{CP_1}.\mathsf{ctCnd}_S(sk_I^1, \mathbb{C}, T), \\ \mathcal{A}(\mathsf{st}_1) \end{matrix} \right\rangle$

$\Pi_{CP_1}.\mathsf{subTx}(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1)$

$tx_{\mathsf{red}}1 \leftarrow \mathcal{A}(\mathsf{st}_2)$

if $\neg b_2$ **abort**

$w \leftarrow \Pi_{CP_1}.\mathsf{GetWit}(tx_{\mathsf{cnd}}^1, tx_{\mathsf{red}}^1, \mathsf{aux}^1)$

$(tx_{\mathsf{red}}^0, \sigma_{\mathsf{red}}^0) \leftarrow \Pi_{CP_0}.\mathsf{Red}(tx_{\mathsf{cnd}}^0, sk_I^0, w, \mathsf{aux}^0, \ddot{pk}_I^0)$

$\Pi_{CP_0}.\mathsf{subTx}(tx_{\mathsf{red}}^0, \sigma_{\mathsf{red}}^0)$

$a_0 := tx_{\mathsf{cnd}}^0, tx_{\mathsf{cnd}}^1 \notin \mathcal{Q}$

$a_1 := \Pi_{CP_0}.\mathsf{ckTx}(tx_{\mathsf{cnd}}^0) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^0, pk_C^0)$

$a_2 := \mathsf{isCond}(tx_{\mathsf{cnd}}^0, \mathbb{C}) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^0, epk_{C,I}^0)$

$a_3 := \Pi_{CP_1}.\mathsf{ckTx}(tx_{\mathsf{cnd}}^1) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^1, pk_I^1)$

$a_4 := \mathsf{isCond}(tx_{\mathsf{cnd}}^1, \mathbb{C}) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^1, epk_{I,D}^1)$

$b_0 := \nexists\ \overline{tx_{\mathsf{red}}}^0\ s.t.\ \overline{tx_{\mathsf{red}}}^0 \in \mathcal{Q} \wedge \mathsf{ckTx}(\overline{tx_{\mathsf{red}}}^0) \wedge$
$\quad \mathsf{isLinked}(tx_{\mathsf{cnd}}^0, \overline{tx_{\mathsf{red}}}^0)$

$b_1 := \Pi_{CP_0}.\mathsf{ckTx}(tx_{\mathsf{red}}^0) = 0 \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^0, tx_{\mathsf{red}}^0)$

$b_2 := \Pi_{CP_1}.\mathsf{ckTx}(tx_{\mathsf{red}}^1) \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^1, tx_{\mathsf{red}}^1)$

$b_3 := \mathtt{readTime}(\cdot) < T$

return $\bigwedge_{i=0}^{4} a_i \wedge \bigwedge_{i=0}^{3} b_i$

Figure 33: Experiment for $BSI^{CP} - G_0^{\tau < T}(\lambda)$.

on Π_{CP_1}, without locking its own funds on Π_{CP_0}. If this forgery is valid, $BSI^{CP} - G_1^{\tau<T}$ aborts.

Game $BSI^{CP} - G_2^{\tau<T}$: This game, formally defined in Figure 35, works exactly as $BSI^{CP} - G_1^{\Pi_{CP_1}.\tau<T}$ with the following exception (highlighted in the gray lines). If the value w extracted from the pair $(tx_{\mathsf{cnd}}^1, tx_{\mathsf{red}}^1)$ is not a valid witness for the public statement $\mathbb{C}$, $BSI^{CP} - G_2^{\tau<T}$ aborts.

On the path where $\tau \geq T$, we only consider the following game:

Game $BSI^{CP} - G_0^{\tau \geq T}$: This game, formally defined in Figure 36, corresponds to the original game for Balance security for I, restricted to the path where $\tau \geq T$. The game is expanded with the interactions described in our implementation.

Claim 12. *Let* Bad_1 *be the event that* $BSI^{CP} - G_1^{\tau<T}$ *aborts on gray lines of Figure 34. Assume that ledger* Π_{CP_1} *provides CP unforgeability, 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 $\mathcal{B}$ that uses $\mathcal{A}$ to win cndForge. $\mathcal{B}$ interacts with $\mathcal{A}$ over the ledgers Π_{CP_0} and Π_{CP_1} and with the challenger over the ledger Π_{CP_1}.

- Challenger provides $\mathcal{B}$ with a sender public key pk_S. $\mathcal{B}$ sets $pk_I^1 := pk_S$ and generates (pk_I^0, sk_I^0) and $(\ddot{pk}_C^0, \ddot{sk}_C^0)$.
- $\mathcal{B}$ forwards $(pk_I^0, pk_I^1, \ddot{sk}_C^0)$ to $\mathcal{A}$, which returns $(pk_C^0, pk_D^1, \mathbb{C}, T)$ and $\overline{tx_{\mathsf{cnd}}}^1$, the latter being a forgery on Π_{CP_1}.
- $\mathcal{B}$ forwards $(tx_{\mathsf{i}} := \overline{tx_{\mathsf{cnd}}}^1, pk_R := pk_D^1, \mathbb{C}, T)$ to challenger.

If $\mathcal{A}$ makes a query to $\mathsf{Set}_I O$, $\mathcal{B}$ follows all the steps of protocol Set as described in Figure 26. However, $\mathcal{B}$ does not have the private key sk_I^1, so when Set requires this input, $\mathcal{B}$ queries $\mathsf{ctCnd}_S O$ for the output of $\Pi_{CP_1}.\mathsf{ctCnd}_S(sk_I^1, \mathbb{C}, T)$ and relays the answer. Note that

$\mathcal{Q}$ is synchronized in both games. If $\mathcal{A}$ makes a query to $\mathsf{Pay}_I O$, $\mathcal{B}$ follows all the steps of protocol Set, as described in Figure 27.

Our adversary $\mathcal{B}$ perfectly simulates $BSI^{CP} - G_1^{\tau<T}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ can force $\mathcal{B}$ to abort on the gray lines of Figure 34, it implies that (1) $\overline{tx_{\mathsf{cnd}}}^1 \notin \mathcal{Q}$ (condition $\hat{c}_0$); (2) $\overline{tx_{\mathsf{cnd}}}^1 \in \Pi_{CP_1}.TX_L$ and $\mathsf{isSender}(\overline{tx_{\mathsf{cnd}}}^1, pk_C^1) = 1$ (condition $\hat{c}_1$); (3) $\mathsf{isCond}(\overline{tx_{\mathsf{cnd}}}^1, \mathbb{C}) = 1$ (condition $\hat{c}_2$). It is easy to see that these are equivalent to conditions b_0, b_1 and b_2 of game cndForge. Furthermore, we know that $\mathcal{Q}$ is synchronized in both games. However, this result contradicts the assumption that Π_{CP_1} satisfies CP unforgeability. Therefore, such adversary $\mathcal{A}$ cannot exist, thus this claim has been proven. □

Since games $BSI^{CP} - G_0^{\tau<T}$ and $BSI^{CP} - G_1^{\tau<T}$ are equivalent except for event Bad_1 occuring, it holds that

$$\Pr[BSI^{CP} - G_0^{\tau<T}(\lambda) = 1] \leq \Pr[BSI^{CP} - G_1^{\tau<T}(\lambda) = 1] + \mathsf{negl}(\lambda).$$

Claim 13. *Let* Bad_2 *be the event that* $BSI^{CP} - G_2^{\tau<T}$ *aborts on gray lines of Figure 35. Assume that ledger* Π_{CP_1} *provides CP extractability, then* $\Pr[\mathsf{Bad}_2] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{Bad}_2] > \mathsf{negl}(\lambda)$. Then we can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win ExpExtract. $\mathcal{B}$ interacts with $\mathcal{A}$ over the ledgers Π_{CP_0} and Π_{CP_1} and with the challenger over the ledger Π_{CP_1}.

- Challenger provides $\mathcal{B}$ with a sender public key pk_S. $\mathcal{B}$ sets $pk_I^1 := pk_S$ and generates (pk_C^0, sk_C^0) and $(\ddot{pk}_C^0, \ddot{sk}_C^0)$.
- $\mathcal{B}$ forwards $(pk_I^0, pk_I^1, \ddot{sk}_I^0)$ to $\mathcal{A}$, which returns $(pk_C^0, pk_D^1, \mathbb{C}, T)$ and $\overline{tx_{\mathsf{cnd}}}^1$, the latter being a forgery on Π_{CP_1}. Due to $\Pr[\mathsf{Bad}_1] \leq \mathsf{negl}(\lambda)$, this forgery does not make the game to abort.
- $\mathcal{B}$ sets $pk_R := pk_D^1$ and forwards $(pk_R, \mathbb{C}, T)$ to the challenger.
- $\mathcal{B}$ engages with $\mathcal{A}$ on protocol cbl.ctCnd that outputs the tuple $(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0)$. By assumption $\mathcal{A}$ wins the game $BSI^{CP} - G_2^{\tau<T}$, hence conditions a_1 and a_2 are fulfilled and $\mathcal{B}$ does not abort.
- $\mathcal{B}$ engages with the challenger and $\mathcal{A}$ on protocol $\Pi_{CP_1}.\mathsf{ctCnd}$ acting as a relay. This results in $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux}) := (tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1)$. Thereafter, the challenger submits tx_{cnd} to Π_{CP_1}.
- $\mathcal{A}$ outputs tx_{red}^1. By assumption $\mathcal{A}$ wins the game $BSI^{CP} - G_2^{\tau<T}$, hence condition b_2 is fulfilled and $\mathcal{B}$ does not abort. Henceforth, $\mathcal{B}$ forwards $tx_{\mathsf{red}} := tx_{\mathsf{red}}^1$ to the challenger.
- Finally, both $\mathcal{B}$ and the challenger extract w from $(tx_{\mathsf{cnd}} := tx_{\mathsf{cnd}}^1, tx_{\mathsf{red}} := tx_{\mathsf{red}}^1)$.

If $\mathcal{A}$ makes a query to $\mathsf{Set}_I O$, $\mathcal{B}$ follows all the steps of protocol Set as described in Figure 26. However, $\mathcal{B}$ does not have the private key sk_I^1, so when Set requires this input, $\mathcal{B}$ queries $\mathsf{ctCnd}_S O$ for the output of $\Pi_{CP_1}.\mathsf{ctCnd}_S(sk_I^1, \mathbb{C}, T)$ and relays the answer. Note that $\mathcal{Q}$ is synchronized in both games. If $\mathcal{A}$ makes a query to $\mathsf{Pay}_I O$, $\mathcal{B}$ follows all the steps of protocol Set, as described in Figure 27.

Our adversary $\mathcal{B}$ perfectly simulates $BSI^{CP} - G_2^{\tau<T}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ can force $\mathcal{B}$ to abort on the gray lines of Figure 31, this implies that (1) $tx_{\mathsf{cnd}}^1 \notin \mathcal{Q}$ (condition a_0); (2) $tx_{\mathsf{cnd}}^1 \in \Pi_{CP_1}.TX_L$ and $\mathsf{isSender}(tx_{\mathsf{cnd}}^1, pk_I^1) = 1$ (condition a_3); (3) $\mathsf{isCond}(tx_{\mathsf{cnd}}^1, \mathbb{C}) = 1$ and $\mathsf{isRcvr}(tx_{\mathsf{cnd}}^1, epk_{I,D}^1) = 1$ (condition a_4); (4) $tx_{\mathsf{red}}^1 \in \Pi_{CP_1}.TX_L$ and $\mathsf{isLinked}(tx_{\mathsf{cnd}}^1, tx_{\mathsf{red}}^1) = 1$ (condition b_2); (5) $(\mathbb{C}, w) \notin \mathcal{R}$ (condition $\hat{c}_3$). It is easy to see that these are equivalent to conditions b_0, b_1, b_2, b_3 and b_4 of game ExpExtract. Furthermore, we know that $\mathcal{Q}$ is synchronized in both games. However, this result contradicts the assumption that Π_{CP_0} satisfies CP extractability. Therefore, such adversary $\mathcal{A}$ cannot exist, thus this claim has been proven. □

$BSI^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}} - G_1^{\tau<T}(\lambda)$

$\mathcal{Q} := \emptyset$

$(pk_I^0, sk_I^0) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$

$(\ddot{pk}_I^0, \ddot{sk}_I^0) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$

$(pk_I^1, sk_I^1) \leftarrow \Pi_{CP_1}.\mathsf{ctAcc}(1^\lambda)$

$(pk_C^0, pk_D^1, \mathbb{C}, T, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{Set}_I O, \mathsf{Pay}_I O}(pk_I^0, pk_I^1, \ddot{pk}_I^0)$

$T' := T + \delta$

$\overline{tx_{\mathsf{cnd}}}^1 \leftarrow \mathcal{A}(\mathsf{st}_0)$

$\hat{c}_0 := \overline{tx_{\mathsf{cnd}}}^1 \notin \mathcal{Q}$

$\hat{c}_1 := \Pi_{CP_1}.\mathsf{ckTx}(\overline{tx_{\mathsf{cnd}}}^1) \wedge \mathsf{isSender}(\overline{tx_{\mathsf{cnd}}}^1, pk_I^1)$

$\hat{c}_2 := \mathsf{isCond}(\overline{tx_{\mathsf{cnd}}}^1, \mathbb{C})$

if $\hat{c}_0 \wedge \hat{c}_1 \wedge \hat{c}_2$ **abort**

$(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0, \mathsf{st}_1) \leftarrow \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_0), \\ \Pi_{CP_0}.\mathsf{ctCnd}_R(sk_I^0, \mathbb{C}, T') \end{matrix} \right\rangle$

if $\neg(a_1 \wedge a_2)$ **abort**

$(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1, \mathsf{st}_2) \leftarrow \left\langle \begin{matrix} \Pi_{CP_1}.\mathsf{ctCnd}_S(sk_I^1, \mathbb{C}, T), \\ \mathcal{A}(\mathsf{st}_1) \end{matrix} \right\rangle$

$\Pi_{CP_1}.\mathsf{subTx}(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1)$

$tx_{\mathsf{red}}1 \leftarrow \mathcal{A}(\mathsf{st}_2)$

if $\neg b_2$ **abort**

$w \leftarrow \Pi_{CP_1}.\mathsf{GetWit}(tx_{\mathsf{cnd}}^1, tx_{\mathsf{red}}^1, \mathsf{aux}^1)$

$(tx_{\mathsf{red}}^0, \sigma_{\mathsf{red}}^0) \leftarrow \Pi_{CP_0}.\mathsf{Red}(tx_{\mathsf{cnd}}^0, sk_I^0, w, \mathsf{aux}^0, \ddot{pk}_I^0)$

$\Pi_{CP_0}.\mathsf{subTx}(tx_{\mathsf{red}}^0, \sigma_{\mathsf{red}}^0)$

$a_0 := tx_{\mathsf{cnd}}^0, tx_{\mathsf{cnd}}^1 \notin \mathcal{Q}$

$a_1 := \Pi_{CP_0}.\mathsf{ckTx}(tx_{\mathsf{cnd}}^0) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^0, pk_C^0)$

$a_2 := \mathsf{isCond}(tx_{\mathsf{cnd}}^0, \mathbb{C}) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^0, epk_{C,I}^0)$

$a_3 := \Pi_{CP_1}.\mathsf{ckTx}(tx_{\mathsf{cnd}}^1) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^1, pk_I^1)$

$a_4 := \mathsf{isCond}(tx_{\mathsf{cnd}}^1, \mathbb{C}) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^1, epk_{I,D}^1)$

$b_0 := \nexists\, \overline{tx_{\mathsf{red}}}^0 \; s.t. \; \overline{tx_{\mathsf{red}}}^0 \in \mathcal{Q} \wedge \mathsf{ckTx}(\overline{tx_{\mathsf{red}}}^0) \wedge$
$\mathsf{isLinked}(tx_{\mathsf{cnd}}^0, \overline{tx_{\mathsf{red}}}^0)$

$b_1 := \Pi_{CP_0}.\mathsf{ckTx}(tx_{\mathsf{red}}^0) = 0 \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^0, tx_{\mathsf{red}}^0)$

$b_2 := \Pi_{CP_1}.\mathsf{ckTx}(tx_{\mathsf{red}}^1) \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^1, tx_{\mathsf{red}}^1)$

$b_3 := \mathtt{readTime}(\cdot) < T$

return $\bigwedge_{i=0}^{4} a_i \wedge \bigwedge_{i=0}^{3} b_i$

Figure 34: Experiment for $BSI^{CP} - G_1^{\tau<T}(\lambda)$.

Since games $BSI^{CP} - G_1^{\tau<T}$ and $BSI^{CP} - G_2^{\tau<T}$ are equivalent except for event Bad_2 occuring, it holds that

$$\Pr[BSI - G_1^{\tau<T}(\lambda) = 1] \leq \Pr[BSI^{CP} - G_2^{\tau<T}(\lambda) = 1] + \mathsf{negl}(\lambda).$$

Claim 14. *Let ledger Π_{CP_0} provide CP redeemability. Then* $\Pr[BSI^{CP} - G_2^{\tau<T}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[BSI^{CP} - G_2^{\tau<T}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win ExpRedeem. $\mathcal{B}$ interacts with $\mathcal{A}$ over the ledgers Π_{CP_0} and Π_{CP_1} and with the challenger over the ledger Π_{CP_0}.

- Challenger provides $\mathcal{B}$ with public keys $(pk_R, \ddot{pk})$. $\mathcal{B}$ sets $pk_I^0 := pk_R$, $\ddot{pk}_I^0 := \ddot{pk}$ and generates (pk_I^1, sk_I^1).
- $\mathcal{B}$ forwards $(pk_I^0, pk_I^1, \ddot{sk}_I^0)$ to $\mathcal{A}$, which returns $(pk_C^0, pk_D^1, \mathbb{C}, T)$ and $\overline{tx_{\mathsf{cnd}}}^1$, the latter being a forgery on Π_{CP_1}. Due to $\Pr[\mathsf{Bad}_1] \leq \mathsf{negl}(\lambda)$, this forgery does not make the game to abort.
- $\mathcal{B}$ sets $pk_S := pk_C^0$, $T' := T + \delta$ and forwards $(pk_S, \mathbb{C}, T')$ to the challenger.
- $\mathcal{B}$ engages with the challenger and $\mathcal{A}$ on protocol $\Pi_{CP_0}.\mathsf{ctCnd}$ acting as a relay. This results in $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux}) := (tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0)$. By assumption $\mathcal{A}$ wins the game $BSI^{CP} - G_2^{\tau<T}$, hence conditions a_1 and a_2 are fulfilled and $\mathcal{B}$ does not abort.
- $\mathcal{B}$ engages with $\mathcal{A}$ on protocol $\Pi_{CP_1}.\mathsf{ctCnd}$ that outputs the tuple $(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1)$. Henceforth, $\mathcal{B}$ submits the pair $(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1)$ to Π_{CP_1}.
- $\mathcal{A}$ outputs tx_{red}^1. By assumption $\mathcal{A}$ wins the game $BSI^{CP} - G_2^{\tau<T}$, hence condition b_2 is fulfilled and $\mathcal{B}$ does not abort. Thereafter, $\mathcal{B}$ extracts w from $(tx_{\mathsf{cnd}}^1, tx_{\mathsf{red}}^1)$. Due to $\Pr[\mathsf{Bad}_1] \leq \mathsf{negl}(\lambda)$ it holds that $(\mathbb{C}, w) \in \mathcal{R}$. Hence, $\mathcal{B}$ forwards w to the challenger.
- Finally, the challenger uses w to output $(tx_{\mathsf{red}}, \sigma_{\mathsf{red}})$ and submits it to Π_{CP_0}.

If $\mathcal{A}$ makes a query to $\mathsf{Set}_C\mathcal{O}$, $\mathcal{B}$ follows all the steps of protocol Set as described in Figure 26. However, $\mathcal{B}$ does not have the private key sk_I^0, so when Set requires this input, $\mathcal{B}$ queries $\mathsf{ctCnd}_R\mathcal{O}$ for the output of $\Pi_{CP_0}.\mathsf{ctCnd}_R(sk_I^0, \mathbb{C}, T')$ and relays the answer. Similarly, when $\mathcal{A}$ makes a query to $\mathsf{Pay}_C\mathcal{O}$, $\mathcal{B}$ follows all the steps of protocol Pay as described in Figure 27. Nevertheless, $\mathcal{B}$ does not have the private key sk_I^0, hence it queries $\mathsf{Red}\mathcal{O}$ for the output of $\Pi_{CP_0}.\mathsf{Red}(tx_{\mathsf{cnd}}^0, sk_I^0, w, \mathsf{aux}^0, \ddot{pk}_I^0)$ and relays the answer. Note that $\mathcal{Q}$ is synchronized in both games.

Our adversary $\mathcal{B}$ perfectly simulates $BSI^{CP} - G_2^{\tau<T}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ can win $BSI^{CP} - G_2^{\tau<T}$ with non-negligible probability, this implies that (1) $tx_{\mathsf{cnd}}^0 \notin \mathcal{Q}$ (condition a_0); (2) $tx_{\mathsf{cnd}}^0 \in \Pi_{CP_0}.TX_L$ and $\mathsf{isSender}(tx_{\mathsf{cnd}}^0, pk_C^0) = 1$ (condition a_1); (3) $\mathsf{isCond}(tx_{\mathsf{cnd}}^0, \mathbb{C}) = 1$ and $\mathsf{isRcvr}(tx_{\mathsf{cnd}}^0, epk_{C,I}^0) = 1$ (condition a_2); (4) $\mathcal{A}$ did not use $\mathsf{Pay}_I\mathcal{O}$ to put $\overline{tx_{\mathsf{red}}}^0$ on Π_{CP_0} (condition b_0); (5) $tx_{\mathsf{red}}^0 \notin \Pi_{CP_0}.TX_L$ and $\mathsf{isLinked}(tx_{\mathsf{cnd}}^0, tx_{\mathsf{red}}^0) = 1$ (condition b_1); (6) $(\mathbb{C}, w) \in \mathcal{R}$ ($\Pr[\mathsf{Bad}_2] \leq \mathsf{negl}(\lambda)$); (7) timeout T has not expired (condition b_3). It is easy to see that these are equivalent to conditions b_0, b_1, b_2, b_3, b_4, b_5 and b_6 of game ExpRedeem. Furthermore, we know that $\mathcal{Q}$ is synchronized in both games. However, this result contradicts the assumption that Π_{CP} satisfies CP redeemability. Therefore, such $\mathcal{A}$ cannot exist, thus this claim has been proven. □

So far, we have obtained that

$$\begin{aligned}&\Pr[BSI^{CP} - G_0^{\tau<T}(\lambda) = 1] \leq \\ &\quad \Pr[BSI^{CP} - G_1^{\tau<T}(\lambda) = 1] + \Pr[\mathsf{Bad}_1] \leq \\ &\quad \Pr[BSI^{CP} - G_2^{\tau<T}(\lambda) = 1] + \Pr[\mathsf{Bad}_2] + \Pr[\mathsf{Bad}_1] \leq \mathsf{negl}(\lambda).\end{aligned}$$

This means that for the path where $\tau < T$, $\mathcal{A}$ can win with negligible probability.

We now explore the other path.

Claim 15. *Let ledger Π_{CP_1} provide CP refundability. Then* $\Pr[BSI^{CP} - G_0^{\tau\geq T}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[BSI^{CP} - G_0^{\tau\geq T}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win ExpRefund. $\mathcal{B}$ interacts with $\mathcal{A}$ over the ledgers Π_{CP_0} and Π_{CP_1} and with the challenger over the ledger Π_{CP_1}.

- Challenger provides $\mathcal{B}$ with public keys $(pk_S, \ddot{pk})$. $\mathcal{B}$ sets $pk_I^1 := pk_S$, $\ddot{pk}_I^1 := \ddot{pk}$ and generates (pk_I^0, sk_I^0).
- $\mathcal{B}$ forwards $(pk_I^0, pk_I^1, \ddot{pk}_I^1)$ to $\mathcal{A}$, which returns $(pk_C^0, pk_D^1, \mathbb{C}, T)$.
- $\mathcal{B}$ sets $pk_R := pk_D^1$ and forwards $(pk_R, \mathbb{C}, T)$ to the challenger.
- $\mathcal{B}$ engages with $\mathcal{A}$ on protocol $\Pi_{CP_0}.\mathsf{ctCnd}$ that outputs the tuple $(tx_{\mathsf{cnd}}^0, \sigma_{\mathsf{cnd}}^0, \mathsf{aux}^0)$. By assumption $\mathcal{A}$ wins the game $BSI^{CP} - G_2^{\tau<T}$, hence conditions a_1 and a_2 are fulfilled and $\mathcal{B}$ does not abort.
- $\mathcal{B}$ engages with the challenger and $\mathcal{A}$ on protocol $\Pi_{CP_1}.\mathsf{ctCnd}$ acting as a relay. This results in $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux}) := (tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1)$. Thereafter, the challenger submits $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}})$ to Π_{CP_1}.
- Finally, the challenger generates a refund transaction tx_{ref} from tx_{cnd} and submits $(tx_{\mathsf{ref}}, \sigma_{\mathsf{ref}})$ to Π_{CP_1}.

If $\mathcal{A}$ makes a query to $\mathsf{Set}_I\mathcal{O}$, $\mathcal{B}$ follows all the steps of protocol Set as described in Figure 26. However, $\mathcal{B}$ does not have the private key sk_I^1, so when Set requires this input, $\mathcal{B}$ queries $\mathsf{ctCnd}_S\mathcal{O}$ for the output of $\Pi_{CP_1}.\mathsf{ctCnd}_S(sk_I^1, \mathbb{C}, T)$ and relays the answer. If $\mathcal{A}$ makes a query to $\mathsf{Ref}_I\mathcal{O}$, $\mathcal{B}$ does not have the private key sk_I^1, hence it queries $\mathsf{Ref}\mathcal{O}$ and relays the answer. Note that $\mathcal{Q}$ is synchronized in both games.

Our adversary $\mathcal{B}$ perfectly simulates $BSI^{CP} - G_0^{\tau\geq T}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ is successful in wining $BSI^{CP} - G_0^{\tau\geq T}$ with non-negligible probability, this implies that (1) $tx_{\mathsf{cnd}}^1 \notin \mathcal{Q}$ (condition a_0); (2) $tx_{\mathsf{cnd}}^1 \in \Pi_{CP_1}.TX_L$ and $\mathsf{isSender}(tx_{\mathsf{cnd}}^1, pk_I^1) = 1$ (condition a_3); (3) $\mathsf{isCond}(tx_{\mathsf{cnd}}^1, \mathbb{C}) = 1$ and $\mathsf{isRcvr}(tx_{\mathsf{cnd}}^1, epk_{I,D}^1) = 1$ (condition a_4); (4) $\mathcal{A}$ did not put $\overline{tx_{\mathsf{red}}}^1$ on Π_{CP_1} (condition c_0); (5) $\mathcal{A}$ did not use $\mathsf{Ref}_I\mathcal{O}$ to put $\overline{tx_{\mathsf{ref}}}^1$ on Π_{CP_1} (condition c_1); (6) $tx_{\mathsf{ref}}^1 \notin \Pi_{CP_1}.TX_L$ and $\mathsf{isLinked}(tx_{\mathsf{cnd}}^1, tx_{\mathsf{ref}}^1) = 1$ (condition c_2); (7) timeoutn T has expired (condition c_3). , It is easy to see that these are equivalent to conditions b_0, b_1, b_2, b_3, b_4, b_5 and b_6 of game ExpRefund. Furthermore, we know that $\mathcal{Q}$ is synchronized in both games. However, this result contradicts the assumption that Π_{CP_1} satisfies CP refundability. Therefore, such $\mathcal{A}$ cannot exist, thus this claim has been proven. □

$BSI^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}} - G_2^{\tau<T}(\lambda)$

$\mathcal{Q} := \emptyset$

$(pk_I^0, sk_I^0) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$

$(\ddot{pk}_I^0, \ddot{sk}_I^0) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$

$(pk_I^1, sk_I^1) \leftarrow \Pi_{CP_1}.\mathsf{ctAcc}(1^\lambda)$

$(pk_C^0, pk_D^1, \mathbb{C}, T, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{Set}_I O, \mathsf{Pay}_I O}(pk_I^0, pk_I^1, \ddot{pk}_I^0)$

$T' := T + \delta$

$\overline{tx_{\mathsf{cnd}}}^1 \leftarrow \mathcal{A}(\mathsf{st}_0)$

$\hat{c}_0 := \overline{tx_{\mathsf{cnd}}}^1 \notin \mathcal{Q}$

$\hat{c}_1 := \Pi_{CP_1}.\mathsf{ckTx}(\overline{tx_{\mathsf{cnd}}}^1) \wedge \mathsf{isSender}(\overline{tx_{\mathsf{cnd}}}^1, pk_I^1)$

$\hat{c}_2 := \mathsf{isCond}(\overline{tx_{\mathsf{cnd}}}^1, \mathbb{C})$

if $\hat{c}_0 \wedge \hat{c}_1 \wedge \hat{c}_2$ **abort**

$(tx^0_{\mathsf{cnd}}, \sigma^0_{\mathsf{cnd}}, \mathsf{aux}^0, \mathsf{st}_1) \leftarrow \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_0), \\ \Pi_{CP_0}.\mathsf{ctCnd}_R(sk_I^0, \mathbb{C}, T') \end{matrix} \right\rangle$

if $\neg(a_1 \wedge a_2)$ **abort**

$(tx^1_{\mathsf{cnd}}, \sigma^1_{\mathsf{cnd}}, \mathsf{aux}^1, \mathsf{st}_2) \leftarrow \left\langle \begin{matrix} \Pi_{CP_1}.\mathsf{ctCnd}_S(sk_I^1, \mathbb{C}, T), \\ \mathcal{A}(\mathsf{st}_1) \end{matrix} \right\rangle$

$\Pi_{CP_1}.\mathsf{subTx}(tx^1_{\mathsf{cnd}}, \sigma^1_{\mathsf{cnd}})$

$tx_{\mathsf{red}}1 \leftarrow \mathcal{A}(\mathsf{st}_2)$

if $\neg b_2$ **abort**

$w \leftarrow \Pi_{CP_1}.\mathsf{GetWit}(tx^1_{\mathsf{cnd}}, tx^1_{\mathsf{red}}, \mathsf{aux}^1)$

$\hat{c}_3 := (\mathbb{C}, w) \notin \mathcal{R}$

if $\hat{c}_3$ **abort**

$(tx^0_{\mathsf{red}}, \sigma^0_{\mathsf{red}}) \leftarrow \Pi_{CP_0}.\mathsf{Red}(tx^0_{\mathsf{cnd}}, sk_I^0, w, \mathsf{aux}^0, \ddot{pk}_I^0)$

$\Pi_{CP_0}.\mathsf{subTx}(tx^0_{\mathsf{red}}, \sigma^0_{\mathsf{red}})$

$a_0 := tx^0_{\mathsf{cnd}}, tx^1_{\mathsf{cnd}} \notin \mathcal{Q}$

$a_1 := \Pi_{CP_0}.\mathsf{ckTx}(tx^0_{\mathsf{cnd}}) \wedge \mathsf{isSender}(tx^0_{\mathsf{cnd}}, pk_C^0)$

$a_2 := \mathsf{isCond}(tx^0_{\mathsf{cnd}}, \mathbb{C}) \wedge \mathsf{isRcvr}(tx^0_{\mathsf{cnd}}, epk^0_{C,I})$

$a_3 := \Pi_{CP_1}.\mathsf{ckTx}(tx^1_{\mathsf{cnd}}) \wedge \mathsf{isSender}(tx^1_{\mathsf{cnd}}, pk_I^1)$

$a_4 := \mathsf{isCond}(tx^1_{\mathsf{cnd}}, \mathbb{C}) \wedge \mathsf{isRcvr}(tx^1_{\mathsf{cnd}}, epk^1_{I,D})$

$b_0 := \nexists\, \overline{tx_{\mathsf{red}}}^0 \; s.t. \; \overline{tx_{\mathsf{red}}}^0 \in \mathcal{Q} \wedge \mathsf{ckTx}(\overline{tx_{\mathsf{red}}}^0) \wedge$
$\quad \mathsf{isLinked}(tx^0_{\mathsf{cnd}}, \overline{tx_{\mathsf{red}}}^0)$

$b_1 := \Pi_{CP_0}.\mathsf{ckTx}(tx^0_{\mathsf{red}}) = 0 \wedge \mathsf{isLinked}(tx^0_{\mathsf{cnd}}, tx^0_{\mathsf{red}})$

$b_2 := \Pi_{CP_1}.\mathsf{ckTx}(tx^1_{\mathsf{red}}) \wedge \mathsf{isLinked}(tx^1_{\mathsf{cnd}}, tx^1_{\mathsf{red}})$

$b_3 := \mathtt{readTime}(\cdot) < T$

return $\bigwedge_{i=0}^{4} a_i \wedge \bigwedge_{i=0}^{3} b_i$

Figure 35: Experiment for $BSI^{CP} - G_2^{\tau<T}(\lambda)$.

Therefore, we reach to:

$$\begin{aligned}
&\Pr[BSI^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda)] \\
&\quad = \Pr[BSI^{CP} - G_0^{\tau<T}(\lambda) = 1 \vee BSI^{CP} - G_0^{\tau\geq T}(\lambda) = 1] \\
&\quad \leq \Pr[BSI^{CP} - G_0^{\tau<T}(\lambda) = 1] + \Pr[BSI^{CP} - G_0^{\tau\geq T}(\lambda) = 1] \\
&\quad \leq \mathsf{negl}(\lambda).
\end{aligned}$$

$BSI^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}} - G_0^{\tau\geq T}(\lambda)$

$\mathcal{Q} := \emptyset$

$(pk_I^0, sk_I^0) \leftarrow \Pi_{CP_0}.\mathsf{ctAcc}(1^\lambda)$

$(pk_I^1, sk_I^1) \leftarrow \Pi_{CP_1}.\mathsf{ctAcc}(1^\lambda)$

$(\ddot{pk}_I^1, \ddot{sk}_I^1) \leftarrow \Pi_{CP_1}.\mathsf{ctAcc}(1^\lambda)$

$(pk_C^0, pk_D^1, \mathbb{C}, T, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{Set}_I O, \mathsf{Ref}_I O}(pk_I^0, pk_I^1, \ddot{pk}_I^0)$

$T' := T + \delta$

$(tx^0_{\mathsf{cnd}}, \sigma^0_{\mathsf{cnd}}, \mathsf{aux}^0, \mathsf{st}_1) \leftarrow \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_0), \\ \Pi_{CP_0}.\mathsf{ctCnd}_R(sk_I^0, \mathbb{C}, T') \end{matrix} \right\rangle$

if $\neg(a_1 \wedge a_2)$ **abort**

$(tx^1_{\mathsf{cnd}}, \sigma^1_{\mathsf{cnd}}, \mathsf{aux}^1) \leftarrow \left\langle \begin{matrix} \Pi_{CP_1}.\mathsf{ctCnd}_S(sk_I^1, \mathbb{C}, T), \\ \mathcal{A}(\mathsf{st}_1) \end{matrix} \right\rangle$

$\Pi_{CP_1}.\mathsf{subTx}(tx^1_{\mathsf{cnd}}, \sigma^1_{\mathsf{cnd}})$

$(tx^1_{\mathsf{ref}}, \sigma^1_{\mathsf{ref}}) \leftarrow \Pi_{CP_1}.\mathsf{Ref}(tx^1_{\mathsf{cnd}}, sk_I^1, \mathsf{aux}^1, \ddot{pk}_I^1)$

$\Pi_{CP_1}.\mathsf{subTx}(tx^1_{\mathsf{ref}}, \sigma^1_{\mathsf{ref}})$

$a_0 := tx^0_{\mathsf{cnd}}, tx^1_{\mathsf{cnd}} \notin \mathcal{Q}$

$a_1 := \Pi_{CP_0}.\mathsf{ckTx}(tx^0_{\mathsf{cnd}}) \wedge \mathsf{isSender}(tx^0_{\mathsf{cnd}}, pk_C^0)$

$a_2 := \mathsf{isCond}(tx^0_{\mathsf{cnd}}, \mathbb{C}) \wedge \mathsf{isRcvr}(tx^0_{\mathsf{cnd}}, epk^0_{C,I})$

$a_3 := \Pi_{CP_1}.\mathsf{ckTx}(tx^1_{\mathsf{cnd}}) \wedge \mathsf{isSender}(tx^1_{\mathsf{cnd}}, pk_I^1)$

$a_4 := \mathsf{isCond}(tx^1_{\mathsf{cnd}}, \mathbb{C}) \wedge \mathsf{isRcvr}(tx^1_{\mathsf{cnd}}, epk^1_{I,D})$

$c_0 := \nexists\, \overline{tx_{\mathsf{red}}}^1 \; s.t. \; \mathsf{ckTx}(\overline{tx_{\mathsf{red}}}^1) \wedge \mathsf{isLinked}(tx^1_{\mathsf{cnd}}, \overline{tx_{\mathsf{red}}}^1)$

$c_1 := \nexists\, \overline{tx_{\mathsf{ref}}}^1 \; s.t. \; \overline{tx_{\mathsf{ref}}}^1 \in \mathcal{Q} \wedge \mathsf{ckTx}(\overline{tx_{\mathsf{ref}}}^1) \wedge$
$\quad \mathsf{isLinked}(tx^1_{\mathsf{cnd}}, \overline{tx_{\mathsf{ref}}}^1)$

$c_2 := \Pi_{CP_1}.\mathsf{ckTx}(tx^1_{\mathsf{ref}}) = 0 \wedge \mathsf{isLinked}(tx^1_{\mathsf{cnd}}, tx^1_{\mathsf{ref}})$

$c_3 := \mathtt{readTime}(\cdot) \geq T$

return $\bigwedge_{i=0}^{4} a_i \wedge \bigwedge_{i=0}^{3} c_i$

Figure 36: Experiment for $BSI^{CP} - G_0^{\tau\geq T}(\lambda)$.

This concludes the proof of Lemma 5. □

Lemma 6. *(Balance security for D) For all PPT adversaries* $\mathcal{A}$*, it holds that* $\Pr[BSD^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$

Proof. The game BSD^{CP} contains two mutually exclusive paths that the adversary $\mathcal{A}$ could exploit. We will prove Lemma 6 by performing reductions on the two possible paths since:

$$\begin{aligned}
&\Pr[BSD^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda)] = \\
&\quad \Pr[BSD^{CP} - G_0^{\tau<T}(\lambda) = 1 \vee BSD^{CP} - G_0^{\tau\geq T+2\delta}(\lambda) = 1] \leq \\
&\quad \Pr[BSD^{CP} - G_0^{\tau<T}(\lambda) = 1] + \Pr[BSD^{CP} - G_0^{\tau\geq T+2\delta}(\lambda) = 1]
\end{aligned}$$

In the path where $\tau < T$, we perform the following game hops:

Game $BSD^{CP} - G_0^{\tau<T}$: This game, formally defined in Figure 37, corresponds to the original game for Balance security for D, restricted to the path where $\tau < T$. The game is expanded with the interactions described in our implementation.

$BSD^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}} - G_0^{\tau<T}(\lambda)$

$\mathcal{Q} := \emptyset$

$(pk_D^1, sk_D^1) \leftarrow \Pi_{CP_1}.\mathsf{ctAcc}(1^\lambda)$

$(\ddot{pk}_D^1, \ddot{sk}_D^1) \leftarrow \Pi_{CP_1}.\mathsf{ctAcc}(1^\lambda)$

$(pk_D^{\mathsf{cbl}}, sk_D^{\mathsf{cbl}}) \leftarrow \mathsf{cbl.ctAcc}(1^\lambda)$

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

$(pk_I^1, pk_C^{\mathsf{cbl}}, T, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{Set}_D O, \mathsf{Pay}_D O}(pk_D^1, pk_D^{\mathsf{cbl}}, \ddot{pk}_D^1, \mathbb{C})$

$T'' := T + 2\delta$

$(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathsf{aux}^{\mathsf{cbl}}, \mathsf{st}_1) \leftarrow \left\langle \begin{matrix} \mathsf{cbl.ctCnd}_S(sk_D^{\mathsf{cbl}}, \mathbb{C}, T''), \\ \mathcal{A}(\mathsf{st}_0) \end{matrix} \right\rangle$

$\mathsf{cbl.subTx}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}})$

$(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1) \leftarrow \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_1), \\ \Pi_{CP_1}.\mathsf{ctCnd}_R(sk_D^1, \mathbb{C}, T) \end{matrix} \right\rangle$

if $\neg(a_1 \wedge a_2)$ **abort**

$(tx_{\mathsf{red}}^1, \sigma_{\mathsf{red}}^1) \leftarrow \Pi_{CP_1}.\mathsf{Red}(tx_{\mathsf{cnd}}^1, sk_D^1, w, \mathsf{aux}^1, \ddot{pk}_D^1)$

$\Pi_{CP_1}.\mathsf{subTx}(tx_{\mathsf{red}}^1, \sigma_{\mathsf{red}}^1)$

$a_0 := tx_{\mathsf{cnd}}^1, tx_{\mathsf{cnd}}^{\mathsf{cbl}} \notin \mathcal{Q}$

$a_1 := \Pi_{CP_1}.\mathsf{ckTx}(tx_{\mathsf{cnd}}^1) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^1, pk_I^1)$

$a_2 := \mathsf{isCond}(tx_{\mathsf{cnd}}^1, \mathbb{C}) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^1, epk_{I,D}^1)$

$a_3 := \mathsf{cbl.ckTx}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, pk_D^{\mathsf{cbl}})$

$a_4 := \mathsf{isCond}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathbb{C}) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, epk_{D,C}^{\mathsf{cbl}})$

$b_0 := \nexists\, \overline{tx_{\mathsf{red}}}^1 \; s.t. \; \overline{tx_{\mathsf{red}}}^1 \in \mathcal{Q} \wedge \mathsf{ckTx}(\overline{tx_{\mathsf{red}}}^1) \wedge$
$\mathsf{isLinked}(tx_{\mathsf{red}}^1, \overline{tx_{\mathsf{red}}}^1)$

$b_1 := \Pi_{CP_1}.\mathsf{ckTx}(tx_{\mathsf{red}}^1) = 0 \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^1, tx_{\mathsf{red}}^1)$

$b_2 := \mathsf{cbl.ckTx}(tx_{\mathsf{red}}^{\mathsf{cbl}}) \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, tx_{\mathsf{red}}^{\mathsf{cbl}})$

$b_3 := \mathtt{readTime}(\cdot) < T$

return $\bigwedge_{i=0}^{4} a_i \wedge \bigwedge_{i=0}^{3} b_i$

Figure 37: Experiment for $BSD^{CP} - G_0^{\tau<T}(\lambda)$.

Game $BSD^{CP} - G_1^{\tau<T}$: This game, formally defined in Figure 38, works exactly as $BSD^{CP} - G_0^{\tau<T}$ with the following exception (highlighted in the gray lines). After the D has locked its funds on cbl, $\mathcal{A}$ produces a forgery $\overline{tx_{\mathsf{red}}}^{\mathsf{cbl}}$. This allows $\mathcal{A}$ to claim D's funds on cbl without first to locking its own funds in Π_{CP_1}. If this forgery is valid, $BSD^{CP} - G_1^{\tau<T}$ aborts.

On the path where $\tau \geq T + 2\delta$, we only consider the following game:

Game $BSD^{CP} - G_0^{\tau \geq T+2\delta}$: This game, formally defined in Figure 39, corresponds to the original game for Balance security for D, restricted to the path where $\tau \geq T + 2\delta$. The game is expanded with the interactions described in our implementation.

Claim 16. *Let* Bad_1 *be the event that* $BSD^{CP} - G_1^{\tau<T}$ *aborts on gray lines of Figure 38. Assume that ledger* Π_{CP_1} *provides CP witness unforgeability, 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 $\mathcal{B}$ that uses $\mathcal{A}$ to win wForge. $\mathcal{B}$ interacts with $\mathcal{A}$ over the ledgers Π_{CP_1} and cbl and with the challenger over the ledger cbl.

- Challenger provides $\mathcal{B}$ with a sender public key pk_S and $\mathbb{C}$. $\mathcal{B}$ sets $pk_D^{\mathsf{cbl}} := pk_S$ and generates (pk_D^1, sk_D^1) and $(\ddot{pk}_D^1, \ddot{sk}_D^1)$.
- $\mathcal{B}$ forwards $(pk_D^1, pk_D^{\mathsf{cbl}}, \ddot{pk}_D^1, \mathbb{C})$ to $\mathcal{A}$, which returns $(pk_I^1, pk_C^{\mathsf{cbl}}, T)$.
- $\mathcal{B}$ sets $pk_R := pk_C^{\mathsf{cbl}}$, $T'' := T + 1\delta$ and forwards (pk_R, T'') to the challenger.
- $\mathcal{B}$ engages with the challenger and $\mathcal{A}$ on protocol cbl.ctCnd acting as a relay. This results in $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux}) := (tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathsf{aux}^{CL})$. Thereafter, the challenger submits $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}})$ to cbl.
- $\mathcal{A}$ produces the forgery $\overline{tx_{\mathsf{red}}}^{\mathsf{cbl}}$ on cbl. Thereafter, $\mathcal{B}$ forwards $\overline{tx_{\mathsf{red}}} := \overline{tx_{\mathsf{red}}}^{\mathsf{cbl}}$ to the challenger.

If $\mathcal{A}$ makes a query to $\mathsf{Set}_D O$, $\mathcal{B}$ follows all the steps of protocol Set as described in Figure 26. However, $\mathcal{B}$ does not have the private key sk_D^{cbl}, so when Set requires this input, $\mathcal{B}$ queries $\mathsf{ctCnd}_S O$ for the output of $\mathsf{cbl.ctCnd}_S(sk_D^{\mathsf{cbl}}, \mathbb{C}, T'')$ and relays the answer. Note that $\mathcal{Q}$ is synchronized in both games. If $\mathcal{A}$ makes a query to $\mathsf{Pay}_D O$, $\mathcal{B}$ follows all the steps of protocol Set, as described in Figure 27.

Our adversary $\mathcal{B}$ perfectly simulates $BSD^{CP} - G_1^{\tau<T}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ can force $\mathcal{B}$ to abort on the gray lines of Figure 34, it implies that (1) $tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \overline{tx_{\mathsf{red}}}^{\mathsf{cbl}} \notin \mathcal{Q}$ (conditions a_0 and $\hat{c}_0$); (2) $tx_{\mathsf{cnd}}^{\mathsf{cbl}} \in \mathsf{cbl}.TX_L$ and $\mathsf{isSender}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, pk_D^{\mathsf{cbl}}) = 1$ (condition a_3); (3) $\mathsf{isCond}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathbb{C}) = 1$ and $\mathsf{isRcvr}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, epk_{D,C}^{\mathsf{cbl}}) = 1$ (condition a_4); (4) $\mathsf{cbl.ckTx}(\overline{tx_{\mathsf{red}}}^{\mathsf{cbl}}) = 0$ and $\mathsf{isLinked}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \overline{tx_{\mathsf{red}}}^{\mathsf{cbl}}) = 1$ (condition $\hat{c}_1$). (5) timeout T has not expired (condition b_3). It is easy to see that these are equivalent to conditions b_0, b_1, b_2, b_3 and b_4 of game wForge. Furthermore, we know that $\mathcal{Q}$ is synchronized in both games. However, this result contradicts the assumption that cbl satisfies CP witness unforgeability. Therefore, such $\mathcal{A}$ cannot exist, thus this claim has been proven. □

Since games $BSD^{CP} - G_0^{\tau<T}$ and $BSD - G_1^{\tau<T}$ are equivalent except for event Bad_1 occuring, it holds that

$$\Pr[BSD^{CP} - G_0^{\tau<T}(\lambda) = 1] \leq \Pr[BSD^{CP} - G_1^{\tau<T}(\lambda) = 1] + \mathsf{negl}(\lambda).$$

Claim 17. *Let ledger* Π_{CP_1} *provide CP redeemability. Then* $\Pr[BSD^{CP} - G_1^{\tau<T}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[BSD^{CP} - G_1^{\tau<T}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win ExpRedeem. $\mathcal{B}$ interacts with $\mathcal{A}$ over the ledgers Π_{CP_1} and cbl and with the challenger over the ledger Π_{CP_1}.

- Challenger provides $\mathcal{B}$ with public keys $(pk_R, \ddot{pk})$. $\mathcal{B}$ sets $pk_D^1 := pk_R$, $\ddot{pk}_D^1 := \ddot{pk}$ and generates $(pk_D^{\mathsf{cbl}}, sk_D^{\mathsf{cbl}})$ and $(\mathbb{C}, w) \in \mathcal{R}$.
- $\mathcal{B}$ forwards $(pk_D^1, pk_D^{\mathsf{cbl}}, \ddot{pk}_D^1, \mathbb{C})$ to $\mathcal{A}$, which returns $(pk_I^1, pk_C^{\mathsf{cbl}}, T)$.
- $\mathcal{B}$ sets $pk_S := pk_I^1$ and forwards $(pk_S, \mathbb{C}, T)$ to the challenger.
- $\mathcal{B}$ engages with $\mathcal{A}$ on protocol cbl.ctCnd that outputs the tuple $(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathsf{aux}^{\mathsf{cbl}})$. Henceforth, $\mathcal{B}$ submits the pair $(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}})$ to cbl.

- $\mathcal{A}$ produces the forgery $\overline{tx_{\mathsf{red}}}^{\mathsf{cbl}}$ on cbl. Due to $\Pr[\mathsf{Bad}_1] \leq \mathsf{negl}(\lambda)$, this forgery does not make the game to abort.
- $\mathcal{B}$ engages with the challenger and $\mathcal{A}$ on protocol $\Pi_{CP_1}.\mathsf{ctCnd}$ acting as a relay. This results in $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux}) := (tx^1_{\mathsf{cnd}}, \sigma^1_{\mathsf{cnd}}, \mathsf{aux}^1)$. By assumption $\mathcal{A}$ wins the game $BSD^{CP} - G_1^{\tau<T}$, hence conditions a_1 and a_2 are fulfilled and $\mathcal{B}$ does not abort.
- Finally, $\mathcal{B}$ forwards w to the challenger. The latter uses w to output $(tx_{\mathsf{red}}, \sigma_{\mathsf{red}})$ and submits it to cbl.

If $\mathcal{A}$ makes a query to $\mathsf{Set}_D\mathcal{O}$, $\mathcal{B}$ follows all the steps of protocol Set as described in Figure 26. However, $\mathcal{B}$ does not have the private key sk^1_D, so when Set requires this input, $\mathcal{B}$ queries $\mathsf{ctCnd}_R\mathcal{O}$ for the output of $\Pi_{CP_1}.\mathsf{ctCnd}_R(sk^1_D, \mathbb{C}, T)$ and relays the answer. Similarly, when $\mathcal{A}$ makes a query to $\mathsf{Pay}_C\mathcal{O}$, $\mathcal{B}$ follows all the steps of protocol Pay as described in Figure 27. Nevertheless, $\mathcal{B}$ does not have the private key sk^1_D, hence it queries $\mathsf{Red}\mathcal{O}$ for the output of $\Pi_{CP_1}.\mathsf{Red}(tx^1_{\mathsf{cnd}}, sk^1_D, w, \mathsf{aux}^1, \ddot{pk}^1_D)$ and relays the answer. Note that $\mathcal{Q}$ is synchronized in both games.

Our adversary $\mathcal{B}$ perfectly simulates $BSI - G_1^{\tau<T}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ can win $BSI - G_1^{\tau<T}$ with non-negligible probability, this implies that (1) $tx^1_{\mathsf{cnd}} \notin \mathcal{Q}$ (condition a_0); (2) $tx^1_{\mathsf{cnd}} \in \Pi_{CP_1}.TX_L$ and $\mathsf{isSender}(tx^1_{\mathsf{cnd}}, pk^1_D) = 1$ (condition a_1); (3) $\mathsf{isCond}(tx^1_{\mathsf{cnd}}, \mathbb{C}) = 1$ and $\mathsf{isRcvr}(tx^1_{\mathsf{cnd}}, epk^1_{I,D}) = 1$ (condition a_2); (4) $\mathcal{A}$ did not use $\mathsf{Pay}_D\mathcal{O}$ to put $\overline{tx_{\mathsf{red}}}^1$ on Π_{CP_1} (condition b_0); (5) $tx^1_{\mathsf{red}} \notin \Pi_{CP_1}.TX_L$ and $\mathsf{isLinked}(tx^1_{\mathsf{cnd}}, tx^1_{\mathsf{red}}) = 1$ (condition b_1); (6) $(\mathbb{C}, w) \in \mathcal{R}$; (7) timeout T has not expired (condition b_3). It is easy to see that these are equivalent to conditions b_0, b_1, b_2, b_3, b_4 and b_5 of game ExpRedeem. Furthermore, we know that $\mathcal{Q}$ is synchronized in both games. However, this result contradicts the assumption that Π_{CP_1} satisfies CP redeemability. Therefore, such $\mathcal{A}$ cannot exist, thus this claim has been proven. □

So far, we have obtained that

$$\Pr[BSD^{CP} - G_0^{\tau<T}(\lambda) = 1] \leq \Pr[BSD^{CP} - G_1^{\tau<T}(\lambda) = 1] + \Pr[\mathsf{Bad}_1] \leq \mathsf{negl}(\lambda).$$

This means that for the path where $\tau < T$, $\mathcal{A}$ can win with negligible probability.

We now explore the other path.

Claim 18. *Let ledger Π_{CP_1} provide CP refundability. Then* $\Pr[BSD^{CP} - G_0^{\tau\geq T+2\delta}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[BSD^{CP} - G_0^{\tau\geq T+2\delta}(\lambda) = 1] > \mathsf{negl}(\lambda)$. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to win ExpRefund. $\mathcal{B}$ interacts with $\mathcal{A}$ over the ledgers Π_{CP_1} and cbl and with the challenger over the ledger cbl.

- Challenger provides $\mathcal{B}$ with public keys $(pk_S, \ddot{pk})$. $\mathcal{B}$ sets $pk^{\mathsf{cbl}}_D := pk_S$, $\ddot{pk}^{\mathsf{cbl}}_D := \ddot{pk}$ and generates (pk^1_D, sk^1_D) and $(\mathbb{C}, w) \in \mathcal{R}$.
- $\mathcal{B}$ forwards $(pk^1_D, pk^{\mathsf{cbl}}_D, \ddot{pk}^{\mathsf{cbl}}_D, \mathbb{C})$ to $\mathcal{A}$, which returns $(pk^1_I, pk^{\mathsf{cbl}}_C, T)$.
- $\mathcal{B}$ sets $pk_R := pk^{\mathsf{cbl}}_C$, $T'' := T + 2\delta$ and forwards $(pk_R, \mathbb{C}, T'')$ to the challenger.
- $\mathcal{B}$ engages with the challenger and $\mathcal{A}$ on protocol cbl.ctCnd acting as a relay. This results in $(tx_{\mathsf{cnd}}, \sigma_{\mathsf{cnd}}, \mathsf{aux}) := (tx^{\mathsf{cbl}}_{\mathsf{cnd}}, \sigma^{\mathsf{cbl}}_{\mathsf{cnd}}, \mathsf{aux}^{\mathsf{cbl}})$. Thereafter, the challenger submits $(paird)$ to cbl.

$BSD^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}} - G_1^{\tau<T}(\lambda)$

$\mathcal{Q} := \emptyset$

$(pk^1_D, sk^1_D) \leftarrow \Pi_{CP_1}.\mathsf{ctAcc}(1^\lambda)$

$(\ddot{pk}^1_D, \ddot{sk}^1_D) \leftarrow \Pi_{CP_1}.\mathsf{ctAcc}(1^\lambda)$

$(pk^{\mathsf{cbl}}_D, sk^{\mathsf{cbl}}_D) \leftarrow \mathsf{cbl.ctAcc}(1^\lambda)$

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

$(pk^1_I, pk^{\mathsf{cbl}}_C, T, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{Set}_D\mathcal{O},\mathsf{Pay}_D\mathcal{O}}(pk^1_D, pk^{\mathsf{cbl}}_D, \ddot{pk}^1_D, \mathbb{C})$

$T'' := T + 2\delta$

$(tx^{\mathsf{cbl}}_{\mathsf{cnd}}, \sigma^{\mathsf{cbl}}_{\mathsf{cnd}}, \mathsf{aux}^{\mathsf{cbl}}, \mathsf{st}_1) \leftarrow \left\langle \begin{matrix} \mathsf{cbl.ctCnd}_S(sk^{\mathsf{cbl}}_D, \mathbb{C}, T''), \\ \mathcal{A}(\mathsf{st}_0) \end{matrix} \right\rangle$

$\mathsf{cbl.subTx}(tx^{\mathsf{cbl}}_{\mathsf{cnd}}, \sigma^{\mathsf{cbl}}_{\mathsf{cnd}})$

$\overline{tx_{\mathsf{red}}}^{\mathsf{cbl}} \leftarrow \mathcal{A}(\mathsf{st}_1)$

$\hat{c}_0 := \overline{tx_{\mathsf{red}}}^{\mathsf{cbl}} \notin \mathcal{Q}$

$\hat{c}_1 := \mathsf{cbl.ckTx}(\overline{tx_{\mathsf{red}}}^{\mathsf{cbl}}) \wedge \mathsf{isLinked}(tx^{\mathsf{cbl}}_{\mathsf{cnd}}, \overline{tx_{\mathsf{red}}}^{\mathsf{cbl}})$

if $\hat{c}_0 \wedge \hat{c}_1$ **abort**

$(tx^1_{\mathsf{cnd}}, \sigma^1_{\mathsf{cnd}}, \mathsf{aux}^1) \leftarrow \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_1), \\ \Pi_{CP_1}.\mathsf{ctCnd}_R(sk^1_D, \mathbb{C}, T) \end{matrix} \right\rangle$

if $\neg(a_1 \wedge a_2)$ **abort**

$(tx^1_{\mathsf{red}}, \sigma^1_{\mathsf{red}}) \leftarrow \Pi_{CP_1}.\mathsf{Red}(tx^1_{\mathsf{cnd}}, sk^1_D, w, \mathsf{aux}^1, \ddot{pk}^1_D)$

$\Pi_{CP_1}.\mathsf{subTx}(tx^1_{\mathsf{red}}, \sigma^1_{\mathsf{red}})$

$a_0 := tx^1_{\mathsf{cnd}}, tx^{\mathsf{cbl}}_{\mathsf{cnd}} \notin \mathcal{Q}$

$a_1 := \Pi_{CP_1}.\mathsf{ckTx}(tx^1_{\mathsf{cnd}}) \wedge \mathsf{isSender}(tx^1_{\mathsf{cnd}}, pk^1_I)$

$a_2 := \mathsf{isCond}(tx^1_{\mathsf{cnd}}, \mathbb{C}) \wedge \mathsf{isRcvr}(tx^1_{\mathsf{cnd}}, epk^1_{I,D})$

$a_3 := \mathsf{cbl.ckTx}(tx^{\mathsf{cbl}}_{\mathsf{cnd}}) \wedge \mathsf{isSender}(tx^{\mathsf{cbl}}_{\mathsf{cnd}}, pk^{\mathsf{cbl}}_D)$

$a_4 := \mathsf{isCond}(tx^{\mathsf{cbl}}_{\mathsf{cnd}}, \mathbb{C}) \wedge \mathsf{isRcvr}(tx^{\mathsf{cbl}}_{\mathsf{cnd}}, epk^{\mathsf{cbl}}_{D,C})$

$b_0 := \nexists\, \overline{tx_{\mathsf{red}}}^1 \; s.t. \; \overline{tx_{\mathsf{red}}}^1 \in \mathcal{Q} \wedge \mathsf{ckTx}(\overline{tx_{\mathsf{red}}}^1) \wedge \mathsf{isLinked}(tx^1_{\mathsf{red}}, \overline{tx_{\mathsf{red}}}^1)$

$b_1 := \Pi_{CP_1}.\mathsf{ckTx}(tx^1_{\mathsf{red}}) = 0 \wedge \mathsf{isLinked}(tx^1_{\mathsf{cnd}}, tx^1_{\mathsf{red}})$

$b_2 := \mathsf{cbl.ckTx}(tx^{\mathsf{cbl}}_{\mathsf{red}}) \wedge \mathsf{isLinked}(tx^{\mathsf{cbl}}_{\mathsf{cnd}}, tx^{\mathsf{cbl}}_{\mathsf{red}})$

$b_3 := \mathtt{readTime}(\cdot) < T$

return $\bigwedge_{i=0}^{4} a_i \wedge \bigwedge_{i=0}^{3} b_i$

Figure 38: Experiment for $BSD^{CP} - G_1^{\tau<T}(\lambda)$.

- $\mathcal{B}$ engages with $\mathcal{A}$ on protocol $\Pi_{CP_1}.\mathsf{ctCnd}$ that outputs the tuple $(tx^1_{\mathsf{cnd}}, \sigma^1_{\mathsf{cnd}}, \mathsf{aux}^1)$. By assumption $\mathcal{A}$ wins the game $BSD^{CP} - G_0^{\tau<T}$, hence conditions a_1 and a_2 are fulfilled and $\mathcal{B}$ does not abort.
- Finally, the challenger generates a refund transaction tx_{ref} from and tx_{cnd} and submits $(tx_{\mathsf{red}}, \sigma_{\mathsf{red}})$ to cbl.

If $\mathcal{A}$ makes a query to $\mathsf{Set}_D\mathcal{O}$, $\mathcal{B}$ follows all the steps of protocol Set as described in Figure 26. However, $\mathcal{B}$ does not have the private key sk^{cbl}_D, so when Set requires this input, $\mathcal{B}$ queries $\mathsf{ctCnd}_S\mathcal{O}$ for the output of $\mathsf{cbl.ctCnd}_S(sk^{\mathsf{cbl}}_D, \mathbb{C}, T'')$ and relays the answer. If $\mathcal{A}$ makes a query to $\mathsf{Ref}_D\mathcal{O}$, $\mathcal{B}$ does not have the private key

$\underline{BSD^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}} - G_0^{\tau \geq T+2\delta}(\lambda)}$

$\mathcal{Q} := \emptyset$

$(pk_D^1, sk_D^1) \leftarrow \Pi_{CP_1}.\mathsf{ctAcc}(1^\lambda)$

$(pk_D^{\mathsf{cbl}}, sk_D^{\mathsf{cbl}}) \leftarrow \mathsf{cbl.ctAcc}(1^\lambda)$

$(\ddot{pk}_D^{\mathsf{cbl}}, \ddot{sk}_D^{\mathsf{cbl}}) \leftarrow \mathsf{cbl.ctAcc}(1^\lambda)$

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

$(pk_I^1, pk_C^{\mathsf{cbl}}, T, \mathsf{st}_0) \leftarrow \mathcal{A}^{\mathsf{Set}_D\mathcal{O},\mathsf{Ref}_D\mathcal{O}}(pk_D^1, pk_D^{\mathsf{cbl}}, \ddot{pk}_D^{\mathsf{cbl}}, \mathbb{C})$

$T'' := T + 2\delta$

$(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathsf{aux}^{\mathsf{cbl}}, \mathsf{st}_1) \leftarrow \left\langle \begin{matrix} \mathsf{cbl.ctCnd}_S(sk_D^{\mathsf{cbl}}, \mathbb{C}, T''), \\ \mathcal{A}(\mathsf{st}_0) \end{matrix} \right\rangle$

$\mathsf{cbl.subTx}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \sigma_{\mathsf{cnd}}^{\mathsf{cbl}})$

$(tx_{\mathsf{cnd}}^1, \sigma_{\mathsf{cnd}}^1, \mathsf{aux}^1) \leftarrow \left\langle \begin{matrix} \mathcal{A}(\mathsf{st}_1), \\ \Pi_{CP_1}.\mathsf{ctCnd}_R(sk_D^1, \mathbb{C}, T) \end{matrix} \right\rangle$

if $\neg(a_1 \wedge a_2)$ **abort**

$(tx_{\mathsf{ref}}^{\mathsf{cbl}}, \sigma_{\mathsf{ref}}^{\mathsf{cbl}}) \leftarrow \mathsf{cbl.Ref}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, sk_D^{\mathsf{cbl}}, \mathsf{aux}^{\mathsf{cbl}}, \ddot{pk}_D^{\mathsf{cbl}})$

$\mathsf{cbl.subTx}(tx_{\mathsf{ref}}^{\mathsf{cbl}}, \sigma_{\mathsf{ref}}^{\mathsf{cbl}})$

$a_0 := tx_{\mathsf{cnd}}^1, tx_{\mathsf{cnd}}^{\mathsf{cbl}} \notin \mathcal{Q}$

$a_1 := \Pi_{CP_1}.\mathsf{ckTx}(tx_{\mathsf{cnd}}^1) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^1, pk_I^1)$

$a_2 := \mathsf{isCond}(tx_{\mathsf{cnd}}^1, \mathbb{C}) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^1, epk_{I,D}^1)$

$a_3 := \mathsf{cbl.ckTx}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}) \wedge \mathsf{isSender}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, pk_D^{\mathsf{cbl}})$

$a_4 := \mathsf{isCond}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathbb{C}) \wedge \mathsf{isRcvr}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, epk_{D,C}^{\mathsf{cbl}})$

$c_0 := \nexists\, \overline{tx_{\mathsf{red}}}^{\mathsf{cbl}}\ s.t.\ \mathsf{ckTx}(\overline{tx_{\mathsf{red}}}^{\mathsf{cbl}}) \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \overline{tx_{\mathsf{red}}}^{\mathsf{cbl}})$

$c_1 := \nexists\, \overline{tx_{\mathsf{ref}}}^{\mathsf{cbl}}\ s.t.\ \overline{tx_{\mathsf{ref}}}^{\mathsf{cbl}} \in \mathcal{Q} \wedge \mathsf{ckTx}(\overline{tx_{\mathsf{ref}}}^{\mathsf{cbl}}) \wedge$
$\quad \mathsf{isLinked}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \overline{tx_{\mathsf{ref}}}^{\mathsf{cbl}})$

$c_2 := \mathsf{cbl.ckTx}(tx_{\mathsf{ref}}^{\mathsf{cbl}}) = 0 \wedge \mathsf{isLinked}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, tx_{\mathsf{ref}}^{\mathsf{cbl}})$

$c_3 := \mathtt{readTime}(\cdot) \geq T + 2\delta$

return $\bigwedge_{i=0}^{4} a_i \wedge \bigwedge_{i=0}^{3} c_i$

Figure 39: Experiment for $BSD^{CP} - G_0^{\tau \geq T+2\delta}(\lambda)$.

sk_D^{cbl}, hence it queries $\mathsf{Ref}\mathcal{O}$ and relays the answer. Note that $\mathcal{Q}$ is synchronized in both games. If $\mathcal{A}$ makes a query to $\mathsf{Pay}_D\mathcal{O}$, $\mathcal{B}$ follows all the steps of protocol Set, as described in Figure 27.

Our adversary $\mathcal{B}$ perfectly simulates $BSD^{CP} - G_0^{\tau \geq T+2\delta}$ to $\mathcal{A}$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Now, if $\mathcal{A}$ is successful in wining $BSD^{CP} - G_0^{\tau \geq T+2\delta}$ with non-negligible probability, this implies that (1) $tx_{\mathsf{cnd}}^{\mathsf{cbl}} \notin \mathcal{Q}$ (condition a_0); (2) $tx_{\mathsf{cnd}}^{\mathsf{cbl}} \in \mathsf{cbl}.TX_L$ and $\mathsf{isSender}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, pk_D^{\mathsf{cbl}}) = 1$ (condition a_3); (3) $\mathsf{isCond}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, \mathbb{C}) = 1$ and $\mathsf{isRcvr}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, epk_{D,C}^{\mathsf{cbl}}) = 1$ (condition a_4); (4) $\mathcal{A}$ did not put $\overline{tx_{\mathsf{red}}}^{\mathsf{cbl}}$ on cbl (condition c_0); (5) $\mathcal{A}$ did not use $\mathsf{Ref}_D\mathcal{O}$ to put $\overline{tx_{\mathsf{ref}}}^{\mathsf{cbl}}$ on cbl (condition c_1); (6) $tx_{\mathsf{ref}}^{\mathsf{cbl}} \notin \mathsf{cbl}.TX_L$ and $\mathsf{isLinked}(tx_{\mathsf{cnd}}^{\mathsf{cbl}}, tx_{\mathsf{ref}}^{\mathsf{cbl}}) = 1$ (condition c_2); (7) timeout $T + 2\delta$ has expired (condition c_3). , It is easy to see that these are equivalent to conditions b_0, b_1, b_2, b_3, b_4, b_5 and b_6 of game ExpRefund. Furthermore, we know that $\mathcal{Q}$ is synchronized in both games. However, this result contradicts the assumption that cbl satisfies CP refundability. Therefore, such $\mathcal{A}$ cannot exist, thus this claim has been proven. □

Therefore, we reach to:

$$\begin{aligned}
&\Pr[BSD^{CP}_{\Pi_{CP_0},\Pi_{CP_1},\mathsf{cbl},\mathcal{A}}(\lambda)] \\
&\quad = \Pr[BSD^{CP} - G_0^{\tau<T}(\lambda) = 1 \vee BSD^{CP} - G_0^{\tau \geq T+2\delta}(\lambda) = 1] \\
&\quad \leq \Pr[BSD^{CP} - G_0^{\tau<T}(\lambda) = 1] + \Pr[BSD^{CP} - G_0^{\tau \geq T+2\delta}(\lambda) = 1] \\
&\quad \leq \mathsf{negl}(\lambda).
\end{aligned}$$

This concludes the proof of Lemma 6. □

And this concludes the proof of Theorem 19. □