P2C2T: Preserving the Privacy of Cross-Chain Transfer

Panpan Han*, Zheng Yan*†✉, Laurence T. Yang‡, and Elisa Bertino§
*The State Key Laboratory on Integrated Services Networks, School of Cyber Engineering, Xidian University, China
†Hangzhou Institute of Technology, Xidian University, China
‡School of Computer Science and Artificial Intelligence, Zhengzhou University, China
§Department of Computer Science, Purdue University, USA
happyafrog@163.com, zyan@xidian.edu.cn, ltyang@ieee.org, bertino@purdue.edu

Abstract—**Blockchain-enabled digital currency systems have typically operated in isolation, lacking necessary mechanisms for seamless interconnection. Consequently, transferring assets across distinct currency systems remains a complex challenge, with existing schemes often falling short in ensuring security, privacy, and practicality. This paper proposes P2C2T – a privacy-preserving cross-chain transfer scheme. It is the first scheme to address atomicity, unlinkability, indistinguishability, non-collateralization, and required functionalities across diverse currency systems. P2C2T is based on *threshold anonymous atomic locks* (TA^{2}L), also proposed by us, serving as the cornerstone for guaranteeing atomic cross-chain transfer while obscuring the payment relationships between users. By combining TA^{2}L with *verifiable timed discrete logarithm* schemes, P2C2T renders cross-chain transactions indistinguishable from regular intra-chain ones. Notably, P2C2T eliminates the collateralization of senders and imposes minimal requirements on underlying blockchains, specifically on the ability to verify signatures. We substantiate the security of TA^{2}L based on a proposed cryptographic notion called *threshold blind conditional signatures* and demonstrate the security of P2C2T through necessary proofs. Additionally, we compare the performance of P2C2T with an existing scheme that has properties closest to P2C2T. The comparison reveals that P2C2T reduces overhead by at least 85.488% in terms of running time, communication cost, and storage cost when completing a cross-chain transfer. We further conduct cross-chain transfers and intra-chain payments using the Bitcoin testnet and Litecoin testnet to illustrate the privacy and practicality of P2C2T.**

1. Introduction

Blockchain-based digital currencies have garnered significant attention across academia, industry, and government, owing to their attractive features, including decentralization and payment unforgeability. As of December 2023, the landscape boasts over 20000 blockchain-based digital currency systems, commonly referred to as blockchains, with more than 8000 actively operational ones [1]. However, a major issue still exists: these blockchains are typically designed in isolation, hindering interconnectivity and limiting the broader potential of blockchain technology.

Managing assets across blockchains is crucial for blockchain interoperability [2]. Users (not service providers) encounter different scenarios based on their specific cross-chain asset management needs. When users exchange ownership of assets across blockchains, it is referred to as cross-chain swap, which involves bidirectional payment relationships between the participants. In contrast, cross-chain transfer occurs when users transfer ownership of assets across blockchains, creating unidirectional payment relationships. In this scenario, the original owners are referred to as senders, and those gaining ownership are called receivers. This paper focuses on the case where each sender corresponds to a receiver. This case necessitates addressing five key requirements, as described below.

Atomicity. Atomicity requires that either the sender relinquishes ownership of assets on one blockchain while the corresponding receiver gains ownership of roughly equivalent assets on another, or the asset ownership of both parties remains unaltered. Failures to uphold atomicity may result in users losing or receiving unexpected assets – clearly an undesirable outcome.

Unlinkability. Unlinkability ensures that the payment relationships between the sender and receiver are hidden from other users. This definition draws from the concept of the anonymity set discussed in prior works [3], [4]. Specifically, unlinkability mirrors the anonymity within the anonymity set, with each sender-receiver pair representing an element. The strength of anonymity increases with the number of elements within the anonymity set. Breaching unlinkability poses risks to both parties involved in the transfer; for instance, the disclosure of the payment relationship between cooperating parties could result in loosing their competitive advantage relative to competitors.

Indistinguishability. Indistinguishability means that cross-chain transactions are indistinguishable from regular intra-chain ones when observed on blockchains. Loss of indistinguishability may hinder asset fluidity related to cross-chain transfer due to potential discriminatory censorship.

Non-collateralization. Non-collateralization ensures that senders only need to lock assets roughly equivalent to those withdrawn by corresponding receivers on another blockchain. Conversely, collateralization mandates senders to lock additional assets, roughly equivalent to the assets withdrawn by corresponding receivers. In this case, the sender locks approximately twice the amount of assets compared to non-collateralization, limiting participation of senders with insufficient assets and making cross-chain transfers less practical.

Required Functionalities. Required functionalities dictates which functionalities on the underlying blockchains are essential for the scheme. It is evident that the fewer functionalities a scheme necessitates, the easier it can be applied into practice. It is important to emphasize that signature verification is fundamental for blockchains [5], and smart contracts can support all other functionalities. Hence, smart contracts can be considered as the only necessary functionality for a cross-chain transfer scheme that requires them.

However, one of the most widely used cross-chain technologies, known as exchanges, either centralized [6], [7], [8] or decentralized [9], [10], [11], is primarily designed for cross-chain swap. Moreover, none of existing cross-chain transfer schemes [12], [13], [14], [15], [16], [17], [18], [19], [20] and their variants [3], [4], [21], [22], [23] satisfactorily addresses all above requirements. Broadly speaking, only non-collateralization and the minimality requirement for signature verification are satisfied in [12]. Only atomicity and non-collateralization are satisfied in [13], [14], [15], [16], [17], [18]. Only atomicity and unlinkability are achieved in [19]. References [3], [4], [21], [22], [23] require adaptation to support cross-chain transfer, but they still fall short in achieving indistinguishability and the minimality requirement for signature verification. In [20], only an additional adapted Zerocash protocol [24] is needed on chain. However, practical constraints limit the applicability of [20]. First, the scheme in [20] introduces shielded addresses similar to Zerocash, rendering it incompatible with most existing blockchains. Second, besides the basic operations of Zerocash, it introduces additional operations such as OR proofs and relaying block headers, imposing significant overhead on users. The shortcomings of existing work create a significant gap in the context of Blockchain 4.0 [25].

To bridge this gap, we propose P2C2T, a privacy-preserving cross-chain transfer scheme that addresses all those requirements in an integrated way. A comparative analysis on P2C2T against existing schemes is presented in Table 1, clearly highlighting its advantages with respect to those requirements.

1.1. Technical Challenges and Solutions

P2C2T builds upon the preliminary discussion in [4], which adapts a construction for *payment channel hubs* (PCHs) [3] to support cross-chain transfer. We further enhance this by replacing the *synchronization puzzle protocol* instance (say A^2L) used in the original construction with A^2L^+, proposed by [23] since A^2L is demonstrated insecure [23]. Although the resulting scheme (say A^2L^+-CC) successfully achieves atomicity and unlinkability due to the PCH design, it faces the following challenges:

Challenge 1: Ensuring Indistinguishability. As shown in Table 1, most existing cross-chain transfer schemes, except for the one in [20], fail to achieve indistinguishability. The scheme in [20] uses OR proofs to mix cross-chain transactions with intra-chain ones, but this approach requires shielded asset addresses, which are not supported by most blockchains. To our knowledge, the only other scheme achieving indistinguishability is that proposed in [5], although it is designed specifically for cross-chain swap. This scheme involves only two users with no third parties and employs a joint signature generated through *two-party adaptor signatures* [26], rather than separate signatures from each user. Additionally, it replaces timelock functionality with verifiable timed discrete logarithm (VTD) schemes [27], making signature verification the sole on-chain functionality required. Consequently, cross-chain transactions in this scheme are indistinguishable from intra-chain transactions. Inspired by this work, we integrate two-party adaptor signatures and the VTD scheme into A^2L^+-CC, meeting the minimality requirement for signature verification. However, we still face the following issues:

Issue 1: How to maintain the security of the adapted A^2L^+? Replacing the original adaptor signatures with two-party adaptor signatures introduces security concerns for the adapted A^2L^+, making security assurance a crucial issue.

Issue 2: How to protect against distinguishable information? Unlike [27], A^2L^+-CC includes an untrusted intermediary (say P_h). To achieve unlinkability against P_h, the amount of transferred coins is fixed for a certain duration. Since this amount is public, on-chain observers can potentially compromise indistinguishability by simply observing coin amounts unless defended against. Moreover, if on-chain observers are aware of the asset addresses of P_h and the users involved in cross-chain transfer, they could undermine indistinguishability by inferring that transactions involving these addresses are cross-chain.

Solutions to Challenge 1. To address Issue 1, we propose threshold blind conditional signatures (TBCS), inspired by the concept of *blind conditional signatures* in [23], to ensure the security of the adapted A^2L^+ (say TA^2L). For Issue 2, to mitigate amount distinguishability, we suggest that P_h or users create intra-chain transactions with coin amounts identical to those used in cross-chain transactions within the same duration, excluding transaction fees if feasible. To address asset address distinguishability, we let P_h and users privately share public keys to derive asset addresses.

Challenge 2: Enabling Non-Collateralization. A^2L^+-CC requires each sender (say P_s) to collateralize supplementary coins to secure a signature from P_h on a request token intended for the receiver (say P_r). The challenge is to reduce the amount of supplementary coins and the coins P_s pays to P_h in order to achieve non-collateralization.

Solutions to Challenge 2. We address Challenge 2 with the following steps:

Step 1: Reducing the Value of Deposits. P_s deposits coins into a shared address controlled by both P_s and P_h to later pay P_h. The deposited amount is generally larger than the payment P_s makes to P_h each time, allowing

TABLE 1: Comparison of Existing Cross-Chain Transfer Schemes and Adapted Schemes with P2C2T Classified Based on Technique Types

Requirements		**TB**	**Proof Relay**				**Notary**							
		[12]	**[13], [15]**	**[17]**	**[18]**	**[20]**	**[14], [16]**	**[19]**	**[4]***	**[21]***	**[22]***	**[3]***	**[23]***	**P2C2T**
atomicity		N	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y
unlinkability		N	N	N	N	Y	N	Y	Y	Y^-	Y^-	Y	Y	Y
indistinguishability		N	N	N	N	Y	N	N	N	N	N	N	N	Y
non-collateralization		Y	Y	Y	Y	Y	Y	N	N	Y	N	Y	N	Y
required functionalities	SV	○	●	○	○	○	●	○	○	●	○	○	○	○
	MTPV	●	●	●	○	●	●	●	●	●	●	●	●	●
	SPVPV	●	●	○	●	●	●	●	●	●	●	●	●	●
	AZCP	●	●	●	●	○	●	●	●	●	●	●	●	●
	HL	●	●	●	●	●	●	●	●	●	●	○	●	●
	TL	●	●	●	●	●	●	○	○	●	○	○	○	●
	WT	●	●	○	●	●	●	●	●	●	●	●	●	●
	SC	●	○	●	●	●	○	●	●	○	●	●	●	●

TB: third blockchain; * denotes adapted variants of original schemes are compared; Y (resp., N) denotes the corresponding criterion is (resp., is not) fulfilled; Y^- denotes the corresponding criterion is fulfilled in some cases but not consistently; SV: signature verification; MTPV: the verification of Merkle-tree proofs; SPVPV: the verification of simplified payment verification proofs; AZCP: adapted Zerocash protocol; HL: hashlocks; TL: timelocks; WT: wrapped tokens; SC: smart contracts; ○ (resp., ●) denotes the corresponding functionality is (resp., is not) required on chain.

multiple off-chain payments. We let the deposited amount be roughly equivalent to the coins withdrawn by P_r on another blockchain. This unifies the deposit amounts from senders, excluding transaction fees, and enhances unlinkability since each sender can only pay P_h for one time.

Step 2: Replacing Supplementary Coins with Deposits. Since the deposits are now equivalent to the collateralized coins for P_s, we eliminate collateralization-related steps and shift the deposit-related steps to replace them.

A^2L^+-CC is transformed into P2C2T by solving Challenge 1 and 2.

1.2. Our Contributions

Our contributions can be summarized as follows.

(1) We propose P2C2T, the first cross-chain transfer scheme that simultaneously addresses atomicity, unlinkability, indistinguishability, non-collateralization, and minimization of required functionalities. P2C2T primarily leverages the proposed TA^2L scheme to achieve atomicity, unlinkability, and indistinguishability. It ensures non-collateralization by transforming all assets sent from senders to shared addresses into roughly equivalent ones available to corresponding receivers without considering transaction fees. Additionally, P2C2T incorporates the VTD scheme to remove the dependency on timelocks, thus only requiring the fundamental signature verification functionality on chain.

(2) We provide rigorous definitions of security properties and formal security proofs for TA^2L, based on the proposed TBCS. For unlinkability and indistinguishability of P2C2T, we present rigorous definitions and proof sketches with a game-based setting. Additionally, we offer proof sketches or detailed discussions on atomicity, *griefing attack*[1] *resistance*, non-collateralization, and the minimality requirement for signature verification in P2C2T.

(3) We conduct a comparison between P2C2T and a cutting-edge scheme [20], regarding running time, communication cost, and storage cost. Our comparison demonstrates that P2C2T achieves a minimum of 85.488% overhead reduction when achieving a cross-chain transfer. To further illustrate its privacy and practicality, we also conduct cross-chain transfers and intra-chain payments using the Bitcoin testnet [28] and Litecoin testnet [29]. For interested readers, we present a performance comparison between A^2L^+ and TA^2L in Appendix C.

2. Related Work

This section reviews related schemes that facilitate cross-chain transfer based on their technique types. They are generally categorized into three types: notary, proof relay, and third blockchain, as reviewed below.

(1) Notary. Schemes using this technique typically rely on a third party or committee to assist in cross-chain transfer, such as [14], [16], [19] and their variants [3], [4], [21], [22], [23]. Zamyatin et al. [14] leveraged multiple off-chain entities known as vaults and a specialized smart contract called iSC to enable trustless cross-chain transfer. While only one blockchain requires the signature verification functionality, the other must be compatible with iSC, which is used to verify cross-chain proofs and issue wrapped tokens. Yin et al. [16] employed hidden committees to enable bidirectional asset transfers across two blockchains. Only one blockchain requires signature verification, while the other uses smart contracts to create wrapped tokens. Hanzlik et al. [19] introduced two protocols, the redeem and exchange protocols, to achieve cross-chain transfer. Unlinkability is satisfied but timelocks are needed. Non-collateralization is undermined for the same reason as the variant of [23], as outlined in Section 1.1. All above schemes provide some cross-chain verification mechanisms to guarantee atomicity and do not require collateral for senders except for [19].

1. Griefing attack is described in [4]: P_r initiates numerous requests in some phase, each requiring P_h to lock a specific amount of coins, while the corresponding interactions in the next phase are never executed. This type of attack depletes the resources of P_h, resulting in a form of denial-of-service attack.

However, they share a common requirement: at least one interconnected blockchain should support more functionalities than signature verification. This dependency undermines not only indistinguishability but also unlinkability except for [19], as on-chain nodes can easily identify a sender and its corresponding receiver. In addition, as explained in Section 1.1, PCH schemes [3], [4], [21], [22], [23] can be adapted to achieve cross-chain transfer. However, such variants often require additional on-chain functionalities, such as hashlocks [3], timelocks [3], [4], [22], [23], and smart contracts [21]. Depending on these functionalities undermines indistinguishability of such schemes. Atomicity is compromised in [4] as users can exploit a security vulnerability to steal coins from the intermediary, as demonstrated in [23]. Collateralization is essential in [4], [22] for the same reason as the variant of [23], as described in Section 1.1. Furthermore, while both variants of [21], [22] support variable-amount cross-chain transfer, their unlinkability is compromised when payment channels between the hub and the receivers are closed.

(2) Proof Relay. Schemes employing this technique generally require parties to submit proofs from the source chain to the destination chain [13], [15], [17], [18], [20]. Back et al. [17] pioneered a sidechain technology, allowing assets to move bidirectionally between a parent chain and a sidechain. This scheme requires one blockchain to generate a simplified payment verification proof and another to verify the proof on chain. If the verification passes, the other blockchain creates or unlocks corresponding wrapped tokens [30]. Subsequent sidechain schemes aim to reduce verification costs [13] or explore innovative approaches to sidechain initialization [18]. However, smart contracts are needed for a sidechain to verify cross-chain proofs from the parent chain and create wrapped tokens [13]. Merkle-tree [31] proofs need to be verified on a parent chain in the scheme proposed by Gavzi et al. [18]. Xie et al. [15] introduced a block header relay network for relaying block headers and corresponding zero-knowledge proofs [32], [33] to prove locking transactions on the sender chain. In this work, smart contracts are needed to transfer coins and verify zero-knowledge proofs. Baldimtsi et al. [20] proposed a scheme for asset transfers across Zerocash-based blockchains [24]. All above schemes provide some cross-chain verification mechanisms to guarantee atomicity and do not require collateral for senders. In addition, The approach in [20] achieves indistinguishability by mixing cross-chain transactions with intra-chain ones via OR proofs. Unlinkability is maintained by hiding all transaction states, including asset sources, destinations, and amount. However, practical constraints as described in Section 1 limit its applicability.

(3) Third Blockchain. Schemes utilizing this technique typically use a third blockchain to facilitate cross-chain transfer [12]. Tian et al. [12] introduced a scheme that relies on a validation committee, two qualified intermediaries, and Ethereum [34] as a coordinator to facilitate cross-chain transfer. Senders are not required to provide collateral, as they only need to transfer assets to intermediaries that are approximately equivalent to the assets the receivers will eventually receive. Only signature verification is required on interconnected blockchains. However, the receiver gets ethers rather than expected native coins on the receiver blockchain as compensation if one intermediary fails to fulfill its obligation, which undermines atomicity. Unlinkability is not assured since the committee can correlate a sender and a corresponding receiver by simply observing the relationships of transactions on Ethereum and interconnected blockchains. Indistinguishability is compromised since the committee can differentiate cross-chain transfer transactions and regular intra-chain ones by linking a sender and its corresponding receiver.

Table 1 compares P2C2T with these schemes and shows that the existing literature still lacks a cross-chain transfer scheme that can simultaneously addresses atomicity, unlinkability, indistinguishability, and non-collateralization, with minimization of required functionalities.

3. Problem Statement

This section states our problem, followed by necessary preliminaries.

3.1. Problem

We describe the problem with respect to four aspects, i.e., scenario description, system model, threat model, and design goals.

Scenario Description. We primarily consider a cross-chain transfer scenario. A party P_s on blockchain B_1 wishes to transfer some amount of coins residing on B_2 to another party P_r on B_2. However, P_s may not have enough coins in its asset address on B_2. Or P_s may not even have an asset address on B_2. In such a scenario, it is evident that P_s cannot achieve its goal without the assistance of a third party.

System Model. The system model is depicted in Fig. 1 where a third party P_h is introduced to aid P_s in the described scenario. P_h is required to possess asset addresses (corresponding to A_h^1 and A_h^2 shown in Fig. 1) on both B_1 and B_2, along with sufficient coins in an asset address on B_2. Without loss of generality, we consider the following case: the sender P_s aims to transfer y coins from B_2 to the receiver P_r, also on B_2. To achieve this, P_s pays P_h x coins (from the address A_s^1 shown in Fig. 1) on B_1 in exchange for P_h sending y coins to P_r (possessing the address A_r^2 shown in Fig. 1) on B_2. It is important to note that a single entity can control both P_s and P_r to obtain coins on B_2 using roughly equivalent coins on B_1 via P2C2T.

Threat Model. We consider four threats that could undermine the security of P2C2T. The first three threats are similar to those of [4] and the fourth one corresponds to indistinguishability.

(1) P_h is rational and curious. P_h may attempt to get some coins from P_s without sending enough coins to P_r. Additionally, P_h may seek to link P_s with the corresponding P_r.

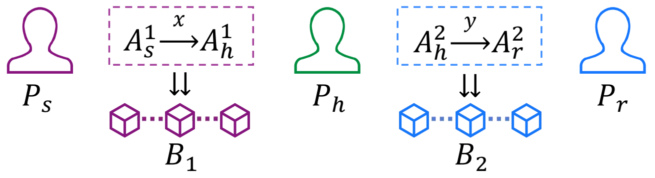

Figure 1: System model of P2C2T. In this figure, $\xrightarrow{x}$ (resp., $\xrightarrow{y}$) represents the transfer of x (resp., y) coins from one address to another. $\downdownarrows$ indicates the posting of a transaction (represented by the part within the dashed box) on a blockchain.

(2) P_r may launch a griefing attack on P_h. It is worth noting that P_s cannot initiate a griefing attack on P_h since P_h locks a specific amount of coins only after receiving a request from P_r, not P_s.

(3) P_s and the corresponding P_r may collude to steal coins from P_h.

(4) On-chain observers are curious about cross-chain transfer transactions and may manage to distinguish these transactions from regular intra-chain ones.

Design Goals. The design of P2C2T must achieve atomicity, unlinkability, and indistinguishability with griefing attack resistance and non-collateralization. In addition, P2C2T must have minimal requirements with respect to signature verification for enhancing practicality.

3.2. Preliminaries

Let $n \in \mathbb{N}$ denote the security parameter, $[n]$ the set $\{1, 2, ..., n\}$, $\mathbb{Z}_n$ the set $\{0, 1, ..., n-1\}$, and $x \leftarrow_\$ \mathcal{X}$ the uniform sampling of x from the set $\mathcal{X}$. We write $y \leftarrow A(x)$ to denote that a probabilistic polynomial time (PPT) algorithm A on input x, outputs y. If A is a deterministic polynomial time (DPT) algorithm, we use the notation $y := A(x)$. We also use $:=$ to denote tuples. For example, we write $x := (x_1, x_2)$ for a tuple x composed of two elements. We write $\mathbb{G}$ to denote a cyclic group of prime order p with generator g. We assume that the discrete logarithm (DLOG) problem is hard on $\mathbb{G}$. We use a PPT algorithm $GenR(1^n)$ to generate an instance of a hard relationship based on DLOG problem. Specifically, $(Y, y) \leftarrow GenR(1^n)$ where $Y := y \cdot g$ and $y \leftarrow_\$ \mathbb{Z}_p$. We say that a function is negligible if it vanishes faster than any polynomial.

Two-Party Adaptor Signatures. A two-party adaptor signature scheme with respect to a hard relationship R and a digital signature $\Pi^S := (KeyGen, Sign, Verify)$ with a message space $\mathcal{M}$ satisfying strong existential unforgeability [35] generally consists of five algorithms $\Pi^{2AS} := (JointKeyGen, JointPreSign, PreVerify, Adapt, Ext)$. Two parties (say P_1 and P_2) get (sk_1, pk_2, pk) and (sk_2, pk_1, pk) respectively by running an interactive protocol $JointKeyGen\langle P_1(\ ), P_2(\ )\rangle$. Here (pk_1, sk_1) and (pk_2, sk_2) are key pairs consistent with those of Π^S from the view of any PPT distinguishers. pk is the joint verification key corresponding to the secret key shares sk_1 and sk_2. Then both P_1 and P_2 receive the pre-signature $\tilde{\sigma}$ by executing $JointPreSign\langle P_1(sk_1, pk, m, Y), P_2(sk_2, pk, m, Y)\rangle$. Here m is the message to be signed and Y is a statement associated with some witness y such that $(Y, y) \in R$. Anyone holding the witness y can transform $\tilde{\sigma}$ into a final signature σ by running $Adapt(y, \tilde{\sigma})$. Conversely, anyone with σ can extract y by executing $Ext(\sigma, \tilde{\sigma}, Y)$. Both $\tilde{\sigma}$ and σ can be verified publicly by invoking $PreVerify(pk, m, Y, \tilde{\sigma})$ and $Verify(pk, m, \sigma)$ respectively. Instantiation of Π^{2AS} with respect to ECDSA and Schnorr signature schemes have been defined [26], satisfying properties such as two-party pre-signature correctness, two-party signature unforgeability, two-party pre-signature adaptability, and two-party witness extractability similar to [36], [37]. For formal definitions of these properties, please refer to Appendix A.1.

Commitment Scheme. A commitment scheme consists of two algorithms $\Pi^C := (Commit, Verify)$. The prover can commit to a message m by running commitment algorithm $(com, decom) \leftarrow Commit(m)$. Subsequently, a verifier can employ the verification algorithm $Verify(com, decom, m)$ to determine whether the message m has been committed. The Pedersen commitment scheme [38] adopted in this paper is unconditionally-hiding and computationally-biding [39].

Non-Interactive Zero-Knowledge. A non-interactive zero-knowledge proof scheme Π^{NIZK} [32], [33] consists of a tuple of three algorithms $\Pi^{NIZK} := (Setup, Prove, Verify)$. The setup algorithm takes as input a relationship R and then outputs a common reference string crs and a trapdoor td. A prover can generate a proof π to show the validity of a statement x with a witness w (i.e., $(x, w) \in R$) by running the prover algorithm $\pi \leftarrow Prove(crs, x, w)$. The proof π can be efficiently checked by invoking the verification algorithm $Verify(crs, x, \pi)$. We require Π^{NIZK} to satisfy the following properties.

Soundness. It is infeasible for an adversary to output a valid proof for a statement x where $(x, \cdot) \notin R$.

Zero-Knowledgeness. There exists a simulator that on input td and x, outputs a valid proof π without the knowledge of w. That is, the prover does not leak any information about w except the truth of x. For brevity, we omit crs in writing $Prove$ and $Verify$ in the following text.

Randomizable Signatures. A randomizable signature scheme, denoted as Π^{RS}, allows for both signature randomization and signing of committed messages. In addition to the usual algorithms of a regular signature scheme, Π^{RS} includes four additional algorithms: $Commit$, $SignCom$, $ExtSign$, and $RandSign$. To obtain a signature on a message m, a user first computes a commitment $(com, decom) \leftarrow Commit(m)$. The user then submits com along with a proof of knowledge π for verifying com to the signer. If π is valid, the signer generates σ_{com} by running $SignCom$. The user can then extract the signature σ on m using $\sigma \leftarrow ExtSign(decom, \sigma_{com})$. Additionally, anyone can produce a randomized signature σ' by executing $\sigma' \leftarrow RandSign(\sigma)$. Π^{RS} is instantiated with the signature scheme [40] which works over type 3 pairings $(p, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ in this paper. For more details about type 3 pairings, please refer to Appendix A.2.

Linear-only Encryption. Linear-only encryption (LOE) is a generic homomorphic cryptosystem modeled by giving access to oracles instead of their corresponding algorithms [41]. With reference to [23], the cryptosystem can be instantiated with a LOE scheme Π^{LOE} given a linearly homomorphic encryption scheme $\Pi^{LHE} := (KeyGen^*, Enc^*, Dec^*, RandEnc^*)$ over $\mathbb{Z}_p$ as follows.

$KeyGen(1^n)$: Run $(sk^*, pk^*) \leftarrow KeyGen^*(1^n)$ and $\alpha \leftarrow_\$ \mathbb{Z}_p$. Return $sk^L := (sk^*, \alpha)$ as the decryption key and $pk^L := (pk^*, Enc^*(pk^*, \alpha))$ as the encryption key.

$Enc(pk^L, x)$: Compute ciphertext c as $(Enc^*(pk^*, x), Enc^*(pk^*, \alpha{\cdot}x))$, where $Enc^*(pk^*, \alpha{\cdot}x) := (Enc^*(pk^*, \alpha))^x$.

$Dec(sk^L, c)$: Parse c as (c_0, c_1) and compute $x_0 \leftarrow Dec^*(sk^*, c_0)$ and $x_1 \leftarrow Dec^*(sk^*, c_1)$. If $x_1 = \alpha \cdot x_0$, then return x_0, else return $\perp$.

As demonstrated in [23], Π^{LOE} can resist oblivious ciphertext sampling and is one-more CCA-A2L (OM-CCA-A2L) secure under the one-more discrete logarithm (OMDL) assumption [42]. Π^{LHE} is instantiated with the encryption scheme based on a class group with $q \in \mathbb{N}$ as the upper bound of its order [43] in this paper. An algorithm $RandEnc$ randomizing a ciphertext can be obtained trivially. Specifically, given input $c := (c_0, c_1)$ and $r \leftarrow_\$ \mathbb{Z}_q$, $RandEnc$ outputs c' by running $RandEnc(c, r) := (RandEnc^*(c_0, r), RandEnc^*(c_1, r))$. For additional details about OM-CCA-A2L security, please refer to Appendix A.3.

Verifiable Timed Discrete Logarithm. A VTD scheme [5], [27] with respect to $\mathbb{G}$ involves two parties: a committer and a verifier. Specifically, a VTD scheme Π^{VTD} comprises four algorithms $\Pi^{VTD} := (Commit, Verify, Open, ForceOp)$. The committer generates a timed commitment of timing hardness T for a value $x \in \mathbb{Z}_p$ and a proof π by running $(C, \pi) \leftarrow Commit(x, T)$. Then the committer sends (H, C, π) to the verifier, where $H := x \cdot g$. Next the verifier checks if the value x embedded in C satisfies $H = x \cdot g$ by invoking $Verify(H, C, \pi)$. The committer can open the commitment C by running $(x, r) := Open(C)$, where r is used in generating C. Finally, the verifier can learn the value x in time T by running $x \leftarrow ForceOp(C)$. A secure VTD scheme must satisfy the following properties.

Soundness. The verifier is convinced that, given C, the algorithm $ForceOp$ can produce the value x in time T.

Privacy. All polynomial time algorithms whose running time is at most t (where $t < T$) succeed in learning x from C and π with at most negligible probability.

Synchronization Puzzles. A synchronization puzzle protocol [3], [4], [23] involves three parties (say P_s, P_r and P_h). This protocol enables P_r to get a signature from P_h on some message (say m_{HR}) only if P_h obtains a signature from P_s on the other message (say m_{SH}) with the following properties.

Blindness. P_h cannot link some P_s with the corresponding P_r when multiple sender-receiver pairs are involved.

Unlockability. If P_h obtains a signature from P_s on m_{SH}, P_r can get a signature from P_h on m_{HR}.

Unforgeability. If P_h fails to obtain a signature from P_s on m_{SH}, P_r cannot get a signature from P_h on m_{HR}.

4. TA^{2}L

In this section, we first introduce TBCS with TA^{2}L serving as an instance. Then we describe TA^{2}L, the cornerstone of P2C2T. We finally prove the security of TA^{2}L.

4.1. Threshold Blind Conditional Signatures

TBCS is performed among P_s, P_h, and P_r. Its interfaces and associated security properties are defined below.

Definition 1 (Threshold Blind Conditional Signatures). A threshold blind conditional signature $\Pi^{TBCS} := (Setup, Promise, Solving, Open)$ is defined with respect to a two-party adaptor signature scheme $\Pi^{2AS} := (JointKeyGen, JointPreSign, PreVerify, Adapt, Ext)$ and a linear-only encryption scheme $\Pi^{LOE} := (KeyGen, Enc, Dec, RandEnc)$ as follows.

(1) $Setup(1^n)$ runs the following algorithms.

1) $(sk^L, pk^L) \leftarrow \Pi^{LOE}.KeyGen(1^n)$.

2) $(\{sk_s^0, pk_{h1}^0, pk_1\}, \{sk_{h1}^0, pk_s^0, pk_1\}) \leftarrow \Pi^{2AS}.JointKeyGen\langle P_s(\ ), P_h(\ )\rangle$.

3) $(\{sk_r^0, pk_{h2}^0, pk_2\}, \{sk_{h2}^0, pk_r^0, pk_2\}) \leftarrow \Pi^{2AS}.JointKeyGen\langle P_r(\ ), P_h(\ )\rangle$.

(2) $(\perp, \{\tau, \perp\}) \leftarrow$ $Promise\left\langle \begin{array}{c} P_h(sk^L, pk^L, sk_{h2}^0, pk_2, m_{HR}) \\ P_r(sk_r^0, pk^L, pk_2, m_{HR}) \end{array} \right\rangle$ is an interactive protocol between P_h (with inputs the decryption key sk^L,the encryption key pk^L, the signing key share sk_{h2}^0, the joint verification key pk_2, and a message m_{HR}) and P_r (with inputs the signing key share sk_r^0, the encryption key pk^L, the joint verification key pk_2, and a message m_{HR}). It returns $\perp$ to P_h and either a puzzle τ or $\perp$ to P_r.

(3) $(\{(\sigma_{SH}, s_r), \perp\}, \{\sigma_{SH}, \perp\}) \leftarrow$ $Solving\left\langle \begin{array}{c} P_s(sk_s^0, pk_1, \tau_r, pk^L, m_{SH}) \\ P_h(sk_{h1}^0, sk^L, pk_1, m_{SH}) \end{array} \right\rangle$ is an interactive protocol between P_s (with inputs the signing key share sk_s^0, the joint verification key pk_1, a puzzle τ_r, the encryption key pk^L, and a message m_{SH}) and P_h (with inputs the signing key share sk_{h1}^0, the decryption key sk^L, the joint verification key pk_1, and a message m_{SH}). It returns either a signature σ_{SH} (P_s additionally receives a secret s_r) or $\perp$ to both P_s and P_h.

(4) $\{\sigma_{HR}, \perp\} := Open(\tau, s_r)$ is a DPT algorithm that takes as input a puzzle τ and a secret s_r. It returns a signature σ_{HR} or $\perp$.

We defer the formal definitions of correctness (Definition 2) and security properties in a game-based setting for Π^{TBCS} to Appendix B. These properties include blindness (Definition 3), unlockability (Definition 4), and unforgeability (Definition 5).

Public parameters: group description $(\mathbb{G}, g, p)$, message m_{HR}	
$Promise_{P_h}(sk^L, pk^L, sk^0_{h2}, pk_2)$	$Promise_{P_r}(sk^0_r, pk^L, pk_2)$
1: $(Y, s) \leftarrow GenR(1^n)$	
2: $c \leftarrow \Pi^{LOE}.Enc(pk^L, s)$	
3: $\pi^E \leftarrow \Pi^{NIZK}.Prove(st^E := (Y, c), s, sk^L)$	
4: $\xrightarrow{(Y,c,\pi^E)}$	
5: Execute $\Pi^{2AS}.JointPreSign\langle P_h(sk^0_{h2}, pk_2, m_{HR}, Y), P_r(sk^0_r, pk_2, m_{HR}, Y)\rangle$ jointly. Both P_h and P_r receive $\tilde{\sigma}_{HR}$.	
6:	If $\Pi^{NIZK}.Verify(st^E := (Y, c), \pi^E) \neq 1$, return $\perp$
7:	$r' \leftarrow_{\$} \mathbb{Z}_q, Y' := r' \cdot Y$
8:	$c' \leftarrow \Pi^{LOE}.RandEnc(c, r')$
9: return $\perp$	return $\tau := (m_{HR}, \tilde{\sigma}_{HR}, r', (Y, c), (Y', c'))$

Figure 2: Promise protocol of TA^{2}L

Definition 6 (Security). Π^{TBCS} *is secure if it satisfies blindness, unlockability, and unforgeability.*

4.2. TA^{2}L Design

As an instance of both TBCS and synchronization puzzle protocols, TA^{2}L involves four distinct phases, i.e., setup, promise, solving, and open, each featuring corresponding protocols or algorithms outlined below.

Setup Phase. In setup phase, P_s, P_r, and P_h need to complete the following steps.

(1) P_h runs $\Pi^{LOE}.KeyGen$ to generate its keys and publishes the encryption key.

(2) P_s invokes $\Pi^{2AS}.JointKeyGen$ by interacting with P_h, then P_s and P_h receive $(sk^0_s, pk^0_{h1}, pk_1)$ and $(sk^0_{h1}, pk^0_s, pk_1)$ respectively.

(3) P_r invokes $\Pi^{2AS}.JointKeyGen$ by interacting with P_h, then P_r and P_h receive $(sk^0_r, pk^0_{h2}, pk_2)$ and $(sk^0_{h2}, pk^0_r, pk_2)$ respectively.

Promise Phase. In promise phase, there are two required procedures presented below.

(1) P_h and P_r execute the promise protocol shown in Fig. 2. Specifically, P_h acquires the random s and then obtains a partial puzzle (Y, c) and its proof π^E (Lines 1-3 in Fig. 2). P_h shares the puzzle and proof, and invokes $\Pi^{2AS}.JointPreSign$ with P_r. They compute a pre-signature $\tilde{\sigma}_{HR}$ on m_{HR} (Line 5 in Fig. 2). Next P_r verifies the puzzle and randomizes it if the verification is successful (Lines 6-8 in Fig. 2). As a result, P_r gets a complete puzzle $\tau := (m_{HR}, \tilde{\sigma}_{HR}, r', (Y, c), (Y', c'))$ (Line 9 in Fig. 2).

(2) P_r sends a partial puzzle $\tau_r := (Y', c')$ to P_s.

Solving Phase. In solving phase, there are two required procedures presented below.

(1) P_s and P_h perform the solving protocol shown in Fig. 3. Specifically, P_s re-randomizes τ_r from P_r and gets (Y'', c'') (Lines 1-2 in Fig. 3). Then P_s shares (Y'', c'') with P_h. Next P_h verifies the validity of (Y'', c'') (Lines 4-5 in Fig. 3). If the verification passes, P_h invokes $\Pi^{2AS}.JointPreSign$ with P_s. Then they receive a pre-signature $\tilde{\sigma}_{SH}$ on m_{SH} (Line 6 in Fig. 3). P_h adapts $\tilde{\sigma}_{HR}$ into a final signature σ_{SH} and sends σ_{SH} to P_s (Lines 7-8 in Fig. 3). Next P_s extracts and verifies the witness corresponding to Y'' by performing Lines 9-10 in Fig. 3. Last P_s acquires the solution s_r of τ_r (Line 11 in Fig. 3).

(2) P_s sends s_r to P_r.

Open Phase. In open phase, P_r follows the open algorithm shown in Fig. 7. Specifically, P_r first acquires the witness s by computing $s := s_r \cdot (r')^{-1}$. Then P_r transforms the pre-signature $\tilde{\sigma}_{HR}$ into the final signature by invoking $\sigma_{HR} \leftarrow \Pi^{2AS}.Adapt(s, \tilde{\sigma}_{HR})$.

4.3. Security Proofs

Theorem 1. *Let* Π^{LOE} *be a linear-only encryption scheme,* Π^{2AS} *a secure two-party adaptor signature scheme, and* Π^{NIZK} *a sound and zero-knowledge proof scheme. Assuming the hardness of OMDL, TA*^{2}*L is a secure threshold blind conditional signature.*

The formal proof of Theorem 1 is given in Appendix D.

5. P2C2T

In this section, we respectively present design assumptions, scheme description, and security analysis of P2C2T.

5.1. Design Assumptions

The following assumptions are pre-set to allow us to focus on the target issues and for ease of presentation.

(1) The communication channel is secure and authenticated. In addition, the communication channel between P_s and P_r are anonymous.

(2) The interconnected blockchains can securely manage a sequence of transactions based on some intra-chain security assumptions. For example, we assume that a majority hashing power or stake used to maintain interconnected blockchains is never controlled by any adversaries.

(3) The scheme operates in epochs, with protocols and algorithms executed in phases. The duration of an epoch is the sum of the duration of all phases, each of which should be longer than the time required for the corresponding protocol or algorithm. The time required varies based on factors such as transaction finalization time and computing node performance. All parties have access to a global clock and are aware of the epochs and phases they are in.

(4) P_h cannot be undermined by centralization risks such as single point failure. To achieve this as much as possible, we can equip P_h with robust infrastructure, including powerful machines and multiple backup servers. Note that multiple

Public parameters: group description $(\mathbb{G}, g, p)$, message m_{SH}		
$Solving_{P_s}(sk_s^0, pk_1, \tau_r := (Y', c'), pk^L)$		$Solving_{P_h}(sk_{h1}^0, sk^L, pk_1)$
1: $r'' \leftarrow_{\$} \mathbb{Z}_q, Y'' := r'' \cdot Y'$		
2: $c'' \leftarrow \Pi^{LOE}.RandEnc(c', r'')$		
3:	$\xrightarrow{(Y'', c'')}$	
4:		$s'' := \Pi^{LOE}.Dec(sk^L, c'')$
5:		If $Y'' \neq s'' \cdot g$, return $\perp$
6: Execute $\Pi^{2AS}.JointPreSign\langle P_h(sk_{h1}^0, pk_1, m_{SH}, Y''), P_s(sk_s^0, pk_1, m_{SH}, Y'')\rangle$ jointly. Both P_s and P_h receive $\tilde{\sigma}_{SH}$.		
7:		$\sigma_{SH} \leftarrow \Pi^{2AS}.Adapt(s'', \tilde{\sigma}_{SH})$
8:	$\xleftarrow{\sigma_{SH}}$	
9: $s'' \leftarrow \Pi^{2AS}.Ext(\sigma_{SH}, \tilde{\sigma}_{SH}, Y'')$		
10: If $s'' = \perp$, return $\perp$		
11: $s_r := s'' \cdot (r'')^{-1}$		
12: return (σ_{SH}, s_r)		return σ_{SH}

Figure 3: Solving protocol of TA^{2}L

P_h could exist in the system to offer the same service, thus it is possible for the senders to select the most reputable one to serve cross-chain transfer, mitigating single P_h failure.

(5) Coins received by P_h and P_r remain constant within a given epoch. This requires that x and y remain constant during the given epoch. To incentivize P_h, the value of x coins residing on B_1 is no less than that of the sum of y coins and transaction fees on B_2.

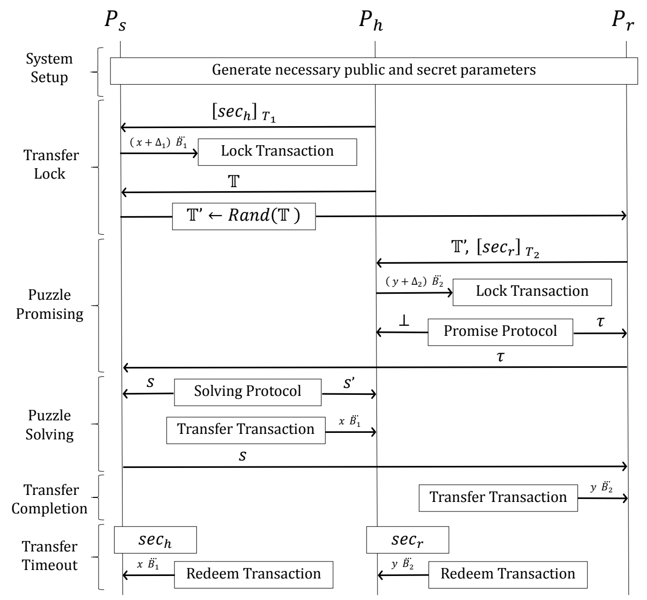

Figure 4: Overview of P2C2T. Here, $\dddot{B}_i$ refers to coins on B_i, $i \in \{1, 2\}$.

5.2. Scheme Description

We present the overview of P2C2T and then introduce P2C2T in detail.

P2C2T Overview. P2C2T operates in epochs, consisting of six phases: system setup, transfer lock, puzzle promising, puzzle solving, transfer completion, and transfer timeout. The key local and interactive operations among P_s, P_h, and P_r in P2C2T are illustrated in Fig. 4. During the system setup phase, P_s, P_h, and P_r generate necessary parameters. In the transfer lock phase, P_s locks $x + \Delta_1$ coins on B_1, with Δ_1 coins reserved for transaction fees for subsequent transactions involving these locked coins, after receiving a commitment (corresponding to $[sec_h]_{T_1}$ in Fig. 4) on secret sec_h from P_h. This secret, preventing perpetual locking, can be disclosed to P_s in time T_1. Subsequently, P_h sends a valid request token $\mathbb{T}$ to P_s, who randomizes $\mathbb{T}$ (corresponding to $Rand(\mathbb{T})$ in Fig. 4) and forwards the result $\mathbb{T}'$ to P_r. During the puzzle promising phase, P_h locks $y + \Delta_2$ coins on B_2, with Δ_2 coins reserved for transaction fees for subsequent transactions involving these locked coins, after receiving a valid token $\mathbb{T}'$ and a commitment (corresponding to $[sec_r]_{T_2}$ in Fig. 4) on secret sec_r from P_r. Similar to sec_h, sec_r prevents perpetual locking and can be revealed to P_h in time T_2. P_h then initiates $\Pi^{TA^2L}.Promise$ with P_r, generating a puzzle τ for P_r, which is subsequently sent to P_s. During the puzzle solving phase, P_s collaborates with P_h using $\Pi^{TA^2L}.Solving$, producing a solution s' to aid P_h in withdrawing x coins on B_1. Simultaneously, P_s obtains a solution s to τ and transmits it to P_r. In the transfer completion phase, P_r can withdraw y coins on B_2 by posting a transfer transaction with the aid of s. Finally, in the transfer timeout phase, P_s and P_h can access sec_h and sec_r, respectively, after a predefined duration (T_1 and T_2, respectively). If coins locked on B_1 (resp., B_2) by a lock transaction are not withdrawn by the corresponding transfer transaction, P_s (resp., P_h) can reclaim these coins by submitting the appropriate redeem transaction with the assistance of sec_h (resp., sec_r).

P2C2T Design Details. We introduce specific algorithms or protocols invoked in corresponding phases, as shown below.

(1) System Setup Phase. In addition to the setup phase of TA^{2}L, P_h and each user must exchange their public keys privately for deriving asset addresses. Specifically, P_s and P_r share pk_s and pk_r with P_h, respectively, while P_h sends pk_{h1} and pk_{h2} to P_s and P_r, respectively. Here, asset addresses derived from pk_s and pk_{h1} are on B_1. Asset addresses derived from pk_r and pk_{h2} are on B_2. In addition, P_h runs $\Pi^{RS}.KeyGen$ to generate (sk^{RS}, pk^{RS})

Public parameters: type 3 pairings description $(p, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$, message tx_s^{lock}		
$TLock_{P_s}(sk_s, pk^{RS})$		$TLock_{P_h}(sk_{h1}^0, sk^{RS})$
1:		$(C_1, \pi_1) \leftarrow \Pi^{VTD}.Commit(sk_{h1}^0, T_1)$
2:	$\xleftarrow{(C_1,\pi_1)}$	
3: If $\Pi^{VTD}.Verify(pk_{h1}^0, C_1, \pi_1) \neq 1$, return $\perp$		
4: Start solving $\Pi^{VTD}.ForceOp(C_1)$		
5: $\sigma_s^{lock} \leftarrow \Pi^S.Sign(sk_s, tx_s^{lock})$		
6: Post $(tx_s^{lock}, \sigma_s^{lock})$ on B_1 and get the transaction identifier id_s^{lock}		
7: $tid \leftarrow_\$ \mathbb{Z}_p$		
8: $(com, decom) \leftarrow \Pi^{RS}.Commit(tid)$		
9: $\pi^B \leftarrow \Pi^{NIZK}.Prove(st^B := com, decom, tid)$		
10:	$\xrightarrow{(\pi^B, com, id_s^{lock})}$	
11:		If $\Pi^{NIZK}.Verify(st^B := com, \pi^B) \neq 1$, return $\perp$
12:		If id_s^{lock} is not finalized on B_1, return $\perp$
13:		$\sigma_{com} \leftarrow \Pi^{RS}.SignCom(sk^{RS}, com)$
14:	$\xleftarrow{\sigma_{com}}$	
15: $\sigma_{tid} \leftarrow \Pi^{RS}.ExtSign(decom, \sigma_{com})$		
16: If $\Pi^{RS}.Verify(pk^{RS}, tid, \sigma_{tid}) \neq 1$, return $\perp$		
17: $\sigma'_{tid} \leftarrow \Pi^{RS}.RandSign(\sigma_{tid})$		
18: Send (tid, σ'_{tid}) to P_r		
19: return $\perp$		return $\perp$

Figure 5: Transfer lock protocol of P2C2T

and publishes pk^{RS}. P_h also performs the setup algorithm of time-lock puzzles [44] utilized in the VTD scheme [5], [27] to generate necessary public parameters.

(2) Transfer Lock Phase. In this phase, P_s and P_h executes the transfer lock (TLock) protocol shown in Fig. 5. For convenience of notation, we use $pk_i \xrightarrow{v} pk_j$ to denote that some sender sends v coins residing in the asset address derived from its public key pk_i to the asset address derived from public key pk_j of some receiver. The lock transaction tx_s^{lock} in public parameters signifies $pk_s \xrightarrow{x+\Delta_1} pk_1$, where Δ_1 is the amount of transaction fees. Broadly speaking, $(tx_s^{lock}, \sigma_s^{lock})$ can be utilized to verify the validity of tx_s^{lock}, as all necessary data except signatures for authenticating tx_s^{lock} is publicly accessible[2]. It can be seen that P_s has to first lock enough assets to get the signature under pk^{RS} on the request token if both parties follow this protocol.

(3) Puzzle Promising Phase. In this phase, P_r and P_h run the puzzle promising (PPromising) protocol shown in Fig. 6. Here the set $\mathcal{T}$ is maintained locally by P_h. The lock transaction tx_{h2}^{lock} in public parameters means $pk_{h2} \xrightarrow{y+\Delta_2} pk_2$, where Δ_2 is the amount of transaction fees. The transfer transaction tx_r^{trans} means $pk_2 \xrightarrow{y} pk_r$. It can be seen that this protocol tie together authentication of tx_r^{trans} for P_r and leakage of a solution to the partial puzzle (Y, c) from P_h.

(4) Puzzle Solving Phase. In this phase, P_s and P_h invoke the puzzle solving protocol. Specifically, P_s and P_h first execute $(\{(\sigma_{h1}^{trans}, s_r)\}, \{\sigma_{h1}^{trans}\}) \leftarrow \Pi^{TA^2L}.Solving\left\langle \begin{array}{c} P_s(sk_s^0, pk_1, \tau_r, pk^L, tx_{h1}^{trans}) \\ P_h(sk_{h1}^0, sk^L, pk_1, tx_{h1}^{trans}) \end{array} \right\rangle$ with a difference that P_h needs to post $(tx_{h1}^{trans}, \sigma_{h1}^{trans})$ on B_1 before sending σ_{h1}^{trans} to P_s. Here σ_{h1}^{trans} denotes the signature under pk_1 on tx_{h1}^{trans}, indicating $pk_1 \xrightarrow{x} pk_{h1}$. Then P_s sends s_r to P_r. It can be seen that P_s and P_r get the solution s_r to the puzzle τ_r after P_h achieves authentication of tx_{h1}^{trans} if both P_s and P_h follow this protocol.

2. Our design goals do not consider transactions requiring more on-chain functionalities than signature verification.

(5) Transfer Completion Phase. In this phase, P_r executes the transfer completion algorithm. Specifically, P_r first executes $\Pi^{TA^2L}.Open(\tau, s_r)$ to get the signature σ_r^{trans}. Then P_r posts $(tx_r^{trans}, \sigma_r^{trans})$ on B_2.

(6) Transfer Timeout Phase. If there are still some coins in the asset address derived from pk_1 (resp., pk_2) in this phase, P_s (resp., P_h) runs the transfer timeout (TTimeout) algorithm shown in Fig. 8 (resp., Fig. 9) to redeem these coins. Here $\Pi^{VTD}.ForceOp(C_1)$ (resp., $\Pi^{VTD}.ForceOp(C_2)$) can output a secret key share of P_h (resp., P_r) if the earliest time for forcibly opening C_1 (resp., C_2) is set at the beginning of this phase by carefully choosing a timing hardness T_1 (resp., T_2). $\diamond$ denotes a binary operator. For example, it is the multiplication operator (resp., addition operator) with respect to ECDSA (resp., Schnorr) signature scheme-based Π^{2AS} [26]. The redeem transaction tx_s^{redeem} (resp., tx_{h2}^{redeem}) means that $pk_1 \xrightarrow{x} pk'_s$ (resp., $pk_2 \xrightarrow{y} pk'_{h2}$) where pk'_s (resp., pk'_{h2}) is newly generated by P_s (resp., P_h). If no coin remains in the asset address derived from pk_1 (resp., pk_2) in this phase, P_s (resp., P_h) does nothing.

Discussion. Non-collateralization is satisfied since the coins sent from P_s to a shared address (Line 6 in Fig. 5) are exchanged for roughly equivalent coins from P_h to P_r (Line 8 in Fig. 6 and the step of posting a signed transaction in the transfer completion algorithm) without considering transaction fees. The minimality requirement for signature verification is ensured because the only on-chain operation involves posting signed transactions awaiting authentication (Line 6 in Fig. 5, Line 8 in Fig. 6, the step of posting a signed transaction in puzzle solving protocol, transfer

```
Public parameters: group description $(\mathbb{G}, g, p)$, messages $tx_{h2}^{lock}$, $tx_{r}^{trans}$
$PPromising_{P_h}((pk^L, sk^L), sk_{h2}, pk^{RS})$                    $PPromising_{P_r}(tid, \sigma'_{tid})$
1:                                                                   $(C_2, \pi_2) \leftarrow \Pi^{VTD}.Commit(sk_r^0, T_2)$
2:                                   $\xleftarrow{(tid, \sigma'_{tid}, C_2, \pi_2)}$
3: If $tid \in \mathcal{T}$ or $\Pi^{RS}.Verify(pk^{RS}, tid, \sigma'_{tid}) \neq 1$, return $\perp$
4: Add $tid$ into $\mathcal{T}$
5: If $\Pi^{VTD}.Verify(pk_r^0, C_2, \pi_2) \neq 1$, return $\perp$
6: Start solving $\Pi^{VTD}.ForceOp(C_2)$
7: $\sigma_{h2}^{lock} \leftarrow \Pi^S.Sign(sk_{h2}, tx_{h2}^{lock})$
8: Post $(tx_{h2}^{lock}, \sigma_{h2}^{lock})$ on $B_2$ and get the transaction identifier $id_{h2}^{lock}$
9:                                   $\xrightarrow{id_{h2}^{lock}}$
10: $P_h$ (with inputs $(sk^L, pk^L, sk_{h2}^0, pk_2, tx_r^{trans})$) invokes $\Pi^{TA^2L}.Promise$ with $P_r$ (with inputs $(sk_r^0, pk^L,$
    $pk_2, tx_r^{trans})$). $P_r$ receives $\tau := (tx_r^{trans}, \tilde{\sigma}_r^{trans}, r', (Y, c), (Y', c'))$.
11:                                                                  If $id_{h2}^{lock}$ is not finalized on $B_2$, return $\perp$
12:                                                                  Send $\tau_r := (Y', c')$ to $P_s$
13: return $\mathcal{T}$                                              return $\tau$
```

Figure 6: Puzzle promising protocol of P2C2T

```
$Open(\tau, s_r)$
Parse $\tau := (\cdot, \tilde{\sigma}_{HR}, r', \cdot, \cdot)$
$s := s_r \cdot (r')^{-1}$
$\sigma_{HR} \leftarrow \Pi^{2AS}.Adapt(s, \tilde{\sigma}_{HR})$
return $\sigma_{HR}$
```

Figure 7: Open algorithm of TA^{2}L

```
$TTimeout(sk_s^0)$
$sk_{h1}^0 := \Pi^{VTD}.ForceOp(C_1)$
$sk_1 := sk_s^0 \diamond sk_{h1}^0$
$\sigma_s^{redeem} \leftarrow \Pi^S.Sign(sk_1, tx_s^{redeem})$
Post $(tx_s^{redeem}, \sigma_s^{redeem})$ on $B_1$
return $(tx_s^{redeem}, \sigma_s^{redeem})$
```

Figure 8: Transfer timeout algorithm of P2C2T on P_s side

```
$TTimeout(sk_{h2}^0)$
$sk_r^0 := \Pi^{VTD}.ForceOp(C_2)$
$sk_2 := sk_{h2}^0 \diamond sk_r^0$
$\sigma_{h2}^{redeem} \leftarrow \Pi^S.Sign(sk_2, tx_{h2}^{redeem})$
Post $(tx_{h2}^{redeem}, \sigma_{h2}^{redeem})$ on $B_2$
return $(tx_{h2}^{redeem}, \sigma_{h2}^{redeem})$
```

Figure 9: Transfer timeout algorithm of P2C2T on P_h side

completion algorithm, and transfer timeout algorithm).

5.3. Security Analysis

In the following, we present the definitions and proof sketches of atomicity, unlinkability, and indistinguishability. We also provide a comprehensive discussion focusing on scenarios where unlinkability is compromised and elucidating the mechanisms by which P2C2T effectively withstands griefing attacks.

Definition 7 (Atomicity). *P2C2T satisfies atomicity if for all PPT parties P_s, P_h, and P_r, the following conditions hold:*

(1) If P_s pays P_h, P_h pays P_r (The PPT adversary plays the role of P_h.).

(2) If P_h pays P_r, P_s pays P_h (The PPT adversary plays the role of P_s and P_r.).

(3) If some coins still remain in the asset address derived from pk_1 (resp., pk_2) in the transfer timeout phase, P_s (resp., P_h) can redeem these coins (The PPT adversary plays the role of P_h (resp., P_r).).

Theorem 2. *Assuming the unlockability and unforgeability of TA^{2}L, the security of Π^{VTD}, P2C2T satisfies atomicity.*

Proof (Sketch). If the atomicity of P2C2T is broken, there are the following possible cases according to Definition 7.

(1) P_s pays P_h, but P_r cannot receive coins from P_h. This means either P_h gets signature from P_s on tx_{h1}^{trans} while P_r cannot get signature from P_h on tx_r^{trans} or P_h opens C_2 before transfer completion phase. The former case implies the unlockability of TA^{2}L is broken and the latter case implies the privacy of Π^{VTD} is broken.

(2) P_h pays P_r, but P_h cannot receive coins from P_s. This means either P_r gets signature from P_h on tx_r^{trans} while P_h cannot get signature from P_s on tx_{h1}^{trans} or P_s opens C_1 before puzzle solving phase. The former case implies the unforgeability of TA^{2}L is broken and the latter case implies the privacy of Π^{VTD} is broken.

(3) There remain some coins in the asset address derived from pk_1 (resp., pk_2) in transfer timeout phase, but P_s (resp., P_h) cannot redeem these coins. This means P_s (resp., P_h) cannot get sk_{h1}^0 (resp., sk_r^0) by running $\Pi^{VTD}.ForceOp(C_1)$ (resp., $\Pi^{VTD}.ForceOp(C_2)$) in time T_1 (resp., T_2), which violates the soundness of Π^{VTD}.

□

Unlinkability Game. Let the adversary $\mathcal{A}$ playing the role of P_h choose two candidate senders $P_{s,0}$, $P_{s,1}$ and receivers $P_{r,0}$, $P_{r,1}$, and inform the challenger. Then the challenger runs $b \leftarrow_\$ \{0, 1\}$. If $b = 0$, the challenger lets the sender-receiver pairs $(P_{s,0}, P_{r,0})$ and $(P_{s,1}, P_{r,1})$ run P2C2T with $\mathcal{A}$. Otherwise, $(P_{s,0}, P_{r,1})$ and $(P_{s,1}, P_{r,0})$ are directed to invoke P2C2T with $\mathcal{A}$. Let K_i denote the transcript (e.g., exchanged messages and intermediate values or computations) of $\mathcal{A}$ after the cross-chain transfer is completed successfully when $b = i$, where $i \in \{0, 1\}$. Then the advantage of $\mathcal{A}$ winning the game is $\Pr_{\text{unl}} := \Pr[b' = b : b' \leftarrow \mathcal{A}^{K_b}, b \leftarrow_\$ \{0, 1\}]$.

Definition 8 (Unlinkability). *P2C2T satisfies unlinkability if there exists a negligible function* $\mathsf{negl}(n)$ *such* *that for all $n \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}$,* $\Pr_{\text{unl}} \leq \frac{1}{2} + \mathsf{negl}(n)$.

Theorem 3. *Assuming the relatively long duration of phases, the security of Π^{RS}, the blindness of TA^{2}L, and the anonymity of communication channel between P_s and P_r, P2C2T satisfies unlinkability.*

Proof (Sketch). If the unlinkability of P2C2T is compromised, it means $\mathcal{A}$ can link $P_{s,i}$ with the counterparty $P_{r,j}$ through some transcripts, where $i, j \in \{0, 1\}$ and their values are determined by the internal coin tosses of unlinkability game. Hence, there are four potential cases as follows according to possible transcripts.

(1) $\mathcal{A}$ associates $P_{s,i}$ with the corresponding $P_{r,j}$ by associating (C_1, π_1) from transfer lock phase with (C_2, π_2) from puzzle promise phase. It implies linkability between the timing hardnesses of these two message pairs. This is contradictory, as both timing hardnesses can be freely set as relatively random values during phases, given the assumption of the phases having a relatively long duration.

(2) $\mathcal{A}$ associates $P_{s,i}$ with the corresponding $P_{r,j}$ by associating (com, σ_{com}) from transfer lock phase with (tid, σ'_{tid}) from puzzle promise phase. This means $\mathcal{A}$ can learn some related information of tid from its commitment com, which implies the unconditionally-hiding property of commitment scheme Π^{RS} uses is broken.

(3) $\mathcal{A}$ associates $P_{s,i}$ with the corresponding $P_{r,j}$ by associating $(Y'', c'', tx^{lock}_{s,i}, \sigma^{lock}_{s,i}, tx^{trans}_{h1}, \sigma^{trans}_{h1})$ with $(Y, c, tx^{lock}_{h2}, \sigma^{lock}_{h2}, tx^{trans}_{r,j}, \sigma^{trans}_{r,j})$. This means $\mathcal{A}$ can either discern the number of transferred coins in tx^{trans}_{h1} compared to other transfer transactions on B_1, and similarly for $tx^{trans}_{r,j}$ on B_2, or succeed in the blindness experiment of TA^{2}L. However, the former case is infeasible since the number of transferred coins in all transfer transactions on each interconnected blockchain remains constant within an epoch, making them indistinguishable. The latter case contradicts with the blindness of TA^{2}L.

(4) $\mathcal{A}$ associates $P_{s,i}$ with the corresponding $P_{r,j}$ by monitoring communication messages (e.g., τ_r and s_r) between $P_{s,i}$ and $P_{r,j}$. This means $\mathcal{A}$ can perceive the sessions between $P_{s,i}$ and $P_{r,j}$, which violates the assumed anonymity of the communication channel between $P_{s,i}$ and $P_{r,j}$.

□

Discussion. If parties act irrationally, the following cases may happen, which undermine the unlinkability of P2C2T.

(1) At least one of P_s and P_r is corrupted and colludes with $\mathcal{A}$. The corrupt party just needs to reveal the identity of the counterparty to $\mathcal{A}$. Then $\mathcal{A}$ can link P_s with the corresponding P_r.

(2) $\mathcal{A}$ launches an abort attack, which is common for epoch-based synchronization puzzle protocols. Then the sender and the receiver fail to complete cross-chain transfer can be linked, provided that all other senders and receivers succeed. The abort attack includes two cases as listed below.

1) $\mathcal{A}$ refuses to interact with some P_r in the puzzle promising phase. It results in the abort of the puzzle solving phase for some P_s that has run the transfer lock protocol successfully with $\mathcal{A}$ before.

2) $\mathcal{A}$ refuses to interact with some P_s in the puzzle solving phase. It results in the abort of the transfer completion phase for some P_r that has invoked the puzzle promising protocol successfully with $\mathcal{A}$ before.

We note that in our threat model, we have assumed P_h to be rational. Therefore, P_h would refrain from launching abort attacks in order to preserve its prestige.

Indistinguishability Game. Let the challenger playing the role of a PPT party (P_s or P_h or P_r) randomly select two transaction-signature pairs (tx_0, σ_0) and (tx_1, σ_1) and send them to the adversary $\mathcal{A}$ playing the role of an on-chain observer. Let $D(\cdot)$ be a discriminant function of $\mathcal{A}$. Given transaction-signature pair (tx, σ), let $D(tx, \sigma) = 0$ (resp., $D(tx, \sigma) = 1$) denote the assertion of $\mathcal{A}$ that tx is a cross-chain transfer (resp., regular intra-chain) transaction. Then the advantage of $\mathcal{A}$ winning the game is $\Pr_{\text{ind}} := |\Pr[D(tx_0, \sigma_0) = 0] - \Pr[D(tx_1, \sigma_1) = 0]|$.

Definition 9 (Indistinguishability). *P2C2T satisfies indistinguishability if there exists a negligible function* $\mathsf{negl}(n)$ *such that for all $n \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}$,* $\Pr_{\text{ind}} \leq \mathsf{negl}(n)$.

Theorem 4. *Assuming the security of Π^S and Π^{2AS}, P2C2T satisfies indistinguishability.*

Proof (Sketch). Regular intra-chain transaction-signature pairs are used to compare with cross-chain transfer transaction-signature pairs in the indistinguishability game. As can be seen in description of P2C2T, there are six transaction-signature pairs related to cross-chain transfer, i.e., $(tx^{lock}_s, \sigma^{lock}_s)$, $(tx^{lock}_{h2}, \sigma^{lock}_{h2})$, $(tx^{trans}_{h1}, \sigma^{trans}_{h1})$, $(tx^{trans}_r, \sigma^{trans}_r)$, $(tx^{redeem}_s, \sigma^{redeem}_s)$, and $(tx^{redeem}_{h2}, \sigma^{redeem}_{h2})$.

P2C2T introduces several defense measures. It requires only on-chain signature verification functionality. Additionally, P_h and users are encouraged to create intra-chain transactions with identical coin amounts as cross-chain transactions within the same epoch. They must also privately share public keys for deriving asset addresses. If, despite these precautions, transaction-signature pairs remain distinguishable by an on-chain observer, two possible scenarios may arise, as outlined below.

(1) Two kinds of transactions themselves are distinguishable. It means shared addresses are distinguishable from non-shared ones, which implies the distinguishability of joint verification keys and regular verification keys. It further implies the distinguishability of two statistically indistinguishable elements in $\mathbb{G}$ since two kinds of keys above are generated randomly in $\mathbb{G}$, which is a contraction.

(2) Signatures on cross-chain transfer transactions and the verification of them are distinguishable from those of regular intra-chain transactions. It means either signing keys or verification algorithms of these two kinds of signatures are distinguishable. The former implies the distinguishability of two random values in $\mathbb{Z}_p$, which is impossible. The latter does not hold since these two kinds of signatures adopt the same verification algorithms, i.e., $\Pi^S.Verify$.

□

Discussion on griefing attack resistance. We argue that assuming the soundness and zero-knowledgeness of Π^{NIZK}, the strong existential unforgeability of digital signature derived by Π^{RS}, the security of transactions on interconnected blockchains, and the security of set $\mathcal{T}$, P2C2T satisfies griefing attack resistance. Specially, P_h can verify the commitment of the request token by running the verification algorithm of Π^{NIZK} and determine whether tx_s^{lock} is finalized on B_1 by querying id_s^{lock}. If both verifications are successful, P_h signs the request token for P_s. In addition, P_h can check if the request token is replayed maliciously by verifying if $tid \in \mathcal{T}$. Hence, if P_r successfully initiates a request to P_h in the puzzle promising phase, P_s must have locked corresponding coins on B_1 in the transfer lock phase. And then if P_r does not honestly cooperate with P_h, P_s has to wait for unlocking its locked coins until the transfer timeout phase. Thus, a griefing attack leads to a loss for P_s, which is not desirable for P_r.

6. Performance Evaluation

In this section, we first evaluate the performance of P2C2T by comparing it experimentally with a baseline scheme described in [20] in terms of running time (RT), communication cost (CC), and storage cost (SC). Then, we illustrate the privacy and practicality of P2C2T by conducting cross-chain transfers and intra-chain payments using the Bitcoin testnet and Litecoin testnet.

6.1. Performance Comparison

In this subsection, we set up experiments, present experimental results, and compare them with a baseline scheme described in [20]. We choose the scheme in [20] as the baseline for comparison because it closely aligns with P2C2T in terms of specified requirements, as indicated in Table 1.

6.1.1. Experimental Setup. We now introduce the experimental setup in terms of evaluation metrics and experimental settings.

Evaluation Metrics. We use RT, CC, and SC as evaluation metrics based on the following factors:

(1) RT and CC are widely accepted metrics for assessing the performance of the synchronization puzzle protocol [3], [4]. SC represents a critical area of improvement for Zerocash-based blockchains; indeed the baseline scheme [20] is specifically designed for interconnecting these blockchains.

(2) These metrics not only directly gauge off-chain performance but are also significant for on-chain assessments. For instance, transaction finalization time is integral to the overall RT, and CC encompasses transaction byte size, which influences transaction fees on some blockchains such as Bitcoin. Parties responsible for finalizing blocks have to store all transaction-related data locally, leading to significant SC.

Experimental Settings. For the scheme in [20], we adopt a Merkle tree depth of 32 to store coin commitments, aligning with the depths used in Sapling and Orchard shielded pools of Zcash blockchain [45], an implementation of Zerocash with security fixes and performance improvements. We incorporate a bit 1 into coin commitments as a flag to indicate cross-chain transactions and a bit 0 for intra-chain transactions.

For P2C2T, we employ the $secp256k1$ curve [46], which is also used in Bitcoin. With reference to [27], the parameters for Π^{VTD} are set as follows: a statistical parameter-threshold pair of (30,15) for the cut-and-choose proof, a security parameter of 30 for range proofs, a 320-bit integer for the interval parameter in range proofs, a timing hardness $T = 1000000$ and a 1024-bit integer for the RSA integer in time-lock puzzles [44].

Experimental tests are conducted with the following simplifications:

(1) For the scheme in [20], we focus on testing its primary overhead, particularly related to zk-SNARKs schemes [47] (say Π^{ZKS}, a kind of Π^{NIZK}) and relayed block headers. We exclude additional overhead from OR proofs, signing, and encrypting operations, as these items are not discussed or instantiated in [20].

(2) Regarding P2C2T, we do not measure the transfer timeout algorithm, which primarily acts as a deterrent for dishonest participants. In addition, we focus exclusively on schemes utilizing the ECDSA signature scheme, as it is the most widely used signature scheme in mainstream blockchains and supports two-party adaptor signatures.

(3) Comparing the on-chain overhead of these two schemes is inconclusive because they target at different blockchains in cross-chain transfer. P2C2T operates on blockchains with unshielded asset addresses, like Bitcoin addresses, while the scheme in [20] requires operation on shielded addresses, such as those used in Zcash. Thus, we cannot compare their operation performance through cross-chain transfer between the Bitcoin testnet and the Litecoin testnet. In our assessment of RT, we still identify necessary on-chain overhead for a more comprehensive comparison. Specifically, we use the time taken for verifying transactions from the perspective of cryptographic operations to simulate the on-chain RT. This verification process includes verifying signatures in P2C2T and zk-SNARKs proofs in [20][3]. All operations other than these verifications belong to the off-chain part. When evaluating CC and SC, we do not differentiate between the off-chain and on-chain parts to simplify our performance analysis. We focus on reusable data in multiple cross-chain transfers when evaluating SC.

We implement the scheme in [20] using a C++ library [48] for the Zerocash protocol. Additionally, we develop P2C2T in the C language, relying on the RELIC library [49] and the GMP library [50] for cryptographic operations, the PARI library [51] for arithmetic operations in class groups, and the ZeroMQ library [52]

3. In reality, on-chain signature verification is also needed in [20] to ensure transaction non-malleability [24]. We do not measure RT of it based on the simplifications discussed before.

for message delivery. The source code is available at https://github.com/smallfrog/ptoctot. Our experiments are conducted on a single machine running Ubuntu 20.04 LTS, equipped with a 2.50GHz Intel Core i5-3210M processor, 4 cores, and 12GB of RAM.

6.1.2. Experimental Results. We denote by m the number of finalized cross-chain transfers between senders and receivers. The experimental results presented below are the averages obtained from 20 implementations to ensure robustness.

For the scheme in [20], we test the performance of the zk-SNARKs scheme for proving and verifying transactions with any type flag, as different flags do not affect the results. The results indicate that RT of $\Pi^{ZKS}.Setup$, $\Pi^{ZKS}.Prove$, and $\Pi^{ZKS}.Verify$ take 247.957 seconds, 72.476 seconds, and 0.039 seconds, respectively. The size of the proving key and verification key generated through $\Pi^{ZKS}.Setup$ are 252192.732 kilobytes and 0.638 kilobytes, respectively. The size of the proof generated through $\Pi^{ZKS}.Prove$ is 0.280 kilobytes. Furthermore, the scheme [20] requires relaying block headers from the source chain (say B_s) to the destination chain (say B_d). Let $H.size$ denote the average size of block headers from B_s, h represent the height of B_s, and l indicate the number of miners for B_s. Hence, RT of the off-chain part for [20] is at least $247.957 + m \cdot 144.952$ seconds[4], while RT of the on-chain part is at least $m \cdot 0.078$ seconds[5]. CC is at least $504386.740+m\cdot 0.560+h\cdot l\cdot H.size$ kilobytes [6] and SC at least $504386.740 + h \cdot l \cdot H.size$ kilobytes [7].

For P2C2T, only $\Pi^{LOE}.KeyGen$, $\Pi^{RS}.KeyGen$, and the setup algorithm of time-lock puzzles [44] in system setup phase are one-time operations, with RT of 0.540 seconds, 0.002 seconds, and 11.494 seconds respectively. The parameters generated have sizes of 3.449 kilobytes, 0.192 kilobytes and 0.423 kilobytes, respectively. All other operations for a cross-chain transfer in P2C2T has RT of the off-chain part of 44.983 seconds, RT of the on-chain part of 0.004 seconds, CC of 148.443 kilobytes, and no storage cost.

6.1.3. Performance Comparison. Building upon the previous description, the performance comparison between [20] and P2C2T for m cross-chain transfers is presented in Table 2. P2C2T demonstrates significant improvements in various aspects:

(1) P2C2T saves at least 68.967% of RT of the off-chain part compared to [20]. Even when $m = 1$, RT of the off-chain part of P2C2T accounts for only 14.512% of that of [20].

(2) RT of the on-chain part of P2C2T is only 5.128% of that of [20].

(3) P2C2T eliminates the need for miners to relay block headers, resulting in significant savings in CC and SC, compared to [20]. For instance, assuming a single miner ($l = 1$), a block height of $h = 2447490$ that is also the height of some finalized block on Zcash blockchain[8], and a block header size of $H.size = 1.452$ kilobytes that is also the block header size of Zcash blockchain, both CC and SC for the miner to relay block headers amount to 3553755.480 kilobytes. Moreover, both CC and SC for the miner linearly increase with the height of source chain.

(4) Without considering miner overhead, CC of P2C2T accounts for only 0.030% of that of [20] for a pair of sender and receiver when $m = 1$. Although CC of P2C2T increases faster than that of [20] with the number of cross-chain transfers, it remains favorable unless $m \geq 3411$, which is uncommon in practice. Notably, P2C2T incurs no SC for a pair of sender and receiver, as only 4.064 kilobytes of data are stored and maintained by P_h. In contrast, [20] requires a sender-receiver pair to store a total of 504386.740 kilobytes.

In summary, P2C2T reduces overhead by at least 85.488% in terms of running time, communication cost, and storage cost when completing a cross-chain transfer. We note that a significant portion of RT and CC in P2C2T is attributed to the cut-and-choose style proof of Π^{VTD}. Since the cryptographic operations adopted by P2C2T impose significantly less overhead than those used by [20], P2C2T emerges as a more efficient choice compared to the main part of [20] and hence [20] itself.

Discussion. If P2C2T were adapted to support cross-chain transfer across Zerocash-based blockchains, it would require four on-chain transactions – double of those of [20]. Consequently, transaction fees for the adapted scheme would also be doubled compared to [20]. However, such an adapted scheme currently does not exist and is left for future work.

TABLE 2: Performance Comparison between [20] and P2C2T for m Cross-Chain Transfers

	RT		**CC**	**SC**
	OFF	**ON**		
[20]	$247.957 + m \cdot 144.952$	$m \cdot 0.078$	$504386.740 + m \cdot 0.560 + h \cdot l \cdot H.size$	$504386.740 + h \cdot l \cdot H.size$
P2C2T	$12.036 + m \cdot 44.983$	$m \cdot 0.004$	$4.064 + m \cdot 148.443$	4.064
ROR	85.488%	94.872%	99.970%	99.999%

RT: running time (in seconds); CC: communication cost (in kilobytes); SC: storage cost (in kilobytes); OFF: off-chain part; ON: on-chain part; h, l, and $H.size$ denote the height, the number of miners, and the average size of all block headers of source chain, respectively; $ROR = (o_1 - o_2)/o_1 \times 100\%$, where o_1 and o_2 represent the overhead of the scheme in [20] and P2C2T, respectively, when $m = 1$, excluding miner overhead (i.e., $h \cdot l \cdot H.size := 0$).

4. It comprises the time taken for $\Pi^{ZKS}.Setup$ as a one-time operation, along with two runs of $\Pi^{ZKS}.Prove$ for two intra-chain transactions on B_s and B_d, respectively.

5. It comprises the time taken for two runs of $\Pi^{ZKS}.Verify$.

6. It includes the message size for transferring two proving-verification key pairs on B_s and B_d, respectively, as a one-time operation, two proofs for two intra-chain transactions on B_s and B_d, respectively, and h block headers relayed by every miner.

7. It involves the storage size of two proving-verification key pairs for the sender and corresponding receiver, respectively, along with h block headers for every miner.

8. The block hash is 0000000000ce2ef802d1f7754fb1901f1068836e1ea0 8f5753655c15e0105a34.

TABLE 3: Transaction Identifiers in Experimental Examples of Cross-Chain Transfers and Intra-Chain Payments

	txid_1	**txid_2**
$tx^{cc}_{1,1}$	a17ea464a0392e481d16f751e8e875ed27ce408391b8fa35ea778c1ab1f5f8b8	3772f0caf8878599d2cbb7e1ef8ae660720baea95061bb5b0a68dc810236d8ad
$tx^{cc}_{1,2}$	f30e1495a1b1d2af049860ef3b34d58c867f322b43ce15443e375194a313538e	751dbafcb61829b0930e201085b16c0a4d2210f0d454a6bf2d911331ffb5c134
$tx^{cc}_{2,1}$	fb01023235fd023d533af60082bad032c32aaba063348bccec322106b02dab33	9f8e85fa0d13118bb0f53e358bc6945ed924b10f6633e87409b6efa8ad43fa05
$tx^{cc}_{2,2}$	c3693e62dfb8a84bb6f2dd79721be67c6abda4caae02e8b556a7d48350de008c	40830eadfc2e0ec849a73f21a4c1798b595a0ee8bd92439fb8990182a1235dc5
tx^{ic}_{1}	d1c1df85c880874b7925ee09c9d3f12888a7269207878ca5d9cd5d45bc038ad5	160c0774b55f4f673486477b287220cdd802d63aa4a814e1f409b638568a608a
tx^{ic}_{2}	13569aa27aa11cdedd9e371dd69ecfe44af440140dea5f1b0a5cc96f0d674c31	19280e09f1ffc06a873d827338d94e7340ee4548dda55b3be84e4a0e235b5bce

txid_1 and txid_2 represent the transaction identifiers of two consecutive payments on one testnet, respectively.

6.2. Privacy and Practicality Illustration

The privacy and practicality of P2C2T have been rigorously addressed in Sections 5, providing theoretical evidence. This subsection offers experimental examples to further illustrate the privacy and practicality of P2C2T, serving as supplementary verification. Specifically, we conduct cross-chain transfers and intra-chain payments using the Bitcoin testnet (B_1) and the Litecoin testnet (B_2). To finalize posted transactions, we let users wait for a minimum of 6 block confirmations on the Bitcoin testnet and at least 12 block confirmations on the Litecoin testnet[9]. Setting $x = 0.0001$ and $y = 0.074$ as described in Subsection 5.1 based on exchange rates observed in April 2024 [1][10], we execute two complete cross-chain transactions, $tx^{cc}_1 = (tx^{cc}_{1,1}, tx^{cc}_{1,2})$ and $tx^{cc}_2 = (tx^{cc}_{2,1}, tx^{cc}_{2,2})$ where $tx^{cc}_{i,j}$ involves two payments (i.e., a lock transaction and a transfer transaction) on B_j ($i, j \in \{1, 2\}$). The transaction identifiers of these payments are presented in Table 3. Notably, linking $tx^{cc}_{i,j}$ to $tx^{cc}_{i,3-j}$ rather than $tx^{cc}_{3-i,3-j}$ is challenging, which illustrates the unlinkability of P2C2T in terms of on-chain transactions. Additionally, we conduct two intra-chain payments on each testnet, where both payments involve the same address, with one depositing coins and the other withdrawing coins. We denote by tx^{ic}_1 the two payments on the Bitcoin testnet and tx^{ic}_2 the two payments on the Litecoin testnet. Distinguishing between payments related to $tx^{cc}_{i,j}$ and tx^{ic}_j, as indicated by their transaction identifiers in Table 3, presents an intuitive challenge ($i, j \in \{1, 2\}$). Importantly, tx^{cc}_1 and tx^{cc}_2 do not require sender coin collateralization and only utilize signature verification functionalities, highlighting the practicality of P2C2T.

7. Conclusion

In this paper, we proposed P2C2T, the first cross-chain transfer scheme that satisfies a set of critical criteria in an integrated way, including atomicity, unlinkability, indistinguishability, non-collateralization, and minimization of required functionalities. At the core of P2C2T, we presented a novel synchronization puzzle protocol instance, TA^{2}L. We established the security of TA^{2}L based on the threshold blind conditional signatures and highlighted the compelling properties of P2C2T through necessary discussions and proofs. We demonstrated the superior performance of P2C2T in terms of running time, communication cost, and storage cost by comparing it with an existing scheme with properties closest to P2C2T. Additionally, we illustrated the privacy and practicality of P2C2T by conducting cross-chain transfers and intra-chain payments using the Bitcoin testnet and the Litecoin testnet. In the future, we are going to extend our scheme to support variable-amount cross-chain transfers and accommodate signature schemes that feature unique signatures.

9. 6-block-confirmation represents a fundamental security measure on Bitcoin against common attacks, including double-spending attacks. However, due to the smaller block rewards and higher stale block rate of Litecoin compared to Bitcoin, it requires additional block confirmations to achieve equivalent security levels [53]. Following the recommendation by [54], 12-block-confirmation is adopted on the Litecoin testnet to enhance security.

10. We aim to simulate real cross-chain transactions, despite the fact that coins on testnets hold no real-world value.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China under Grants U23A20300 and 62072351; in part by the Key Research Project of Shaanxi Natural Science Foundation under Grant 2023-JC-ZD-35; in part by the Concept Verification Funding of Hangzhou Institute of Technology of Xidian University under Grant GNYZ2024XX007, and in part by the 111 Project under Grant B16037.

References

[1] B. Chez. (2013) Coinmarketcap. Coinmarketcap. [Online]. Available: https://coinmarketcap.com/

[2] P. Han, Z. Yan, W. Ding, S. Fei, and Z. Wan, “A survey on cross-chain technologies,” *Distributed Ledger Technologies: Research and Practice*, vol. 2, no. 2, pp. 1–30, 2023.

[3] E. Heilman, L. AlShenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg, “Tumblebit: an untrusted bitcoin-compatible anonymous payment hub,” 2017. [Online]. Available: https://open.bu.edu/handle/2144/29224

[4] E. Tairi, P. Moreno-Sanchez, and M. Maffei, “A 2 l: Anonymous atomic locks for scalability in payment channel hubs,” in *2021 IEEE Symposium on Security and Privacy (SP)*, IEEE. San Francisco, CA, USA: IEEE, 2021, pp. 1834–1851.

[5] S. A. Thyagarajan, G. Malavolta, and P. Moreno-Sanchez, “Universal atomic swaps: Secure exchange of coins across all blockchains,” in *2022 IEEE Symposium on Security and Privacy (SP)*, IEEE. San Francisco, CA, USA: IEEE, 2022, pp. 1299–1316.

[6] C. Zhao. (2017) Binance exchange. Binance. [Online]. Available: https://www.binance.com/en

[7] I. Bentov, Y. Ji, F. Zhang, L. Breidenbach, P. Daian, and A. Juels, "Tesseract: Real-time cryptocurrency exchange using trusted hardware," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 1521–1538.

[8] H. Xie and Z. Yan, "Spcex: Secure and privacy-preserving cryptocurrency exchange," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–14, 2024.

[9] W. Warren and A. Bandeali, "0x: An open protocol for decentralized exchange on the ethereum blockchain," *URl: https://github.com/0xProject/whitepaper*, pp. 04–18, 2017.

[10] S. Chu, Q. Xia, and Z. Zhang, "Manta: Privacy preserving decentralized exchange," Cryptology ePrint Archive, Paper 2020/1607, 2020. [Online]. Available: https://eprint.iacr.org/2020/1607

[11] C. Baum, B. David, and T. K. Frederiksen, "P2dex: privacy-preserving decentralized cryptocurrency exchange," in *International Conference on Applied Cryptography and Network Security*, Springer. Cham: Springer International Publishing, 2021, pp. 163–194.

[12] H. Tian, K. Xue, X. Luo, S. Li, J. Xu, J. Liu, J. Zhao, and D. S. Wei, "Enabling cross-chain transactions: A decentralized cryptocurrency exchange protocol," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3928–3941, 2021.

[13] A. Kiayias and D. Zindros, "Proof-of-work sidechains," in *Financial Cryptography and Data Security: FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*, Springer. Cham: Springer International Publishing, 2020, pp. 21–34.

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

[15] T. Xie, J. Zhang, Z. Cheng, F. Zhang, Y. Zhang, Y. Jia, D. Boneh, and D. Song, "zkbridge: Trustless cross-chain bridges made practical," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2022, pp. 3003–3017.

[16] Z. Yin, B. Zhang, J. Xu, K. Lu, and K. Ren, "Bool network: An open, distributed, secure cross-chain notary platform," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3465–3478, 2022.

[17] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, "Enabling blockchain innovations with pegged sidechains," *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, vol. 72, pp. 201–224, 2014.

[18] P. Gaži, A. Kiayias, and D. Zindros, "Proof-of-stake sidechains," in *2019 IEEE Symposium on Security and Privacy (SP)*, IEEE. San Francisco, CA, USA: IEEE, 2019, pp. 139–156.

[19] L. Hanzlik, J. Loss, S. A. Thyagarajan, and B. Wagner, "Sweep-uc: Swapping coins privately," Cryptology ePrint Archive, Paper 2022/1605, 2022. [Online]. Available: https://eprint.iacr.org/2022/1605

[20] F. Baldimtsi, I. Miers, and X. Zhang, "Anonymous sidechains," in *International Workshop on Data Privacy Management*, Springer. Cham: Springer International Publishing, 2021, pp. 262–277.

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

[22] X. Qin, S. Pan, A. Mirzaei, Z. Sui, O. Ersoy, A. Sakzad, M. F. Esgin, J. K. Liu, J. Yu, and T. H. Yuen, "Blindhub: Bitcoin-compatible privacy-preserving payment channel hubs supporting variable amounts," in *2023 IEEE symposium on security and privacy (SP)*. IEEE, 2023, pp. 2462–2480.

[23] N. Glaeser, M. Maffei, G. Malavolta, P. Moreno-Sanchez, E. Tairi, and S. A. K. Thyagarajan, "Foundations of coin mixing services," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2022, pp. 1259–1273.

[24] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE symposium on security and privacy*, IEEE. Berkeley, CA, USA: IEEE, 2014, pp. 459–474.

[25] A. Lohachab, S. Garg, B. Kang, M. B. Amin, J. Lee, S. Chen, and X. Xu, "Towards interconnected blockchains: A comprehensive review of the role of interoperability among disparate blockchains," *ACM Computing Surveys (CSUR)*, vol. 54, no. 7, pp. 1–39, 2021.

[26] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, "Anonymous multi-hop locks for blockchain scalability and interoperability," Cryptology ePrint Archive, Paper 2018/472, 2018. [Online]. Available: https://eprint.iacr.org/2018/472

[27] S. A. K. Thyagarajan, A. Bhat, G. Malavolta, N. Döttling, A. Kate, and D. Schröder, "Verifiable timed signatures made practical," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1733–1750.

[28] S. Nakamoto. (2009) Bitcoin core. The Bitcoin Core project. [Online]. Available: https://bitcoincore.org/en/download/

[29] L. project development team. (2011) Litecoin. Litecoin Foundation. [Online]. Available: https://litecoin.org/

[30] R. K. Lyons and G. Viswanath-Natraj, "What keeps stablecoins stable?" *Journal of International Money and Finance*, vol. 131, p. 102777, 2023.

[31] R. C. Merkle, "Protocols for public key cryptosystems," in *1980 IEEE symposium on security and privacy*, IEEE. Oakland, CA, USA: IEEE, 1980, pp. 122–122.

[32] J. Groth, "On the size of pairing-based non-interactive arguments," in *Annual international conference on the theory and applications of cryptographic techniques*, Springer. Cham: Springer International Publishing, 2016, pp. 305–326.

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

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

[35] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal on computing*, vol. 17, no. 2, pp. 281–308, 1988.

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

[37] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostakova, M. Maffei, P. Moreno-Sanchez, and S. Riahi, "Generalized bitcoin-compatible channels," E192-06 - Forschungsbereich Security and Privacy, Tech. Rep., 2020.

[38] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Annual international cryptology conference*, Springer. Cham: Springer International Publishing, 1991, pp. 129–140.

[39] I. Damgård, "Commitment schemes and zero-knowledge protocols," in *School organized by the European Educational Forum*, Springer. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 63–86.

[40] D. Pointcheval and O. Sanders, "Short randomizable signatures," in *Cryptographers' Track at the RSA Conference*, Springer. Cham: Springer International Publishing, 2016, pp. 111–126.

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

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

[43] G. Castagnos and F. Laguillaumie, "Linearly homomorphic encryption from ddh," in *Topics in Cryptology — CT-RSA 2015*, Springer. Cham: Springer International Publishing, 2015, pp. 487–505.

[44] G. Malavolta and S. A. K. Thyagarajan, "Homomorphic time-lock puzzles and applications," in *Advances in Cryptology – CRYPTO 2019*, A. Boldyreva and D. Micciancio, Eds., Springer. Cham: Springer International Publishing, 2019, pp. 620–649.

[45] D. Hopwood, S. Bowe, T. Hornby, N. Wilcox *et al.*, "Zcash protocol specification," *GitHub: San Francisco, CA, USA*, vol. 4, no. 220, p. 32, 2016.

[46] Secp256k1. (2011) Secp256k1. MediaWiki. [Online]. Available: https://en.bitcoin.it/wiki/Secp256k1

[47] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct non-interactive zero knowledge for a von neumann architecture," in *23rd USENIX Security Symposium (USENIX Security 14)*, USENIX AssociatiDon. San Francisco, CA, USA: USENIX Association, 2014, pp. 781–796.

[48] T. H. Eran Tromer, Madars Virza. (2015) libzerocash: a c++ library implementing the zerocash protocol. Zerocash. [Online]. Available: https://github.com/Zerocash/libzerocash

[49] D. F. Aranha, C. P. L. Gouvêa, T. Markmann, R. S. Wahby, and K. Liao, "RELIC is an Efficient LIbrary for Cryptography," https://github.com/relic-toolkit/relic, 2014.

[50] T. Granlund. (2004) Gnu mp: The gnu multiple precision arithmetic library. ZeroMQ. [Online]. Available: https://gmplib.org/

[51] H. Cohen *et al.* (2003) Pari. Laboratoire A2X. [Online]. Available: http://pari.math.u-bordeaux.fr/

[52] ZeroMQ. (2009) The zeromq project. ZeroMQ. [Online]. Available: https://github.com/zeromq/libzmq

[53] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 3–16.

[54] Kraken. (2011) Kraken's confirmations requirements. Kraken. [Online]. Available: https://support.kraken.com/hc/en-us/articles/203325283-Cryptocurrency-deposit-processing-times

[55] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for cryptographers," *Discrete Applied Mathematics*, vol. 156, no. 16, pp. 3113–3121, 2008.

Appendix A. Additional Preliminaries

A.1. Two-Party Adaptor Signatures

Definition 10 (Two-Party Adaptor Signatures). A two-party adaptor signature scheme with respect to a hard relationship R and a digital signature $\Pi^S := (KeyGen, Sign, Verify)$ with a message space $\mathcal{M}$ generally consists of five algorithms $\Pi^{2AS} := (JointKeyGen, JointPreSign, PreVerify, Adapt, Ext)$. Here $JointKeyGen$ and $JointPreSign$ are run between two parties (say P_1 and P_2). These algorithms are defined as follows.

(1) $(\{sk_1, pk_2, pk\}, \{sk_2, pk_1, pk\}) \leftarrow JointKeyGen\langle P_1(\ ), P_2(\ )\rangle$ is an interactive protocol between P_1 (with inputs some public parameters) and P_2 (with inputs some public parameters). It returns sk_1, pk_2, and pk to P_1 and sk_2, pk_1, and pk to P_2.

(2) $(\{\tilde{\sigma}, \perp\}, \{\tilde{\sigma}, \perp\}) \leftarrow JointPreSign\langle P_1(sk_1, pk, m, Y), P_2(sk_2, pk, m, Y)\rangle$ is an interactive protocol between P_1 (with inputs the signing key share sk_1, the joint verification key pk, a message m, and a statement Y) and P_2 (with inputs the signing key share sk_2, the joint verification key pk, a message m, and a statement Y). It returns either a pre-signature $\tilde{\sigma}$ or $\perp$ to both parties.

(3) $b := PreVerify(pk, m, Y, \tilde{\sigma})$ is a DPT algorithm that takes as input the joint verification key pk, the message m, the statement Y, and the pre-signature $\tilde{\sigma}$. It returns a bit b.

(4) $\sigma := Adapt(y, \tilde{\sigma})$ is a DPT algorithm that takes as input the witness y and the pre-signature $\tilde{\sigma}$. It returns a signature σ.

(5) $y := Ext(\sigma, \tilde{\sigma}, Y)$ is a DPT algorithm that takes as input the signature σ, the pre-signature $\tilde{\sigma}$, and the statement Y. It returns a witness y.

Next, we can define two-party adaptor signature correctness, two-party signature unforgeability, two-party pre-signature adaptability, and two-party witness extractability respectively.

Definition 11 (Two-Party Adaptor Signature Correctness). A two-party adaptor signature scheme Π^{2AS} is correct if for all $n \in \mathbb{N}$, all messages $m \in \mathcal{M}$, all hard relationships $(Y, y) \leftarrow GenR(1^n)$, and all $(\{sk_1, pk_2, pk\}, \{sk_2, pk_1, pk\}) \in JointKeyGen\langle P_1(\ ), P_2(\ )\rangle$, the following holds: $PreVerify(pk, m, Y, \tilde{\sigma}) = 1$, $\Pi^S.Verify(pk, m, \sigma) = 1$, and $(Y, y') \in R$ where $\tilde{\sigma} \leftarrow JointPreSign\langle P_1(sk_1, pk, m, Y), P_2(sk_2, pk, m, Y)\rangle$, $\sigma := Adapt(y, \tilde{\sigma})$, and $y' := Ext(\sigma, \tilde{\sigma}, Y)$.

Definition 12 (Two-Party Signature Unforgeability). Π^{2AS} satisfies two-party signature unforgeability if there exists a negligible function $\mathsf{negl}(n)$ such that for all $n \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}$ playing the role of P_i $(i \in \{1, 2\})$, the following holds: $\Pr[\mathsf{ExpSigForge}^{\mathcal{A}-P_i}_{\Pi^{2AS}}(n) = 1] \leq \mathsf{negl}(n)$, where $\mathsf{ExpSigForge}^{\mathcal{A}-P_i}_{\Pi^{2AS}}$ is defined in Fig. 10.

Definition 13 (Two-Party Pre-Signature Adaptability). Π^{2AS} satisfies two-party pre-signature adaptability if for all $n \in \mathbb{N}$, all messages $m \in \mathcal{M}$, all hard relationships $(Y, y) \leftarrow GenR(1^n)$, all $(\{sk_1, pk_2, pk\}, \{sk_2, pk_1, pk\}) \in JointKeyGen\langle P_1(\ ), P_2(\ )\rangle$, all pre-signatures $\tilde{\sigma}$ satisfying $PreVerify(pk, m, Y, \tilde{\sigma}) = 1$, the following holds: $\Pr[\Pi^S.Verify(pk, m, Adapt(y, \tilde{\sigma})) = 1] = 1$.

Definition 14 (Two-Party Witness Extractability). Π^{2AS} satisfies two-party signature extractability if there exists a negligible function $\mathsf{negl}(n)$ such that for all $n \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}$ playing the role of P_i $(i \in \{1, 2\})$, the

```
$\mathsf{ExpSigForge}^{\mathcal{A}-P_i}_{\Pi^{2AS}}(n)$
$\mathcal{Q} := \emptyset$
$(\{sk_{3-i}, pk_i, pk\}, \{sk_i, pk_{3-i}, pk\})$
        $\leftarrow JointKeyGen\langle P_{3-i}(\ ), \mathcal{A}(\ )\rangle$
$m^* \leftarrow \mathcal{A}^{\mathcal{O}^S(\cdot), \mathcal{O}^{PreS}(\cdot)}$
$(Y, y) \leftarrow GenR(1^n)$
$\tilde{\sigma} \leftarrow JointPreSign\langle P_{3-i}(sk_{3-i}, pk, m^*, Y), \mathcal{A}(Y)\rangle$
$\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}^S(\cdot), \mathcal{O}^{PreS}(\cdot)}$
return $(m^* \notin \mathcal{Q}) \wedge (\Pi^S.Verify(pk, m^*, \sigma^*))$

$\underline{\mathcal{O}^S(m)}$
$\sigma \leftarrow \Pi^S.Sign(sk, m)$ where $sk$ is the signing key
corresponding to $pk$.
$\mathcal{Q} := \mathcal{Q} \cup \{m\}$

$\underline{\mathcal{O}^{PreS}(m, Y)}$
$\tilde{\sigma} \leftarrow JointPreSign\langle P_{3-i}(sk_{3-i}, pk, m, Y), \mathcal{A}\rangle$
$\mathcal{Q} := \mathcal{Q} \cup \{m\}$
```

Figure 10: Two-party signature unforgeability experiment

```
$\mathsf{ExpWitExt}^{\mathcal{A}-P_i}_{\Pi^{2AS}}(n)$
$\mathcal{Q} := \emptyset$
$(\{sk_{3-i}, pk_i, pk\}, \{sk_i, pk_{3-i}, pk\})$
        $\leftarrow JointKeyGen\langle P_{3-i}(\ ), \mathcal{A}(\ )\rangle$
$(m^*, Y^*) \leftarrow \mathcal{A}^{\mathcal{O}^S(\cdot), \mathcal{O}^{PreS}(\cdot)}$
$\tilde{\sigma} \leftarrow JointPreSign\langle P_{3-i}(sk_{3-i}, pk, m^*, Y^*), \mathcal{A}\rangle$
$\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}^S(\cdot), \mathcal{O}^{PreS}(\cdot)}$
$y^* := Ext(\sigma, \tilde{\sigma}, Y^*)$
return $(m^* \notin \mathcal{Q}) \wedge ((Y^*, y^*) \notin R)$
            $\wedge (\Pi^S.Verify(pk, m^*, \sigma^*))$

$\underline{\mathcal{O}^S(m)}$
$\sigma \leftarrow \Pi^S.Sign(sk, m)$ where $sk$ is the signing key
corresponding to $pk$.
$\mathcal{Q} := \mathcal{Q} \cup \{m\}$

$\underline{\mathcal{O}^{PreS}(m, Y)}$
$\tilde{\sigma} \leftarrow JointPreSign\langle P_{3-i}(sk_{3-i}, pk, m, Y), \mathcal{A}\rangle$
$\mathcal{Q} := \mathcal{Q} \cup \{m\}$
```

Figure 11: Two-party witness extractability experiment

following holds: $\Pr[\mathsf{ExpWitExt}^{\mathcal{A}-P_i}_{\Pi^{2AS}}(n) = 1] \leq \mathsf{negl}(n)$, where $\mathsf{ExpWitExt}^{\mathcal{A}-P_i}_{\Pi^{2AS}}$ is defined in Fig. 11.

A.2. Bilinear Groups

Bilinear Groups. Bilinear groups are a set of three cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ of prime order p along with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. These groups satisfy the following properties.

(1) The map e is efficiently computable.

(2) For all $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.

(3) For $g_1 \neq 1_{\mathbb{G}_1}$ and $g_2 \neq 1_{\mathbb{G}_2}$, $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_1}$, $1_{\mathbb{G}_2}$, and $1_{\mathbb{G}_T}$ are the neutral elements of the group $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ respectively.

Bilinear groups are classified as type 3 pairings if $\mathbb{G}_1 \neq \mathbb{G}_2$ and no efficiently computable homomorphism exists between $\mathbb{G}_1$ and $\mathbb{G}_2$ [55].

```
$\text{OM-CCA-A2L}^{\mathcal{A}}_{\Pi^{LOE}, d}(n)$
$D := 0$
$(sk^L, pk^L) \leftarrow \Pi^{LOE}.KeyGen(1^n)$
$r_1, ..., r_{d+1} \leftarrow_\$ \mathbb{Z}_p$
$c_i \leftarrow \Pi^{LOE}.Enc(pk^L, r_i)$
$r'_1, ..., r'_{d+1} \leftarrow \mathcal{A}^{\mathcal{O}^{A^2L}(\cdot)}(pk^L, (c_1, g^{r_1}), ..., (c_{d+1}, g^{r_{d+1}}))$
if $(r_i = r'_i, \forall i \in [d+1]) \wedge (D \leq d)$
    return 1
else
    return 0

$\underline{\mathcal{O}^{A^2L}(pk, m, h, c, \tilde{\sigma})}$
if $(\cdot, pk) \in \Pi^{2AS}.KeyGen(1^n)$
    $x \leftarrow \Pi^{LOE}.Dec(sk^L, c)$
else
    return $\perp$
if $(\Pi^{2AS}.PreVerify(pk, m, h, \tilde{\sigma}) = 1) \wedge (h = g^x)$
    $D := D + 1$
    return $\sigma \leftarrow \Pi^{2AS}.Adapt(x, \tilde{\sigma})$
else
    return $\perp$
```

Figure 12: OM-CCA-A2L game

A.3. OM-CCA-A2L Security

Definition 15 (OM-CCA-A2L) [23]. An encryption scheme Π^E is OM-CCA-A2L secure if there exists a negligible function $\mathsf{negl}(n)$ such that for all $n \in \mathbb{N}$, all polynomials d, and all PPT adversaries $\mathcal{A}$, the following holds: $\Pr[OM - CCA - A2L^{\mathcal{A}}_{\Pi^E, d}(n) = 1] \leq \mathsf{negl}(n)$, where $OM - CCA - A2L^{\mathcal{A}}_{\Pi^E, d}$ is defined in Fig. 12.

Appendix B. Properties of TBCS

Definition 2 (Correctness). *A threshold blind conditional signature Π^{TBCS} is correct if for all $n \in \mathbb{N}$, all $(sk^L, pk^L) \in \Pi^{LOE}.KeyGen(1^n)$, all $(\{sk_s^0, pk_{h1}^0, pk_1\}, \{sk_{h1}^0, pk_s^0, pk_1\}) \in \Pi^{2AS}.JointKeyGen\langle P_s(\quad), P_h(\quad)\rangle$, all $(\{sk_r^0, pk_{h2}^0, pk_2\}, \{sk_{h2}^0, pk_r^0, pk_2\}) \in \Pi^{2AS}.JointKeyGen\langle P_r(\ ), P_h(\ )\rangle$, and all pairs of messages (m_{HR}, m_{SH}), the following holds: $\Pr[\Pi^S.Verify(pk_2, m_{HR}, Open(\tau, s_r)) = 1] = 1$ and $\Pr[\Pi^S.Verify(pk_1, m_{SH}, \sigma_{SH}) = 1] = 1$, where $(\perp, \{\tau\}) \leftarrow Promise\left\langle \begin{array}{c} P_h(sk^L, pk^L, sk_{h2}^0, pk_2, m_{HR}) \\ P_r(sk_r^0, pk^L, pk_2, m_{HR}) \end{array} \right\rangle$ and $(\{(\sigma_{SH}, s_r)\}, \{\sigma_{SH}\}) \leftarrow Solving\left\langle \begin{array}{c} P_s(sk_s^0, pk_1, \tau_r, pk^L, m_{SH}) \\ P_h(sk_{h1}^0, sk^L, pk_1, m_{SH}) \end{array} \right\rangle$.*

With respect to blindness, we can consider the following case: a attacker $\mathcal{A}$ playing the role of P_h attempts to distinguish between two sets of message pairs, $\{(m_{HR,0}, m_{SH,0}), (m_{HR,1}, m_{SH,1})\}$ and $\{(m_{HR,0}, m_{SH,1}), (m_{HR,1}, m_{SH,0})\}$. Obviously, if $\mathcal{A}$ can distinguish them with non-negligible probability, it implies the sender and corresponding receiver can be associated even when there are other senders and receivers

```
$\mathsf{ExpBld}^{\mathcal{A}}_{\Pi^{TBCS}}(n)$
$(pk^L, (m_{HR,0}, m_{SH,0}), (m_{HR,1}, m_{SH,1})) \leftarrow \mathcal{A}(1^n)$
$(\{sk^0_{s,0}, pk^0_{h1,0}, pk_{1,0}\}, \{sk^0_{h1,0}, pk^0_{s,0}, pk_{1,0}\}) \leftarrow$
        $\Pi^{2AS}.JointKeyGen\langle P_s(\ ), \mathcal{A}(\ )\rangle$
$(\{sk^0_{s,1}, pk^0_{h1,1}, pk_{1,1}\}, \{sk^0_{h1,1}, pk^0_{s,1}, pk_{1,1}\}) \leftarrow$
        $\Pi^{2AS}.JointKeyGen\langle P_s(\ ), \mathcal{A}(\ )\rangle$
$(\{sk^0_{r,0}, pk^0_{h2,0}, pk_{2,0}\}, \{sk^0_{h2,0}, pk^0_{r,0}, pk_{2,0}\}) \leftarrow$
        $\Pi^{2AS}.JointKeyGen\langle P_r(\ ), \mathcal{A}(\ )\rangle$
$(\{sk^0_{r,1}, pk^0_{h2,1}, pk_{2,1}\}, \{sk^0_{h2,1}, pk^0_{r,1}, pk_{2,1}\}) \leftarrow$
        $\Pi^{2AS}.JointKeyGen\langle P_r(\ ), \mathcal{A}(\ )\rangle$
$(\cdot, \{\tau_0\}) \leftarrow Promise\left\langle \begin{matrix} \mathcal{A} \\ P_r(sk^0_{r,0}, pk^L, pk_{2,0}, m_{HR,0}) \end{matrix} \right\rangle$
$(\cdot, \{\tau_1\}) \leftarrow Promise\left\langle \begin{matrix} \mathcal{A} \\ P_r(sk^0_{r,1}, pk^L, pk_{2,1}, m_{HR,1}) \end{matrix} \right\rangle$
$b \leftarrow_{\$} \{0,1\}$
$(\sigma^*_0, s^0_r) \leftarrow Solving\left\langle \begin{matrix} P_s(sk^0_{s,0}, pk_{1,0}, [\tau^{0\oplus b}_r]^{a}, pk^L, m_{SH,0}) \\ \mathcal{A} \end{matrix} \right\rangle$
$(\sigma^*_1, s^1_r) \leftarrow Solving\left\langle \begin{matrix} P_s(sk^0_{s,1}, pk_{1,1}, \tau^{1\oplus b}_r, pk^L, m_{SH,1}) \\ \mathcal{A} \end{matrix} \right\rangle$
if $(\sigma^*_0 = \perp) \vee (\sigma^*_1 = \perp) \vee (\tau_0 = \perp) \vee (\tau_1 = \perp)$
    $\sigma_0 := \sigma_1 := \perp$
else
    $\sigma_{0\oplus b} := Open(\tau_{0\oplus b}, s^0_r)$
    $\sigma_{1\oplus b} := Open(\tau_{1\oplus b}, s^1_r)$
$b' \leftarrow \mathcal{A}(\sigma_0, \sigma_1)$
return $(b = b')$
```

[a] τ^b_r is extracted from τ_b where $b \in \{0,1\}$.

Figure 13: Blindness experiment

performing Π^{TBCS} during the same phase. Thus, the definition of blindness can be given as below.

Definition 3 (Blindness). Π^{TBCS} *is blind if there exists a negligible function* $\mathsf{negl}(n)$ *such that for all* $n \in \mathbb{N}$ *and all PPT adversaries* $\mathcal{A}$*, the following holds:* $\Pr[\mathsf{ExpBld}^{\mathcal{A}}_{\Pi^{TBCS}}(n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n)$*, where* ExpBld *is defined in Fig. 13.*

With respect to unlockability, we impose more restrictions on the consequences of attacks launched by $\mathcal{A}$ playing the role of P_h. That is, in addition to the restriction described in the property of unlockability of a synchronization puzzle protocol, $\mathcal{A}$ is unable to produce a valid signature on any message except m_{SH} if promise protocol is completed normally and a valid signature on any message if promise protocol is not completed normally. The additional restrictions prevent $\mathcal{A}$ from independently forging signatures which should be generated through the cooperation of $\mathcal{A}$ and P_s. Thus, the definition of unlockability can be presented as below.

Definition 4 (Unlockability). Π^{TBCS} *is unlockable if there exists a negligible function* $\mathsf{negl}(n)$ *such that for all* $n \in \mathbb{N}$ *and all PPT adversaries* $\mathcal{A}$*, the following holds:* $\Pr[\mathsf{ExpUnlock}^{\mathcal{A}}_{\Pi^{TBCS}}(n) = 1] \leq \mathsf{negl}(n)$*, where* $\mathsf{ExpUnlock}$ *is defined in Fig. 14.*

With respect to unforgeability, there are two requirements on $\mathcal{A}$ acting as both P_s and P_r. The first requirement is an extension of unforgeability of a synchronization puzzle protocol. That is, $\mathcal{A}$ cannot get d signatures from the interaction with P_h in promise protocol and release only $d-1$ signatures for P_h in solving phase. The second requirement protects $\mathcal{A}$ from forging a joint signature on a message

```
$\mathsf{ExpUnlock}^{\mathcal{A}}_{\Pi^{TBCS}}(n)$
$(pk^L, m_{HR}, m_{SH}) \leftarrow \mathcal{A}(1^n)$
$(\{sk^0_s, pk^0_{h1}, pk_1\}, \{sk^0_{h1}, pk^0_s, pk_1\}) \leftarrow$
        $\Pi^{2AS}.JointKeyGen\langle P_s(\ ), \mathcal{A}(\ )\rangle$
$(\{sk^0_r, pk^0_{h2}, pk_2\}, \{sk^0_{h2}, pk^0_r, pk_2\}) \leftarrow$
        $\Pi^{2AS}.JointKeyGen\langle P_r(\ ), \mathcal{A}(\ )\rangle$
$(\cdot, \{\tau\}) \leftarrow Promise\left\langle \begin{matrix} \mathcal{A} \\ P_r(sk^0_r, pk^L, pk_2, m_{HR}) \end{matrix} \right\rangle$
if $\tau = \perp$
    $(m, \sigma) \leftarrow \mathcal{A}$
    $b_0 := (\Pi^S.Verify(pk_1, m, \sigma) = 1)$
else
    $(\sigma^*, s_r) \leftarrow Solving\left\langle \begin{matrix} P_s(sk^0_s, pk_1, \tau_r, pk^L, m_{SH}) \\ \mathcal{A} \end{matrix} \right\rangle$
    $(m, \sigma) \leftarrow \mathcal{A}$
    $b_1 := (\Pi^S.Verify(pk_1, m, \sigma) = 1) \wedge (m \neq m_{SH})$
    $b_2 := (\Pi^S.Verify(pk_1, m_{SH}, \sigma^*) = 1)$
    $b_3 := (\Pi^S.Verify(pk_2, m_{HR}, Open(\tau, s_r)) \neq 1)$
return $b_0 \vee b_1 \vee (b_2 \wedge b_3)$
```

Figure 14: Unlockability experiment

```
$\mathsf{ExpUnforg}^{\mathcal{A}}_{\Pi^{TBCS}}(n)$
$\mathcal{P} := \emptyset, D := 0$
$(sk^L, pk^L) \leftarrow \Pi^{LOE}.KeyGen(1^n)$
$(pk_{2,1}, m_1, \sigma_1), ..., (pk_{2,d}, m_d, \sigma_d) \leftarrow \mathcal{A}^{\mathcal{O}_p(\cdot), \mathcal{O}_s(\cdot)}(pk^L)$
$b_0 := \exists i \in [d]\ s.t.\ (pk_{2,i}, \cdot) \in \mathcal{P} \wedge (pk_{2,i}, m_i) \notin \mathcal{P}$
        $\wedge\ \Pi^S.Verify(pk_{2,i}, m_i, \sigma_i) = 1$
$b_1 := \forall i \in [d]\ s.t.\ (pk_{2,i}, m_i) \in \mathcal{P}$
        $\wedge\ \Pi^S.Verify(pk_{2,i}, m_i, \sigma_i) = 1$
$b_2 := \bigwedge_{i,j \in [d], i \neq j} (pk_{2,i}, m_i, \sigma_i) \neq (pk_{2,j}, m_j, \sigma_j)$
$b_3 := (D \leq d - 1)$
return $b_0 \vee (b_1 \wedge b_2 \wedge b_3)$

$\underline{\mathcal{O}_p(m)}$
$(\{sk^0_r, pk^0_{h2}, pk_2\}, \{sk^0_{h2}, pk^0_r, pk_2\}) \leftarrow$
        $\Pi^{2AS}.JointKeyGen\langle \mathcal{A}(\ ), P_h(\ )\rangle$
$\mathcal{P} := \mathcal{P} \cup \{(pk_2, m)\}$
$(\cdot, \{\tau\}) \leftarrow Promise\left\langle \begin{matrix} P_h(sk^L, pk^L, sk^0_{h2}, pk_2, m) \\ \mathcal{A} \end{matrix} \right\rangle$

$\underline{\mathcal{O}_s(m')}$
$(\{sk^0_s, pk^0_{h1}, pk_1\}, \{sk^0_{h1}, pk^0_s, pk_1\}) \leftarrow$
        $\Pi^{2AS}.JointKeyGen\langle \mathcal{A}(\ ), P_h(\ )\rangle$
$\sigma^* \leftarrow Solving\left\langle \begin{matrix} \mathcal{A} \\ P_h(sk^0_{h1}, sk^L, pk_1, m') \end{matrix} \right\rangle$
if $\sigma^* \neq \perp$ then $D := D + 1$
```

Figure 15: Unforgeability experiment

which is not processed in promise protocol. To formalize the requirements, we allow $\mathcal{A}$ to access an oracle $\mathcal{O}_p$ which invokes promise protocol between $\mathcal{A}$ and P_h and an oracle $\mathcal{O}_s$ which invokes solving protocol between $\mathcal{A}$ and P_h. Thus, the definition of unforgeability can be proposed as below.

Definition 5 (Unforgeability). Π^{TBCS} *is unforgeable if there exists a negligible function* $\mathsf{negl}(n)$ *such that for all* $n \in \mathbb{N}$ *and all PPT adversaries* $\mathcal{A}$*, the following holds:* $\Pr[\mathsf{ExpUnforg}^{\mathcal{A}}_{\Pi^{TBCS}}(n) = 1] \leq \mathsf{negl}(n)$*, where* $\mathsf{ExpUnforg}$ *is defined in Fig. 15.*

TABLE 4: Performance Comparison between A^2L^+ and TA^2L for Exchanging Signatures m Times

	RT	**CC**	**SC**
A^2L^+ [23]	$0.542 + m \cdot 2.978$	$m \cdot 15.487$	3.449
TA^2L (ours)	$0.546 + m \cdot 5.858$	$m \cdot 31.925$	3.449

RT: running time (in seconds); CC: communication cost (in kilobytes); SC: storage cost (in kilobytes).

Appendix C. Performance Comparison between A^2L^+ and TA^2L

Using the same evaluation metrics and experimental settings as those in Section 6, we obtain performance results for A^2L^+ and TA^2L when using them to exchange signatures m times. The performance comparison is summarized in Table 4, leading to the following conclusions:

(1) RT and CC of TA^2L are approximately double of those of A^2L^+.

(2) SC of TA^2L is identical to that of A^2L^+.

Appendix D. Formal Proof of Theorem 1

Proof of Theorem 1. We establish proofs for blindness, unlockability, and unforgeability. For ease of notation, we denote the ensemble of executions for a protocol Π from the view of an adversary $\mathcal{A}$ as $\mathsf{EXEC}_{\Pi,\mathcal{A}}$. We use $\mathsf{EXEC}_{\Pi_1,\mathcal{A}} \approx \mathsf{EXEC}_{\Pi_2,\mathcal{A}}$ (resp, $\mathsf{EXEC}_{\Pi_1,\mathcal{A}} \equiv \mathsf{EXEC}_{\Pi_2,\mathcal{A}}$) to signify that $\mathsf{EXEC}_{\Pi_1,\mathcal{A}}$ and $\mathsf{EXEC}_{\Pi_2,\mathcal{A}}$ are computationally (resp, statistically) indistinguishable. The proofs follow the methodology of Theorem 4.9 in [23], with modifications in the reduction from hybrid experiment $\mathcal{H}_2$ to $\mathcal{H}_3$ regarding the setting of verification keys for the promise protocol oracle in demonstrating unforgeability. Specifically, the reduction simulates verification keys for the promise protocol oracle by forwarding messages between $\mathcal{A}$ and the challenger during some query of $\mathcal{A}$ to the solving protocol oracle in this paper. However, the reduction uses the verification key from the witness extractability game as the verification key in some query of $\mathcal{A}$ to the solving protocol oracle in [23].

Lemma 1 (Blindness). *Assuming Π^{NIZK} is sound, TA^2L is blind in the LOE model.*

Proof. We establish, through a series of hybrid experiments, that the cases where $b = 0$ and $b = 1$ are statistically indistinguishable.

Hybrid $\mathcal{H}_0$: Run ExpBld with $b = 0$.

Hybrid $\mathcal{H}_1$: In both runs of the solving protocol, set $Y'' := r \cdot g$, and $c'' \leftarrow \Pi^{LOE}.Enc(pk^L, r)$, where $r \leftarrow_{\$} \mathbb{Z}_p$.

Hybrid $\mathcal{H}_2$: Compute Y'' and c'' using τ_r^1 in the first run of the solving protocol and τ_r^0 in the second run.

Hybrid $\mathcal{H}_3$: Run ExpBld with $b = 1$.

Claim 1. *For all PPT adversaries $\mathcal{A}$, $\mathsf{EXEC}_{\mathcal{H}_0,\mathcal{A}} \equiv \mathsf{EXEC}_{\mathcal{H}_1,\mathcal{A}}$.*

Proof. If the ciphertext provided by P_h is well-formed, Y'' is g raised to a uniform element, and c'' is an encryption of the same uniform element in both experiments. Any distinguishing advantage implies a violation of the soundness of Π^{NIZK}, making the executions statistically indistinguishable. □

Claim 2. *For all PPT adversaries $\mathcal{A}$, $\mathsf{EXEC}_{\mathcal{H}_1,\mathcal{A}} \equiv \mathsf{EXEC}_{\mathcal{H}_2,\mathcal{A}}$.*

Proof. This holds with the same logic as Claim 1. □

Claim 3. *For all PPT adversaries $\mathcal{A}$, $\mathsf{EXEC}_{\mathcal{H}_2,\mathcal{A}} \equiv \mathsf{EXEC}_{\mathcal{H}_3,\mathcal{A}}$.*

Proof. The change is only syntactical and the executions are identical. □

Hence, the cases of $b = 0$ and $b = 1$ are statistically indistinguishable. □

Lemma 2 (Unlockability). *Assuming the two-party signature unforgeability, two-party pre-signature adaptability, and two-party witness extractability of Π^{2AS}, TA^2L is unlockable.*

Proof. We just need to examine two cases as follows.

$b_0 \vee b_1$: In this scenario, $\mathcal{A}$ can produce a valid signature on a message without having seen any pre-signature on it. This occurrence is only possible with negligible probability, relying on two-party signature unforgeability.

$b_2 \wedge b_3$: We first consider a situation in which $\mathcal{A}$ outputs a valid signature σ^* on m_{SH} while simultaneously $s'' \leftarrow \Pi^{2AS}.Ext(\sigma^*, \tilde{\sigma}_{h1}^{trans}, Y'')$ is not a valid witness for Y''. Then we can give a reduction running $\mathcal{A}$ which breaks two-party witness extractability with non-negligible probability. The reduction begins by sampling a uniform element $r \leftarrow_{\$} \mathbb{Z}_p$. It sets $Y'' := r \cdot g$ and employs pk^L generated by $\mathcal{A}$ to compute $c'' \leftarrow \Pi^{LOE}.Enc(pk^L, r)$. During the solving phase, it sends Y'', c'', and the two-party witness extractability challenge $\tilde{\sigma}$ to $\mathcal{A}$ and responds to the challenger with σ received from $\mathcal{A}$. This simulation is perfectly indistinguishable from an honest run of the solving protocol. Consequently, $\Pi^{2AS}.Ext(\sigma, \tilde{\sigma}, Y'')$ is not a valid witness for Y'', which contradicts two-party witness extractability. Therefore, s'' is a valid witness for Y'' with all but negligible probability. Since $Y'' = r' \cdot r'' \cdot Y = r' \cdot r'' \cdot s \cdot g$, the only valid witness for Y'' is $r' \cdot r'' \cdot s$. Hence $s'' = r' \cdot r'' \cdot s$, which implies $s = (r' \cdot r'')^{-1} \cdot s''$ is a valid witness for Y. We conclude that $\sigma_{HR} \leftarrow \Pi^{2AS}.Adapt(s, \tilde{\sigma}_{HR})$ is a valid signature on m_{HR} with a probability of 1. Therefore, $\mathcal{A}$ succeeds in this case with negligible probability. □

Lemma 3 (Unforgeability). *Assuming the hardness of OMDL, the zero-knowledgeness of Π^{NIZK}, two-party signature unforgeability and two-party witness extractability of Π^{2AS}, TA^2L is unforgeable.*

Proof. We present a series of hybrid experiments to demonstrate their indistinguishability. Finally, we establish, through reduction to OM-CCA-A2L, the absence of any adversary with a non-negligible advantage against the final hybrid.

Hybrid $\mathcal{H}_0$: This is the normal game ExpUnforg shown as Fig. 15.

Hybrid $\mathcal{H}_1$: Simulate all proofs of Π^{NIZK}.

Hybrid $\mathcal{H}_2$: If $\exists i \in [d]$ such that $(pk_{2,i}, \cdot) \in \mathcal{P}$, $(pk_{2,i}, m_i) \notin \mathcal{P}$ and $\Pi^S.Verify(pk_{2,i}, m_i, \sigma_i) = 1$, return 0.

Hybrid $\mathcal{H}_3$: If $\exists i \in [d]$ such that $\Pi^S.Verify(pk_{2,i}, m_i, \sigma_i) = 1$ and $\Pi^{2AS}.Ext(\sigma_i, \tilde{\sigma}_i, Y_i) = \perp$, return 0.

Claim 4. *For all PPT adversaries $\mathcal{A}$,* $\mathsf{EXEC}_{\mathcal{H}_0,\mathcal{A}} \approx \mathsf{EXEC}_{\mathcal{H}_1,\mathcal{A}}$.

Proof. This holds by the zero-knowledgeness of Π^{NIZK}. □

Claim 5. *For all PPT adversaries $\mathcal{A}$,* $\mathsf{EXEC}_{\mathcal{H}_1,\mathcal{A}} \approx \mathsf{EXEC}_{\mathcal{H}_2,\mathcal{A}}$.

Proof. The hybrids differ only in the case where $\mathcal{A}$ return a valid signature on a message that is not part of the transcript. This happens only with negligible probability by two-party signature unforgeability. □

Claim 6. *For all PPT adversaries $\mathcal{A}$,* $\mathsf{EXEC}_{\mathcal{H}_2,\mathcal{A}} \approx \mathsf{EXEC}_{\mathcal{H}_3,\mathcal{A}}$.

Proof. Any distinguishing advantage equals the occurrence probability of the event that $\mathcal{A}$ outputs some tuple $(pk_{2,i}, m_i, \sigma_i)$ such that, for the corresponding pre-signature $\tilde{\sigma}_i$, $\Pi^{2AS}.Ext(\sigma_i, \tilde{\sigma}_i, Y_i) = \perp$. We provide a reduction to the two-party witness extractability of Π^{2AS}. The reduction runs the setup as in $\mathcal{H}_3$ and picks some guess $i^* \leftarrow_{\$} [d-1]$ (where $d-1$ is the number of non-$\perp$ queries of $\mathcal{A}$ to the solving protocol oracle) for the distinguishing index. It starts $\mathcal{A}$ on pk^L and behaves the same way as $\mathcal{H}_3$ for all oracle queries, except for the i^*-th queries, in which it forwards messages between $\mathcal{A}$ and the challenger to simulate verification keys of the promise protocol oracle. In the execution of the promise protocol oracle, it sends m_{i^*} to its game and sends $\tilde{\sigma}$ received from the game to $\mathcal{A}$. Once $\mathcal{A}$ terminates and outputs $\{(pk_{2,i}, m_i, \sigma_i)\}_{i\in[d]}$, the reduction sends σ_{i^*} to its game. If it guesses the distinguishing index i^* correctly, σ_{i^*} is a winning signature. Suppose the distinguishing advantage is non-negligible. Since the guess is correct with probability $1/(d-1)$, the reduction violates two-party witness extractability also with non-negligible advantage, which is a contradiction. Hence, the two experiments must be computationally indistinguishable. □

We proceed with a reduction from hybrid $\mathcal{H}_3$ to OM-CCA-A2L. Assuming the existence of an adversary $\mathcal{A}$ with non-negligible success probability in $\mathcal{H}_3$, we construct a reduction that employs $\mathcal{A}$ to win the OM-CCA-A2L game. The reduction is provided with $\{(c_i, h_i)\}_{i\in[d+1]}$ and generates $(sk^L, pk^L) \leftarrow \Pi^{LOE}.KeyGen(1^n)$ as in $\mathcal{H}_3$, initiating $\mathcal{A}$ with the input pk^L. For $\mathcal{O}_p$ queries, the reduction follows the same steps as $\mathcal{H}_3$ but uses a distinct statement h_j each time it generates a pre-signature $\tilde{\sigma}_j$. Concerning $\mathcal{O}_s$ queries, the reduction takes the final signature σ^* as the output of the $\mathcal{O}^{A^2L}$ run on inputs $(pk_2, m', Y'', c'', \tilde{\sigma})$ from $\mathcal{A}$. As $\mathcal{O}_s$ and $\mathcal{O}^{A^2L}$ return $\perp$ in the exact same cases, and since $\mathcal{A}$ makes at most d non-$\perp$ queries to $\mathcal{O}_s$, the reduction also makes at most d non-$\perp$ queries to $\mathcal{O}^{A^2L}$.

Upon receiving $d+1$ tuples $(pk_{2,i}, m_i, \sigma_i)$ from $\mathcal{A}$, the reduction computes $r_j \leftarrow \Pi^{2AS}.Ext(\sigma_i, \tilde{\sigma}_j, Y_i)$ where $i, j \in [d+1]$. According to the definition of $\mathcal{H}_3$, the reduction can obtain $d+1$ non-$\perp$ r_j such that $h_j := g^{r_j}$ through at most $(d+1)^2$ invocations of $\Pi^{2AS}.Ext$ with non-negligible probability. This contradicts the OM-CCA-A2L security of Π^{LOE}. Thus, no adversary exists that can win the ExpUnforg game with non-negligible probability. □

Theorem 1 follows directly from Lemmas 1 to 3. □