AuditPCH: Auditable Payment Channel Hub with Privacy Protection

Yuxian Li, Jian Weng, Junzuo Lai, Yingjiu Li, Jianfei Sun, Jiahe Wu, Ming Li, Pengfei Wu, Robert H. Deng

***Abstract*—Anonymous Payment Channel Hub (PCH), one of the most promising layer-two solutions, settles the scalability issue in blockchain while guaranteeing the unlinkability of transacting parties. However, such developments bring conflicting requirements, i.e., hiding the sender-to-receiver relationships from any third party but opening the relationship to the auditor. Existing works do not support these requirements simultaneously since off-chain transactions are not recorded in the blockchain. Further, the privacy protection strategies hinder auditors from capturing the payment relationships. Thus, it is still a challenge to audit the finance activities of PCH transacting parties.**

This paper proposes a novel anonymous PCH solution called AuditPCH to achieve privacy and auditability. Concretely, we design a Linkable Randomizable Puzzle scheme for constructing conditional transactions, allowing a sender to pay for a receiver via the hub. As such, AuditPCH, with the new LRP scheme, ensures that (1) payment relationships can be protected from the hub and (2) an auditor with necessary trapdoors can associate the sender and receiver of a payment. We prove the security of AuditPCH under the Global Universal Composability framework. The extensive experimental evaluations on AuditPCH are established to demonstrate its functionality and flexibility.

***Index Terms*—Blockchain, Payment channel hub, Auditability.**

I. Introduction

PAYMENT Channel Hub (PCH) is a popular solution to facilitate blockchain off-chain transactions with the help of an intermediary hub (also known as a tumbler or mixing service) [1–5]. In a PCH, each user maintains a payment channel with the hub, and each sender-to-receiver payment is conducted as a relay payment with two hub-involved payments, including a sender-to-hub payment and a hub-to-receiver payment. Generally, the hub is operated by large trading institutions (e.g., PayPal) or banks that follow the regulation of auditors and simultaneously enforce the transacting parties to follow the payment protocol.

Since interacting with a hub will leak sender-to-receiver relationships to the hub, PCH schemes typically induce non-auditable privacy protection strategies to protect payment relationships. However, such privacy protection of payment relationships hinders auditors from auditing illegal financial activities (e.g., money launching and tax avoidance) [6]. Specifically, the off-chain payments with the support of auditing are essential since (1) it enables auditors to monitor the direction of every fund transfer in real-time. In security breaches such as the Mt.Gox incident, timely monitoring of fund transfers by auditors is critical for preventing further financial losses, and (2) it can help deter illegal financial activities as such activities are subject to scrutiny by auditors. In the real world, professional financial audit entities and governmental regulatory bodies may serve as auditors.

Many efforts [2, 6, 7] have been proposed to match privacy and audit requirements in blockchain systems. For the privacy requirements, the latest PCH proposals such as A^2L [2], A^2L^+ [8], BlindHub [9], and Tumblebit [7] leverage puzzle-promise protocols to hide the sender-to-receiver payment relationship. For the audit requirement, previous works [10–13] mainly focus on on-chain transactions. Elli *et. al* [12] and Gaby *et. al* [13] leverage zero-knowledge protocols and commitment schemes [14, 15] to hide the transaction details while allowing auditors to perform fine-grained audits of transactions. Prcash [11] and zkLedger [10] construct auditable blockchain systems which support an auditing interface and privacy protection in a new blockchain system using non-interactive zero-knowledge proofs [15]. *However, the existing on-chain solutions are difficult to extend to sender-to-receiver payment audits directly since the hub-involved payments in PCHs are not recorded in public ledgers.* In addition, the auditability of sender-to-receiver payments has received minimal attention to date. Further, due to the apparent conflict between the privacy concerns of PCHs and the requirements for auditability, the privacy protection solutions [3, 8] in PCH seem incompatible with the payment audit. It thus calls for new solutions to handle auditability and privacy challenges in PCH simultaneously.

Since the PCH scheme named A^2L^+ [8] has been compatible with most underlying blockchains, we start from it to construct an auditable and privacy-protection PCH scheme. The core idea of A^2L^+ is that via conditional hub-involved payments, and the hub pays a receiver *iff* a sender helps the receiver to solve a cryptographic challenge so-called *puzzle*, implying that the sender has sent necessary assets to the hub. Specifically, the *Randomizable Puzzle* (RP) scheme used in A^2L^+ enables users to encode a solution into a puzzle and randomize the puzzle later. A^2L^+ hides sender-to-receiver relationships due to the unlinkability between puzzles and randomizable puzzles used in conditional hub-involved payments. Nevertheless, realizing auditability and privacy in A^2L^+ is still a problem to be resolved.

Yuxian Li, Jianfei Sun, Robert H. Deng (Fellow, IEEE), and Pengfei Wu are with the School of Computing and Information Systems, Singapore Management University, Singapore 188065. Jianfei Sun is the corresponding author. (email: {yuxianli, jfsun, pfwu, robertdeng}@smu.edu.sg).
Jian Weng are with College of Cyber Security, Jinan University, Guangzhou 510632, Machine Learning and Cyber Security Interdiscipline Research Engineering Center of Jiangsu Province, Soochow University, Suzhou 215021, China (email: cryptjweng@gmail.com).
Junzuo Lai, Jiahe Wu, Ming Li are with College of Cyber Security, Jinan University, Guangzhou 510632, China. (email: {laijunzuo, jiahewu001, limjnu}@gmail.com).
Yingjiu Li is with Computer Science Department, University of Oregon, OR 97403, USA. (email: yingjiul@uoregon.edu).

Challenges. Realizing this goal hinges on settling two crucial challenges. The first *challenge* is to enable auditors to capture sender-to-receiver relationships while safeguarding the confidentiality of the association among senders and receivers from the hub. This challenge is primarily due to the unlinkability of puzzles, which hinders auditors from associating the sender and receiver of each relay payment. Besides, since off-chain hub-involved payments, which occur outside the blockchain but involve a hub for processing, are not submitted to blockchains, the immutability of blockchains cannot guarantee transaction integrity. Verifying the integrity of off-chain hub-involved payments, including the comprehensive inclusion and accurate ordering of all transactions within the audit scope, presents another significant *challenge*.

Solutions. To address these challenges, we present AuditPCH with privacy and auditability with the following explanations. Specifically, *(a) to resolve the contradiction between privacy and auditability*, a Linkable Randomizable Puzzle (LRP) scheme is designed to substitute the RP scheme in A^2L^+. To maintain privacy while supporting puzzle linking, we retain the puzzle randomization feature and ensure that only users with a trapdoor can link puzzles by encrypting the randomness. Additionally, to ensure the correctness of linkability, malleable proofs [16] are employed to generate the necessary puzzle correctness proofs during the randomization process, as traditional NIZK schemes are not feasible without access to the initial solution. AuditPCH, with the new LRP scheme, enables the auditor to link randomized puzzles to initial puzzles, thereby associating the sender and receiver of relay payments through mappings between puzzles and conditional hub-involved payments; *(b) to verify the integrity of hub-involved payments in each payment channel,* AuditPCH enforces transacting parties and hubs to append a hash of the latest payment to a newly generated hub-involved payment. Once the latest hub-involved payment in an audited payment channel is submitted to the blockchain, the hash chain will be immutable, and the auditor can verify the integrity of hub-involved payments of the audited payment channel via the hash chain. Our work makes the following contributions:

- **Linkable randomizable puzzle for conditional transactions.** We design the LRP scheme to construct conditional hub-involved payments, allowing users who own a trapdoor to associate initial puzzles with randomized puzzles. Our LRP restricts the visibility of the sender-to-receiver payment relationship to only the auditor by associating the used puzzles, which greatly mitigates the contradiction between payment relationship privacy and auditablity.
- **A novel auditable solution for PCH.** We propose a PCH solution called AuditPCH, achieving auditability and privacy by constructing a hash chain of off-chain transactions and utilizing the proposed LRP scheme. It innovatively allows the auditor to identify the integrity of off-chain payments and enforce payment audits on the closed channel.
- **Formal security proof.** We provide an ideal functionality for the auditable PCH and prove that AuditPCH is secure in the Global Universal Composability (GUC) framework [17].
- **Extensive experiments and evaluation.** We evaluate the effectiveness of AuditPCH by measuring its communication and computation costs. The evaluation results show that (1) the communication cost of each party to generate a sender-to-receiver payment is less than 8.6s, (2) the communication cost, except for the audit operation, is less than 26.5KB, and (3) the computation and communication cost of the auditor is linear with the number of sender-to-receiver payments in the audited channel.

II. Preliminaries

A. Cryptographic Building Blocks

We now introduce the building blocks used in our solution. Here, we denote λ as the security parameter and write $out \leftarrow \mathsf{A}(in)$ as an algorithm A with an input in and an output out.

Public Key Encryption. The encryption scheme includes the algorithms (Gen, Enc, Dec) [18]. $(pk, sk) \leftarrow \mathsf{Gen}(\lambda)$ is the key generation algorithm to produce a key pair (sk, pk), where sk is a selected value. $c \leftarrow \mathsf{Enc}(pk, m)$ is the encryption algorithm to encrypt a message m with the public key pk as a ciphertext c. $m' \leftarrow \mathsf{Dec}(c, sk)$ decrypts the ciphertext c as the plaintext m' via the private key sk. We utilize the ElGamal encryption scheme Π_{El} [18] and the Castagnos-Laguillaumie (CL) encryption scheme Π_{CL} which satisfy the Indistinguishability under Chosen Plaintext Attacks (IND-CPA).

Commitment scheme. Our construction requires a commitment scheme that allows users to commit a message (e.g., token identification) and verify its correctness. A commitment scheme Π_{com} is composed by three algorithms ($\mathsf{CMSetup}$, Com, $\mathsf{CMVerify}$). $pp \leftarrow \mathsf{CMSetup}(\lambda)$ inputs λ and generates the public parameter pp. $(com, r) \leftarrow \mathsf{Com}(pp, m, r)$ commits the message m in the commitment com with the randomness coin r. $\{0, 1\} \leftarrow \mathsf{CMVerify}(com, r, m)$ verifies if the message m is committed in com. Our solution introduces the Pedersen commitment scheme [14], satisfying the information-theoretically hiding and computationally binding properties.

Malleable proof scheme. The malleable proof schemes [16, 19] support users to transform their receiving proofs into new proofs against the converted witness (e.g., a randomized solution of a randomized puzzle) and statement. Let $R(x, w)$ be a relation related to the language $L := \{x \mid \exists w$ such that $(x, w) \in R\}$, where x is a statement and w is a witness of the statement. Two transformation functions ($w' = \mathcal{T}_{wit}(w)$, $x' = \mathcal{T}_{stmt}(x)$) are defined to restrict the allowed transformation of users. The malleable proof scheme is formulated as: Π_{MP} = ($\mathsf{CRSSetup}$, Vry, Prove, ZKEval). $crs \leftarrow \mathsf{CRSSetup}(\lambda)$ generates the Common Reference Strings (CRS) crs. $\pi \leftarrow \mathsf{Prove}(w, crs, x)$ is the prover algorithm with the witness w, the CRS crs, and the statement x as inputs to produce a proof π stating $(x, w) \in R$. $\{0, 1\} \leftarrow \mathsf{Vry}(\pi, crs, x)$ is the verifier algorithm to check whether the existence of w and x satisfies the relation R. $\pi' \leftarrow \mathsf{ZKEval}(crs, x, \{\mathcal{T}_{wit}, \mathcal{T}_{stmt}\}, \pi)$ produces a transformed proof π' for stating $(x', w') \in R$, where x' and w' come from the transformation functions $\{\mathcal{T}_{wit}, \mathcal{T}_{stmt}\}$. Note that the transformed proof π' still can be verified by the algorithm Vry. Here, a malleable proof scheme [16] is initialized by the Groth-Sahai proof scheme [19] and satisfies the Witness Indistinguishability (WI) property.

Adaptor signature scheme. Different from traditional signature schemes, the adaptor signature scheme Π_{ad}, which consists

of five algorithms (SKeyGen, PSign, Adapt, PVrfy, Ext), enables signers to give a *pre-signature* concerning the revelation of a secret value. We define a statement $Y = g^y$, where g is the generator of the group. $(pk, sk) \leftarrow \mathsf{SKeyGen}(\lambda)$ initializes a key pair (pk, sk) for signing. $\widetilde{\sigma} \leftarrow \mathsf{PSign}(m, Y, sk)$ inputs the secret key sk, a message m, and a statement Y to generate a pre-signature $\widetilde{\sigma}$. $\{0,1\} \leftarrow \mathsf{PVrfy}(pk, m, \widetilde{\sigma}, Y)$ checks the validity of the pre-signature $\widetilde{\sigma}$. $\sigma \leftarrow \mathsf{Adapt}(y, \widetilde{\sigma})$ takes the witness y and the pre-signature $\widetilde{\sigma}$ as inputs to produce a valid signature σ. $y \leftarrow \mathsf{Ext}(\widetilde{\sigma}, \sigma, Y)$ computes the witness y via the inputs $\widetilde{\sigma}$ and σ. The adaptor signature scheme satisfies pre-signature adaptability, which guarantees parties collect a valid signature from a valid pre-signature, and witness extractability, which ensures parties extract a valid witness via a valid signature and its pre-signature. Here, the scheme is formalized in [20, 21] and matches the security property against Existential Unforgeability under Chosen Message Attack (EUF-CMA) [18].

B. The High-level Design of A^2L^+

The A^2L^+ protocol [8] ensures secure and private off-chain payments through its puzzle promise and solver mechanisms. Specifically, it leverages randomizable puzzles and adaptor signatures to provide unlinkability of payment relationships.

Puzzle-promise phase. During this phase, the receiver P_r starts by sending a valid signature σ_r' on a transaction message m' to the hub P_h. The hub generates a statement/witness pair (A, α) and creates a randomizable puzzle Z along with a zero-knowledge proof π_α [15, 22] that proves α is a valid solution to Z. The hub then produces an adaptor signature $\hat{\sigma}_h'$ over the transaction m' using α and shares both the puzzle and the adaptor signature with P_r. The receiver pre-verifies the signature and randomizes the puzzle Z to Z', which is then shared with the sender P_s, completing the puzzle promise protocol.

Puzzle-solver phase. Here, the sender further randomizes the puzzle Z' to Z'' and generates a pre-signature $\hat{\sigma}_s$ on the transaction m' using Z''. This randomized puzzle and the pre-signature are sent to the hub, which then solves the puzzle Z'' using the trapdoor information to obtain α''. The hub uses α'' to adapt $\hat{\sigma}_s$ into a valid signature σ_s and signs the transaction m' with its secret key, producing σ_h. After verifying the signature σ_s, the hub publishes both σ_s and σ_h. Finally, the secret α'' is extracted and shared with the receiver, allowing them to finalize the transaction by revealing the secret α.

After that, the hub receives a valid transaction transferring assets from the sender, and simultaneously, the receiver obtains a valid transaction transferring the assets from the hub.

III. Models, Goals, and Definition

A. Security Model

In this section, we describe the security model of the AuditPCH protocol. We first briefly describe the security model and then formalize the security of AuditPCH in the Global Universal Composability (GUC) framework [17]. Additionally, we summarize the used notations in Table I.

Generally, both hub and users can be malicious in a PCH [2, 23, 24]. That is, the hub might steal money from users [2, 25, 26] or associate the sender with the receiver for any relay payment [2, 7, 23, 24]. The auditor plays a pivotal role in the PCH system, entrusted with auditing any payment channel and verifying sender-to-receiver relationships from hub-involved payments. Securing off-chain transactions against a malicious hub is a complex task, necessitating the achievement of atomicity and unlinkability in PCHs, as identified in previous studies. Atomicity protects users' tokens. It ensures that for each relay payment, either both hub-involved payments of the relay payment are completed or none of them is completed, thereby safeguarding the users' assets. Unlinkability ensures that any sender-to-receiver relationship is hidden from the hub, providing users with high privacy and confidentiality. Besides atomicity and unlinkability, we further identify the necessity of achieving auditability in PCHs, which ensures that an auditor can determine whether any pair of hub-involved payments belong to the same relay payment.

Attack model. Transacting parties are modeled as Interactive Turing Machines (ITMs) to interact with an ideal functionality $\mathcal{F}$ via secure and authenticated channels. The adversary $\mathcal{A}$ is a Probabilistic Polynomial-Time (PPT) machine that can corrupt any transacting party and a hub via the given interfaces. Assume that the adversary $\mathcal{A}$ adopts *static corruption*, meaning that the inputs and outputs of corrupted parties can be controlled by $\mathcal{A}$, and the corrupted parties are chosen before protocol execution.

Communication model. Communication among transacting parties and a hub in the system will be conducted via a synchronous communication network, and thus, our protocol proceeds in discrete rounds. The parties are aware of the present rounds, and the messages of the i-th round are sent at the beginning of the (i+1)-th round. Here, we follow the definition in [17, 27] to model the synchronous communication network as an ideal functionality $\mathcal{F}_{\mathsf{syn}}$. To prevent the adversary from reading or rewriting the messages, we further model the secure transmission as a functionality $\mathcal{F}_{\mathsf{smt}}$ [27]. We refer the reader to [17, 27] to see more details of the functionalities $\mathcal{F}_{\mathsf{syn}}$ and $\mathcal{F}_{\mathsf{smt}}$.

Payment channel hub system assumption. Each transacting party pair can open one channel identified by a unique identification $< p_i, p_j >$. When creating a new channel, we assume each party has enough assets in their transactions. The auditor is trusted and takes charge of auditing the given channel. We use the definition in [28] to model the blockchain to give a global ledger ideal functionality $\mathcal{F}_L$. Especially, the ideal functionalities in the real world or other protocols in the hybrid world have access to $\mathcal{F}_L$ to read the validity of channels and update the state.

TABLE I: Notations.

Notation	Definition
p_r, p_h, p_s, p_o	Receiver, hub, sender, auditor in the protocol.
γ_1, γ_2	Payment channels.
τ_0, τ_1, τ_1	Hub-involved payments.
pid, pid'	Identifiers for a puzzle pair in payments.
η	A receiver-hub payment.
η'	A sender-hub payment.
$\mathbb{P}$	Set of public-key-puzzle pairs.
$\mathbb{T}$	Set of hub-involved payment pairs.
TxSet	Set of channels for auditing.

Operations. The operations of AuditPCH are modeled as

an ideal functionality $\mathcal{F}_{\mathsf{AuditPCH}}$ (the details shown in), where the sender p_s and the receiver p_r establish sender-to-receiver payments via the hub p_h with the help of pairs of puzzles. After the party p_i closes his channel, the ideal functionality $\mathcal{F}_{\mathsf{AuditPCH}}$ allows the auditor p_o to audit the party p_i via puzzle pairs used in the sender-to-receiver payments happened in this channel. Concretely, $\mathcal{F}_{\mathsf{AuditPCH}}$ defines five operations: setup, open, pay, close, and audit. The auditor p_o can execute the setup operation only once to generate the trapdoor and the corresponding public key pk_0.

The open operation aims to create a payment channel γ through a commitment transaction $txid_p$. By executing the open operation, p_s and p_r respectively create a new payment channel with p_h. The pay operation enables the party p_s to make a sender-to-receiver payment to p_r via p_h. In detail, the party p_s first makes a registration claiming that he owns enough assets for one payment and then sends the registration oid to the receiver p_r. After that, a temporary hub-involved payment τ_0 that sends ϵ coins from p_s to p_h is generated. Waiting for the party p_r to send a registration oid' he received before, the operation checks the existence of the registration. If the registration oid' exists, an average-sampled puzzles pair (pid, pid') is generated to p_s and p_r so that they can update the sender-to-receiver payment to the channels γ_1 and γ_2. When p_s and p_r provide the valid puzzle pair, the payment τ_0 for p_s and p_h as well as another payment for updating the channel between p_r and p_h will respectively update to the channel γ_1 and γ_2. After that a record $(pk_0, pid, pid', (\gamma_2, \eta), (\gamma_1, \eta'))$ will be stored for the audit operation. The close operation closes the aimed channel and notifies the channel users.

The audit operation audits the sender-to-receiver payments of the party p_i. Through this operation, the auditor p_o can capture the sender-to-receiver payment relationships in the channel of the audited party p_i in TxSet provided by p_h and check if p_h provides a complete hub-involved payment set. Specifically, this operation first checks if the audited channel is closed. If yes, it interacts with the simulator to check if each channel in TxSet is including complete off-chain payment. That is, for the audited channel γ_i, the operation will check if the hash chain of the payments in a channel is complete. Note that the order of the hub-involved payments will be revealed by the order of the hub-involved payments in their hash chain. Further, the operation checks whether each payment in the closed channel can be linked to another payment provided by p_h. Then, the operation captures the payment relationship recorded before and adds the sender-to-receiver payment to the set $\mathbb{A}$. Finally, the operation will output the set $\mathbb{A}$ and the flag flag which flag=1 that represents γ_i in TxSet is complete and flag=0 represents that γ_i in TxSet is incomplete or there are some payments in γ_i can not be linked to another payment.

B. Security and Privacy Goals

Next, we demonstrate how the ideal functionality $\mathcal{F}_{\mathsf{AuditPCH}}$ achieves atomicity, unlinkability, and auditability.

Atomicity. This property guarantees that honest parties (e.g., p_r, p_h, and p_s) do not lose their assets even though some are corrupted. The ideal functionality $\mathcal{F}_{\mathsf{AuditPCH}}$ achieves atomicity if transacting parties call the pay operation with two channels (γ_1, γ_2) to transfer assets, and finally the operation simultaneously updates or does not update the channels (γ_1, γ_2). That is, a successful sender-to-receiver payment only happens when the following conditions hold: (1) the sender owns a valid token identification oid to claim that he/she has enough assets in the channel for a new payment and (2) the new hub-involved payments for establishing a sender-to-receiver payment will be updated to γ_1 and γ_2 simultaneously. The above conditions are enforced by $\mathcal{F}_{\mathsf{AuditPCH}}$ since a token identification is only provided to a sender owning assets for at least one payment. $\mathcal{F}_{\mathsf{AuditPCH}}$ will terminate if the receiver cannot provide an unspent token identification of a sender. It means payment must be stopped if the sender does not have enough assets. Through tracing the state of the generated puzzle pair (pid, pid') in the list $\mathbb{P}$, $\mathcal{F}_{\mathsf{AuditPCH}}$ will terminate when receiving a puzzle that is excluded in $\mathbb{P}$. The channels (γ_1, γ_2) will be updated *iff* the received puzzles are included in $\mathbb{P}$ and have been solved by a corresponding execution of puzzle solver at the previous round; Otherwise, none of the channels will be updated.

Unlinkability. It guarantees that the corrupted hub p_h cannot associate the sender p_s and the receiver p_r of a sender-to-receiver payment. More specifically, given any triple of hub-involved payments (τ_0, τ_1, τ_2), where τ_1 and τ_2 are executed in the same round and the same payment direction (either to p_h or from p_h), and τ_0 is selected from, either the relay payment involving τ_1 or the relay payment involving τ_2 with the same probability $1/2$. The unlinkability requires that the probability, for the corrupted party p_h to output the correct relay payment between $\{\tau_0, \tau_1\}$ and $\{\tau_0, \tau_2\}$, is no higher than $1/2+\epsilon(\lambda)$, where $\epsilon(\lambda)$ is negligible in terms of a security parameter λ. In the ideal functionality $\mathcal{F}_{\mathsf{AuditPCH}}$, the linkability may be revealed via the correlation between pid and pid'. Nevertheless, p_h cannot output the correct puzzle pair (pid, pid') among the puzzles used in multiple relay payments because pid and pid' are provided to p_h in separate rounds while the relay payments occur concurrently. Further, p_h intends to use network information to connect p_s and p_r in a relay payment involving τ_0, such that p_h can output the correct relay payment $\{\tau_0, \tau_{1/2}\}$. However, under the assumption that p_s and p_r make the communication via a secure and anonymous network, p_h cannot use the network information to establish a link between p_s and p_r. This is a common assumption adopted in the PCH-related works [2, 23]. It is emphasized that only p_o can be allowed to obtain the hub-involved payment relationships in the audit operation. The corrupted hub p_h cannot leverage the audit operation to output the correct hub-involved payment pairs.

Auditability. This property ensures that given a closed channel γ_i with the complete hub-involved payment hash-chain, the auditor can link the hub-involved payments in γ_i to another hub-involved payment generated by one sender-to-receiver payment if another hub-involved payment is included in TxSet. Note that γ_i is also provided by p_h and is included in TxSet. First, the payment hash chain of γ_i is committed in the last payment submitted to the blockchain. If the payment hash chain of γ_i includes all hub-involved payments from the first payment (i.e. the payment for opening the channel) to the last payment (i.e. the payment for closing the channel), the functionality considers the

given channel is complete; Otherwise, it outputs 0 for notifying p_o that the given channel is incomplete. Second, if the above check succeeds, the functionality will retrieve payment pairs related to γ_i from the set $\mathbb{T}$ that stores hub-involved payment pairs. Then, $\mathcal{F}_{\mathsf{AuditPCH}}$ adds all the payment pairs including a payment from γ_i and another payment from TxSet to the set $\mathbb{A}$ and outputs the set $\mathbb{A}$ to show the sender-to-receiver payment relationship of p_i to the auditor p_o. Note that the ideal functionality we formulate only enables p_o to output all possible hub-involved payments pair of the aimed channel and enforce p_h submits all hub-involved payments committed in the payment hash-chain of the aimed channel but cannot prevent the hub from providing incomplete channels set.

Remark. The A^{2}L$^+$ protocol addresses two theoretical attacks: the key recovery attack and the one-more signature attack, which may not be practical or extant in real-world applications as explained in [8]. Nonetheless, A^{2}L$^+$ introduces several enhancements over the original A^{2}L protocol to improve security and restrict adversaries' capabilities, thereby providing stronger security assurances in practical implementations and ensuring robust security proofs for the revised protocol. To facilitate clearer security proof in our approach, limiting the adversary's capabilities by assuming that the hub cannot act as a decryption oracle is essential. This assumption enhances the protocol's security because it prevents adversaries from using the hub to decrypt information, thus addressing the key vulnerability issue in previous versions.

C. GUC-Security Definition

Suppose that a protocol Π exists in the real world and an ideal functionality $\mathcal{F}$ exists in the ideal world. We denote an environment $\mathcal{E}$ that distinguishes the protocol execution in the real world from the ideal functionality execution in the ideal world. In the *real world*, the protocol Π is executed by parties $\mathcal{P}$=$\{p_1, \ldots, p_n\}$. The protocol Π has access to ideal functionalities when Π is in the hybrid world. The adversary in the protocol Π is denoted as $\mathcal{A}$, capturing the outputs and inputs of corrupted parties. In the *ideal world*, the parties $\mathcal{P}$ interact with the interface of the ideal functionality $\mathcal{F}$, which can be an abstract specification achieving the security properties that the protocol Π desires to satisfy. An ideal adversary *Sim* (i.e., the simulator) performs the attack on the ideal functionality via its interface. The protocol GUC-realizes the ideal functionality if the environment cannot distinguish the execution of the protocol Π in $(\mathcal{F}_{\mathsf{syn}}, \mathcal{F}_{\mathsf{smt}}, \mathcal{F}_L)$-hybrid world from the execution of the ideal functionality $\mathcal{F}$ in the ideal world.

In detail, the ensemble of the protocol execution can be denoted as $\mathsf{EXEC}_{\Pi_{\mathsf{AuditPCH}}, Sim, \mathcal{E}}$. Besides, $\mathsf{EXEC}_{\mathcal{F}_{\mathsf{AuditPCH}}, \mathcal{A}, \mathcal{E}}$ refers to the ensemble of the ideal functionality execution. Based on the aforementioned descriptions, the protocol Π_{AuditPCH} GUC-realizing the ideal functionality $\mathcal{F}_{\mathsf{AuditPCH}}$ is defined as:

Definition III.1 (GUC-security)**.** *The protocol Π_{AuditPCH} GUC-realizes the functionality $\mathcal{F}_{\mathsf{AuditPCH}}$ with respect to a global ledger ideal functionality $\mathcal{F}_L$ in the following condition: for any PPT adversary $\mathcal{A}$, there is a simulator Sim so that no environment $\mathcal{E}$ can distinguish $\mathsf{EXEC}_{\mathcal{F}_{\mathsf{AuditPCH}}, \mathcal{A}, \mathcal{E}}$ from $\mathsf{EXEC}_{\Pi_{\mathsf{AuditPCH}}, Sim, \mathcal{E}}$. The condition is denoted as:*

$$\mathsf{EXEC}^{\mathcal{F}_L}_{\mathcal{F}_{\mathsf{AuditPCH}}, Sim, \mathcal{E}} \approx \mathsf{EXEC}^{\mathcal{F}_L}_{\Pi_{\mathsf{AuditPCH}}, \mathcal{A}, \mathcal{E}}$$

IV. Linkable Randomizable Puzzle (LRP)

In this section, we introduce the construction of the LRP scheme and then give the security goals of the LRP scheme.

A. Construction

Compared with the RP scheme [8], the LRP scheme not only allows users to generate, solve, and randomize puzzles but also enables users to link the initial puzzle to the randomized puzzle with a trapdoor. Supporting unlinkability while maintaining the randomizing puzzle functionality is crucial for privacy. We introduce the attribute adversary-linkability in the LRP scheme, meaning only users with a trapdoor can see puzzle relationships. To achieve this requirement, the randomness in the RP scheme must be encrypted. Only an auditor with the decryption trapdoor can link the initial and randomized puzzles. Further, a puzzle correctness proof must be generated during randomization to ensure the correct relationship of puzzles. Since users cannot access the original puzzle's solution, traditional NIZK schemes [15] cannot be used. Instead, malleable proofs [16] can derive the proof for the randomized puzzle from the existing proof. The LRP scheme is denoted as seven algorithms Π_{LRP} = (Setup, KGen, PGen, PVerify, PRand, PSol, PLink).

The Linkable Randomizable Puzzle (LRP) Scheme

$(pp, sk_0) \leftarrow$ **Setup**(λ): is a PPT algorithm (used by the auditor) that on input security parameter λ, outputs public parameters $pp = (G, g, q, r_0, \Pi_{\mathsf{MP}}, crs, \Pi_{\mathsf{El}}, pk_0)$ and private key sk_0. The details of the outputs are as follows:
- G is an elliptic curve group of order q with generator g.
- r_0 is a random number in Z_q.
- Π_{MP} is a secure malleable non-interactive zero-knowledge proof scheme [16] under a common reference string crs.
- Π_{El} is the ElGamal encryption scheme [18] of which $\mathrm{Enc}(pk_0, \cdot)$ is for encryption and $\mathrm{Dec}(pk_0, sk_0, \cdot)$ is for decryption, and (pk_0, sk_0) is a key pair of Π_{El}.

$(pk_1, sk_1) \leftarrow$ **KGen**(λ) : is a PPT algorithm (used by the hub) that on input security parameter λ, outputs a key pair (pk_1, sk_1) from the Castagnos-Laguillaumie (CL) encryption scheme Π_{CL} [29], of which $\mathrm{Enc}(pk_1, \cdot)$ is for encryption and $\mathrm{Dec}(pk_1, sk_1, \cdot)$ is for decryption.

$(pz, \pi) \leftarrow$ **PGen**$(pp, pk_1, \aleph)$: is a PPT algorithm (used by the hub) that on input public parameters pp, public key pk_1, and a puzzle solution $\aleph \in Z_q$, outputs a puzzle $pz = (\alpha, \beta, \gamma)$ and associated proof π. The details of the outputs are as follows:
- $\alpha = \Pi_{\mathsf{El}} \cdot \mathrm{Enc}(pk_0, r_0)$.
- $\beta = \Pi_{\mathsf{CL}} \cdot \mathrm{Enc}(pk_1, \aleph)$.
- $\gamma = g^{\aleph}$.
- π is a malleable proof of existence of witness $(\aleph, r_0)$ for a statement $\alpha \wedge \beta \wedge \gamma * g^{r_0}$ using $\Pi_{\mathsf{MP}}{}^a$.

$0/1 \leftarrow$ **PVerify**(pp, pz, π): is a Deterministic Polynomial Time (DPT) algorithm that on input public parameters pp, a puzzle pz, and a proof π, outputs either 0 (failure) or 1 (success) for verifying pz. The process of generating the outputs is as follows:
- Parse $pz = (\alpha, \beta, \gamma)$.
- Check if proof π is correct for statement $\alpha \wedge \beta \wedge \gamma \wedge \gamma * g^{r_0}$ using scheme Π_{MP}.
- Return 1 if the proof is correct and 0 otherwise.

$\aleph \leftarrow$ **PSol**(pk_1, sk_1, pz): is a DPT algorithm (used by hub) that on input public key pk_1, private key sk_1, and a puzzle pz, outputs $\aleph = \Pi_{\mathsf{CL}}.\mathrm{Dec}(pk_1, sk_1, \beta)$, where pz is parsed as $pz = (\alpha, \beta, \gamma)$.

$(pz', \pi', r) \leftarrow$ **PRand**(pp, pk_1, pz, π): is a PPT algorithm (used by users) that on input public parameters pp, pubic key pk_1, a puzzle pz, and a malleable proof π, outputs a randomized puzzle pz', randomized proof π', and associated random number r. The process of generating the outputs is as follows:

- Parse $pz = (\alpha, \beta, \gamma)$.
- Sample a random number $r \in Z_q$.
- Randomize puzzle pz to $pz' = (\alpha', \beta', \gamma')$ using r such that $\alpha' = \Pi_{\mathsf{EI}} \cdot \mathrm{Enc}(pk_0, r_0 + r), \beta' = \Pi_{\mathsf{CL}} \cdot \mathrm{Enc}(pk_1, \aleph + r)$, and $\gamma' = g^{\aleph + r}$ (note that both Π_{CL} and Π_{EI} are homomorphic).
- Randomize π to π' using r to prove the existence of witness $(\aleph + r, r_0 + r)$ for statement $\alpha' \wedge \beta' \wedge \gamma' \wedge \gamma' * g^{r_0}$. Note that $T_{wit}((\aleph, r_0)) = (\aleph + r, r_0 + r)$ and $T_{stmt}(\alpha \wedge \beta \wedge \gamma \wedge \gamma * g^{r_0}) = \alpha' \wedge \beta' \wedge \gamma' \wedge \gamma' * g^{r_0}$
- Return $pz' = (\alpha', \beta', \gamma')$, π', and r.

$0/1 \leftarrow$ **PLink**(pp, sk_0, pz, pz'): is a DPT algorithm (used by the auditor) that on input public parameters pp, (auditor's) private key sk_0, and two puzzles pz and pz', outputs 1 (success) or 0 (failure). Respectively, pz and pz' are parsed as $pz = (\alpha, \beta, \gamma)$ and $pz' = (\alpha', \beta', \gamma')$. The process of generating the output is as follows:

- If $g^{\Pi_{\mathsf{EI}} \cdot \mathrm{Dec}(pk_0, sk_0, \alpha')} / g^{\Pi_{\mathsf{EI}} \cdot \mathrm{Dec}(pk_0, sk_0, \alpha)} = \gamma' / \gamma$, return 1.
- Otherwise, output 0.

[a] Including $\gamma * g^{r_0}$ in the statement is to bind $\aleph$ to r_0.

B. Property Statement

We require the LRP scheme to achieve the properties: *security*, *correctness*, and *adversary-unlinkability*.

(1) Security. It implies that the adversary cannot extract the solution of a puzzle with nonnegligible probability even though he/she owns the puzzle and the public parameter.

Definition IV.1 (Security)**.** *An LRP scheme is secure if the formula holds:*

$$\Pr\left[\aleph \leftarrow \mathcal{A}(pp, pk_1, pz, \pi) \,\middle|\, \begin{array}{l}(pp, sk_0) \leftarrow \mathsf{Setup}(\lambda), \\ (sk_1, pk_1) \leftarrow \mathsf{KGen}(\lambda), \\ (pz, \pi) \leftarrow \mathsf{PGen}(pp, pk_1, \aleph)\end{array}\right] \leq \mathsf{negl}(\lambda) \tag{1}$$

where negl *is a negligible function.*

(2) Correctness. It ensures the correct puzzles can be solved with the secret key sk_1 and the relationship between initial puzzles and randomized puzzles can be identified via the trapdoor sk_0.

Definition IV.2 (Correctness)**.** *An LRP scheme satisfies correctness for every* $\aleph$*, we have that*

$$\Pr[\mathsf{PSol}(pk_1, sk_1, pz) = \aleph \wedge \mathsf{PLink}(pp, sk_0, pz, pz') = 1] = 1 \tag{2}$$

where $(pz, \pi) \leftarrow \mathsf{PGen}(pp, pk_1, \aleph)$ *and* $(pz', \pi', r) \leftarrow \mathsf{PRand}(pp, pk_1, pz, \pi)$.

(3) Adversary-unlinkability. It states that the adversary without the trapdoor cannot reveal the linkability of the given puzzles with a non-negligible advantage.

Definition IV.3 (Adversary-unlinkability)**.** *If no PPT adversary can win the following game with a non-negligible advantage, the LRP scheme satisfies adversary-unlinkability. We define the advantage to be:* $\mathsf{ADV}^{\mathsf{PLink}}_{\mathcal{A},\mathsf{LRP}} = |\Pr[\mathsf{Link}_{\mathcal{A},\mathsf{LRP}}(\lambda)] - 1/2| \leq \mathsf{negl}(\lambda)$*, where* negl *is a negligible function.*

$\mathrm{Link}_{\mathcal{A},\mathrm{CP}}(\lambda)$

(1) $(pp, sk_0) \leftarrow \mathrm{Setup}(\lambda)$.
(2) $(sk_1, pk_1) \leftarrow \mathrm{KGen}(\lambda)$.
(3) $(pz_0, \pi_0) \leftarrow \mathrm{PGen}(pp, pk_1, \aleph_0)$.
(4) $(pz_1, \pi_1) \leftarrow \mathrm{PGen}(pp, pk_1, \aleph_1)$.
(5) $(pz'_0, \pi'_0, r_0) \leftarrow \mathrm{PRand}(pp, pk_1, pz_0, \pi_0)$.
(6) $(pz'_1, \pi'_1, r_1) \leftarrow \mathrm{PRand}(pp, pk_1, pz_1, \pi_1)$.
(7) $\mathrm{b}' \leftarrow \mathcal{A}(pp, pk_1, pz_0, \pi_0, pz_1, \pi_1, pz'_b, \pi'_b)$ *(where* $\mathrm{b} \leftarrow \{0, 1\}$*).*
(8) Return $\mathrm{b} = \mathrm{b}'$.

Theorem IV.1. *The LRP scheme satisfies security, correctness, and adversary-unlinkability if the encryption schemes* Π_{EI} *and* Π_{CL} *satisfy the indistinguishability under a chosen-plaintext attack and the malleable proof scheme* Π_{MP} *satisfies witness indistinguishability.*

V. Construction of AuditPCH

A. Overview

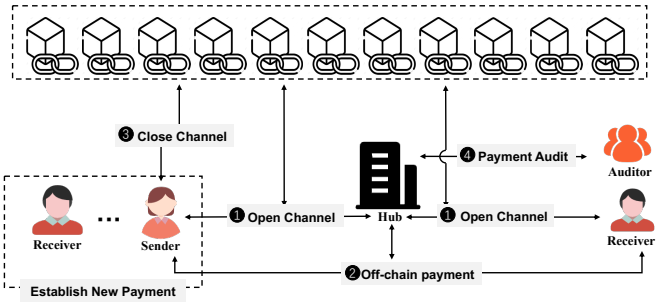

Fig. 1: Overview of AuditPCH.

Inspired by A^{2}L$^+$, we design an AuditPCH with the following phases: *open channel, off-chain payment, close payment*, and *payment audit*. Intuitively, Sender S and Receiver R first open a channel with Hub H, respectively, by locking assets in the shared account. Second, each pair of Sender-Receiver cooperates with the hub to establish an off-chain payment. A naive protocol of making a relay payment from S to R through H is that S makes an off-chain payment $\tau_{S \to H}$ to H first, requesting H to make the further off-chain payment $\tau_{H \to R}$ to R and thus complete the relay payment. This naive protocol, however, guarantees no atomicity since $\tau_{H \to R}$ may fail after $\tau_{S \to H}$ succeeds. It guarantees no unlinkability either since H can link $\tau_{S \to H}$ to $\tau_{H \to R}$ for sure. Our idea of achieving a relay payment from sender S to receiver R is partly inspired by TumbleBit [7] and A^{2}L$^+$ [8] which establishes a payment $\tau_{H \to R}$ from hub H to receiver R first but its success is conditioned on the success of a payment $\tau_{S \to H}$ from sender S to hub H. Third, after multiple off-chain payments, a party can close their channel to submit the latest balance of their channels to the blockchain. Finally, Auditor O can use a trapdoor to output the correct off-chain payment pair $(\tau_{S \to H}, \tau_{H \to R})$ and further audit the payment details of the certain closed channel. This phase enables O to authenticate the financial payments that transpired within the closed channel.

B. Concrete Design

Following TumbleBit [7] and A^{2}L$^+$ [8], we assume fixed payment amounts and constant transaction fees in PCHs.

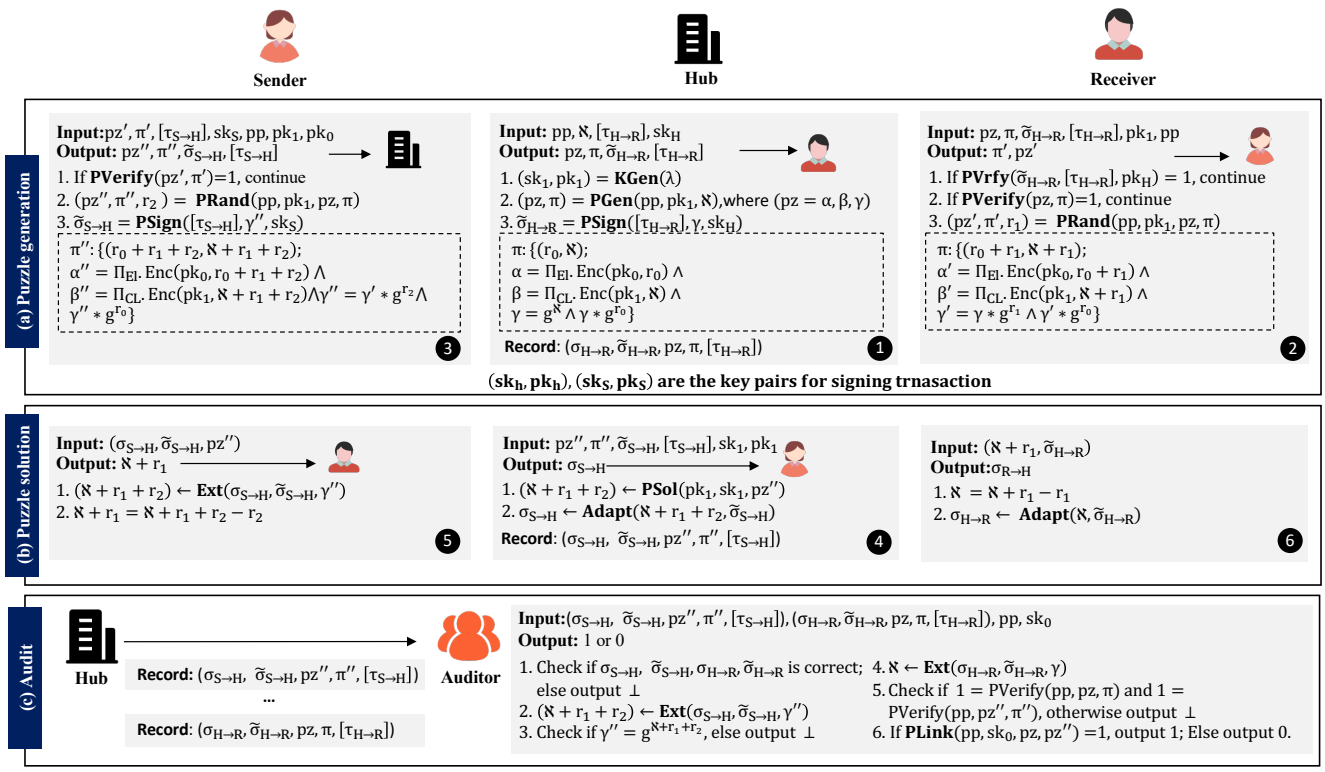

Fig. 2: The description of the pay phase in the AuditPCH protocol: (a) Puzzle generation, where puzzles are created to initiate sender-to-receiver payments; (b) Puzzle solution, which involves solving the puzzle to enable the effectiveness of off-chain payments; and (c) Audit, allowing authorized auditors to audit off-chain transactions.

Phase I: Open channel. At this stage, the sender S (receiver R) and a hub H create a commitment transaction to lock a certain amount of digital assets on the blockchain, establishing a payment channel. Additionally, an auditor O generates public parameters $pp = (G, g, q, r_0, \Pi_{\mathsf{MP}}, crs, \Pi_{\mathsf{EI}}, pk_0)$ and private key sk_0 using Π_{LRP}.Setup. A hub H generates its key pair (pk_1, sk_1) using Π_{LRP}.KGen and computes a zero-knowledge proof stating that $(pk_1, sk_1) \in \Pi_{\mathsf{LRP}}$.KGen. Following A^{2}L$^+$, all parties verify the proof to ensure the correctness of pk_1. A hub-involved off-chain transaction τ is available in a payment channel once it is signed by the parties who set up the payment channel using a blockchain-adopted signature scheme Σ (e.g., ECDSA), where each party has their secret signing key and public verification key bound to the payment channel. In the following, we will use $[\tau]$ to denote the body of off-chain transaction τ excluding its signatures.

Phase II: Off-chain payment. Each Sender-Receiver pair cooperates with the hub to establish hub-involved payments in this phase. In this work, the token registration mechanism used in A^{2}L$^+$ is employed, enabling a hub H to verify whether a receiver R interacts with a Sender S that possesses adequate assets for a one-time payment. Specifically, S samples a token identification oid and commits it into a commitment. Then, H signs in this commitment and returns the blinded signature σ^* generated by Π_{br} to S if S owns enough assets in the channel between S and H. Next, S opens and randomizes the blinded signature σ^*, sending σ and oid to R. Finally, the signature σ will be passed to H. If the validation is successful, H can act as an intermediary to enable hub-involved payments between the Sender-Receiver pair.

As shown in Figure 2(a), H generates a pre-signature $\tilde{\sigma}_{H\to R}$ on a payment $[\tau_{H\to R}]$ using an adaptor signature scheme Π_{ad} [30] with respect to a blockchain-adopted signature scheme Σ and a witness-statement pair $(\aleph, g^{\aleph})$, where witness $\aleph$ is chosen randomly from Z_q. Then, H sends $[\tau_{H\to R}]$ and $\tilde{\sigma}_{H\to R}$ to R. Since $\tilde{\sigma}_{H\to R}$ is not a full valid signature from H, R cannot commit $[\tau_{H\to R}]$ by simply adding its signature unless R can adapt the pre-signature $\tilde{\sigma}_{H\to R}$ to a full valid signature $\sigma_{H\to R}$. To enable R to do so later, H calls $(pz, \pi) \leftarrow \Pi_{\mathsf{LRP}}.\mathrm{PGen}(pp, pk_1, \aleph)$ and sends the puzzle pz and malleable proof π to R. Specifically, the solution $\aleph$ can be used to transfer $\tilde{\sigma}_{H\to R}$ to the valid signature $\sigma_{H\to R}$.

After verifying pz using $\Pi_{\mathsf{LRP}}.\mathrm{PVerify}(pp, pz, \pi)$, the receiver R verifies the correctness of $\tilde{\sigma}_{H\to R}$ on $[\tau_{H\to R}]$ with statement $\gamma = g^{\aleph}$, where γ is taken from $pz = (\alpha, \beta, \gamma)$. Then, R calls $(pz', \pi', r_1) \leftarrow \Pi_{\mathsf{LRP}}.\mathrm{PRand}(pp, pk_1, pz, \pi)$, records random number r_1, and sends pz' and π' to sender S. Similarly, after using $\Pi_{\mathsf{LRP}}.\mathrm{PVerify}(pp, pz', \pi')$ to verify the correctness of pz', the sender S calls $(pz'', \pi'', r_2) \leftarrow \Pi_{\mathsf{LRP}}.\mathrm{PRand}(pp, pk_1, pz', \pi')$, stores r_2 for R. Additionally, S computes a pre-signature $\tilde{\sigma}_{S\to H}$ on a payment $[\tau_{S\to H}]$ with statement $\gamma'' = g^{\aleph''} = g^{\aleph+r_1+r_2}$, where γ'' is taken from $pz'' = (\alpha'', \beta'', \gamma'')$. Then, S sends $[\tau_{S\to H}]$, $\tilde{\sigma}_{S\to H}$, pz'', π'' to H. The hub verifies pz'' with π'' and also verifies $\tilde{\sigma}_{S\to H}$ on $[\tau_{S\to H}]$ with statement γ'' which is taken from $pz'' = (\alpha'', \beta'', \gamma'')$.

If no verification fails, H proceeds in the puzzle so-

lution (cf., Figure 2(b)). Specifically, H calls $\aleph'' \leftarrow \Pi_{\mathsf{LRP}}.\mathrm{PSol}(pk_1, sk_1, pz'')$. If $g^{\aleph''}$ belongs to pz'', the hub uses $\aleph''$ as the witness to adapt pre-signature $\tilde{\sigma}_{S\to H}$ to a full signature $\sigma_{S\to H}$. The hub commits the payment $\tau_{S\to H}$ by adding $\sigma_{S\to H}$ and its signature and sends the committed payment back to sender S. Meanwhile, H records audit trail ($\tilde{\sigma}_{S\to H}$, $\sigma_{S\to H}$, pz'', π'', $[\tau_{S\to H}]$) for off-chain transaction $\tau_{S\to H}$.

Via the adaptor signature scheme Π_{ad}, the sender S extracts witness $\aleph''$ from its pre-signature $\tilde{\sigma}_{S\to H}$ and the received full signature $\sigma_{S\to H}$ in $\tau_{S\to H}$. Then S de-randomizes $\aleph''$ to $\aleph' = \aleph'' - r_2$ and sends $\aleph'$ to R. Finally, the receiver R de-randomizes $\aleph'$ to $\aleph = \aleph' - r_1$, uses $\aleph$ as witness to adapt the hub's pre-signature $\tilde{\sigma}_{H\to R}$ to a full signature $\sigma_{H\to R}$ on payment $[\tau_{H\to R}]$. R adds it together with its own signature to the payment to commit it sends the committed payment $\tau_{H\to R}$ to H to complete the relay payment. After that, H records audit trail ($\tilde{\sigma}_{H\to R}$, $\sigma_{H\to R}$, pz, π, $[\tau_{H\to R}]$) for off-chain transaction $\tau_{H\to R}$.

Phase III: Close channel. In this phase, the Sender/Receiver or Hub submits the latest transactions of their channels to the blockchain for releasing the assets.

Phase IV: Payment audit. Before analyzing sender-to-receiver payments in the aimed channel, the auditor O first checks if it is closed. Subsequently, O requires H to provide all audit trails that he/she involves and checks if H submits all transactions $\{[\tau]\}$ of the aimed channel via the transaction hash chain. Completing the hash chain means H honestly submits all hub-involved transactions respecting the aimed channel.

If the check is successful, the auditor can audit the sender-to-receiver payments of the aimed channel (cf., Figure 2(c)). Given audit trails ($\tilde{\sigma}_0$, σ_0, pz_0, π_0, $[\tau_0]$) and ($\tilde{\sigma}_1$, σ_1, pz_1, π_1, $[\tau_1]$) for off-chain payments τ_0 and τ_1, respectively, auditor O can determine whether τ_0 and τ_1 form a relay payment by verifying the following conditions. (i) The pre-signatures and signatures on $[\tau_0]$ and $[\tau_1]$ in the audit trails are correct. (ii) The auditor O parses $pz_0 = (\alpha_0, \beta_0, \gamma_0)$ and $pz_1 = (\alpha_1, \beta_1, \gamma_1)$, extracts witness $\aleph_0$ from $(\tilde{\sigma}_0, \sigma_0)$ using the adaptor signature scheme Π_{ad} with statement γ_0, and similarly extracts witness $\aleph_1$ from $(\tilde{\sigma}_1, \sigma_1)$. The auditor O verifies that $g^{\aleph_0} = \gamma_0$ and $g^{\aleph_1} = \gamma_1$ hold, and also verifies $1 \leftarrow \Pi_{\mathsf{LRP}}.\mathrm{PVerify}(pp, pz_0, \pi_0)$ and $1 \leftarrow \Pi_{\mathsf{LRP}}.\mathrm{PVerify}(pp, pz_1, \pi_1)$. (iii) The auditor O verifies $1 \leftarrow \Pi_{\mathsf{LRP}}.\mathrm{PLink}(pp, sk_0, pz_0, pz_1)$.

If the hub manipulates the audit trails for τ_0 and/or τ_1, then not all of the first two conditions are met; thus, the auditor can detect such manipulations in the audit. Under the assumption that the hub uses a fresh witness $\aleph$ for each sender-to-receiver payment, it is not difficult to prove that τ_0 and τ_1 form a relay payment *iff* all the above conditions are met. In addition, the auditor can detect whether the hub uses a fresh witness for each relay payment in any audit epoch by collecting all audit trails generated in that epoch, retrieving a witness from each audit trail, and checking the freshness of each witness. This process is similar to the hub checking the freshness of each token identification in $\mathrm{A^2L^+}$, which is a typical unlinkable PCH supporting fixed payment amounts with high interoperability [2]. After linking the audit trails, if some audit trails respect the aimed channel and cannot be linked to another trail, O can suggest a doubt that the hub may submit incomplete audit trails.

Remark: It should be noted that our methodology primarily addresses the auditability of closed channels, constrained by the hash chain mechanism that depends on transactions to open and close channels for constructing and verifying the offline payment hash chain. If a verification method, irrespective of on-chain transactions, appears, our audit method could also be compatible with the unclosed channel. Our methodology satisfies the atomicity requirement by ensuring that both off-chain transactions, $\tau_{S\to H}$ and $\tau_{H\to R}$, complete together. The completion of the off-chain transactions relies on the hub solving the puzzle associated with $[\tau_{S\to H}]$, enabling the receiver to solve the puzzle associated with $[\tau_{H\to R}]$ with the sender's help. Our solution satisfies the unlinkability requirement as the puzzle associated with $[\tau_{S\to H}]$ is randomized from and thus unlinkable to the puzzle associated with $[\tau_{H\to R}]$.

VI. Security Analysis

This section demonstrates the security analysis of AuditPCH. We first give an informal analysis of how AuditPCH satisfies *atomicity*, *unlinkability*, and *auditability*. Second, we define a theorem that the protocol Π_{AuditPCH} *GUC-realizes* the ideal functionality $\mathcal{F}_{\mathsf{AuditPCH}}$. Finally, we give the formal security proof under the GUC framework [17].

A. Informal Analysis

Atomicity. It means that the honest parties (i.e., sender, receiver, or hub) do not suffer from any loss of their assets, and meanwhile, no corrupted parties can obtain more assets than they own. Atomicity will be broken if the following situations hold: (1) the honest hub cannot receive assets from the sender after forwarding his/her assets to the receiver and (2) the honest receiver cannot gain assets from the hub that has received assets from the sender. The first holds *iff* the receiver can collect the solution without the secret key sk_1 and use the solution to obtain a valid signature of the conditional transaction that pays assets to himself. Under the LRP scheme's security property, dishonest receivers cannot open the puzzle without the secret key sk_1. The second holds *iff* the dishonest hub publishes a signature $\sigma_{S\to H}$ that can pass the verification of the sender, but the sender cannot extract a correct solution $\aleph + r_1 + r_2$. The happening of this situation implies that the adversary can win the signature adaptability and witness-extractability game of the adaptor signature scheme. However, as proven in [2], the adversary wins the game with negligible possibility. Thus, the second situation cannot occur with a nonnegligible possibility.

Unlinkability. It means any other parties except the auditor cannot associate the sender and receiver of each payment. In AuditPCH, a Hub-Receiver transactions $\tau_{H\to R}$ are independent of the Sender-Hub transaction $\tau_{S\to H}$, even if these two transactions are generated through a relay payment. The hub can link the sender-to-receiver payment relationship *iff* one of the conditions happens: (1) the hub is capable of linking the blinded signature σ^* to the unblinded signature σ or (2) the hub can output the correct puzzle pairs (pz, pz'') where pz'' originates from pz. First, under the hiding property of the commitment scheme, the hub cannot collect the identification oid that the hub is authenticated via a signature $\sigma\hat{*}$. The hub cannot associate σ^* with σ even though the hub knows the identification tid at that

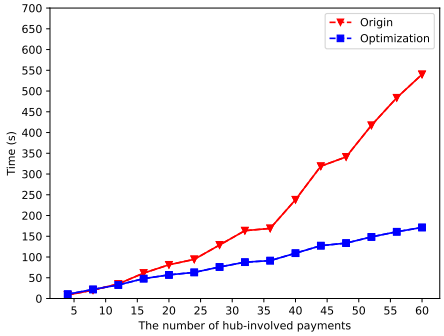

Fig. 3: The time cost of executing the AuditPCH protocol in the origin and optimization versions. The number of hub-involved payments ranges from 4 to 60.

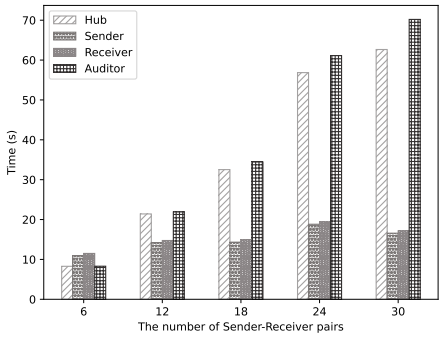

Fig. 4: The computation cost of each role while executing the AuditPCH protocol. The number of Sender-Receiver pairs ranges from 6 to 30.

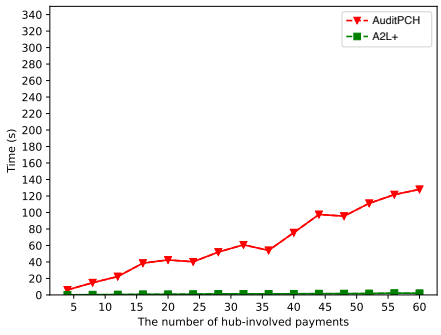

Fig. 5: The time cost comparison between AuditPCH and A^{2}L$^+$. The number of hub-involved payments ranges from 4 to 60.

time. Second, the hub p_h cannot link the randomized puzzle pz'' with the initial puzzle pz since the adversary wins the adversary-unlinkability game of the LRP scheme with negligible advantage. Thus, the corrupted hub makes it difficult to capture the Sender-Receiver relationship of each relay payment.

Auditability. It means the auditor can collect the order of transactions and the payment relationship in the aimed party's channel if the given channel with the complete payment hash chain. Under the adversary-unlinkability and correctness property of the LRP scheme, the auditor O can reveal the information that the randomized puzzle pz'' comes from the initial puzzle pz. Since the extractability of the adaptor signature scheme, O can extract the solution from pre-signatures and signatures and compare the solutions with the puzzle to check the validity of the puzzle. Then, the relationship between the transactions $(\tau_{S\to H}, \tau_{H\to R})$ can be found via a one-to-one mapping between the puzzle pair (pz'', pz) and the transaction pair $(\tau_{S\to H}, \tau_{H\to R})$. Further, by requiring the opened and closed transactions from the blockchain, O can check the completeness of the hash chain of the given channel from the hub. Since each transaction must append the last transaction in the channel, the auditor can collect the order of the transactions if the hash chain is completed in the given channel.

B. Security Analysis

The GUC security notion captures the following theorem.

Theorem VI.1. *Assume that the global ideal functionality $\mathcal{F}_L$, the anonymous communication network functionality $\mathcal{F}_{\mathsf{syn}}$, and the secure transmission functionality $\mathcal{F}_{\mathsf{smt}}$ are given, it exists a secure protocol Π_{AuditPCH} that GUC-realizes the ideal auditable PCH functionality $\mathcal{F}_{\mathsf{AuditPCH}}$ in the $(\mathcal{F}_L, \mathcal{F}_{\mathsf{syn}}, \mathcal{F}_{\mathsf{smt}})$-hybrid model.*

Proof. Due to the space limitation, here we omit this proof and present the comprehensive proofs in the Appendix **??**. □

VII. Performance Analysis

AuditPCH is implemented in Java and C languages with JPBC and RELIC library [31, 32]. All experiments are conducted on a 1.6GHz Laptop (Intel Core i5-8265U) with 8-Core and 8GB RAM. Over the curve *secp256kl*, we initialize the adaptor signatures scheme [30] and the blind signature [14, 33]. The malleable proof utilizes the Groth-Sahai proof [34]. To demonstrate the expressiveness of AuditPCH, we run and record the performance of each party in each phase. Since the main concern is the performance effect when applying our solution to blockchains, we discuss only the extra resource consumption when using AuditPCH and exclude the resources of generating on-chain transactions.

Computation Cost. As shown in Table II and Table III, for the sender-to-receiver payment, the total computation cost of each role is no more than 7.6s. Table III shows the computation cost when one Sender-Receiver pair establishes one relay payment via a hub. The main computation cost is spent on the Pay operation, specifically in the puzzle generation stage. The puzzle generation stage calls the puzzle generation and verification algorithm of the LRP scheme, in which computation times are $\sim$ 0.62s and $\sim$ 6.50s, respectively. This cost is caused by the Groth-Sahai proof [19] that is used to construct the malleable proof and the NIZK proof [22] that states the correctness of the puzzle. In the Open operation, the computation cost is caused by preparing the public parameters of the LRP scheme. In particular, this operation does not require the sender/receiver or hub to consume extra computation resources. The Close operation is identical to the close channel in the general PCH; thus, parties do not consume additional resources. The Audit operation requires the auditor to utilize computational resources to connect puzzles. This involves implementing the PLink algorithm, which incurs a low cost of approximately $\sim$ 0.01s per puzzle pair linked. Before linking received puzzles, the PVerify algorithm must be executed to ensure their accuracy. The primary cost for the auditor in the Audit operation is attributed to puzzle verification. To optimize the Audit operation, the auditor can verify all malleable proof at the beginning and then check the linkability of receiving puzzles.

Further, Figure 3 and Figure 4 illustrate the impact of the number of sender-receiver pairs and hub-involved payments in each channel. To investigate the impact of the number of hub-involved payments on computation cost, we conducted experiments (cf., Figure 3) involving four to sixty relay payments among two Sender-Receiver pairs while recording the total computation cost. As depicted by the red line in Figure 3, the number of hub-involved payments and relay payments exhibit a

roughly linear growth trend. Additionally, optimizing the audit process for the auditor, as indicated by the blue line in Figure 3, by uniformly validating puzzles and then using linked puzzles to output the correct hub-involved payment pair, can significantly reduce computation cost.

To demonstrate the impact of the number of sender-receiver pairs on computation cost, we conducted five sets of experiments (cf., Figure 4) with six to thirty sender-receiver pairs, with an interval of six pairs. We recorded the computation cost for each role, with each sender-receiver pair performing two relay payments. As shown in Figure 4, the computation cost for each sender and receiver was almost consistent across all experimental groups, as the relay payments for each sender-receiver pair were independent. However, as the number of sender-receiver pairs increased, the cost for the hub and auditor increased, as they invested more computational resources to verify the correctness of the received puzzles.

Moreover, since AuditPCH was inspired by A^2L^+, we expanded our comparison with A^2L^+. As A^2L^+ lacks an auditing mechanism, we restricted our comparison to the PCH processing off-chain payments. The comparison result is shown in Figure 5. As the number of payments increases, the time consumption gap between A^2L^+ and AuditPCH gradually widens. The payment happens in two sender-receiver pairs, and the number of hub-involved payments ranges from four to sixty.

Discussion. The consumption gap between $A2L^+$ and AuditPCH is primarily due to using the malleable proof scheme in AuditPCH. We instantiate the malleable proof using the GS-proof system instead of other systems like ZK-SNARKs [15] because GS-proof uniquely offers the required malleability, even though it is not the most efficient zero-knowledge proof system. Developing more efficient constructions for malleable proofs would significantly boost the efficiency of AuditPCH. For example, if we could optimize the malleable proof computations by 50%, reducing their computation time to 6.5 seconds, the total audit time per payment pair would decrease to approximately 6.758 seconds. This improvement would enhance the overall audit efficiency by about 49%, substantially narrowing the performance gap caused by the malleable proof overhead.

TABLE II: The computation cost of the LRP scheme.

	Setup	KGen	PVerify	PGen	PSol	PRand	PLink
Cost(s)	0.419	0.074	6.504	0.615	0.070	0.622	0.013

TABLE III: The computation cost of AuditPCH. Time is shown in seconds.

	Pay			Open	Close	Audit	Total
	Channel Authentication	Puzzle Generation	Puzzle Solution				
Sender	0.002	7.279	0.008	–	–	–	7.289
Hub	0.004	0.768	6.582	0.004	–	–	7.358
Receiver	–	7.583	0.008	–	–	–	7.591
Auditor	–	–	–	0.485	–	13.258	13.716

Communication cost. Table IV shows the additional information that needs to be exchanged. The most resource-consuming operation is Pay, where each party needs to exchange the puzzles ($\sim 5.76KB$). Only the auditor bears the communication cost of the Open operation, as they broadcast the public parameters. The Close operation does not involve any additional communication costs. The total communication cost of one payment (*i.e.*, the total cost of the first three operations) is no greater than $26.5KB$. During the Audit phase, the communication cost increases linearly with the number n of payments in the channel as the hub submits all transactions, puzzles, and signatures.

TABLE IV: The communication cost of AuditPCH. n is the number of payments. Size is shown in KB.

	Pay			Open	Close	Audit
	Channel Authentication	Puzzle Generation	Puzzle Solution			
Sender	0.969	5.764	0.281	–	–	–
Hub	0.203	5.764	0.562	–	–	$11.52n + 2n[TX]$
Receiver	–	5.404	–	–	–	–
Auditor	–	–	–	7.230	–	–

In summary, the computational cost for sender-receiver payments remains low, and the communication overhead is manageable, with the total cost for one payment not exceeding 26.5 KB. Besides, the audit process requires an acceptable computational cost ($\sim$ 13.7), making AuditPCH a practical audit solution for off-chain payments.

VIII. Conclusion

This paper presented an AuditPCH methodology to address both privacy protection and auditability issues via the LRP technique and a self-design hashchain approach. In detail, a novel linkable randomizable puzzle scheme is leveraged to construct conditional transactions guaranteeing the unlinkability of transacting parties and offers a trapdoor for auditors to reveal the parties' relationship. Further, the hashchain mechanism is designed to enable auditors to verify the integrity of the off-chain payment provided by the hub. Also, we prove the security of AuditPCH in the GUC framework and demonstrate the functionality and efficiency of AuditPCH in experiments. In future work, we will extend our solution to the new PCH-solution named "BlindHub" [9], which overcomes the payment amount limitation of A^2L^+ and thus supports flexible payments and privacy protection simultaneously.

IX. Acknowledgements

The work was supported in part by the National Natural Science Foundation of China (Nos. 62332007, U22B2028, U2001205, 62102165, 62472198), the Science and Technology Major Project of Tibetan Autonomous Region of China (No. XZ202201ZD0006G), Open Research Fund of Machine Learning and Cyber Security Interdiscipline Research Engineering Center of Jiangsu Province (No. SDGC2131), National Joint Engineering Research Center of Network Security Detection and Protection Technology, Guangdong Key Laboratory of Data Security and Privacy Preserving, Guangdong Hong Kong Joint Laboratory for Data Security and Privacy Protection, and Engineering Research Center of Trustworthy AI, Ministry of Education, Guangzhou Science and Technology Plan Project (No.2024A03J0464).

References

[1] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski, "Perun: Virtual payment hubs over cryptocurrencies," in *2019 IEEE SP*, 2019, pp. 106–123.

[2] E. Tairi, P. Moreno-Sanchez, and M. Maffei, "A 2l: Anonymous atomic locks for scalability in payment channel hubs," in *IEEE SP*, 2021, pp. 1834–1851.

[3] L. Hanzlik, J. Loss, S. Thyagarajan, and B. Wagner, "Sweep-uc: Swapping coins privately," in *IEEE SP*, 2024.

[4] 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 *CCS*. ACM, 2023, pp. 1541–1555.

[5] Y. Li, J. Weng, W. Wu, M. Li, Y. Li, H. Tu, Y. Wu, and R. H. Deng, "PRI: PCH-based privacy-preserving with reusability and interoperability for enhancing blockchain scalability," *JPDC*, vol. 180, p. 104721, 2023.

[6] C. Garman, M. Green, and I. Miers, "Accountable privacy for decentralized anonymous payments," in *FC*. Springer, 2016, pp. 81–98.

[7] E. Heilman, L. Alshenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg, "Tumblebit: An untrusted bitcoin-compatible anonymous payment hub," in *NDSS*. ISOC, 2017.

[8] N. Glaeser, M. Maffei, G. Malavolta, P. Moreno-Sanchez, E. Tairi, and S. A. K. Thyagarajan, "Foundations of coin mixing services," in *CCS*, 2022, pp. 1259–1273.

[9] 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 *IEEE SP*, 2023.

[10] N. Narula, W. Vasquez, and M. Virza, "zkledger: Privacy-preserving auditing for distributed ledgers," in *NSDI 18*. USENIX Association, 2018, pp. 65–80.

[11] K. Wüst, K. Kostiainen, V. Čapkun, and S. Čapkun, "Prcash: Fast, private and regulated transactions for digital currencies," in *FC*. Springer, 2019, pp. 158–178.

[12] E. Androulaki, J. Camenisch, A. D. Caro, M. Dubovitskaya, K. Elkhiyaoui, and B. Tackmann, "Privacy-preserving auditable token payments in a permissioned blockchain system," in *AFT*. ACM, 2020, pp. 255–267.

[13] G. G. Dagher, B. Bünz, J. Bonneau, J. Clark, and D. Boneh, "Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges," in *CCS*. ACM, 2015, pp. 720–731.

[14] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *CRYPTO*. Springer, 1991, pp. 129–140.

[15] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *IEEE SP*, 2014.

[16] M. Chase, M. Kohlweiss, A. Lysyanskaya, and S. Meiklejohn, "Malleable proof systems and applications," in *EUROCRYPT*. Springer, 2012, pp. 281–300.

[17] R. Canetti, Y. Dodis, R. Pass, and S. Walfish, "Universally composable security with global setup," in *TCC*. Springer, 2007, pp. 61–85.

[18] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *CRYPTO*. Springer, 2004, pp. 41–55.

[19] J. Groth and A. Sahai, "Efficient non-interactive proof systems for bilinear groups," in *EUROCRYPT*. Springer, 2008, pp. 415–432.

[20] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostakova, M. Maffei, P. Moreno-Sanchez, and S. Riahi, "Generalized bitcoin-compatible channels." *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 476, 2020.

[21] P. Gerhart, D. Schröder, P. Soni, and S. A. K. Thyagarajan, "Foundations of adaptor signatures," in *EUROCRYPT*, ser. LNCS, M. Joye and G. Leander, Eds., vol. 14652. Springer, 2024, pp. 161–189.

[22] M. Blum, P. Feldman, and S. Micali, "Non-interactive zero-knowledge and its applications," in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*. ACM, 2019, pp. 329–349.

[23] M. Green and I. Miers, "Bolt: Anonymous payment channels for decentralized currencies," in *CCS*. ACM, 2017, pp. 473–489.

[24] J. Lind, O. Naor, I. Eyal, F. Kelbert, E. G. Sirer, and P. Pietzuch, "Teechain: a secure payment network with asynchronous blockchain access," in *SOSP*. ACM, 2019, pp. 63–79.

[25] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, "Mixcoin: Anonymity for bitcoin with accountable mixes," in *FC*. Springer, 2014, pp. 486–504.

[26] L. Valenta and B. Rowan, "Blindcoin: Blinded, accountable mixes for bitcoin," in *FC*. Springer, 2015, pp. 112–126.

[27] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *FOCS*. IEEE, 2001, pp. 136–145.

[28] M. Graf, D. Rausch, V. Ronge, C. Egger, R. Küsters, and D. Schröder, "A security framework for distributed ledgers," in *CCS*. ACM, 2021, pp. 1043–1064.

[29] G. Castagnos and F. Laguillaumie, "Linearly homomorphic encryption from ddh," in *RSA*. Springer, 2015, pp. 487–505.

[30] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostáková, M. Maffei, P. Moreno-Sanchez, and S. Riahi, "Generalized channels from limited blockchain scripts and adaptor signatures," in *ASIACRYPT*. Springer, 2021, pp. 635–664.

[31] A. D. Caro, "The java pairing based cryptography library (jpbc)," http://libeccio.di.unisa.it/projects/jpbc/index.html Accessed: 2022-06-13, 2022.

[32] D. F. Aranha and C. Gouvea, "Relic is a modern research-oriented cryptographic meta-toolkit with emphasis on efficiency and flexibility," https://github.com/relic-toolkit/relic Accessed: 2022-06-13, 2022.

[33] D. Pointcheval and O. Sanders, "Short randomizable signatures," in *CT-RSA*. Springer, 2016, pp. 111–126.

[34] G. V. Laer, "An implementation of the groth-sahai nizk protocol," https://github.com/gijsvl/groth-sahai Accessed: 2022-06-13, 2022.

I. Problem Statement

Following the notations in A^2L^+, a PCH is defined as a graph ($\mathbb{V}$, $\mathbb{E}$). Expressly, the vertex set $\mathbb{V}$ represents a set of payer addresses, and the edge set $\mathbb{E}$ denotes a set of payment channels $\{\gamma_i\}$ among the hub p_h and transacting parties $\{p_i\}$. A blockchain $\mathbb{B}$ stores the element (p_i, v, s), in which p_i is a blockchain address, v denotes his on-chain balance, and s denotes the number of off-chain payments in the channel. In detail, γ is composed by a tuple ($\gamma.id$, $\gamma.users$, $\gamma.bal$, $\gamma.cash$, $\gamma.st$, $\gamma.closed$, $\gamma.s$). $\gamma.id$ is the channel identification and $\gamma.users$ represents the blockchain addresses (i.e. p_i and p_j) of the channel owners. Further, $\gamma.bal$ is the channel capacity and $\gamma.cash$ is a mapping from the payer address to his/her current balance. $\gamma.st$ is a tuple (η_1, ..., η_n) for storing hub-involved payments, where $\eta_k = (k, \gamma.cash_k(p_i), \gamma.cash_k(p_j), \text{Hash}(\eta_{k-1}))$ is the k-th hub-involved payment, where ϵ is the payment amount at the k-th payment and the hash of last payment will be appended. $\gamma.closed$ represents whether the channel is closed, where $\gamma.closed$ is 1 for the closed channel and 0 for the opened channel. $\gamma.s$ is the number of hub-involved payments established in the channel. Additionally, to store hub-involved payments caused by same sender-to-receiver payments, we denote an oracle $\mathbb{R}$ where an element is composed by the tuple: ($\gamma.id$, $\gamma'.id$, $\gamma.\eta_i$, $\gamma'.\eta_j$). Further, we omit the fee the senders and receivers pay for the hub.

Definition I.1 (AuditPCH)**.** *AuditPCH is represented as a graph ($\mathbb{V}$, $\mathbb{E}$) with four operations (*open*, *pay*, *close*, *audit*). First, transacting parties $\{p_i\}$ use the* open *operation to establish a payment channel with the hub p_h. Second, the* pay *operation is called for updating the channel states. Third, users can use the* close *operation to submit the latest balance in their channel to $\mathbb{B}$ for closing their channel. Finally, the* audit *operation is specific to the audit requirement of auditors p_o. The four operations are as follows:*

- open(p_i, p_h, ϵ_i, ϵ_h): On input the payer addresses p_i, $p_h \in \mathbb{V}$ and the deposits (ϵ_i, ϵ_h), if the operation is executed by payers p_i and p_h, the operation opens a new channel γ with a fresh identification $\gamma.id$. Note that p_i owns at least ϵ_i and p_h that owns at least ϵ_h in the blockchain. First, this operation updates (p_i, $v_i = v_i - \epsilon_i$, 0) and (p_h, $v_h = v_h - \epsilon_h$, 0) to $\mathbb{B}$, where v_i and v_h are the balances of p_i and p_h in $\mathbb{B}$. Second, the operation initializes $\gamma.users = (p_i, p_h)$, $\gamma.bal = \epsilon_i + \epsilon_h$, $\gamma.cash(p_i) = \epsilon_i$, $\gamma.cash(p_h) = \epsilon_h$, $\gamma.s$=0, $\gamma.st = \{\eta_0 = (\gamma.s, \epsilon_i, \epsilon_h)\}$, and $\gamma.closed = 0$. If the operation succeeds, it adds γ to $\mathbb{E}$; otherwise, it terminates.
- pay(γ_1, γ_2, ϵ): There are two different channels (γ_1, γ_2), one for a sender p_s and a hub p_h, and one for a receiver p_r and the hub p_h. Let ϵ represent a value that p_s wants to pay for p_r. If the operation is executed by parties p_s, p_h, and p_r, and the condition $\gamma_1.cash(p_s) \geq \epsilon \wedge \gamma_2.cash(p_h) \geq \epsilon$ holds, the operation does the following: (1) initializes the hub-involved payment $\eta = (\gamma_1.s + 1, \gamma_1.cash(p_s) - \epsilon, \gamma_1.cash(p_h) + \epsilon, \text{Hash}(\eta_{\gamma_1.s}))$ and $\eta' = (\gamma_2.s+1, \gamma_2.cash(p_h) - \epsilon, \gamma_2.cash(p_r) + \epsilon, \text{Hash}(\eta_{\gamma_2.s})$ and appends η to $\gamma_1.st$ and η' to $\gamma_2.st$, (2) updates $\gamma_1.cash(p_s) = \gamma_1.cash(p_s) - \epsilon$ and $\gamma_1.cash(p_h) = \gamma_1.\text{cash}(p_h)+\epsilon$, (3) updates $\gamma_2.cash(p_h) = \gamma_2.cash(p_h) - \epsilon$ and $\gamma_2.cash(p_r) = \gamma_2.\text{cash}(p_r)+ \epsilon$, (4) stores ($\gamma_1.id$, $\gamma_2.id$, η, η') in $\mathbb{R}$, and (5) $\gamma_1.s = \gamma_1.s + 1$ and $\gamma_2.s = \gamma_2.s + 1$, if one of steps in (1)-(5) fails, the operation recalls (1)-(5). Note that multiple sender-to-receiver payments are established concurrently so that p_h cannot reveal the sender-to-receiver relationships directly.
- close(p_i, p_h, γ): If the operation is authorized by the parties p_i and p_h, it will set $\gamma.closed$=1 and updates (p_i, v_i=v_i + $\gamma.cash(p_i)$, $\gamma.s$) as well as (p_h, v_h=v_h+$\gamma.cash(p_h)$, $\gamma.s$), where v_i and v_h are the balances of p_i and p_h in $\mathbb{B}$. Then, the operation removes the channel γ from $\mathbb{V}$.
- audit(TxSet, p_i, p_h, p_o): On input the channel set $TxSet$, if p_o and p_h authorize the operation, $\gamma_i.closed = 1$, and the hash chain of hub-involved payments of γ_i is completed, this operation will retrieve the states pair ($\gamma_i.id$, $\cdot$, $\cdot$, $\cdot$) from $\mathbb{R}$ to the set $\mathbb{A}$. Finally, it returns $\mathbb{A}$.

II. The Security Proof of LRP

Here, we give sketch proof that the LRP scheme satisfies the security, correctness, and adversary-unlinkability properties.

Proof(sketch). We detail the proof as follows:

Security. The security holds if the adversary cannot output the correct solution $\aleph$ of a puzzle pz even if it owns the public parameter pp, the puzzle pz, and a proof π of the puzzle. Specifically, due to the encryption scheme Π_{CL} satisfying IND-CPA security, the adversary cannot distinguish between the ciphertext of 0 and the ciphertext of $\aleph$. Next, since the group G is a Discrete-Logarithm(DLOG) hard group, the adversary cannot output the discrete logarithm of γ. Finally, the zero-knowledge proof Π_{MP} satisfies the zero-knowledge property so that the proof π would not reveal any information about the solution $\aleph$. Thus, the security holds.

Correctness. The correctness is guaranteed by two conditions: (1) $\aleph \leftarrow$ **PSol**(pk_1, sk_1, pz) and (2) PLink(pp, sk_0, pz, pz')= 1. Concretely, pz can be parsed as (α, β, γ) and pz' can be parsed as $(\alpha', \beta', \gamma')$. The first condition holds **PSol** decrypt β, a ciphertext of the solution $\aleph$ under the public key pk_1, with the secret key sk_1. Due to the correctness of the encryption scheme Π_{CL}, **PSol** can collect the correct solution $\aleph$. Next, based on the construction of the algorithm PLink, we consider the second condition and demonstrate why the condition holds as follows:

$$\begin{aligned} g^{\Pi_{\text{El}}.\,\text{Dec}(pk_0, sk_0, \alpha')} / g^{\Pi_{\text{El}}.\,\text{Dec}(pk_0, sk_0, \alpha)} &= g^{r_0+r+\aleph} / g^{r_0} \\ &= \gamma' / \gamma \end{aligned} \tag{1}$$

where α' and γ' originates from α and γ. Thus, the correctness of the scheme has been proved.

Adversary-unlinkability. Based on the construction of the LRP scheme, a puzzle pz consists of a ciphertext α from the encryption scheme Π_{El}, a ciphertext β from the encryption scheme Π_{CL}, and an element $h^{\aleph}$ belonging to a DLOG-hard group G. The construction of the randomized puzzle pz' is similar to the puzzle pz. The advantage of the adversaries to win the game satisfies the formula: $\text{ADV}^{\text{PLink}}_{\mathcal{A},\text{LRP}} = \text{ADV}_{\mathcal{A},\text{El}} + \text{ADV}_{\mathcal{A},\text{CL}} + \text{ADV}_{\mathcal{A},\text{DLOG}} + \text{ADV}_{\mathcal{A},\text{MP}}$.

First, the advantage $\text{ADV}_{\mathcal{A},\text{El}}$ refers to the advantage of the adversaries distinguishes which ciphertexts (α_0, α_1) come from the ciphertext α'_b comes from. Both the encryption scheme Π_{El}

satisfies the *IND-CPA* secure, implying that the adversaries tell which of the two messages a given ciphertext generated from with a negligible advantage neg_1. That is, the advantage satisfies the following formulation:

$$\mathsf{ADV}_{\mathcal{A},\mathsf{EI}} \leq \mathsf{neg}_1 \tag{2}$$

Second, the encryption scheme Π_{CL} also satisfies the *IND-CPA* secure. Thus, the advantage $\mathsf{ADV}_{\mathcal{A},\mathsf{CL}}$ of the adversary to distinguish which ciphertexts (β_0, β_1) come from β'_b satisfies the following formulation:

$$\mathsf{ADV}_{\mathcal{A},\mathsf{CL}} \leq \mathsf{neg}_2 \tag{3}$$

Next, since the construction of γ' follows the formula: $\gamma' = \gamma^r$, the randomized value γ' is complete with a randomness r. Hence, a randomized value γ'_b is equally likely to be the randomized version of the value γ_0 and γ_1. The advantage of the adversary to output the correct puzzle pair (γ'_b, γ_b) satisfies the following relation:

$$\mathsf{ADV}_{\mathcal{A},DLOG} \leq \mathsf{neg}_3 \tag{4}$$

Further, there are malleable proofs sent to the adversary. Under *witness-distinguishability*, given two zero-knowledge proofs (π_0, π_1) and their malleable proofs (π'_0, π'_1), adversaries are capable of identifying that the chosen malleable proof π'_b comes from which zero-knowledge proof with a negligible advantage neg_3. That is because the malleable proof scheme satisfies the *witness distinguishability* property so that the malleable proof π'_b is nearly similar to the version of any zero-knowledge proof. Hence, the advantage here satisfies the following relation:

$$\mathsf{ADV}_{\mathcal{A},\mathsf{MP}} \leq \mathsf{neg}_4 \tag{5}$$

Combined the Equation 2, Equation 3, Equation 4, and Equation 5, we demonstrate why the adversary-unlinkability property holds as follows:

$$\begin{aligned}\mathsf{ADV}^{\mathsf{PLink}}_{\mathcal{A},\mathsf{LRP}} &= \mathsf{ADV}_{\mathcal{A},\mathsf{EI}} + \mathsf{ADV}_{\mathcal{A},\mathsf{CL}} + \mathsf{ADV}_{\mathcal{A},\mathsf{DLOG}} + \mathsf{ADV}_{\mathcal{A},\mathsf{MP}} \\ &\leq \mathsf{negl}_1 + \mathsf{negl}_2 + \mathsf{negl}_3 + \mathsf{neg}_4 \leq \mathsf{negl}\end{aligned} \tag{6}$$

□

III. Formal Security Analysis of AuditPCH

A. Ideal Funcationality

Next, we show the details of the ideal functionality $\mathcal{F}_{\mathsf{AuditPCH}}$.

Ideal Functionality $\mathcal{F}_{\mathsf{AuditPCH}}$

Setup: Upon receiving (setup, sid) from party p_o, do the following:
- If (trapdoor, ·, p_o) exists, abort.
- Otherwise, send (trapGen, sid) to the simulator $\mathcal{S}im$ and wait for (trapRes, sid, pk_0).
- If $pk_0 = \perp$, ignore.
- Record (trapdoor, pk_0, p_o) into $\mathbb{D}$ and return (setupRes, sid, pk_0) to p_o.

Open: Upon receiving (create, sid, γ, $txid_p$) from party p, proceed as follows:
- Send (create, γ, $txid_p$) to the simulator $\mathcal{S}im$.
- Wait for b from $\mathcal{S}im$.
- If $b = \perp$, terminate.
- Add γ into $\mathbb{E}$.
- Output (created, $\gamma.id$) to the parties $\gamma.users$.

Pay: Upon receiving (pay, sid, p_r, pk_0) from party p_s, do as follows:
- Retrieve γ_1 where $\gamma_1.users = (p_s, p_h)$, if $\gamma_1 = \perp$, terminate.
- Retrieve γ_2 where $\gamma_2.users = (p_h, p_r)$, if $\gamma_2 = \perp$, terminate.

Wait for (registration, p_r) from p_s, send (regReq, p_s, sid) to $\mathcal{S}im$ and p_h, do the following:
- Wait for (regRes, b, sid) from p_h.
- If $b = \perp$, then abort.
- Sample oid from $\{0,1\}^\lambda$ and add oid into $\mathbb{O}$.
- Send (registered, oid) to p_s, p_r, and $\mathcal{S}im$.
- Set $\tau_0 = (\eta = (\gamma_2.s+1, \gamma_2.cash(p_h) -= \epsilon, \gamma_2.cash(p_r) += \epsilon, \mathrm{Hash}(\eta_s)), time = now + 2t)$, where t is a constant period.
- Send τ_0 to p_h and p_r.

Wait for (PGen, oid', sid) from p_r, if $oid' = \perp$ or $oid' \notin \mathbb{O}$, abort; otherwise, do the following:
- Remove oid' from $\mathbb{O}$.
- Send (PGenReq, sid, p_r, oid') to p_h and $\mathcal{S}im$.
- Receive (PGenRes, sid, b), if $b = \perp$, terminate.
- Otherwise, sample a puzzle pair (pid, pid') from $\{0,1\}^\lambda$.
- Store $(pk_0, pid, pid', \perp)$ into $\mathbb{P}$.
- Send (puzzle, sid, (pid, pid')) to p_r, (puzzle, sid, pid) to p_h, (puzzle, sid, pid') to p_s and inform $\mathcal{S}im$.
- Set $\tau_0 = (\eta' = (\gamma_1.s + 1, \gamma_1.cash(p_s) -= \epsilon, \gamma_1.cash(p_h) += \epsilon, \mathrm{Hash}(\eta_s)), time = now + t)$, where t is a constant period.
- Send τ_0 to p_h and p_s.
- Record $(pk_0, pid, pid', (\gamma_2.id, \eta), \cdot)$ into $\mathbb{T}$.

Upon receiving (PSol, p_r, pid') from p_s, do the following:
- If $(pk_0, \cdot, pid', \cdot) \notin \mathbb{P}$, abort.
- Send (PSolReq, sid, p_s, pid') to p_h and $\mathcal{S}im$.
- Receive (PSolRes, sid, b) from p_h.
- If $b = \perp$, abort; otherwise, update $(pk_0, \cdot, pid', \top)$ in $\mathbb{P}$.
- Send (solved, pid', $\top$) to p_s, p_r, and $\mathcal{S}im$.

Wait for (open, sid, pid) from p_r, do the following:
- If $(pk_0, pid, \cdot, b) \notin \mathbb{P}$ or $b = \perp$, send (opened, sid, $\perp$, pid) to p_r, remove$(pk_0, pid, pid', (\gamma_2.id, \eta), \cdot)$ terminate.
- Otherwise, send (opened, sid, $\top$, pid) to p_r.
- Send (update. sid, $\gamma_2.id$, η) to $\mathcal{S}im$.
- Send (update. sid, $\gamma_1.id$, $\eta' = (\gamma_1.s + 1, \gamma_1.cash(p_s) -= \epsilon, \gamma_1.cash(p_h) += \epsilon, \mathrm{Hash}(\eta'_s))$) to $\mathcal{S}im$.
- Update $(pk_0, pid, pid', (\gamma_2.id, \eta), (\gamma_1.id, \eta'))$ into $\mathbb{T}$, $\gamma_1.s = \gamma_1.s + 1$, and $\gamma_2.s = \gamma_2.s + 1$.
- Append η to $\gamma_2.st$ and η' to $\gamma_1.st$

Close: Upon receiving (closeChannel, sid, γ, η) from party p, do the following:
- Send (close, sid, $\gamma.id$, η) to the simulator $\mathcal{S}im$.
- Receive (closed, sid, b) from $\mathcal{S}im$.
- If $b = \perp$, terminate.
- Send (closed, sid, $\gamma.id$) to $\gamma.users$.

Audit: Upon receiving (audit, sid, pk_0) from p, do the following:
- Retrieve the record (trapdoor, pk_0, p_o). If the record does not exist or $p_o \neq p$, abort.
- Send (auditReq, sid, pk_0) to p_h and $\mathcal{S}im$.
- Receive (auditRes, sid, p_i, TxSet = $\{\gamma_1 \cdots, \gamma_m\}$) from p_h.
- Send (read sid, γ_i) to $\mathcal{S}im$ for reading the state of γ_i, receive γ'_i.
- Check if $\gamma'_i.closed = 1$, continues; Otherwise, abort.
- Check if there is a completed hash chain from the hub-involved payment $\gamma_i.\eta_0$ to the submitted payment $\gamma'_i.\eta_s$. If fails, abort and set flag = 0; otherwise, set flag = 1.
- If $p_o = p$, (trapdoor, pk_0, p_o) $\in \mathbb{D}$, $\eta_k \in \gamma_i.st$, and $\eta_l \in \gamma_j.st$, retrieve all $(pk_0, pid, pid', (\gamma_i.id, \eta_k), (\gamma_j.id, \eta_l))$ from $\mathbb{T}$ and add it to $\mathbb{A}$. If exist $\eta_l \notin \gamma_j.st$, set flag = 0.
- Output $\mathbb{A}$, flag.

B. Security Proof

In the following, we give a proof of Theorem VI.1. Note that we focus on the security analysis of the audit and pay phases and omit the security analysis of the open and close phases. Because the open and close phases are identical to the other payment channel schemes [1–3]. Here, the proof is constructed upon a series of hybrids built from the original protocol.

Hybrid $\mathcal{H}_0$: It is identical to the formal protocol described in the Construction of AuditPCH Section.

Simulator Sim for pay

Case: $\mathbf{p_h}$ is corrupted

1. Upon receiving (pay, sid, p_r, pk_0) from p_s, proceed as follows:
- Sample a random number r a identification oid.
- Commit tid: $(r, com) \leftarrow \Pi_{com}.\mathrm{Prove}(pp, (r, oid))$.
- Compute a proof of the commitment:
$$\pi \leftarrow \Pi_{MP}.\mathrm{Prove}(com, (oid, r))$$
- Send (regReq, sid, com, π) to p_h instead of p_s.

2. Upon receiving (registered, sid, σ^*) from p_h, proceed as follows:
- Open and randomize the blind signature σ^*as σ
- Check if σ is correct:
  - If yes, store (oid, σ), and send it to p_r.
  - If no, abort in the name of p_s.

3. Upon sending (PGen, oid′, sid) to $\mathcal{F}_{\mathsf{AuditPCH}}$, proceed as follows:
- Retrieve (oid′, σ).
- Generate the signature $\sigma_{R\to H}$ of transaction $[\tau_{H\to R}]$, where includes the hash of last transaction in the channel
- Send (PGen, sid, $\sigma, \sigma_{R\to H}$, oid′, $[\tau_{H\to R}]$) to p_h in the name of p_r.

4. Upon receiving (puzzle, sid, pz, π, $\tilde{\sigma}_{H\to R}$, $[\tau_{H\to R}]$) from p_h, proceed as follows:
- Verify the puzzle pz and the adaptor signature $\tilde{\sigma}_{H\to R}$ via executing $\Pi_{lrp}.\mathrm{PVerify}$ and $\Pi_{ad}.\mathrm{PVrfy}$.
- If all verification succeeds,:
  - Compute the randomizable puzzle pz′:
$$(pz', \pi', r_1) \leftarrow \Pi_{lrp}.\mathrm{PRand}(pp, pk_1, pz, \pi)$$
where $pz = (\alpha, \beta, \gamma)$
  - Send pz′ to p_s in the name of p_r.
- Otherwise, abort.

5. Upon p_r sending (PSol, p_r, pid) to $\mathcal{F}_{\mathsf{AuditPCH}}$, proceed as follows:
- Retrieve the puzzle pz′.
- Randomize the puzzle pz′ as
$$(pz'', \pi'', r_2) \leftarrow \Pi_{lrp}.\mathrm{PRand}(pp, pk_1, pz', \pi')$$
- Sign a pre-signature of the transaction $[\tau_{S\to H}]$,
$\tilde{\sigma}_{S\to H} = \Pi_{ad}.\mathrm{PSign}([\tau_{S\to H}], sk_{ad}, \gamma'')$, $[\tau_{S\to H}]$ where includes the hash of last transaction in the channel.
- Send (PSolReq, sid, pz′′,$\tilde{\sigma}_{S\to H}$,$[\tau_{S\to H}]$) to p_h in the name of p_s.

6. Upon receiving (solve, sid, $\sigma_{S\to H}$) from p_h, proceed as follows:
- Extract the solution $\aleph'' = \Pi_{ad}.\mathrm{Ext}(\tilde{\sigma}_{S\to H}, \sigma_{S\to H}, \gamma'')$.
- If $\aleph'' = \perp$, terminate.
- Otherwise, compute $\aleph' = \aleph'' - r_2$, and sends (solved, sid, $\aleph'$) to p_r in the name of p_s.

Case: $\mathbf{p_r}$ is corrupted

1. Upon receiving (PGen, sid, $\sigma, \sigma_{R\to H}$, oid′, $[\tau_{H\to R}]$) from p_r, proceed as follows:
- If oid′ ∈ 𝕆 or the verification of σ is failed, abort.
- Otherwise, if p_t sends (PGenRes, sid, ⊤) to $\mathcal{F}_{\mathsf{AuditPCH}}$, add oid to 𝕆 , and generate a key pair (pk_1, sk_1) via $\Pi_{lrp}.\mathrm{KGen}$ and use this pair to generate the following parameters:
$$(pz, \pi) \leftarrow \Pi_{lrp}.\mathrm{PGen}(pp, pk_1, \aleph).$$
- Sign a pre-signature of the transaction
$\tilde{\sigma}_{H\to R} = \Pi_{ad}.\mathrm{PSign}([\tau_{H\to R}], sk_{ad}, \gamma)$.
- Send (puzzle, sid, pz, π, $\tilde{\sigma}_{H\to R}$, $[\tau_{H\to R}]$) in the name of p_h.

Case: $\mathbf{p_s}$ is corrupted

1. Upon receiving (regReq, sid, com, π) from p_s, proceed as follows:
- Check if the proof π is correct:
  - If yes, after sending (regRes, sid, ⊤) to $\mathcal{F}_{\mathsf{AuditPCH}}$, compute a blind signature σ^* of the commitment com and send (registered, sid, σ^*) to p_s instead of p_h.
  - Otherwise, abort.

2. Upon receiving (PSolReq, sid, pz′′,$\tilde{\sigma}_{S\to H}$,$[\tau_{S\to H}]$) from p_s, proceed as follows:
- Extract a solution from pz′′:
$$\aleph'' = \Pi_{lrp}.\mathrm{PSlove}(pk_1, sk_1, pz'').$$
- Extract the valid signature $\sigma_{s\to H} = \Pi_{ad}.\mathrm{Adapt}(\aleph'', \tilde{\sigma}_{S\to H})$.
- Generate a signature $\sigma_{H\to S}$ of transaction $[\tau_{S\to H}]$.
- If the verification of $\sigma_{s\to H}$ is correct, after p_h sending (PSolRes, sid, ⊤) to $\mathcal{F}_{\mathsf{AuditPCH}}$, send (solve, sid, $\sigma_{S\to H}$) to p_s in the name of p_h.
- Otherwise, abort.

Simulator Sim for audit

Case: $\mathbf{p_h}$ is corrupted

1. Upon receiving (auditRes, sid, pk_0, p_i, TxSet) from p_h, proceed as follows:
- TxSet is $\{\ldots, c_i = (\sigma_i, \tilde{\sigma}_i, pz_i, \pi_i, \tau, p_i, p_i'), \ldots\}$
- Check if all π_i is valid, if not, abort.
- Check if and the hash chain of p_i in the TxSet is completed, if not, set flag = 0; otherwise, flag=1
- Check if $\{\sigma_i\}$ and $\{\tilde{\sigma}_i\}$ are correct.
- $(\aleph_i + r_{i1} + r_{i2}) \leftarrow \mathbf{Ext}(\sigma_i, \tilde{\sigma}_i, \gamma_i'')$
- Check if $\gamma_i'' = g^{\aleph_i + r_{i1} + r_{i2}}$, else output ⊥
- $\aleph \leftarrow \mathbf{Ext}(\sigma_{H\to R}, \tilde{\sigma}_{H\to R}, \gamma)$
- For each c_i includes p_i
- Collect all element pair (c_i, c_j) into 𝔸, startisfy $\mathbf{PLink}(pp, sk_0, pz_i, pz_j) = 1$;
- Output 𝔸,flag

Fig. 1: The simulator construction.

Hybrid $\mathcal{H}_1$: All the calls to the commitment scheme take place with the corresponding functionality $\mathcal{F}_{com}$ of the commitment scheme (the details are shown in [4]).

Hybrid $\mathcal{H}_2$: All the calls to the commitment scheme take place with the corresponding functionality $\mathcal{F}_{NIZK}$ of the NIZK scheme (the details are shown in [5]).

Hybrid $\mathcal{H}_3$: All the calls to the ElGamal scheme take place with the corresponding ideal functionality $\mathcal{F}_{enc}$ of the ElGamal scheme (the details are shown in [4]).

Hybrid $\mathcal{H}_4$: For an honest hub p_h and sender p_s, and a corrupted receiver p_r, check if there are some pairs (oid, σ) returned from p_r before executing the channel authentication step, such that the hub p_h does not abort during the following operations. The experiment will be aborted and output fail.

Hybrid $\mathcal{H}_5$: For a honest hub p_h and sender p_s, and a corrupted receiver p_r with puzzles (pz, pz'') and pre-signatures $(\tilde{\sigma}_{H\to R}, \tilde{\sigma}_{S\to H})$ generated by executing the puzzle generation stage, if p_r is capable of outputting signature $\sigma_{H\to R}$ such that signatures verification succeeds, before a solution $\aleph'$ is generated. The experiment will be aborted and output fail.

Hybrid $\mathcal{H}_6$: For an honest sender p_s and receiver p_r, if the parties do not terminate when a solution $\aleph'$ is outputted from the execution of the puzzle solution stage cannot be used to generate a valid signature $\sigma_{H\to R}$. The experiment will be aborted and output fail.

1) UC Simulator: This section discusses the construction of simulators in the previous hybrids. The simulator *Sim* takes responsibility for simulating the honest parties and interacting with the environment $\mathcal{E}$ in the name of honest parties. Note that the open and close operations in $\mathcal{F}_{\mathsf{AuditPCH}}$ are identical to the standard payment channel model [2, 6]. Thus our simulator will call the generalized payment channel ideal functionality $\mathcal{F}_{\mathsf{GC}}$ defined in [2, 6] to handle these operations. To simplify, we here omit the description of the construction of the simulator in the open and closed operations. The details of simulators are shown in Figure 1.

2) Simulation Proof: Based on the above hybrids, we prove the environment $\mathcal{E}$ cannot distinguish the neighbor experiments and finally state that the protocol Π_{AuditPCH} is indistinguishable from $\mathcal{F}_{\mathsf{AuditPCH}}$.

Lemma III.1. *For all environment* $\mathcal{E}$, $\mathsf{EXEC}_{\mathcal{H}_0,\mathcal{A},\mathcal{E}} \approx \mathsf{EXEC}_{\mathcal{H}_1,\mathcal{A},\mathcal{E}}$.

Proof. The proof holds since the chosen commitment scheme is secure. □

Lemma III.2. *For all environment* $\mathcal{E}$, $\mathsf{EXEC}_{\mathcal{H}_1,\mathcal{A},\mathcal{E}} \approx \mathsf{EXEC}_{\mathcal{H}_2,\mathcal{A},\mathcal{E}}$.

Proof(sketch). The proof holds since the chosen non-interactive zero-knowledge schemes are secure. □

Lemma III.3. *For all environment* $\mathcal{E}$, $\mathsf{EXEC}_{\mathcal{H}_2,\mathcal{A},\mathcal{E}} \approx \mathsf{EXEC}_{\mathcal{H}_3,\mathcal{A},\mathcal{E}}$.

Proof(sketch). The proof holds since the chosen ElGamal scheme is secure. □

Lemma III.4. *For all environment* $\mathcal{E}$, $\mathsf{EXEC}_{\mathcal{H}_3,\mathcal{A},\mathcal{E}} \approx \mathsf{EXEC}_{\mathcal{H}_4,\mathcal{A},\mathcal{E}}$.

Proof(sketch). Here, we can see that the environment $\mathcal{E}$ can distinguish the hybrids $\mathcal{H}_3$ and $\mathcal{H}_4$ if the hybrid $\mathcal{H}_4$ outputs fail. Thus, if the hybrid $\mathcal{H}_4$ outputs fail with negligible probability, we can state that $\mathcal{H}_3$ is indistinguishable from $\mathcal{H}_4$. Specifically, we can reduce the probability of the signature scheme corresponding to the signature σ. Note that we adopt the signature scheme as the same as the A^2L^+'s. Based on the reduction in A^2L^+ (details in Lemma 3 [7]), we can draw the following conclusion: $\Pr[\mathsf{fail}—\mathcal{H}_3] \leq \mathsf{negl}$. Finally, we can observe that $\mathcal{H}_3$ is identical to $\mathcal{H}_4$ so that the lemma holds. □

Lemma III.5. *For all environment* $\mathcal{E}$, $\mathsf{EXEC}_{\mathcal{H}_4,\mathcal{A},\mathcal{E}} \approx \mathsf{EXEC}_{\mathcal{H}_5,\mathcal{A},\mathcal{E}}$.

Proof(sketch). We denote fail as an event that abort happens in $\mathcal{H}_5$ but not in $\mathcal{H}_4$. If this event happens, the environment $\mathcal{E}$ distinguishes which hybrid it interacts with. To state the lemma holds, we need to prove that the following formula holds: $\Pr[\mathsf{fail}—\mathcal{H}_4] \leq \mathsf{negl}(\lambda)$. Here, we utilize proof by contradiction to show that the event $[\mathsf{fail}—\mathcal{H}_4]$ happens in a negligible probability. First, we make an assumption: $\Pr[\mathsf{fail}—\mathcal{H}_4] \geq 1/\mathsf{poly}(\lambda)$. We reduce it to the security of the LRP scheme, the unforgeability of the adaptor signature scheme, and the hardness of the relation R (where $(Y{=}g^y, y) \in R$). As for the LRP scheme, we have demonstrated its security in section II. It implies that the probability of breaking the security of the LRP scheme is negligible. Therefore, the event fail will only happen if the signature can be forged. Under the unforgeability of the adaptor signature scheme [7], the adversaries only with a public key and some signatures generated before cannot forge a new signature with a non-negligible probability. This fact is contradicts the initial assumption: $\Pr[\mathsf{fail}—\mathcal{H}_4] \geq 1/\mathsf{poly}(\lambda)$ does not hold. Thus, we state that $\Pr[\mathsf{fail}—\mathcal{H}_4] \leq \mathsf{negl}$.

Lemma III.6. *For all environment* $\mathcal{E}$, $\mathsf{EXEC}_{\mathcal{H}_5,\mathcal{A},\mathcal{E}} \approx \mathsf{EXEC}_{\mathcal{H}_6,\mathcal{A},\mathcal{E}}$.

Proof. We denote fail as an event that abort happens in $\mathcal{H}_6$ but not in $\mathcal{H}_5$. If this event happens, the environment $\mathcal{E}$ distinguishes which hybrid it interacts with. Two conditions can trigger such an event. The first condition is that a corrupted hub p_h generates a valid pre-signature $\widetilde{\sigma}_{H\to R}$ but cannot collect a valid signature over $\sigma_{H\to R}$ when the receiver executes the Adapt algorithm. The second condition is that a corrupted hub p_h produces a valid signature $\sigma_{S\to H}$ to the sender but the sender cannot extract a valid witness from the valid signature $\sigma_{S\to H}$ and its pre-signature $\widetilde{\sigma}_{S\to H}$. Based on the security proof in A^2L, we know that the adaptor signature scheme satisfies the *pre-signature adaptability* and *witness extractability*properties. Referring to the analysis in A^2L, we can reduce the probability of the event fail happening to the witness extractability and pre-signature adaptability of the adaptor signature (details in Lemma 5 of A^2L [7]). Thus, we know that $\Pr[\mathsf{fail}—\mathcal{H}_5] \leq \mathsf{negl}$. □

Lemma III.7. *For all environment* $\mathcal{E}$,

$$\mathsf{EXEC}_{\mathcal{H}_6,\mathcal{A},\mathcal{E}} \approx \mathsf{EXEC}_{\mathcal{H}_{\mathsf{AuditPCH}},\mathcal{A},\mathcal{E}}.$$

Proof. The hybrid $\mathcal{H}_6$ is identical to $\mathcal{H}_{\mathsf{AuditPCH}}$. Therefore, the two experiments are indistinguishable. □

Finally, based on the above lemmas, we conclude that the Theorem VI.1 holds.

References

[1] L. Aumayr, M. Maffei, O. Ersoy, A. Erwig, S. Faust, S. Riahi, K. Hostáková, and P. Moreno-Sanchez, "Bitcoin-compatible virtual channels," in *IEEE SP*, 2021, pp. 901–918.

[2] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostáková, M. Maffei, P. Moreno-Sanchez, and S. Riahi, "Generalized channels from limited blockchain scripts and adaptor signatures," in *ASIACRYPT*. Springer, 2021, pp. 635–664.

[3] L. Aumayr, P. Moreno-Sanchez, A. Kate, and M. Maffei, "Blitz: Secure multi-hop payments without two-phase commits," in *USENIX Security*. USENIX Association, 2021, pp. 4043–4060.

[4] R. Canetti, Y. Dodis, R. Pass, and S. Walfish, "Universally composable security with global setup," in *TCC*. Springer, 2007, pp. 61–85.

[5] J. Groth, R. Ostrovsky, and A. Sahai, "New techniques for noninteractive zero-knowledge," *JACM*, vol. 59, no. 3, pp. 1–35, 2012.

[6] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostakova, M. Maffei, P. Moreno-Sanchez, and S. Riahi, "Generalized bitcoin-compatible channels." *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 476, 2020.

[7] E. Tairi, P. Moreno-Sanchez, and M. Maffei, "A 2 l: Anonymous atomic locks for scalability in payment channel hubs," in *IEEE SP*, 2021, pp. 1834–1851.