AUPCH: Auditable Unlinkable Payment Channel Hubs*

Pedro Moreno-Sanchez[1,3], Mohsen Minaei[1], Srinivasan Raghuraman[1,2], Panagiotis Chatzigiannis[1], and Duc V. Le[1]

[1] Visa Research
[2] MIT
[3] IMDEA Software Institute, MPI-SP

Abstract. Cryptocurrencies, which have gained significant adoption in recent years, face ongoing challenges in scalability and privacy. Payment Channel Hubs (PCHs) constitute a solution to both issues by shifting transactions off the public ledger. Various PCH constructions have been proposed, offering different degrees of unlinkability, efficiency, and interoperability. However, regulatory compliance remains a significant concern, particularly under emerging frameworks like the EU's Markets in Crypto-Assets (MiCA) regulation and FATF Travel Rule requirements. This work addresses a gap in existing PCH constructions: the lack of regulatory-compliant auditability mechanisms. While concurrent work AuditPCH attempts to address this challenge, it suffers from fundamental limitations, including reliance on channel closures for auditing, vulnerability to unilateral de-anonymization by the hub, and lack of formal security guarantees for the auditing process. Our approach fundamentally differs by providing targeted, non-disruptive auditability that allows auditability for high-risk payments while preserving unlinkability for the rest. To achieve this, we present Verifiable Linkable Randomizable Puzzles (`VLRP`), a new cryptographic primitive that enables a party to commit to a secret using two distinct keys: a verifiability key (VK) and an auditability key (AK). This primitive provides (i) verifiability that the owner of the VK issued the commitment, (ii) the ability to randomize the commitment to ensure unlinkability, even for the owner of the VK, while still allowing traceability using the AK, and (iii) collaborative auditing that prevents unilateral de-anonymization.
We then present Auditable Unlinkable Payment Channel Hubs, AUPCH, a PCH built on `VLRP` that offers auditability guarantees with stronger security guarantees than existing approaches. AUPCH provides modular integration with existing PCH frameworks (A^2L, BlindHub), operates without requiring channel closures, and ensures that auditing requires collaboration between hub and auditing agent, preventing abuse by either party alone. Crucially, our approach acts as a wrapper around existing PCH implementations, requiring only replacing randomizable puzzle calls with VLRP calls, a minimal change that dramatically reduces deployment complexity compared to building new systems from scratch.

* The extended abstract of this work appeared at the *9th International Workshop on Cryptocurrencies and Blockchain Technology (CBT) 2025*

1 Introduction

Cryptocurrencies have gained prominence as decentralized and publicly verifiable payment systems, attracting interest from banks, leading IT companies, and payment providers [10,13]. Among the off-chain solutions to address their inherent scalability challenges, payment channels [2,14,20] have seen significant academic and industry adoption. A payment channel allows two users to perform multiple off-chain transactions, settling them with just two on-chain transactions.

To overcome the limitation of two-party channels, Payment Channel Hubs (PCHs) were developed. They use a central intermediary (hub) or tumbler to facilitate payments between any two users [11,25,28]. However, this reliance on a single intermediary introduces several concerns: (i) *security* (the hub could steal funds); (ii) *unlinkability* (the hub could link payers to payees); (iii) *value privacy* (the hub could learn transaction amounts); (iv) *interoperability* with different cryptocurrencies; and (v) *amount flexibility* (payments can be of arbitrary amounts). Over the years, various PCH constructions have been proposed to address these issues with different trade-offs (c.f. Table 1). However, *auditability*, that is, the possibility to link a payer and payee in a transaction, is missing from existing PCH constructions, except for AuditPCH, which we overview next.

Comparison with AuditPCH. A recent PCH construction by Li *et al.* [16] also accounts for auditability. However, AuditPCH suffers from fundamental limitations that restrict its practical applicability and security guarantees. Our AUPCH approach differs in several key aspects:

- **Operational Model:** AuditPCH performs auditing through a post-hoc channel closure mechanism. In contrast, AUPCH operates on active channels, enabling auditing without channel termination, which is particularly relevant for use cases requiring pre-transaction compliance verification.
- **Security Model:** We employ a trust model with an explicit separation between the hub and an auditability agent, using nested encryption that requires collaboration for de-anonymization. This prevents unilateral action by the hub, whereas AuditPCH assumes a semi-honest hub model. Our work also provides formal security definitions with complete cryptographic games and proofs.
- **Modularity Approach:** Our design acts as a modular wrapper around existing PCH constructions (A^2L [24], BlindHub [22] or unlinkable/interoperable payment channels [18]). It requires only minimal changes to underlying protocols, essentially replacing randomizable puzzle calls with VLRP calls, while AuditPCH introduces a more integrated design requiring substantial modifications.
- **Compliance Requirements:** Our approach enables targeted auditing of pre-flagged high-risk transactions, which aligns with emerging regulatory frameworks like FATF Travel Rule and EU MiCA requirements for real-time compliance verification, while AuditPCH focuses on post-hoc forensic analysis.

1.1 Goal of this Work

A significant limitation in existing state-of-the-art PCH constructions is the absence of mechanisms for *auditability*. This gap is becoming increasingly critical as regulatory frameworks evolve to mandate traceability for combating illicit financial activities, such as money laundering (AML) and counter-terrorist financing (CTF). Consequently, PCHs require an auditability mechanism to permit the detection of the origin or destination of funds in specific transfers.

The regulatory landscape now imposes stringent requirements on cryptocurrency service providers. The Financial Action Task Force (FATF) Travel Rule [5], for instance, obligates virtual asset service providers (VASPs) to collect and transmit originator and beneficiary information for transactions exceeding certain thresholds. This has direct implications for PCH operators, who must possess the technical means to trace transactions [27] when legally required. Similarly, recent European legislation explicitly identifies technologies that enhance anonymity, such as mixers [15,26] or tumblers [19], as high-risk factors and mandates enhanced due diligence to determine the provenance of crypto-assets [4]. Furthermore, the EU MiCA regulation [4] requires that crypto-asset service providers implement robust systems to prevent market abuse. Non-compliance with these mandates exposes service providers to substantial financial penalties, elevating regulatory adherence from an operational consideration to a critical design requirement.

The objective of this work is to design a PCH architecture that integrates these necessary auditability guarantees. This goal introduces a fundamental tension between auditability, which requires the ability to link transactions, and unlinkability, which is a core privacy-preserving objective of PCHs. We address this conflict by observing that a stringent auditability is typically only required for a subset of transactions identified as high-risk. We therefore propose a design that enables selective auditability for flagged payments, while rigorously preserving the unlinkability of all other transactions.

1.2 Contributions of this Work

First, we introduce *verifiable linkable randomizable puzzles*, `VLRP`, a novel cryptographic primitive (Section 4). A `VLRP` puzzle, Z, commits to a secret with respect to a verifiability key and an auditability key. The puzzle can be randomized to ensure unlinkability, but the auditability key allows it to be traced back to the original. Critically, our construction employs a nested encryption approach where auditing requires collaboration between the hub and an auditing agent, preventing unilateral de-anonymization. We formalize the `VLRP` primitive with security, unlinkability, and auditability notions and provide a construction with cryptographic proofs.

As a second contribution, we design Auditable Unlinkable Payment Channel Hubs, AUPCH, a PCH providing improved auditability with security guarantees than existing approaches (Section 5). AUPCH introduces an independent

[4] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32023R1113

Table 1: Comparison of AUPCH with state-of-the-art PCH. The bottom rows show how existing PCH systems can be enhanced with auditability using our VLRP primitive.

	Security	Unlinkability	Value Privacy	Interoperability	Amount Flexibility	Auditability
BOLT [8]	●	●	●	○ (Blind signatures, Script modifications)	●	○
Perun [3]	●	○	○	○ (Ethereum Virtual Machine)	●	○
Teechain [17]	●	●	●	○ (Trusted Hardware)	●	○
TumbleBit [11]	●	●	N.A.[a]	○ (HTLC-based currencies)	○	○
A2L [25]	●	●	N.A.[a]	● (Digital Signatures and timelocks)	○	○
A2L $^{+}$,A2L $^{\mathsf{UC}}$ [7]	●	●	N.A.[a]	● (Digital Signatures and timelocks)	○	○
BlindHub [21]	●	●	●	● (Digital Signatures and timelocks)	●	○
Accio [6]	●	●	●	○(Ethereum Virtual Machine)	●	○
AuditPCH [16]	●	●	●	○ (Requires channel closure)	●	◐[b]
AUPCH (VLRP +A2L)	●	●	N.A.[a]	● (Digital Signatures and timelocks)	○	●
AUPCH (VLRP + BlindHub)	●	●	●	● (Digital Signatures and timelocks)	●	●

[a] N.A.: not applicable since the amount in these protocols is fixed.

[b] Post-hoc auditability: designed for forensic analysis after channel closure, assumes semi-honest hub model.

auditability agent, responsible for flagging high-risk payments. Users can transact via the hub without the direct involvement of the agent, ensuring normal operations are not disrupted. When a transfer needs to be audited, the agent and hub execute a collaborative protocol to link the puzzles on active channels without requiring closure.

AUPCH follows the puzzle promise, puzzle solve paradigm introduced in [11, 25] and formalized in [7]. By replacing the simpler randomizable puzzles in protocols like A2L [25] and BlindHub [21] with our VLRP puzzles, AUPCH inherits their privacy and interoperability guarantees while providing auditability. This wrapper-like design requires only minimal code changes. Our performance evaluation shows that AUPCH provides auditability with a small computation and storage overhead.

2 Preliminaries

In this section, we first introduce the notation used in this work. We then briefly overview the notion of randomizable puzzles, used as a building block in previous PCH works [7, 21, 25], and other cryptographic primitives required in this work.

Notation. We denote the security parameter by $n \in \mathbb{N}$, by which each cryptographic scheme and adversary is parameterized. We denote by $\mathsf{negl}(n)$ a *negligible* function. A function $f : \mathbb{N} \to \mathbb{R}$ is negligible if its absolute value is smaller than the inverse of any polynomial (i.e., if $\forall d\ \exists k_0\ \forall n \geq k_0 : |\mathsf{negl}(n)| \leq 1/n^d$). We denote by $x \leftarrow_{\$} \mathcal{X}$ the uniform sampling of the variable x from the set $\mathcal{X}$. We write $x \leftarrow \mathcal{A}(y)$ to denote that a probabilistic polynomial time (PPT) algorithm $\mathcal{A}$ on input y outputs x. If $\mathcal{A}$ is a deterministic polynomial time (DPT) algorithm, we use the notation $x := \mathcal{A}(y)$. We use the notation $s \leftarrow s_1 + s_2$ for the assignment of computation results. We use the notation $\sigma := (\sigma_1, \sigma_2)$ for parsing a tuple σ composed of two elements σ_1 and σ_2. We use the dot notation to access the elements of a tuple (e.g., we denote by $\sigma.\sigma_1$ the element σ_1 of σ).

Randomizable Puzzles Scheme. A randomizable puzzle scheme RP is a tuple of algorithms $\mathsf{RP} := (\mathsf{PSetup}, \mathsf{PGen}, \mathsf{PSolve}, \mathsf{PRand})$, where $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PSetup}(1^n)$ is the setup algorithm; $Z \leftarrow \mathsf{PGen}(\mathsf{pp}, \zeta)$ is the puzzle generation algorithm; $\zeta \leftarrow \mathsf{PSolve}(\mathsf{td}, Z)$ is the algorithm to solve a puzzle; and $(Z', r) \leftarrow \mathsf{PRand}(\mathsf{pp}, Z)$ is the puzzle randomization algorithm. A randomizable puzzle scheme must satisfy *randomizability*, *security* and *privacy*.

Digital Signature Scheme. A digital signature DS is a tuple of algorithms $\mathsf{DS} := (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$, where $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^n)$ is the key generation algorithm; $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$ is the signing algorithm on input the signing key sk and a message m; and $\{0,1\} \leftarrow \mathsf{Vrfy}(\mathsf{vk}, m, \sigma)$ is the verification algorithm. A digital signature scheme should satisfy *existential unforgeability under an adaptative chosen-message attack*.

Adaptor Signature Scheme. An *adaptor signature scheme* AS is a tuple of algorithms $\mathsf{AS} := (\mathsf{KeyGen}, \mathsf{PreSign}, \mathsf{PreVrfy}, \mathsf{Adapt}, \mathsf{Extract})$ defined with respect to a hard relation R and a digital signature scheme DS. For every statement/witness pair $(x, w) \in \mathsf{R}$, key pair $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^n)$ and a message m, we have that $\hat{\sigma} \leftarrow \mathsf{PreSign}(\mathsf{sk}, m, x)$ is a pre-signature; and $\sigma \leftarrow \mathsf{Adapt}(\hat{\sigma}, w)$ is a valid signature; and (pre-)verification holds under vk and m for $\hat{\sigma}$ and σ, respectively. Furthermore, it holds that $w \leftarrow \mathsf{Extract}(\hat{\sigma}, \sigma, x)$. An adaptor signature scheme should satisfy the notions of *(unique) extractability*, *unlinkability* and *pre-verify soundness*.

Public Key Encryption Scheme. A public key encryption scheme PKE is a tuple of algorithms $\mathsf{PKE} := (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, where $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{KeyGen}(1^n)$ is the key generation algorithm; $ct \leftarrow \mathsf{Enc}(\mathsf{ek}, m)$ is the encryption algorithm on input a public key ek and a message m; and $\{m, \perp\} \leftarrow \mathsf{Dec}(\mathsf{dk}, ct)$ is the decryption algorithm. A public key encryption scheme should satisfy *indistinguishability under chosen-plaintext attacks*.

Non-Interactive Zero-Knowledge Arguments. A non-interactive zero-knowledge argument system NIZK consists of three algorithms $\mathsf{NIZK} := (\mathsf{CrsGen}, \mathsf{Prove}, \mathsf{Verify})$, where CrsGen is the public parameter (i.e., common reference string) generation algorithm; $\pi \leftarrow \mathsf{Prove}(crs, x, w)$ is the prover algorithm for a statement x and a witness w; and $\{0,1\} \leftarrow \mathsf{Verify}(crs, x, \pi)$ is the verification algorithm. A NIZK argument system allows a prover to convince a verifier, using a proof π, about the existence of a witness w for a statement x without revealing any information apart from the fact that it knows the witness w. We require the NIZK argument system to satisfy the usual properties of *completeness*, *computational soundness* and *computational zero-knowledge*.

We defer the detailed description of the building blocks and their properties to Appendix A.

3 Solution Overview

We outline our solution to integrate auditability into privacy-preserving coin mixing protocols for payment channel hubs (PCHs). We begin with the established concept of randomizable puzzles, which are central to unlinkable transactions, and then detail our extensions to support selective, auditable transactions.

3.1 Coin Mixing Protocol

A coin mixing protocol, as introduced in [11,25], ensures transaction unlinkability through payment channel hubs. The protocol allows a sender (Alice) to send a payment to a receiver (Bob) through the Hub without revealing the connection between them, thus ensuring that transactions remain unlinkable. Refer to Fig. 1 for an illustrative description. The protocol works through four key steps:

1. **Puzzle Creation:** Bob interacts with the Hub to engage in a *puzzle promise protocol*. The Hub generates and sends Bob a puzzle, denoted as Z, which contains a hidden solution ζ. The Hub also provides a pre-signature $\hat{\sigma}_{HB}$ on the transfer from the Hub to Bob, which requires the solution ζ to be completed.

2. **Puzzle Randomization:** Bob forwards the puzzle Z to Alice. To ensure privacy against the Hub, Alice applies her own randomness to the puzzle, transforming it into a new puzzle Z'.

3. **Puzzle Solving:** Alice submits the randomized puzzle Z' to the Hub. Even though the Hub has never seen Z' before, it can still solve it for the randomized solution ζ' using its trapdoor. By completing the payment from Alice, the Hub reveals ζ' to her.

4. **Solution Transfer:** Alice derandomizes the solution ζ' to obtain the original solution ζ and forwards it to Bob. Bob can then complete the payment from the Hub.

The protocol meets specific security and privacy requirements, including unlinkability, which prevents the Hub from linking randomized puzzles back to their originals, thereby concealing payment relationships. However, this essential privacy feature creates a challenge when auditability is needed.

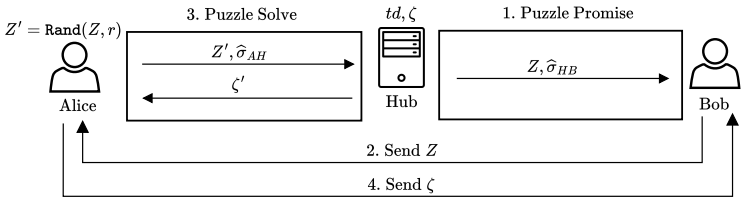

Fig. 1: Overview of coin mixing based on randomizable puzzles. The Hub's trapdoor is denoted by td and the secret, puzzle pair generated by the Hub is denoted by ζ and Z respectively.

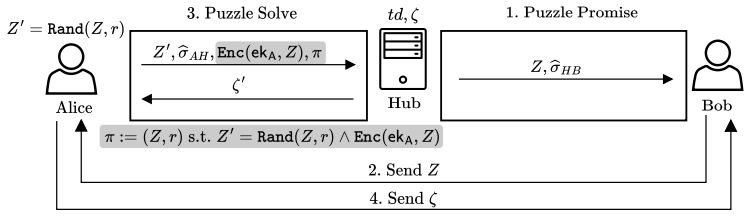

Fig. 2: Naive implementation of the auditability property. During puzzle solving, Alice includes an encryption of the randomness r used to randomize Z' and a zero-knowledge proof π attesting that it is done correctly. The highlighted elements are added to the coin mixing protocol presented in Fig. 1 to provide the auditability property.

3.2 Auditability in Coin Mixing Protocols

To introduce auditability into the coin mixing protocol, we consider an additional party called the *auditability agent*. This agent is responsible for flagging high-risk transactions and assisting the Hub in linking the involved parties, while operating independently to prevent conflicts of interest. Our design enables real-time auditing of active channels, supporting dynamic regulatory compliance requirements such as those emerging under frameworks like FATF Travel Rule and EU MiCA regulation.

The revised protocol, as shown in Fig. 2, follows the same four steps as before, with a key addition during the puzzle-solving step.

– **Step 3 - Enhanced Puzzle Solver:** Along with the randomized puzzle Z', Alice includes an encryption c of the original puzzle Z under the auditability agent's public key. She also provides a zero-knowledge proof attesting that Z' was correctly derived from the puzzle Z encrypted in c.

With these changes, if a transaction is flagged as high-risk, the auditability agent can decrypt c to retrieve the original puzzle Z, allowing the Hub to link the sender and receiver. This process relies on the assumption that the Hub and the auditability agent do not collude.

A notable vulnerability is the *mix-and-match attack* [5] and potential leakage to the auditability agent, where a malicious Alice can manipulate the process to create a false audit trail. To mitigate this, we propose two modifications illustrated in Fig. 3 and Fig. 4:

1. **Adding Authentication Tag:** During puzzle creation, the Hub includes an authentication tag τ, which is a digital signature on Z. This forces Alice to use a puzzle genuinely issued by the Hub.

[5] A malicious sender can fabricate a new puzzle-randomness pair $(\tilde{Z}, \tilde{r})$ such that the same randomized puzzle Z' can be derived from either the original puzzle Z or the fabricated puzzle $\tilde{Z}$. By encrypting $\tilde{Z}$ instead of Z and providing a valid zero-knowledge proof using the fabricated pair, the sender creates a false audit trail that prevents the Hub from correctly linking the transaction back to the intended receiver.

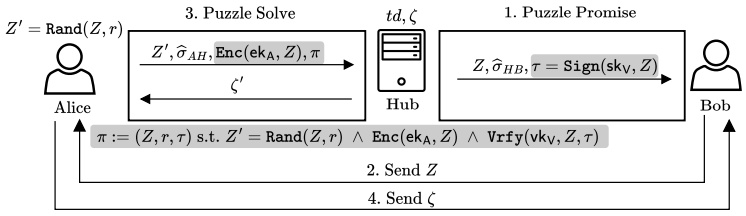

Fig. 3: Illustrative example of applying authentication tag to mitigate the mix-and-match attack. The highlighted elements are additions and modifications to the naive auditable solution provided in Fig. 2.

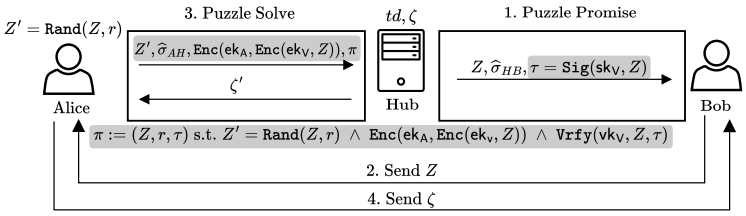

Fig. 4: Illustrative example of our approach. The highlighted elements are additions and modifications to the naive auditable solution provided in Fig. 2.

2. **Mitigating Leakage to Auditability Agent:** To prevent the auditability agent from unilaterally linking transactions, we use nested encryption. The puzzle Z is first encrypted under the Hub's encryption key and then under the auditability agent's key. This ensures that auditing requires collaboration between both parties, preserving the unlinkability of non-flagged transactions.

4 Verifiable Linkable Randomizable Puzzles

In this section, we first describe the notion of Verifiable Linkable Randomizable Puzzles (VLRP) and their security, privacy, and auditability notions. We then describe our cryptographic construction and show that it achieves security, privacy, and auditability. Finally, we discuss the performance of our construction and demonstrate its practical efficiency through detailed benchmarks.

4.1 Problem Definition

Definition 1 (Verifiable Linkable Randomizable Puzzles (VLRP)). *A verifiable linkable randomizable puzzle scheme* $\texttt{VLRP} := (\mathsf{PSetup}, \texttt{PVerifySetup}, \texttt{PAuditSetup}, \texttt{PGen}, \texttt{PVerifyTag}, \texttt{PRand}, \texttt{PVerifyRand}, \texttt{PAudit})$ *with a solution space,* $\mathcal{S}$ *(and a function* ϕ *acting on* $\mathcal{S}$*), is defined as follows:*

– $(pp, td) \leftarrow \mathsf{PSetup}(1^n)$*: is a PPT algorithm that on input a security parameter* 1^n*, outputs public parameters* pp *and a trapdoor* td*.*

– $(\mathsf{vk}_V, \mathsf{ek}_V, \mathsf{sk}_V, \mathsf{dk}_V) \leftarrow \mathtt{PVerifySetup}(1^n)$*: is a PPT algorithm that on input a security parameter* 1^n*, outputs a pair of public and private verification key* $(\mathsf{vk}_V, \mathsf{sk}_V)$ *and a pair of encryption and decryption keys* $(\mathsf{ek}_V, \mathsf{dk}_V)$*.*

– $(\mathsf{ek}_A, \mathsf{dk}_A) \leftarrow \mathtt{PAuditSetup}(1^n)$*: is a PPT algorithm that on input a security parameter* 1^n*, outputs a pair of public and private auditability key* $(\mathsf{ek}_A, \mathsf{dk}_A)$*.*

– $(Z, \tau) \leftarrow \mathtt{PGen}(pp, \mathsf{sk}_V, \zeta)$*: is a PPT algorithm that on input the public parameters* pp*, the private key* sk_V *and a solution* ζ*, outputs a puzzle* Z *and a puzzle tag* τ*.*

– $\zeta := \mathsf{PSolve}(td, Z)$*: is a DPT algorithm that on input a trapdoor* td *and puzzle* Z*, outputs the solution* ζ*.*

– $b := \mathtt{PVerifyTag}(pp, \mathsf{vk}_V, Z, \tau)$*: is a DPT algorithm that on input the public key* vk_V*, a puzzle* Z *and a puzzle tag* τ*, outputs a bit* $b \in \{0, 1\}$*.*

– $(Z', r, \rho) \leftarrow \mathtt{PRand}(pp, \mathsf{ek}_A, \mathsf{ek}_V, \mathsf{vk}_V, Z, \tau)$*: is a PPT algorithm that on input the public parameters* pp*, the public keys* ek_A*,* ek_V *and* vk_V*, a puzzle* Z *and a puzzle tag* τ*, it outputs a randomized puzzle* Z' *and randomness* r *(such that* Z' *has a solution* $\phi(\zeta, r)$*), and a puzzle auditability token* ρ*.*

– $b := \mathtt{PVerifyRand}(pp, \mathsf{ek}_A, \mathsf{ek}_V, \mathsf{vk}_V, Z', \rho)$*: is a DPT algorithm that on input the public keys* ek_A*,* ek_V*,* vk_V*, a puzzle* Z'*, and a puzzle auditability token* ρ*, it outputs a bit* $b \in \{0, 1\}$*.*

– $\delta \leftarrow \mathtt{PFlag}(\mathsf{dk}_A, \rho)$*: is a PPT algorithm that on input the private auditability key* dk_A *and a puzzle auditability token* ρ*, outputs an attestation* δ*.*

– $\{Z, \perp\} := \mathtt{PAudit}(\mathsf{dk}_V, Z', \rho, \delta)$*: is a DPT algorithm that on input a decryption key,* dk_V*, a randomized puzzle* Z'*, an auditability token* ρ *and an attestation* δ*, outputs a puzzle* Z *or bottom* $\perp$*.*

The Verifiable Linkable Randomizable Puzzles (VLRP) primitive is defined by four key properties: *correctness*, *security*, *unlinkability*, and *auditability*. *Correctness* ensures that a legitimately generated and randomized puzzle will always verify properly and yield the correct solution when solved. *Security* guarantees that a malicious party without the private key cannot create a valid puzzle-tag pair or extract a puzzle's solution without the trapdoor. *Unlinkability* provides privacy by ensuring adversaries cannot distinguish between original puzzles when given a randomized puzzle. Finally, *Auditability* ensures that attackers cannot create valid randomized puzzles that subvert the collaborative process. This guarantees that flagged transactions can always be traced back to their original puzzles by authorized parties. We defer the reader to Appendix B for the formal definitions of these properties.

4.2 Our Construction

In this section, we first describe our building blocks, followed by the construction details. Finally, we formally show that it provides the notions of security, unlinkability, and auditability and conclude this section with a performance evaluation.

Building Blocks. As defined in Section 2, we require a randomizable puzzle scheme RP, a digital signature scheme DS, a public-key encryption scheme PKE,

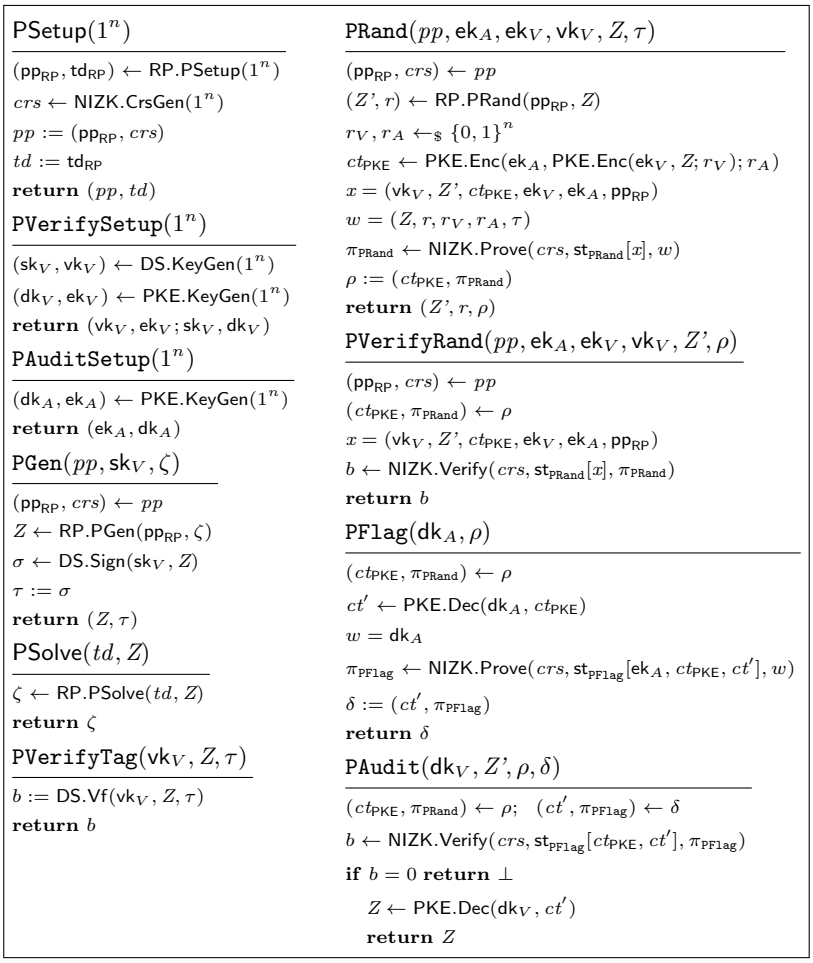

Fig. 5: Our construction for verifiable linkable randomizable puzzles (VLRP).

and a non-interactive zero-knowledge proof scheme NIZK. For the latter, we require the relations $\mathsf{st}_{\mathtt{PRand}}$ (left) and $\mathsf{st}_{\mathtt{PFlag}}$ (right) defined as follows:

$$\left\{\begin{array}{l}(\mathsf{vk}_V, Z', ct_{\mathsf{PKE}}, \mathsf{ek}_V, \mathsf{ek}_A, \mathsf{pp}_{\mathsf{RP}}; Z, r, r_V, r_A, \tau) : \\ \mathtt{PVerifyTag}(\mathsf{vk}_V, Z, \tau) \wedge \\ (Z', r) = \mathsf{RP.PRand}(\mathsf{pp}_{\mathsf{RP}}, Z) \wedge \\ ct_{\mathsf{PKE}} = \mathsf{PKE.Enc}(\mathsf{ek}_A, \mathsf{PKE.Enc}(\mathsf{ek}_V, Z; r_V); r_A)\end{array}\right\} \left\{\begin{array}{l}(\mathsf{ek}_A, ct_{\mathsf{PKE}}, ct'; \mathsf{dk}_A) : \\ ct' = \mathsf{PKE.Dec}(ct_{\mathsf{PKE}}; \mathsf{dk}_A) \wedge \\ (\mathsf{ek}_A, \mathsf{dk}_A) \leftarrow \mathsf{PKE.KeyGen}(1^n)\end{array}\right\}$$

Construction Details. We provide the details of our construction in Fig. 5. The PSetup algorithm generates the public parameters, the trapdoor required for the underlying randomizable puzzle (RP), and initializes the common reference string (crs). The algorithms PVerifySetup and PAuditSetup initialize the verifiability

and auditability keys, respectively. The PGen algorithm creates a new puzzle Z, encoding the solution ζ, and signs it with the verifiability key to produce the tag τ. The validity of the tuple (Z, τ) can be verified by PVerifyTag using the standard verification procedures. Finally, the PSolve algorithm extracts the solution embedded in a puzzle Z, using the corresponding algorithm from the underlying RP scheme.

The PRand and PVerifyRand algorithms function as follows: First, PRand randomizes the input puzzle Z using the randomization algorithm from the underlying RP. Additionally, it creates a nested encryption of Z using both ek_A and ek_V. Finally, it computes a zero-knowledge proof for the relation $\mathsf{st}_{\mathtt{PRand}}$. This proof is then verified by the PVerifyRand algorithm.

The remaining two algorithms are vital for the auditing process. When the auditability agent flags a user as high-risk, PFlag removes one encryption layer from ciphertext c and generates a zero-knowledge proof for $\mathsf{st}_{\mathtt{PFlag}}$ to ensure decryption integrity. PAudit then allows the Hub to extract the associated puzzle Z by verifying the proof and decrypting c'.

Security Analysis. Here, we state our claims and provide intuitions on the notions achieved by our construction. We defer the formal proofs to Appendix E.

Theorem 1. *Assume that the randomizable puzzle* RP *is secure and that the digital signature scheme* DS *is EUF-CMA. Then, our construction is secure.*

Theorem 2. *Assume that the non-interactive zero-knowledge argument system* NIZK *is zero-knowledge, the encryption scheme* PKE *is IND-CPA, and the randomizable puzzle* RP *is randomizable. Then, our construction is unlinkable.*

Theorem 3. *Assume that the encryption scheme* PKE *is correct, the digital signature* DS *is EUF-CMA, and the non-interactive zero-knowledge argument system* NIZK *provides knowledge-soundness. Then our construction provides auditability.*

4.3 Implementation and Performance Analysis

Parameters. We use the non-malleable version of Groth16 zkSnark [9] to instantiate our NIZK proofs. For the signature scheme, we implement EdDSA with the Babyjubjub curve. For the randomizable puzzle, we employ 2048-bit Paillier encryption due to the ease of porting it to our zkSNARK library. In our nested encryption implementation, we separately encrypt two symmetric keys using the respective public keys of the Hub and Judge, then use these keys for layered AES-CTR encryption.

Software and Hardware. The implementation is written in Typescript, and uses the Circom library [12] for writing circuits for all zero-knowledge proof statements and use circomlibjs for other cryptographic typescript implementations such as encryption and digital signatures. The experiments are executed a machine with a 3.5GHz Intel Core i7 processor with 6 cores, and 80GB RAM.

Performance. We measured the average runtimes over 100 runs for each basic operation. `PVerifySetup` takes 39.1 milliseconds to complete. `PAuditSetup` requires 19.4 milliseconds for processing. PGen consumes 202.24 milliseconds. PSolve needs 53.53 milliseconds to execute. `PVerifyTag` verification takes 84.21 milliseconds.

For operations involving zero-knowledge proofs (PRand, `PVerifyRand`, `PFlag`, and `PAudit`), we provide estimated timings based on comparable proof systems in existing implementations: PRand (30s as the number of constraints is 0.6m), `PVerifyRand` (1s), `PFlag` (2s as the number of constraints is 20k), and `PAudit` (1.5s). These NIZK timings are extrapolated from state-of-the-art zero-knowledge proof implementations with similar circuit complexity [1,23]. The size of a puzzle (i.e., one Paillier encryption) is 512 bytes, the size of a NIZK proof is 192 bytes, and the size of the nested ciphertext is approximately 784 bytes. So overall, the Hub may only need to store less than 1.5 KB of data per payment.

5 AUPCH: An Auditable Payment Channel Hub

In this section, we demonstrate how existing Payment Channel Hub (PCH) constructions can be augmented with auditability using our Verifiable Linkable Randomizable Puzzles (`VLRP`) primitive. We begin by outlining the system and threat model, then show how PCH designs based on the puzzle promise and puzzle solve paradigm can be adapted to incorporate auditability via `VLRP`, all while preserving their security and privacy properties. For completeness, our approach is applied to A2L and BlindHub, as described in Appendix C and Appendix D.

5.1 System and Threat Model

We assume a central party, called the Hub, is responsible for mediating payments between senders and receivers. Each sender and receiver has an open payment channel with the Hub. The Hub holds a verifiability key sk_V and a decryption key dk_V, with the corresponding public keys $\mathsf{vk}_V, \mathsf{ek}_V$ known to all parties.

We also introduce an auditability agent, responsible for flagging high-risk payments. The auditability agent holds an auditability key dk_A, with the corresponding public key ek_A known to all parties.

In line with prior PCH constructions, the Hub is trusted for liveness but not for security or privacy. Additionally, senders and receivers may behave arbitrarily. In our model, the auditability agent is trusted to identify and flag high-risk payments by analyzing their senders. We also assume that the auditability agent and Hub do not collude, ensuring unlinkability for non-flagged payments. In this model, the Hub and the auditability agent can collaborate to link the sessions of the flagged payments while maintaining a non-colluding assumption to preserve the unlinkability of non-flagged transactions. This model applies to practical scenarios (e.g., court investigations) where the Hub is required to provide information about the receiver of a flagged payment without compromising the privacy of other payments.

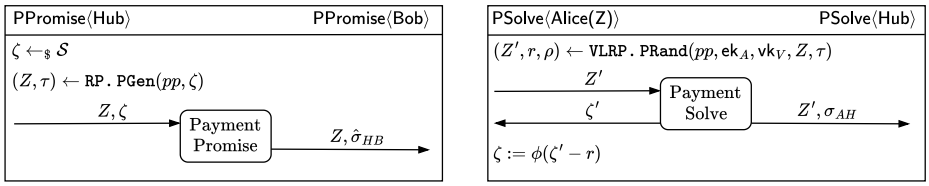

Fig. 6: Puzzle promise (PPromise), puzzle solve (PSolve) paradigm based on randomizable puzzles scheme, RP. PCH-dependent implementation of the PaymentPromise and PaymentSolve subprotocols are abstracted for clarity. Full description for A2L and BlindHub is in Appendix C and Appendix D.

From a system perspective, our protocol satisfies two key requirements: (i) payments can be processed without any direct involvement of the auditability agent, and (ii) auditability can be enforced retrospectively for flagged payments, without requiring further interaction from the sender or receiver.

5.2 AUPCH: A Generic Auditable PCH based on VLRP

To understand how VLRP can be integrated into a PCH to enable auditability, we first review the use of randomizable puzzles in PCH constructions that follow the **puzzle promise**, **puzzle solve** paradigm. The blueprint of these constructions is presented in Fig. 6. Here, we omit the details specific to each PCH construction, focusing instead on how VLRP can be employed to introduce auditability.

In summary, during the puzzle promise phase, the Hub computes a fresh puzzle Z with its corresponding solution ζ. The Hub and Bob then engage in a Payment Promise protocol, where Bob obtains the puzzle Z and a presignature $\hat{\sigma}_{HB}$ on the transfer. Once Bob learns the solution ζ, they can convert the presignature into a valid signature, completing the transfer.

During the puzzle solve phase, Alice randomizes the puzzle Z into Z' and engages with the Hub in the Payment Solve protocol. The Hub receives the randomized puzzle Z' and a valid signature for the transfer, effectively obtaining the coins from Alice. Meanwhile, Alice learns the randomized solution ζ' to the puzzle Z' Alice can then derandomize ζ' into ζ, which is the value Bob needs to complete the transfer and receive the coins from the Hub.

Our approach to constructing AUPCH involves replacing the randomizable puzzle with the VLRP introduced in this work, as shown in Fig. 7. In particular, during the *puzzle promise* phase, the Hub generates a puzzle Z, which is accompanied by an authentication tag τ. In the *puzzle solve* phase, Alice (the sender) randomizes the puzzle Z into Z' and provides an auditability token ρ, which allows linking the transfer to the original puzzle Z when required.

A key feature of AUPCH is the *audit phase*. If a transfer (sender, Z') is flagged for auditing, the Hub must identify the payment receiver. The auditability agent attests to the high-risk transaction by tagging the puzzle Z' with PFlag. The Hub can then use this attestation to run the PAudit algorithm and recover the original puzzle Z, linking sender and receiver.

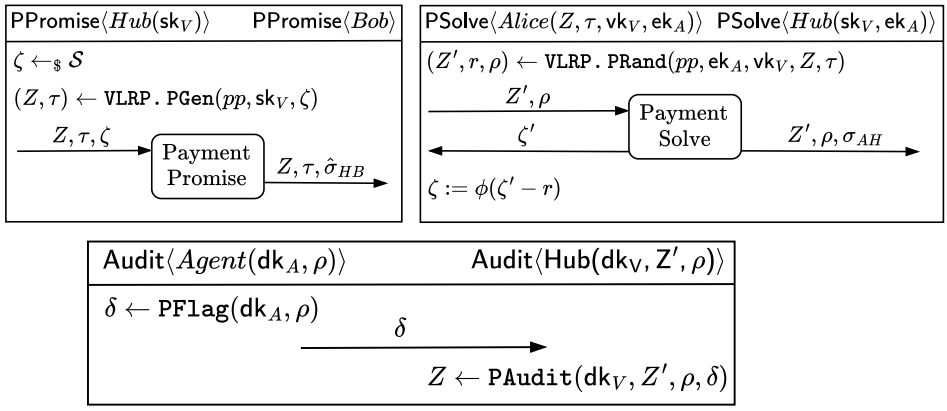

Fig. 7: AUPCH: Puzzle promise, puzzle solve paradigm using VLRP. PCH-specific details of the PaymentPromise and PaymentSolve sub-protocols are abstracted away for clarity.

5.3 Security Discussion

To provide auditability in PCH, we replace the randomizable puzzles from existing schemes with VLRP, introduced in this work. This introduces auditability while retaining the original security and privacy properties of the PCH.

The security and privacy of PCH systems rely on the security of the Randomizable Puzzle (RP) scheme [7][6]. In our case, the security and privacy notions of the VLRP scheme subsume those of the RP scheme, providing stronger guarantees. Thus, by integrating VLRP into the PCH design, we ensure that the auditability property is added without compromising the system's security or privacy. VLRP achieves security through its resistance to forgery (i.e., the inability of an adversary to generate valid puzzles without the corresponding trapdoor and secret key sk_V). The puzzle randomization also guarantees unlinkability, preventing adversaries from correlating sessions of puzzle promise and puzzle solve.

The security implications of VLRP is discussed in Appendix E. Notably, the PCH auditability directly relies on the auditability property of VLRP.

6 Conclusions

We address the need for auditability in Payment Channel Hubs (PCH) transactions, a capability absent from previous PCH constructions. We introduce a new cryptographic primitive (Verifiable Linkable Randomizable Puzzles, VLRP),

[6] More technically, in [7], footnote 4 states that authors use a re-randomizable linearly homomorphic encryption scheme Π_E instead of a re-randomizable puzzle, since the first satisfies the notion of the second; (ii) in [7], Lemma 4.8 states that Π_E achieves the core security notion in A2L (i.e., OM-CCA-A2L); (iii) in [7], Theorem 4.9 states that Π_E along with adaptor signatures and NIZK suffice for the security of A2L+.

which enables commitments to secrets using two keys: a verifiability key and an auditability key. This primitive provides: (i) verification that commitments are issued by the verifiability key owner, and (ii) randomization for unlinkability while preserving traceability via the auditability key. We then present AUPCH, a PCH overlay built on `VLRP` that provides selective auditability guarantees while maintaining the functionality, security, and privacy of existing PCH constructions based on the puzzle promise and puzzle solve paradigm. Our performance evaluation shows that AUPCH imposes only minimal computation and storage overhead.

References

1. zkp application: RSA Circom, https://github.com/zkp-application/circom-rsa-verify/
2. Christodorescu, M., English, E., Gu, W.C., Kreissman, D., Kumaresan, R., Minaei, M., Raghuraman, S., Sheffield, C., Wijeyekoon, A., Zamani, M.: Universal payment channels: An interoperability platform for digital currencies (2021), https://arxiv.org/abs/2109.12194
3. Dziembowski, S., Eckey, L., Faust, S., Malinowski, D.: Perun: Virtual payment hubs over cryptocurrencies. In: 2019 IEEE Symposium on Security and Privacy. pp. 106–123. IEEE Computer Society Press (May 2019). https://doi.org/10.1109/SP.2019.00020
4. European Parliament and Council: Regulation (eu) 2023/1114 of the european parliament and of the council of 31 may 2023 on markets in crypto-assets, and amending regulations (eu) no 1093/2010 and (eu) no 1095/2010 and directives 2013/36/eu and (eu) 2019/1937. Official Journal of the European Union, L 150 (June 2023), available at: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32023R1114
5. Financial Action Task Force (FATF): Updated guidance for a risk-based approach to virtual assets and virtual asset service providers. Tech. rep., FATF/OECD, Paris, France (October 2021), available at: https://www.fatf-gafi.org/en/publications/Fatfrecommendations/Guidance-rba-virtual-assets-2021.html
6. Ge, Z., Gu, J., Wang, C., Long, Y., Xu, X., Gu, D.: Accio: Variable-amount, optimized-unlinkable and NIZK-free off-chain payments via hubs. In: ACM CCS (2023)
7. Glaeser, N., Maffei, M., Malavolta, G., Moreno-Sanchez, P., Tairi, E., Thyagarajan, S.A.K.: Foundations of coin mixing services. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 1259–1273. ACM Press (Nov 2022). https://doi.org/10.1145/3548606.3560637
8. Green, M., Miers, I.: Bolt: Anonymous payment channels for decentralized currencies. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 473–489. ACM Press (Oct / Nov 2017). https://doi.org/10.1145/3133956.3134093
9. Groth, J.: On the size of pairing-based non-interactive arguments. In: EUROCRYPT. pp. 305–326 (2016)
10. Gudgeon, L., Moreno-Sanchez, P., Roos, S., McCorry, P., Gervais, A.: SoK: Layer-two blockchain protocols. In: Bonneau, J., Heninger, N. (eds.) FC 2020. LNCS,

vol. 12059, pp. 201–226. Springer, Heidelberg (Feb 2020). https://doi.org/10.1007/978-3-030-51280-4_12
11. Heilman, E., Alshenibr, L., Baldimtsi, F., Scafuro, A., Goldberg, S.: TumbleBit: An untrusted bitcoin-compatible anonymous payment hub. In: NDSS 2017. The Internet Society (Feb / Mar 2017)
12. Iden3: Circom circuit templates, https://github.com/iden3/circomlib
13. Jourenko, M., Kurazumi, K., Larangeira, M., Tanaka, K.: SoK: A taxonomy for layer-2 scalability related protocols for cryptocurrencies. Cryptology ePrint Archive, Report 2019/352 (2019), https://eprint.iacr.org/2019/352
14. Kumaresan, R., Le, D.V., Minaei, M., Raghuraman, S., Yang, Y., Zamani, M.: Programmable payment channels. In: ACNS. pp. 51–73 (2024)
15. Le, D.V., Gervais, A.: AMR: autonomous coin mixer with privacy preserving reward distribution. In: Proceedings of the 3rd ACM Conference on Advances in Financial Technologies. p. 142–155. AFT '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3479722.3480800, https://doi.org/10.1145/3479722.3480800
16. Li, Y., Weng, J., Lai, J., Li, Y., Wu, J., Li, M., Sun, J., Wu, P., Deng, R.H.: Auditpch: Auditable payment channel hub with privacy protection. IEEE Transactions on Information Forensics and Security **20**, 1251–1261 (2025). https://doi.org/10.1109/TIFS.2024.3515820
17. Lind, J., Naor, O., Eyal, I., Kelbert, F., Sirer, E.G., Pietzuch, P.R.: Teechain: a secure payment network with asynchronous blockchain access. In: ACM SOSP. pp. 63–79 (2019)
18. Minaei, M., Chatzigiannis, P., Jin, S., Raghuraman, S., Kumaresan, R., Zamani, M., Moreno-Sanchez, P.: Unlinkability and interoperability in account-based universal payment channels. Cryptology ePrint Archive, Paper 2023/916 (2023), https://eprint.iacr.org/2023/916
19. Minaei, M., Moreno-Sanchez, P., Fang, Z., Raghuraman, S., Alamati, N., Chatzigiannis, P., Kumaresan, R., Le, D.V.: DTL: Data tumbling layer. a composable unlinkability for smart contracts (2025), https://arxiv.org/abs/2503.04260
20. Poon, J., Dryja, T.: The bitcoin lightning network: Scalable off-chain instant payments (2016)
21. Qin, X., Pan, S., Mirzaei, A., Sui, Z., Ersoy, O., Sakzad, A., Esgin, M.F., Liu, J.K., Yu, J., Yuen, T.H.: BlindHub: Bitcoin-compatible privacy-preserving payment channel hubs supporting variable amounts. Cryptology ePrint Archive, Report 2022/1735 (2022), https://eprint.iacr.org/2022/1735
22. Qin, X., Pan, S., Mirzaei, A., Sui, Z., Ersoy, O., Sakzad, A., Esgin, M.F., Liu, J.K., Yu, J., Yuen, T.H.: BlindHub: Bitcoin-Compatible Privacy-Preserving Payment Channel Hubs Supporting Variable Amounts (2022), https://eprint.iacr.org/2022/1735, publication info: Published elsewhere. Major revision. IEEE S&P 2023
23. Reclaim: AES Circom, https://gitlab.reclaimprotocol.org/reclaim/zk-symmetric-crypto
24. Tairi, E., Moreno-Sanchez, P., Maffei, M.: A^{2}L: Anonymous Atomic Locks for Scalability in Payment Channel Hubs (2019), https://eprint.iacr.org/2019/589, publication info: Published elsewhere. Major revision. IEEE Symposium on Security and Privacy - S&P 2021
25. Tairi, E., Moreno-Sanchez, P., Maffei, M.: A^2L: Anonymous atomic locks for scalability in payment channel hubs. In: 2021 IEEE Symposium on Security and Privacy. pp. 1834–1851. IEEE Computer Society Press (May 2021). https://doi.org/10.1109/SP40001.2021.00111

26. Wang, Z., Cirkovic, M., Le, D.V., Knottenbelt, W., Cachin, C.: Pay Less for Your Privacy: Towards Cost-Effective On-Chain Mixers. In: Bonneau, J., Weinberg, S.M. (eds.) 5th Conference on Advances in Financial Technologies (AFT 2023). Leibniz International Proceedings in Informatics (LIPIcs), vol. 282, pp. 16:1–16:25. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2023). https://doi.org/10.4230/LIPIcs.AFT.2023.16, https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.AFT.2023.16
27. Wicht, F.X., Wang, Z., Le, D.V., Cachin, C.: A transaction-level model for blockchain privacy. In: Financial Cryptography and Data Security: 28th International Conference, FC 2024, Willemstad, Curaçao, March 4–8, 2024, Revised Selected Papers, Part II. p. 293–310. Springer-Verlag, Berlin, Heidelberg (2025). https://doi.org/10.1007/978-3-031-78679-2_16, https://doi.org/10.1007/978-3-031-78679-2_16
28. Zamani, M., English, E., Kumaresan, R., Christodorescu, M., Gu, W., Harper, C., Sheffield, C., Minaei, M., Raghuraman, S.: Cross-border payments for central bank digital currencies via universal payment channels. Retrieved from Bank for International Settlement website: http://www. bis. org/events/cpmi_ptfop/proceedings/paper14. pdf. b (2021)

A Extended Preliminaries

A.1 Randomizable Puzzles Scheme

Definition 2 (Randomizable puzzle scheme). *A* randomizable puzzle scheme $\mathsf{RP} := (\mathsf{PSetup}, \mathsf{PGen}, \mathsf{PSolve}, \mathsf{PRand})$ *is defined with respect to a function* ϕ *as follows:*

$(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PSetup}(1^n)$. *A* PPT *algorithm that takes as input the security parameter* 1^n. *It outputs a pair of public parameters and trapdoor* $(\mathsf{pp}, \mathsf{td})$.

$Z \leftarrow \mathsf{PGen}(\mathsf{pp}, \zeta)$. *A* PPT *algorithm that takes as input the public parameters* pp *and a puzzle soluton* ζ. *It outputs a puzzle* Z.

$\zeta \leftarrow \mathsf{PSolve}(\mathsf{td}, Z)$. *A* DPT *algorithm that takes as input the trapdoor* td *and a puzzle* Z. *It outputs a puzzle solution* ζ.

$(Z', r) \leftarrow \mathsf{PRand}(\mathsf{pp}, Z)$. *A* PPT *algorithm that takes as input the public parameters* pp *and a puzzle* Z. *It outputs a randomized puzzle* Z' *and the randomness value* r. *The randomized puzzle* Z' *has a solution* $\phi(\zeta, r)$.

A randomizable puzzle scheme is correct *if for every* $\zeta \in \mathcal{S}$, *it holds that:*

$$\Pr\left[\begin{array}{c|c} & (\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PSetup}(1^n) \\ \zeta \leftarrow \mathsf{PSolve}(\mathsf{td}, Z) & Z \leftarrow \mathsf{PGen}(\mathsf{pp}, \zeta) \\ \zeta' \leftarrow \mathsf{PSolve}(\mathsf{td}, Z') & (Z', r) \leftarrow \mathsf{PRand}(\mathsf{pp}, Z) \\ & \zeta' := \phi(\zeta, r) \end{array}\right] \geq 1 - \mathsf{negl}(n)$$

We now review the notion of randomizability.

Definition 3 (Randomizability). *A randomizable puzzle scheme* $\mathsf{RP} := (\mathsf{PSetup}, \mathsf{PGen}, \mathsf{PSolve}, \mathsf{PRand})$ *is* randomizable *if for all* $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PSetup}(1^n)$, *and all pairs* (Z', r), *it holds that:*

$$\exists Z | (Z', r) \leftarrow \mathsf{PRand}(\mathsf{pp}, Z)$$

We now review the notion of security and privacy for a randomizable puzzle scheme.

Definition 4 (Security). *A randomizable puzzle scheme* $\mathsf{RP} := (\mathsf{PSetup}, \mathsf{PGen}, \mathsf{PSolve}, \mathsf{PRand})$ *is* secure *if for every* PPT *adversaries* $\mathcal{A}$*, there is a negligible function* $\mathsf{negl}(n)$ *such that:*

$$\Pr\left[\zeta \leftarrow \mathcal{A}(\mathsf{pp}, Z) \middle| \begin{array}{c} (\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PSetup}(1^n) \\ \zeta \leftarrow_{\$} \mathcal{S}, Z \leftarrow \mathsf{PGen}(\mathsf{pp}, \zeta) \end{array}\right] \leq \mathsf{negl}(n)$$

Definition 5 (Privacy). *A randomizable puzzle scheme* RP *is* private *if for every* PPT *adversary* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(n)$ *such that*

$$\Pr[\mathsf{RPRandSec}_{\mathcal{A},\mathsf{RP}}(n) = 1] \leq \mathsf{negl}(n)$$

where the experiment $\mathsf{RPRandSec}_{\mathcal{A},\mathsf{RP}}(n)$ *is defined as follows:*

$\mathsf{RPRandSec}_{\mathcal{A},\mathsf{RP}}(n)$

$(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PSetup}(1^n)$
$((Z_0, \zeta_0), (Z_1, \zeta_1)) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{td})$
$b \leftarrow_{\$} \{0, 1\}$
$(Z'_0, r_0) \leftarrow \mathsf{PRand}(\mathsf{pp}, Z_0)$
$(Z'_1, r_1) \leftarrow \mathsf{PRand}(\mathsf{pp}, Z_1)$
$b^* \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{td}, Z'_b)$
$b_0 := (\mathsf{PSolve}(\mathsf{td}, Z_0) = \zeta_0)$
$b_1 := (\mathsf{PSolve}(\mathsf{td}, Z_1) = \zeta_1)$
$b_2 := (b = b^*)$
return $b_0 \wedge b_1 \wedge b_2$

A.2 Digital Signature Scheme

Definition 6 (Digital signature scheme). *A* digital signature scheme $\mathsf{DS} := (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is defined as follows:*

- $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^n)$*. A* PPT *algorithm that takes as input the security parameter* 1^n*. It outputs a pair of signing and verification keys* $(\mathsf{sk}, \mathsf{vk})$*.*
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$*. A* PPT *algorithm that takes as input the signing key* sk *and a message* m*. It outputs a signature* σ*.*
- $\{0, 1\} \leftarrow \mathsf{Vrfy}(\mathsf{vk}, m, \sigma)$*. A* DPT *algorithm that takes as input the verification key* vk*, a message* m*, and a signature* σ*. It outputs a bit* $b \in \{0, 1\}$*.*

A digital signature scheme is correct *if for every* $m \in \mathcal{M}$ *it holds that:*

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}, m, \sigma) \middle| \begin{array}{c} (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^n) \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m) \end{array}\right] \geq 1 - \mathsf{negl}(n)$$

We now review the notion of security for a digital signature scheme.

Definition 7 (Existential unforgeability under an adaptative chosen-message attack). *A digital signature scheme* $\mathsf{DS} := (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is* existentially unforgeable under an adaptative chosen-message attack *(or just* secure*) if for all* PPT *adversaries* $\mathcal{A}$*, there is a negligible function* $\mathsf{negl}(n)$ *such that:*

$$\Pr[\mathsf{Sig\text{-}Forge}_{\mathcal{A},\mathsf{DS}}(n) = 1] \leq \mathsf{negl}(n)$$

where the experiment $\mathsf{Sig\text{-}Forge}_{\mathcal{A},\mathsf{DS}}(n)$ *is defined as follows:*

$\mathsf{Sig\text{-}Forge}_{\mathcal{A},\mathsf{DS}}(n)$	$\mathsf{OSign}(m)$
$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^n)$	$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
$(m, \sigma) \leftarrow \mathcal{A}^{\mathsf{OSign}}(\mathsf{vk})$	$\mathcal{Q} := \mathcal{Q} \cup m$
$b_0 := m \notin \mathcal{Q}$	**return** σ
$b_1 := \mathsf{Vrfy}(\mathsf{vk}, m, \sigma)$	
return $b_0 \wedge b_1$	

A.3 Adaptor Signature Scheme

Definition 8 (Adaptor signature scheme). *An* adaptor signature scheme *is a tuple of algorithms* $\mathsf{AS} := (\mathsf{KeyGen}, \mathsf{PreSign}, \mathsf{PreVrfy}, \mathsf{Adapt}, \mathsf{Extract})$ *with respect to a digital signature scheme* DS *and a hard relation* R *is defined as follows:*

- $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^n)$*. A* PPT *algorithm that takes as input the security parameter* 1^n*. It outputs a pair of signing and verification keys* $(\mathsf{sk}, \mathsf{vk})$*.*
- $\hat{\sigma} \leftarrow \mathsf{PreSign}(\mathsf{sk}, m, x)$*. A* PPT *algorithm that takes as input the signing key* sk*, a message* m *and a statement* $x \in \mathcal{X}$*. It outputs a pre-signature* $\hat{\sigma}$*.*
- $\{0,1\} \leftarrow \mathsf{PreVrfy}(\mathsf{vk}, m, \hat{\sigma}, x)$*. A* DPT *algorithm that takes as input the verification key* vk*, a message* m*, a pre-signature* σ *and a statement* $x \in \mathcal{X}$*. It outputs a bit* $b \in \{0,1\}$*.*
- $\sigma \leftarrow \mathsf{Adapt}(\hat{\sigma}, w)$*. A* PPT *algorithm that takes as input a pre-signature* $\hat{\sigma}$ *and a witness* $w \in \mathcal{W}$*. It outputs a signature* σ*.*
- $w \leftarrow \mathsf{Extract}(\hat{\sigma}, \sigma, x)$*. A* DPT *algorithm that takes as input a pre-signature* $\hat{\sigma}$*, a signature* σ *and a statement* $x \in \mathcal{X}$*. It outputs a witness* $w \in \mathcal{W}$*.*

An adaptor signature scheme is pre-signature correct *if for every* $m \in \mathcal{M}$ *it holds that:*

$$\Pr\left[\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) \middle| \begin{matrix}(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^n) \\ (x, w) \leftarrow \mathsf{GenR}(1^n) \\ \hat{\sigma} \leftarrow \mathsf{PreSign}(\mathsf{sk}, m, x)\end{matrix}\right] \geq 1 - \mathsf{negl}(n)$$

We now review the notion of security for an adaptor signature scheme.

Definition 9 (Existential unforgeability under a chosen-message attack). *An adaptor signature scheme* $\mathsf{AS} := (\mathsf{KeyGen}, \mathsf{PreSign}, \mathsf{PreVrfy}, \mathsf{Adapt}, \mathsf{Extract})$ *is* existentially unforgeable under a chosen-message attack *if for all* PPT *adversaries* $\mathcal{A}$*, there is a negligible function* $\mathsf{negl}(n)$ *such that:*

$$\Pr[\mathsf{ASig\text{-}EUF\text{-}CMA}_{\mathcal{A},\mathsf{AS}}(n) = 1] \leq \mathsf{negl}(n)$$

where the experiment $\mathsf{ASig\text{-}EUF\text{-}CMA}_{\mathcal{A},\mathsf{AS}}(n)$ *is defined as follows:*

$\mathsf{ASig\text{-}EUF\text{-}CMA}_{\mathcal{A},\mathsf{AS}}(n)$

1 : $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^n)$
2 : $(x, w) \leftarrow \mathsf{GenR}(1^n)$
3 : $m \leftarrow \mathcal{A}^{\mathsf{OSign},\mathsf{OPreSign}}(\mathsf{vk})$
4 : $\hat{\sigma} \leftarrow \mathsf{PreSign}(\mathsf{sk}, m, x)$
5 : $\sigma \leftarrow \mathcal{A}^{\mathsf{OSign},\mathsf{OPreSign}}(\hat{\sigma}, x)$
6 : $b_0 := m \notin \mathcal{Q}$
7 : $b_1 := \mathsf{Vrfy}(\mathsf{vk}, m, \sigma)$
8 : **return** $b_0 \wedge b_1$

$\mathsf{OSign}(m)$

1 : $\sigma \leftarrow \mathsf{Sign}(\sigma, m)$
2 : $\mathcal{Q} := \mathcal{Q} \cup m$
3 : **return** σ

$\mathsf{OPreSign}(m, x)$

1 : $\hat{\sigma} \leftarrow \mathsf{PreSign}(\mathsf{sk}, m, x)$
2 : $\mathcal{Q} := \mathcal{Q} \cup m$
3 : **return** $\hat{\sigma}$

Definition 10 (Pre-signature adaptability). *An adaptor signature scheme* $\mathsf{AS} := (\mathsf{KeyGen}, \mathsf{PreSign}, \mathsf{PreVrfy}, \mathsf{Adapt}, \mathsf{Extract})$ *is* pre-signature adaptable *if for any message* $m \in \mathcal{M}$*, any statement/witness pair* $(x, w) \in \mathsf{R}$*, any key pair* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^n)$ *and any pre-signature* $\hat{\sigma}$ *such that* $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$ *it holds that:*

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}, m, \sigma) \middle| \sigma \leftarrow \mathsf{Adapt}(\hat{\sigma}, w)\right] \geq 1 - \mathsf{negl}(n)$$

Definition 11 (Witness extractability). *An adaptor signature scheme* $\mathsf{AS} := (\mathsf{KeyGen}, \mathsf{PreSign}, \mathsf{PreVrfy}, \mathsf{Adapt}, \mathsf{Extract})$ *is* witness extractable *if for all* PPT *adversaries* $\mathcal{A}$*, there is a negligible function* $\mathsf{negl}(n)$ *such that:*

$$\Pr[\mathsf{ASig\text{-}WE}_{\mathcal{A},\mathsf{AS}}(n) = 1] \leq \mathsf{negl}(n)$$

where the experiment $\mathsf{ASig\text{-}WE}_{\mathcal{A},\mathsf{AS}}(n)$ *is defined as follows:*

$\mathsf{ASig\text{-}WE}_{\mathcal{A},\mathsf{AS}}(n)$

1 : $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^n)$
2 : $(m, x) \leftarrow \mathcal{A}^{\mathsf{OSign},\mathsf{OPreSign}}(\mathsf{vk})$
3 : $\hat{\sigma} \leftarrow \mathsf{PreSign}(\mathsf{sk}, m, x)$
4 : $\sigma \leftarrow \mathcal{A}^{\mathsf{OSign},\mathsf{OPreSign}}(\hat{\sigma}, x)$
5 : $w' \leftarrow \mathsf{Extract}(\hat{\sigma}, \sigma, x)$
6 : $b_0 := m \notin \mathcal{Q}$
7 : $b_1 := \mathsf{Vrfy}(\mathsf{vk}, m, \sigma)$
8 : $b_2 := (x, w') \notin \mathsf{R}$
9 : **return** $b_0 \wedge b_1$

$\mathsf{OSign}(m)$

1 : $\sigma \leftarrow \mathsf{Sign}(\sigma, m)$
2 : $\mathcal{Q} := \mathcal{Q} \cup m$
3 : **return** σ

$\mathsf{OPreSign}(m, x)$

1 : $\hat{\sigma} \leftarrow \mathsf{PreSign}(\mathsf{sk}, m, x)$
2 : $\mathcal{Q} := \mathcal{Q} \cup m$
3 : **return** $\hat{\sigma}$

A.4 Public Key Encryption Scheme

Definition 12 (Public key encryption scheme). *A* public key encryption scheme $\mathsf{PKE} := (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is defined as follows:*

$(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{KeyGen}(1^n)$. *A* PPT *algorithm that takes as input the security parameter* 1^n. *It outputs a pair of encryption and decryption keys* $(\mathsf{dk}, \mathsf{ek})$.

$ct \leftarrow \mathsf{Enc}(\mathsf{ek}, m)$. *A* PPT *algorithm that takes as input the encryption key* ek *and a message* m. *It outputs a ciphertext* ct.

$\{m, \perp\} \leftarrow \mathsf{Dec}(\mathsf{dk}, ct)$. *A* DPT *algorithm that takes as input the decryption key* dk *and a ciphertext* ct. *It outputs a message* m *or a special symbol* $\perp$ *denoting failure.*

A public key encryption scheme is correct *if for every message* $m \in \mathcal{M}$ *it holds that:*

$$\Pr\left[m = m' \middle| \begin{matrix}(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{KeyGen}(1^n) \\ m' := \mathsf{Dec}(\mathsf{dk}, \mathsf{Enc}(\mathsf{ek}, m))\end{matrix}\right] = 1$$

We now review the notion of security for a public key encryption scheme.

Definition 13 (Indistinguishable encryptions under chosen-plaintext attacks).

A public key encryption scheme $\mathsf{PKE} := (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *has* indistinguishable encryptions under chosen-plaintext attacks *(or is* CPA-secure*) if for all* PPT *adversaries* $\mathcal{A}$, *there is a negligible function* $\mathsf{negl}(n)$ *such that:*

$$\Pr\left[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{PKE}}(n) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(n)$$

where the experiment $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{PKE}}(n)$ *is defined as follows:*

$\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{PKE}}(n)$

$(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{KeyGen}(1^n)$
$(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{ek})$
$b \leftarrow_{\$} \{0,1\}$
$ct_b \leftarrow \mathsf{Enc}(\mathsf{ek}, m_b)$
$b^* \leftarrow \mathcal{A}(ct_b)$
$b_0 := (|m_0| = |m_1|)$
$b_1 := (b = b^*)$
return $b_0 \wedge b_1$

A.5 Non-Interactive Zero-Knowledge Proof Scheme

Definition 14 (Non-Interactive Zero Knowledge Proof). *A* non-interactive zero knowledge proof *for a relation* R *is a tuple of algorithms* $\mathsf{NIZK} := (\mathsf{CrsGen}, \mathsf{Prove}, \mathsf{Verify})$ *defined as follows:*

- $crs \leftarrow \mathsf{CrsGen}(1^n)$. *A* PPT *algorithm that takes as input the security parameter* 1^n. *It outputs a common reference string crs.*
- $\pi \leftarrow \mathsf{Prove}(crs, x, w)$. *A* PPT *algorithm that takes as input the common reference string crs, a statement x and a witness w. It outputs a proof* π
- $\{0,1\} \leftarrow \mathsf{Verify}(crs, x, \pi)$. *A* DPT *algorithm that takes as input the common reference string, a statement x and a proof* π. *It outputs a bit* $b \in \{0,1\}$.

A NIZK *is* complete *if for all* $(x, w) \in \mathsf{R}$ *and all* $crs \leftarrow \mathsf{CrsGen}(1^n)$ *it holds that:*

$$\Pr\left[\mathsf{Verify}(crs, x, \pi) = 1 \middle| \begin{matrix} crs \leftarrow \mathsf{CrsGen}(1^n) \\ \pi \leftarrow \mathsf{Prove}(crs, x, w) \end{matrix}\right] \geq 1 - \mathsf{negl}(n)$$

We now review the notion of security for a non-interactive zero-knowledge proof.

Definition 15 (Zero-knowledge). *A non-interactive zero knowledge proof* $\mathsf{NIZK} := (\mathsf{CrsGen}, \mathsf{Prove}, \mathsf{Verify})$ *is* zero-knowledge *if there exist two* PPT *algorithms* $(\mathsf{CrsSim}, \mathsf{ProveSim})$ *such that for every* PPT *adversary* $\mathcal{A}$, *there is a negligible function* $\mathsf{negl}(n)$ *such that:*

$$\left| \Pr\begin{bmatrix} crs \leftarrow \mathsf{CrsGen}(1^n) \\ 1 \leftarrow \mathcal{A}^{\mathsf{OProve}}(crs) \end{bmatrix} - \Pr\begin{bmatrix} (crs, \tau) \leftarrow \mathsf{CrsSim}(1^n) \\ 1 \leftarrow \mathcal{A}^{\mathsf{OProveSim}}(crs) \end{bmatrix} \right| \leq \mathsf{negl}(n)$$

where the oracles OProve *and* $\mathsf{OProveSim}$ *are defined as follows:*

$\mathsf{Prove}(x, w)$	$\mathsf{ProveSim}(x, w)$
$\pi \leftarrow \mathsf{Prove}(crs, x, w)$	**if** $(x, w) \notin \mathsf{R}$ **then**
return π	**abort**
	endif
	$\pi \leftarrow \mathsf{ProveSim}(crs, x)$
	return π

Definition 16 (Knowledge-soundness). *A non-interactive zero knowledge proof* $\mathsf{NIZK} := (\mathsf{CrsGen}, \mathsf{Prove}, \mathsf{Verify})$ *is* knowledge-sound *if there exists a* PPT *algorithm* Ext *such that for all adversaries* $\mathcal{A}$*, it holds that:*

$$\Pr\left[\begin{array}{c} 1 \leftarrow \mathsf{Verify}(crs, x, \pi) \\ (x, w) \notin \mathsf{R} \end{array} \middle| \begin{array}{c} crs \leftarrow \mathsf{CrsGen}(1^n) \\ (x, \pi) \leftarrow \mathcal{A}(crs) \\ w \leftarrow \mathsf{Ext}^{\mathcal{A}}(crs, x, \pi) \end{array}\right] \leq \mathsf{negl}(n)$$

B Properties of VLRP

Definition 17 (VLRP correctness). *A verifiable linkable randomizable puzzle is correct if for every security parameter* 1^n*, every solution* $\zeta \in \mathcal{S}$*, every randomness* $r \in \mathcal{R}$*, it holds that:*

$$\Pr\left[\begin{array}{c} \mathtt{PVerifyTag}(pp, \mathsf{vk}_V, Z, \tau) = 1 \\ \wedge \\ \mathtt{PVerifyRand}(pp, \mathsf{ek}_A, \mathsf{ek}_V, \mathsf{vk}_V, Z', \rho) = 1 \\ \wedge \\ \mathsf{PSolve}(td, Z') = \phi(\zeta, r) \end{array} \middle| \begin{array}{c} (\mathsf{vk}_V, \mathsf{ek}_V, \mathsf{sk}_V, \mathsf{dk}_V) \leftarrow \mathtt{PVerifySetup}(1^n), \\ (\mathsf{ek}_A, \mathsf{dk}_A) \leftarrow \mathtt{PAuditSetup}(1^n), \\ (pp, td) \leftarrow \mathsf{PSetup}(1^n), \\ (Z, \tau) \leftarrow \mathtt{PGen}(pp, \mathsf{sk}_V, \zeta), \\ (Z', r, \rho) \leftarrow \mathtt{PRand}(pp, \mathsf{ek}_A, \mathsf{ek}_V, \mathsf{vk}_V, Z, \tau) \end{array}\right] = 1$$

Security Intuitively, a malicious Alice or Bob that does not know the private key sk_V, should not be able to create a pair of puzzle and tag (Z, τ) that correctly verifies. Moreover, the adversary should not be able to extract the solution ζ to a puzzle Z without access to the trapdoor td. A similar definition for $\mathtt{PRand}$ is not required since it only requires publicly available information.

Definition 18 (VLRP Security). *A* VLRP *scheme is secure, if there exists a negligible function* $\mathsf{negl}(n)$*, such that* $\Pr[\mathsf{VLRP\text{-}Security}^{\mathcal{A}}_{\Pi_{\mathtt{VLRP}}}(1^n) = 1] \leq \mathsf{negl}(n)$ *where* $\mathsf{VLRP\text{-}Security}^{\mathcal{A}}_{\Pi_{\mathtt{VLRP}}}$ *is defined in Fig. 8.*

Unlinkability Intuitively, an adversary that observes an honestly created pair of randomized puzzle and auditability tag (Z', ρ) should not be able to distinguish better than guessing whether the tuple has been created from puzzle Z_0 or Z_1, even when the latter both puzzles are created by the adversary itself.

Definition 19 (VLRP Unlinkability). *A* VLRP *scheme is unlinkable, if there exists a negligible function* $\mathsf{negl}(n)$*, such that* $\Pr[\mathsf{VLRP\text{-}Unlinkability}^{\mathcal{A}}_{\Pi_{\mathtt{VLRP}}}(1^n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n)$ *where* $\mathsf{VLRP\text{-}Unlinkability}^{\mathcal{A}}_{\Pi_{\mathtt{VLRP}}}$ *is defined in Fig. 8.*

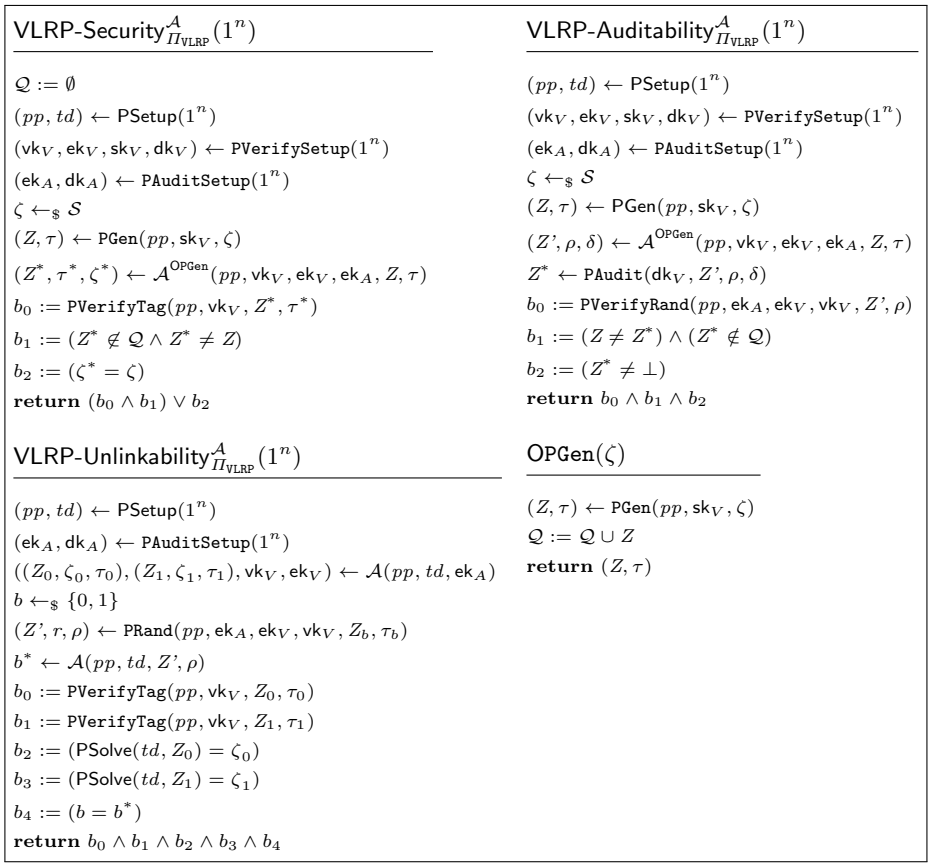

Fig. 8: Definition of experiments VLRP-Security$^{\mathcal{A}}_{\Pi_{\text{VLRP}}}$, VLRP-Unlinkability$^{\mathcal{A}}_{\Pi_{\text{VLRP}}}$, and VLRP-Auditability$^{\mathcal{A}}_{\Pi_{\text{VLRP}}}$.

Auditability Intuitively, an adversary should not be able to create a tuple of randomized puzzle and auditability token (Z', ρ) from a puzzle Z, possibly of its choice, such that the tuple correctly verifies and yet Z' cannot be linked to Z if the auditability agent flags the corresponding payment as high-risk.

Definition 20 (VLRP Auditability). *A* VLRP *scheme is auditable, if there exists a negligible function* $\text{negl}(n)$*, such that* $\Pr[\text{VLRP-Auditability}^{\mathcal{A}}_{\Pi_{\text{VLRP}}}(1^n) = 1] \leq \text{negl}(n)$ *where* VLRP-Auditability$^{\mathcal{A}}_{\Pi_{\text{VLRP}}}$ *is defined in Fig. 8.*

C An Auditabile PCH from VLRP and A2L

In this section, we outline how to enhance the PCH construction in A2L with auditability, using the VLRP scheme introduced in this work. This process is

illustrated in Fig. 9, where the A2L protocol, initially based on the randomizable puzzle scheme (RP)[7], is updated with $\mathtt{VLRP}$ function calls. By transforming the A2L protocol with $\mathtt{VLRP}$ scheme, we achieve a secure, privacy-preserving PCH protocol with auditability.

Beyond the RP and $\mathtt{VLRP}$ schemes, the following key components are required:

1. A zero-knowledge proof NIZK (as described in Section 2) for the relation:

$$\mathsf{st}_{A2L} : \left\{ (Y, Z; \zeta, td) : Y := g^{\zeta} \wedge \zeta \leftarrow \mathsf{RP.PSolve}(td, Z) \right\}$$

2. An adaptor signature scheme Π_{ADP} (as described in Section 2), with respect to a hard relation. In the description of A2L here, we assume the discrete logarithm relation.

Certain steps, such as registration, are omitted for readability, as they do not rely on the RP or $\mathtt{VLRP}$ schemes. While we focus on A2L, the same approach applies to A2L $^{+}$ and A2L $^{\mathsf{UC}}$, as they follow the paradigm in Fig. 6, with technical differences in the PaymentSolve subprotocol. This further illustrates the composability of AUPCH.

D Auditabile PCH from VLRP and BlindHub

Here we show how to add auditability to BlindHub using our VLRP scheme, as shown in Fig. 11.

Apart from the RP and $\mathtt{VLRP}$ schemes, this protocol relies on several key building blocks:

1. a zero-knowledge proof (Definition 14) for the relation st_{A2L} described in Appendix C.
2. a blind adaptor signature scheme BAS. Like an adaptor signature scheme, BAS permits to presign a message, but with the key difference that it presigns a hash of the message, not the message itself. In BlindHub, it is assumed that all parties have been convinced they know the preimage of the hash used in the blind presignature. As in A2L, we assume the discrete logarithm relation, that is, tuples (Y, ζ) such that $Y := g^{\zeta}$, as the one used for BAS.
3. The randomizable signature over randomizable commitments RSoRC scheme. This scheme is composed of the following algorithms:
   - RCSign: a signing algorithm that takes as input a signing key, a hard relation statement, and a commitment, and outputs a signature, binding the statement and commitment.
   - Vf: a verification algorithm that checks the correctness of the signature by verifying the key, signature, statement, and commitment.

[7] The original A2L protocol is presented using an encryption scheme instead of RP. However, in [7] it is shown that a linear-homomorphic encryption scheme is a secure instance of randomizable puzzle.

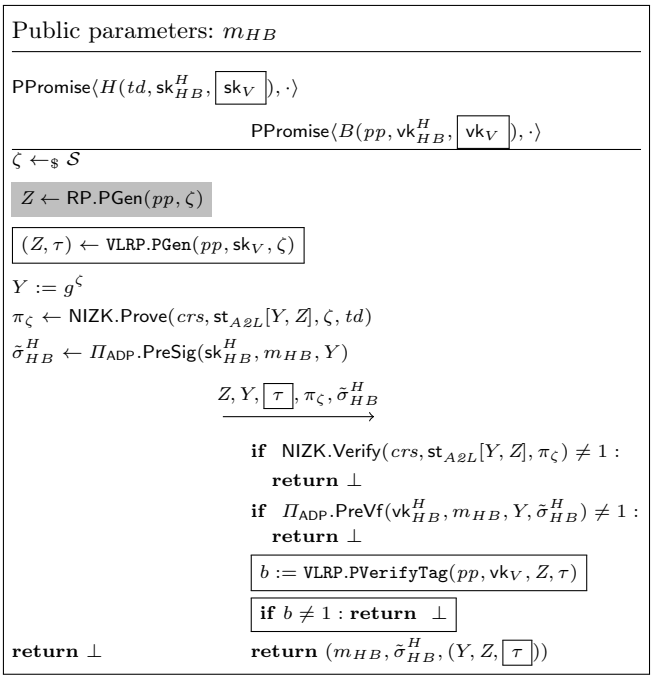

Fig. 9: Enhancing A2L with auditability property using the VLRP scheme. In particular, the calls to the randomizable puzzle RP scheme (in grey) are updated by the corresponding calls to VLRP scheme (in boxes). The second part of this protocol is shown in Fig. 10.

– Rand: A randomization algorithm, outputs randomized versions of signature, statement, and commitment while maintaining signature's validity.

As with A2L, some details are omitted for clarity, as they are independent of the RP or VLRP schemes.

E Correctness and Security Proofs

E.1 Proof of Correctness

Theorem 4 (VLRP Correctness). *Assume that the digital signature scheme* DS*, the encryption scheme* PKE*, the non-interactive zero-knowledge argument system* NIZK*, and the randomizable puzzle scheme* RP *are correct. Then, our construction is correct.*

Proof. In order to prove correctness, we need to prove three cases:
Case (i): $\mathtt{PVerifyTag}(pp, \mathsf{vk}_V, Z, \tau) = 1$

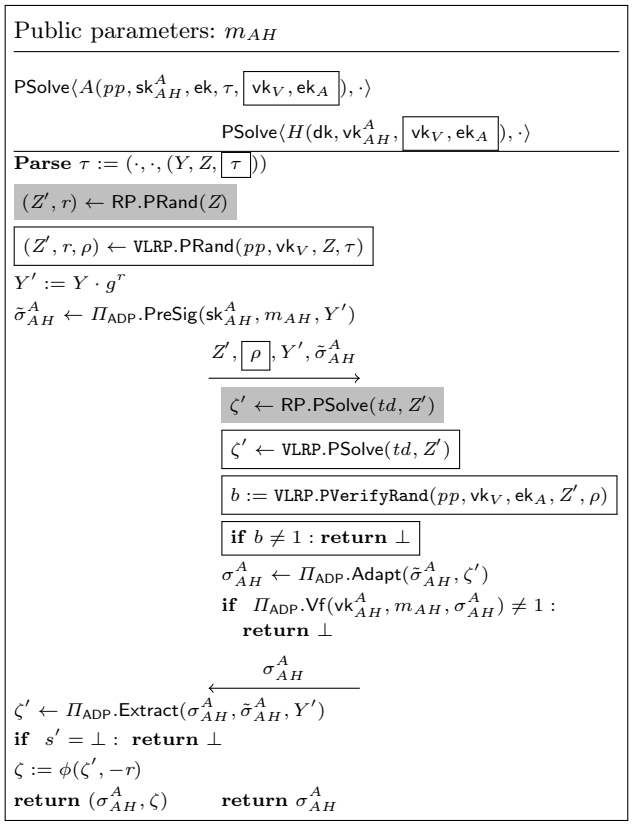

Fig. 10: Continuation from Fig. 9.

For this case, we can observe the following

$$\mathtt{PVerifyTag}(pp, \mathsf{vk}_V, Z, \tau) = \mathsf{DS.Vf}(\mathsf{vk}_V, Z, \tau) = \mathsf{DS.Vf}(\mathsf{vk}_V, Z, \mathsf{DS.Sig}(\mathsf{sk}_V, Z)) = 1$$

Case (ii): $\mathtt{PVerifyRand}(pp, \mathsf{ek}_A, \mathsf{vk}_V, \mathsf{ek}_V, Z', \rho) = 1$
For this case, we observe that

$$\mathtt{PVerifyRand}(pp, \mathsf{ek}_A, \mathsf{ek}_V, \mathsf{vk}_V, Z', \rho) = \mathsf{NIZK.Verify}(crs, \mathsf{st}_{\mathtt{PRand}}[x], \pi_{\mathtt{PRand}}) = 1$$

with

$$x = (\mathsf{vk}_V, Z', ct_{\mathsf{PKE}}, \mathsf{ek}_V, \mathsf{ek}_A, \mathsf{pp}_{\mathsf{RP}})$$

and given that

$$\pi_{\mathtt{PRand}} \leftarrow \mathsf{NIZK.Prove}(crs, \mathsf{st}_{\mathtt{PRand}}[x], w)$$
$$w = (Z, r, r_V, r_A, \tau)$$
$$r_V, r_A \leftarrow_{\$} \{0,1\}^n$$
$$ct_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_A, \mathsf{PKE.Enc}(\mathsf{ek}_V, Z; r_V); r_A)$$

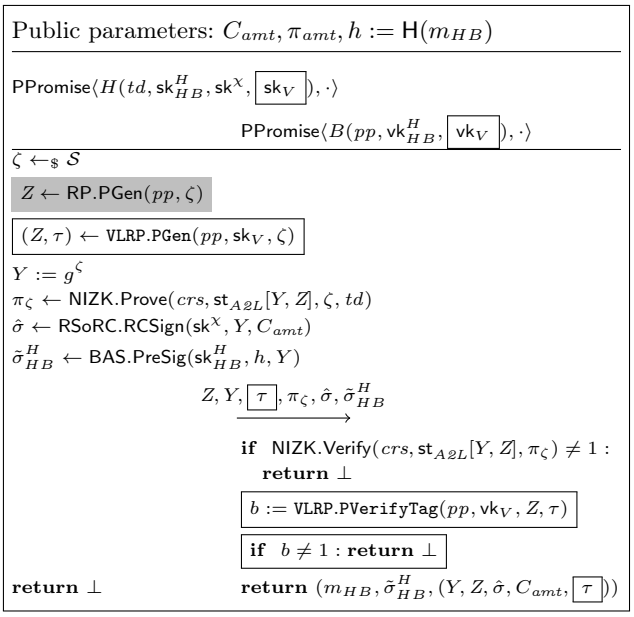

Fig. 11: Enhancing BlindHub with auditability property using the VLRP scheme. In particular, the calls to randomizable puzzle RP scheme (in grey) are updated by the corresponding calls to VLRP scheme (in boxes). The continuation of this protocol is shown in Fig. 12.

Case (iii): $\mathsf{PSolve}(td, Z') = \phi(Z, r)$
This holds follows from correctness of the randomizable puzzle RP scheme.

E.2 Proof of VLRP Security

Proof. To prove this theorem, we consider the following game hops:

Hybrid $\mathcal{H}_0$: This hybrid corresponds to the original game of $\mathsf{VLRP\text{-}Security}^{\mathcal{A}}_{\Pi_{\mathtt{VLRP}}}$ as defined in Fig. 8.

Hybrid $\mathcal{H}_1$: This hybrid, formally defined in Fig. 13, works exactly the same as $\mathcal{H}_0$ but with the highlighted grey line. The challenge checks if the adversary wins because of condition b_2 and in that case, aborts.

Let $\mathsf{VLRP\text{-}Security}^{\mathcal{A}}_{\Pi_{\mathtt{VLRP}}}(1^n)_{\mathcal{H}_i}$ be the game $\mathsf{VLRP\text{-}Security}^{\mathcal{A}}_{\Pi_{\mathtt{VLRP}}}$ as defined in hybrid $\mathcal{H}_i$. The proof for this theorem follows from Lemmas 1 and 2 below.

Lemma 1. *Assume that the randomizable puzzle* RP *is secure. For all PPT adversaries* $\mathcal{A}$*, it holds that*

$$\Pr[\mathsf{VLRP\text{-}Security}^{\mathcal{A}}_{\Pi_{\mathtt{VLRP}}}(1^n)_{\mathcal{H}_0} = 1] \leq \Pr[\mathsf{VLRP\text{-}Security}^{\mathcal{A}}_{\Pi_{\mathtt{VLRP}}}(1^n)_{\mathcal{H}_1} = 1] + \mathsf{negl}(n)$$

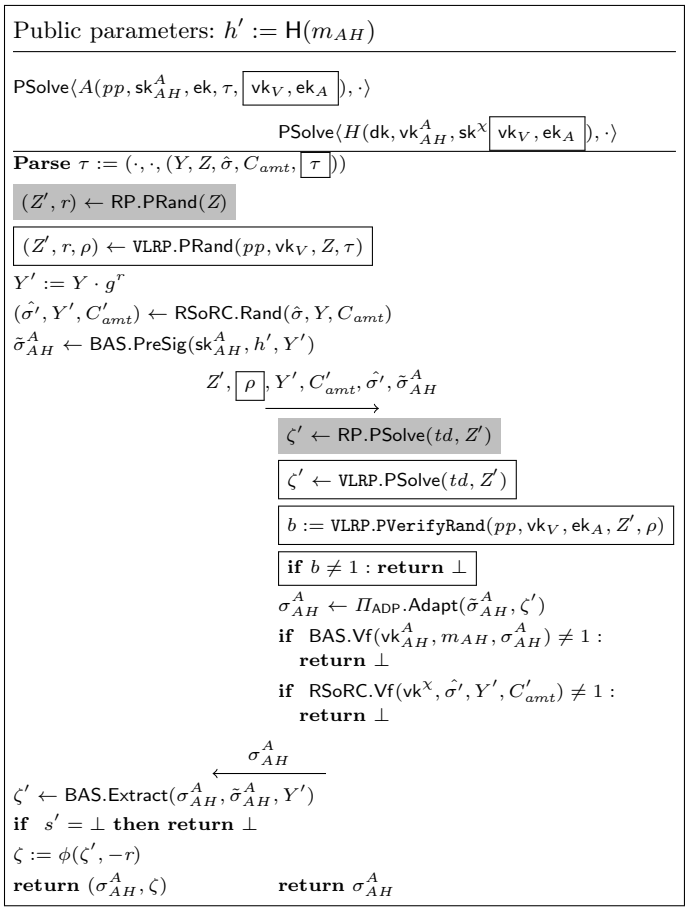

Fig. 12: Continuation from Fig. 11.

Proof. Claim. Let Bad_1 be the event that the game in $\mathcal{H}_1$ aborts. Assume that the randomizable puzzle scheme RP is secure. Then $\Pr[\mathsf{Bad}_1] \leq \mathsf{negl}(n)$.

Proof. Assume, for the sake of contradiction, that the claim does not hold. This implies that there exists an adversary $\mathcal{A}$ such that $\mathcal{H}_1$ aborts with probability higher than $\mathsf{negl}(n)$. We can use $\mathcal{A}$ to build an adversary $\mathcal{B}$ that breaks the security of the randomizable puzzle scheme RP as follows: On input the pair (pp_{RP}, Z), $\mathcal{B}$ executes $crs \leftarrow \mathsf{NIZK.CrsGen}(1^n)$, $(\mathsf{vk}_V, \mathsf{ek}_V; \mathsf{sk}_V, \mathsf{dk}_V) \leftarrow \mathtt{PVerifySetup}(1^n)$, $(\mathsf{ek}_A, \mathsf{dk}_A) \leftarrow \mathtt{PAuditSetup}(1^n)$ and $\sigma \leftarrow \mathsf{DS.Sign}(\mathsf{sk}_V, Z)$. Then $\mathcal{B}$ calls $\mathcal{A}$ on input $((pp_{\mathsf{RP}}, crs), \mathsf{vk}_V, \mathsf{ek}_V, \mathsf{ek}_A, Z, \sigma)$, that in turns returns (Z^*, τ^*, ζ^*). Finally $\mathcal{B}$ outputs ζ^*.

Every time that $\mathcal{A}$ queries the oracle $\mathtt{OPGen}(\zeta)$, $\mathcal{B}$ simply honestly executes $\mathtt{PGen}(pp, \mathsf{sk}_V, \zeta)$ and forwards the output to $\mathcal{A}$.

$\mathsf{VLRP\text{-}Security}^{\mathcal{A}}_{\Pi_{\mathsf{VLRP}}}(1^n)$

$\mathcal{Q} := \emptyset$
$(pp, td) \leftarrow \mathsf{PSetup}(1^n)$
$(\mathsf{vk}_V, \mathsf{ek}_V, \mathsf{sk}_V, \mathsf{dk}_V) \leftarrow \mathtt{PVerifySetup}(1^n)$
$(\mathsf{ek}_A, \mathsf{dk}_A) \leftarrow \mathtt{PAuditSetup}(1^n)$
$\zeta \leftarrow_{\$} \mathcal{S}$
$(Z, \tau) \leftarrow \mathtt{PGen}(pp, \mathsf{sk}_V, \zeta)$
$(Z^*, \tau^*, \zeta^*) \leftarrow \mathcal{A}^{\mathsf{OPGen}}(pp, \mathsf{vk}_V, \mathsf{ek}_V, \mathsf{ek}_A, Z, \tau)$
$b_0 := \mathtt{PVerifyTag}(pp, \mathsf{vk}_V, Z^*, \tau^*)$
$b_1 := Z^* \notin \mathcal{Q} \wedge Z^* \neq Z$
$b_2 := \zeta^* = \zeta$
If b_2 then **abort**
return $(b_0 \wedge b_1) \vee b_2$

$\mathsf{OPGen}(\zeta)$

$(Z, \tau) \leftarrow \mathtt{PGen}(pp, \mathsf{sk}_V, \zeta)$
$\mathcal{Q} := \mathcal{Q} \cup Z$
return (Z, τ)

Fig. 13: Hybrid $\mathcal{H}_1$ for the game $\mathsf{VLRP\text{-}Security}^{\mathcal{A}}_{\Pi_{\mathsf{VLRP}}}$.

Our adversary $\mathcal{B}$ faithfully simulates the $\mathcal{H}_1$ experiment to $\mathcal{A}$. Moreover, $\mathcal{B}$ is efficient. Finally, whenever $\mathcal{H}_1$ aborts, it must be the case that b_2 holds, meaning that $\zeta^* = \zeta$ is a valid solution to the puzzle Z. Given that $\mathcal{H}_1$ aborts with probability higher than $\mathsf{negl}(n)$, $\mathcal{B}$ wins the security game for the randomizable puzzle with the same (non-negligible) probability as $\mathcal{A}$, which is a contradiction to the assumption of our lemma.

Lemma 2. *Assume that the digital signature scheme* DS *is EUF-CMA. For all PPT adversaries* $\mathcal{A}$*, it holds that our construction is secure in* $\mathcal{H}_1$*, i.e.,* $\Pr[\mathsf{VLRP\text{-}Security}^{\mathcal{A}}_{\Pi_{\mathsf{VLRP}}}(1^n)_{\mathcal{H}_1} = 1] = \mathsf{negl}(n)$.

Proof. Assume, for the sake of contradiction, that our construction is not secure in $\mathcal{H}_1$. Then, there exists an adversary $\mathcal{A}$ that wins $\mathcal{H}_1$ with non-negligible probability. Then, we can build an adversary $\mathcal{B}$ that breaks the EUF-CMA of DS, as follows. On input pk, $\mathcal{B}$ computes $((pp_{\mathsf{RP}}, crs), td) \leftarrow \mathsf{PSetup}(1^n)$, $(\mathsf{dk}_V, \mathsf{ek}_V) \leftarrow \mathsf{PKE.KeyGen}(1^n)$, $(\mathsf{ek}_A, \mathsf{dk}_A) \leftarrow \mathtt{PAuditSetup}(1^n)$, $\zeta \leftarrow_{\$} \mathcal{S}$, $Z \leftarrow \mathsf{RP.PGen}(\mathsf{pp}_{\mathsf{RP}}, \zeta)$. Then, $\mathcal{B}$ queries its oracle $\mathsf{OSign}(Z)$ obtaining τ. Finally, $\mathcal{B}$ calls $\mathcal{A}$ on input $((pp_{\mathsf{RP}}, td), \mathsf{pk}, \mathsf{ek}_V, \mathsf{ek}_A, Z, \tau)$. When $\mathcal{A}$ outputs (Z^*, τ^*, ζ^*), $\mathcal{B}$ outputs (Z^*, τ^*) to its challenger.

Every time that $\mathcal{A}$ queries the oracle $\mathsf{OPGen}(\zeta')$, $\mathcal{B}$ processes this query by computing $Z' \leftarrow \mathsf{RP.PGen}(\mathsf{pp}_{\mathsf{RP}}, \zeta')$, queries its oracle $\mathsf{OSign}(Z')$ to obtain τ', and forwards the pair (Z', τ') to $\mathcal{A}$.

The reduction is efficient and faithful. In particular, note that $Z^* \neq Z$ according to b_1, and therefore $\mathcal{B}$ still wins the EUF-CMA game even if its challenger contains Z in its query set. The probability of $\mathcal{B}$ winning the game is the same as $\mathcal{A}$, therefore reaching a contradiction.

E.3 Proof of VLRP Unlinkability

Proof. Consider the following hybrids:

Hybrid $\mathcal{H}_0$: This hybrid is the same as the VLRP unlinkability game with bit $b = 0$.

Hybrid $\mathcal{H}_1$: This hybrid works exactly as $\mathcal{H}_0$ but the challenger changes PRand's calls to NIZK.Prove to instead call the corresponding simulator of the NIZK.

Hybrid $\mathcal{H}_2$: This hybrid works exactly as $\mathcal{H}_1$ but the challenger changes the encryption call $ct_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_A, \mathsf{PKE.Enc}(\mathsf{ek}_V, Z_0; r_V); r_A)$ to $ct_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_A, \mathsf{PKE.Enc}(\mathsf{ek}_V, Z_1; r_V); r_A)$.

Hybrid $\mathcal{H}_3$: This hybrid works exactly as $\mathcal{H}_2$ with bit $b = 1$.

Hybrid $\mathcal{H}_4$: This hybrid works exactly as $\mathcal{H}_3$ but the challenger changes PRand's calls to the simulator of NIZK to instead call NIZK.Prove. This hybrid is the same as the VLPR unlinkability game with bit $b = 1$.

Let $\mathsf{VLRP\text{-}Unlinkability}^{\mathcal{A}}_{\Pi_{\mathtt{VLRP}}}(1^n)_{\mathcal{H}_i}$ be the game $\mathsf{VLRP\text{-}Unlinkability}^{\mathcal{A}}_{\Pi_{\mathtt{VLRP}}}$ as defined in hybrid $\mathcal{H}_i$. The proof for this theorem follows from Lemmas 3 to 6 below.

Lemma 3. *Assume that* NIZK *is zero-knowledge. For all PPT adversaries* $\mathcal{A}$*, it holds that*

$$\Pr[\mathsf{VLRP\text{-}Unlinkability}^{\mathcal{A}}_{\Pi_{\mathtt{VLRP}}}(1^n)_{\mathcal{H}_0} = 1] \leq \Pr[\mathsf{VLRP\text{-}Unlinkability}^{\mathcal{A}}_{\Pi_{\mathtt{VLRP}}}(1^n)_{\mathcal{H}_1} = 1] + \mathsf{negl}(n)$$

Proof. The lemma follows from the zero-knowledge property of NIZK. Assume, for the sake of contradiction, that there exists a PPT adversary $\mathcal{A}$ that can distinguish between the two executions. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break the zero-knowledge of NIZK as follows:

- $\mathcal{B}$ initializes its challenger, who will flip a bit and decide if it uses NIZK.Prove or the simulator $\mathcal{S}$. The challenger sets the *crs* that will be used in proofs.
- $\mathcal{B}$ executes $(pp, td) \leftarrow \mathsf{PSetup}(1^n)$, $(\mathsf{ek}_A, \mathsf{dk}_A) \leftarrow \mathtt{PAuditSetup}(1^n)$ and calls $\mathcal{A}$ on input (pp, td, ek_A).
- $\mathcal{A}$ returns $((Z_0, \zeta_0, \tau_0), (Z_1, \zeta_1, \tau_1), \mathsf{vk}_V)$.
- $\mathcal{B}$ executes PRand on input $(pp, \mathsf{ek}_A, \mathsf{ek}_V, \mathsf{vk}_V, Z_0, \tau_0)$. This call to PRand is executed as defined in our construction except for the step where $\mathcal{B}$ queries its challenger with the corresponding inputs to get either a honestly created proof or a simulated one. In the end, $\mathcal{B}$ obtains a tuple (Z', r, ρ).
- $\mathcal{B}$ forwards the tuple (pp, td, Z', ρ) to $\mathcal{A}$, who in turn responds with b^*. $\mathcal{B}$ forwards b^* to its challenger.

Our adversary $\mathcal{B}$ faithfully simulates $\mathcal{A}$. To see that, note that conditions b_0 to b_3 make sure that the inputs provided by $\mathcal{A}$ are correct, and consequently the proof provided by the challenge of $\mathcal{B}$ is correct. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Finally, if $\mathcal{A}$ can distinguish between the two hybrids, then so can $\mathcal{B}$ because the only difference is the computation of the zero-knowledge proof. However, this contradicts the assumption that NIZK is zero-knowledge.

Lemma 4. *Assume that* RP *is IND-CPA and* RP *is randomizable. For all PPT adversaries* $\mathcal{A}$*, it holds that*

$$\Pr[\mathsf{VLRP\text{-}Unlinkability}^{\mathcal{A}}_{\Pi_{\mathsf{VLRP}}}(1^n)_{\mathcal{H}_1} = 1] \leq \\ \Pr[\mathsf{VLRP\text{-}Unlinkability}^{\mathcal{A}}_{\Pi_{\mathsf{VLRP}}}(1^n)_{\mathcal{H}_2} = 1] + \mathsf{negl}(n)$$

Proof. The lemma follows from the IND-CPA property of PKE. Assume, for the sake of contradiction, that there exists a PPT adversary $\mathcal{A}$ that can distinguish between the two executions. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break the IND-CPA property of the PKE as follows:

- $\mathcal{B}$ obtains a public key pk from its challenger, that is used hereby as ek_A.
- $\mathcal{B}$ executes $(pp, td) \leftarrow \mathsf{PSetup}(1^n)$ and calls $\mathcal{A}$ on input (pp, td, ek_A).
- $\mathcal{A}$ returns $((Z_0, \zeta_0, \tau_0), (Z_1, \zeta_1, \tau_1), \mathsf{vk}_V, \mathsf{ek}_V)$.
- $\mathcal{B}$ executes PRand on input $(pp, \mathsf{ek}_A, \mathsf{ek}_V, \mathsf{vk}_V, Z_0, \tau_0)$. This call to PRand is executed as defined in our construction except for changing the encryption call $ct_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_A, \mathsf{PKE.Enc}(\mathsf{ek}_V, Z_0; r_V); r_A)$ to the following $ct_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_A, \mathsf{PKE.Enc}(\mathsf{ek}_V, Z_1; r_V); r_A)$, and where $\mathcal{B}$ uses the simulator of the NIZK scheme to obtain a simulated proof (as in $\mathcal{H}_1$). In the end, $\mathcal{B}$ obtains a tuple (Z', r, ρ).
- $\mathcal{B}$ forwards the tuple (pp, td, Z', ρ) to $\mathcal{A}$, who in turn responds with b^*. $\mathcal{B}$ forwards b^* to its challenger.

Our adversary $\mathcal{B}$ faithfully simulates $\mathcal{A}$. To see that, note that conditions b_0 to b_3 make sure that the inputs provided by $\mathcal{A}$ are correct, and consequently the proof provided by the challenge of $\mathcal{B}$ is correct. Furthermore, since RP is randomizable, the simulated proof is correct despite changing the encryption call $ct_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_A, \mathsf{PKE.Enc}(\mathsf{ek}_V, Z_0; r_V); r_A)$ to the following $ct_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_A, \mathsf{PKE.Enc}(\mathsf{ek}_V, Z_1; r_V); r_A)$. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Finally, if $\mathcal{A}$ can distinguish between the two hybrids, then so can $\mathcal{B}$ because the only difference is the computation of the ciphertext. However, this contradicts the assumption that PKE is IND-CPA.

Lemma 5. *Assume that* RP *is randomizable and private. For all PPT adversaries* $\mathcal{A}$*, it holds that*

$$\Pr[\mathsf{VLRP\text{-}Unlinkability}^{\mathcal{A}}_{\Pi_{\mathsf{VLRP}}}(1^n)_{\mathcal{H}_2} = 1] \leq \\ \Pr[\mathsf{VLRP\text{-}Unlinkability}^{\mathcal{A}}_{\Pi_{\mathsf{VLRP}}}(1^n)_{\mathcal{H}_3} = 1] + \mathsf{negl}(n)$$

Proof. The lemma follows from the privacy property of RP. Assume, for the sake of contradiction, that there exists a PPT adversary $\mathcal{A}$ that can distinguish between the two executions. We can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break the privacy property of RP as follows:

- $\mathcal{B}$ obtains a $(\mathsf{pp}, \mathsf{td})$ from its challenger.
- $\mathcal{B}$ executes $(\mathsf{ek}_A, \mathsf{dk}_A) \leftarrow \mathsf{PAuditSetup}(1^n)$ and $crs \leftarrow \mathsf{NIZK.CrsGen}(1^n)$, and then calls $\mathcal{A}$ on input $((\mathsf{pp}, crs), \mathsf{td}, \mathsf{ek}_A)$.
- $\mathcal{A}$ returns $((Z_0, \zeta_0, \tau_0), (Z_1, \zeta_1, \tau_1), \mathsf{vk}_V, \mathsf{ek}_V)$.
- $\mathcal{B}$ sends $((Z_0, \zeta_0), (Z_1, \zeta_1))$ to its challenger and receives Z'. $\mathcal{B}$ then executes PRand on input $(pp, \mathsf{ek}_A, \mathsf{ek}_V, \mathsf{vk}_V, Z_1, \tau_1)$, except that it uses Z' instead of invoking $\mathsf{RP.PRand}$. This call to PRand still uses the simulator of the NIZK scheme to obtain a simulated proof (as in $\mathcal{H}_1$). In the end, $\mathcal{B}$ obtains a pair (Z', ρ).
- $\mathcal{B}$ forwards the tuple $((\mathsf{pp}, crs), \mathsf{td}, Z', \rho)$ to $\mathcal{A}$, who in turn responds with b^*. $\mathcal{B}$ forwards b^* to its challenger.

Our adversary $\mathcal{B}$ faithfully simulates $\mathcal{A}$. To see that, note that conditions b_0 to b_3 make sure that the inputs provided by $\mathcal{A}$ are correct, and consequently the proof provided by the challenge of $\mathcal{B}$ is correct. Furthermore, since RP is randomizable, the simulated proof is correct. Moreover, it is easy to see that $\mathcal{B}$ is a PPT algorithm. Finally, if $\mathcal{A}$ can distinguish between the two hybrids, then so can $\mathcal{B}$ because the only difference is the computation of Z'. However, this contradicts the assumption that RP is private.

Lemma 6. *Assume that* NIZK *is zero-knowledge. For all PPT adversaries* $\mathcal{A}$*, it holds that*

$$\Pr[\mathsf{VLRP\text{-}Unlinkability}^{\mathcal{A}}_{\Pi_{\mathsf{VLRP}}}(1^n)_{\mathcal{H}_3} = 1] \leq \Pr[\mathsf{VLRP\text{-}Unlinkability}^{\mathcal{A}}_{\Pi_{\mathsf{VLRP}}}(1^n)_{\mathcal{H}_4} = 1] + \mathsf{negl}(n)$$

Proof. The proof of this lemma is essentially identical to that of Lemma 3.

E.4 Proof of VLRP Auditability

Proof. Consider the following hybrids:

Hybrid $\mathcal{H}_0$: The same as the VLRP Auditability game in Fig. 8.

Hybrid $\mathcal{H}_1$: VLRP Auditability game modified as follows:

$\mathsf{VLRP\text{-}Auditability}^{\mathcal{A}}_{\Pi_{\mathsf{VLRP}}}(1^n)$

$(pp, td) \leftarrow \mathsf{PSetup}(1^n)$
$(\mathsf{vk}_V, \mathsf{ek}_V, \mathsf{sk}_V, \mathsf{dk}_V) \leftarrow \mathtt{PVerifySetup}(1^n)$
$(\mathsf{ek}_A, \mathsf{dk}_A) \leftarrow \mathtt{PAuditSetup}(1^n)$
$\zeta \leftarrow_{\$} \mathcal{S}$
$(Z, \tau) \leftarrow \mathsf{PGen}(pp, \mathsf{sk}_V, \zeta)$
$(Z', \rho, \delta) \leftarrow \mathcal{A}^{\mathsf{OPGen}}(pp, \mathsf{vk}_V, \mathsf{ek}_V, \mathsf{ek}_A, Z, \tau)$
$Z^* \leftarrow \mathtt{PAudit}(\mathsf{dk}_V, Z', \rho, \delta)$
$(\mathsf{pp}_{\mathsf{RP}}, crs) \leftarrow pp$
$(ct_{\mathsf{PKE}}, \pi_{\mathtt{PRand}}) \leftarrow \rho$
$(\tilde{Z}, \cdot, \cdot, \cdot, \cdot) \leftarrow \mathsf{Ext}^{\mathcal{A}}(crs, (\mathsf{vk}_V, Z', ct_{\mathsf{PKE}}, \mathsf{ek}_V, \mathsf{ek}_A, \mathsf{pp}_{\mathsf{RP}}), \pi_{\mathtt{PRand}})$
if $\tilde{Z} \neq Z^*$ **abort**
$b_0 := \mathtt{PVerifyRand}(pp, \mathsf{ek}_A, \mathsf{ek}_V, \mathsf{vk}_V, Z', \rho)$
$b_1 := (Z \neq Z^*) \wedge (Z^* \notin \mathcal{Q})$
$b_2 := (Z^* \neq \bot)$
return $b_0 \wedge b_1 \wedge b_2$

Let $\mathsf{VLRP\text{-}Auditability}^{\mathcal{A}}_{\Pi_{\mathsf{VLRP}}}(1^n)_{\mathcal{H}_i}$ be the game $\mathsf{VLRP\text{-}Auditability}^{\mathcal{A}}_{\Pi_{\mathsf{VLRP}}}$ as defined in hybrid $\mathcal{H}_i$. The proof for this theorem follows from Lemmas 7 and 8 below.

Lemma 7. *Assume that the encryption scheme* PKE *is correct and the* NIZK *scheme achieves soundness. For all PPT adversaries* $\mathcal{A}$*, it holds that*

$$\Pr[\mathsf{VLRP\text{-}Auditability}^{\mathcal{A}}_{\Pi_{\mathsf{VLRP}}}(1^n)_{\mathcal{H}_0} = 1] \leq \Pr[\mathsf{VLRP\text{-}Auditability}^{\mathcal{A}}_{\Pi_{\mathsf{VLRP}}}(1^n)_{\mathcal{H}_1} = 1] + \mathsf{negl}(n)$$

Proof. Claim. Let Bad_2 be the event that the game in $\mathcal{H}_1$ aborts. Assume that the encryption scheme PKE is correct and the NIZK scheme achieves soundness. Then $\Pr[\mathsf{Bad}_2] \leq \mathsf{negl}(n)$.

Proof. This claim is immediate. The soundness of the NIZK scheme guarantees that ct_{PKE} is a double-encryption of $\tilde{Z}$ and that the first element of δ is the decryption of ct_{PKE}, i.e., encryption of Z^* with all but $\mathsf{negl}(n)$ probability. The correctness of PKE further guarantees that $\tilde{Z} = Z^*$.

Lemma 8. *Assume that the digital signature scheme* DS *is EUF-CMA. For all PPT adversaries* $\mathcal{A}$*, it holds that i.e.,* $\Pr[\mathsf{VLRP\text{-}Auditability}^{\mathcal{A}}_{\Pi_{\mathsf{VLRP}}}(1^n)_{\mathcal{H}_1} = 1] = \mathsf{negl}(n)$.

Proof. The proof of this lemma is essentially identical to that of Lemma 2.

E.5 Proof of Relation of Security Notions

Theorem 5 (VLRP Security implies RP Security). *The security of* VLRP *as defined in Definition 18 implies the security of the* RP *as defined in Definition 4.*

Proof. Assume, for the sake of contradiction, that the implication does not hold. Assume then an adversary $\mathcal{A}$ against the security of the RP scheme that wins with non-negligible probability. We then can build an adversary $\mathcal{B}$ against the security of VLRP scheme as follows.

- On input $(pp, \mathsf{vk}_V, \mathsf{ek}_V, \mathsf{ek}_A, Z, \tau)$, $\mathcal{B}$ parses $(pp^*, crs) \leftarrow pp$ and invokes $\mathcal{A}$ on input (pp^*, Z).
- $\mathcal{B}$ obtains ζ from $\mathcal{A}$. Moreover, $\mathcal{B}$ obtains a fresh Z^* and τ^*.
- Finally, $\mathcal{B}$ outputs the tuple (Z^*, τ^*, ζ) to its challenger.

The algorithm $\mathcal{B}$ faithfully simulates $\mathcal{A}$ and it is efficient. Moreover, while the pair (Z^*, τ^*) are not going to pass the check b_0 and b_1 with high probability, the condition b_2 holds with the same probability as $\mathcal{A}$ wins the RP security experiment, therefore reaching a contradiction.

E.6 Proof of Relation of Privacy Notions

Theorem 6 (VLRP Privacy implies RP Privacy). *The privacy of* VLRP *as defined in Definition 19 implies the privacy of the* RP *as defined in Definition 5.*

Proof. Assume, for the sake of contradiction, that the implication does not hold. Assume then an adversary $\mathcal{A}$ against the privacy of the RP scheme that wins with non-negligible probability. We then can build an adversary $\mathcal{B}$ against the privacy of VLRP scheme as follows.

- On input (pp, td, ek_A), algorithm $\mathcal{B}$ parses $(pp^*, crs) \leftarrow pp$, and computes $(\mathsf{vk}_V, \mathsf{ek}_V, \mathsf{sk}_V, \mathsf{dk}_V) \leftarrow \mathtt{PVerifySetup}(1^n)$ and invokes $\mathcal{A}$ on input (pp^*, td).
- $\mathcal{B}$ obtains two pairs $(Z_0, \zeta_0), (Z_1, \zeta_1)$. Then $\mathcal{B}$ computes the following: $\hat{Z}_0, \tau_0 \leftarrow \mathtt{PGen}(pp, \mathsf{sk}_V, \zeta_0)$ and $\hat{Z}_1, \tau_1 \leftarrow \mathtt{PGen}(pp, \mathsf{sk}_V, \zeta_1)$. Finally $\mathcal{B}$ outputs to its challenger $((\hat{Z}_0, \zeta_0, \tau_0), (\hat{Z}_1, \zeta_0, \tau_0), \mathsf{vk}_V)$.
- $\mathcal{B}$ is invoked again on input (pp, td, Z'_b, ρ'_b). Then, $\mathcal{B}$ parses $(pp^*, crs) \leftarrow pp$ and invokes $\mathcal{A}$ on input (pp^*, td, Z'_b)
- $\mathcal{A}$ answers with a bit b^*, which $\mathcal{B}$ forwards to its challenger.

It is easy to see that $\mathcal{B}$ is efficient. To see that $\mathcal{B}$ faithfully simulates $\mathcal{A}$, see that the puzzles that $\mathcal{B}$ forwards to its challenger have the same solutions that those received from $\mathcal{A}$. Moreover, the puzzles output by ρ are correctly randomized versions of those previously created by $\mathcal{B}$. Therefore, from $\mathcal{A}$, the challenge puzzle Z'_b is a correctly randomized puzzle of either a puzzle for solution ζ_0 or a puzzle for solution ζ_1, as specified in the privacy game for RP. Finally, $\mathcal{B}$ wins the privacy game for VLRP with the same probability as $\mathcal{A}$ wins the privacy game for RP, which is a contradiction.