

Issuer-Independent Decentralized Exchange for E-Cash

Roozbeh Sarenche and Omid Mirzamohammadi

COSIC, KU Leuven, Leuven, Belgium
roozbeh.sarenche@esat.kuleuven.be
omid.mirzamohammadi@esat.kuleuven.be

Abstract. Despite decades of cryptographic advances, electronic cash (e-cash) remains largely confined to payments and lacks the broader decentralized financial applications built around cryptocurrencies. As a step toward broadening its application layer, we present an issuer-independent e-cash exchange in which a liquidity provider holds inventories of two independently issued e-cash currencies and users trade between them without requiring either issuer to participate in individual swaps. Our design combines a blockchain settlement layer, which automates the exchange and makes its execution publicly verifiable, with an exchange-compatible e-cash protocol. Classical e-cash must be adapted to this setting in two respects: payments must become publicly verifiable by the exchange, and the additional privacy risks created by public execution must be addressed. We formalize these new privacy requirements and construct a compact e-cash realization with publicly verifiable payments and aggregate anonymous reissue. We prove the security and privacy properties of the resulting exchange protocol under the stated cryptographic and issuer-functionality assumptions.

Keywords: E-cash, automated market maker, decentralized exchange, anonymous reissue, zero knowledge, privacy

1 Introduction

Since Chaum’s blind-signature proposal [12], electronic cash (e-cash) has developed into a rich cryptographic primitive for private digital payments. A user can obtain spendable value from an issuer, make payments without revealing the wallet from which they originate, and still be held accountable for double spending. Modern constructions further support features such as compact wallets, divisible payments, and threshold issuance [14,7,9,10,22]. Yet this cryptographic maturity has not translated into a practical deployment comparable to that of cryptocurrencies. Many factors contribute to this gap. One important difference is the lack of permissionless applications built on e-cash: cryptocurrencies support broad financial ecosystems, while e-cash remains primarily a payment primitive.

Among the most important permissionless applications built on cryptocurrencies are decentralized exchanges, particularly automated market makers (AMMs),

where liquidity providers supply assets to a pool and users can exchange between them according to a publicly defined pricing rule without relying on a trusted intermediary. In this work, we ask whether an analogous exchange can be built for two independently issued e-cash currencies. Specifically, can a liquidity provider offer liquidity in both currencies and allow users to exchange between them through a public protocol, while the two issuing banks remain entirely outside the operation of individual swaps?

Prior e-cash systems and implementations have already explored cross-domain payments and exchange, but in the prominent existing designs the issuing infrastructure remains involved in the conversion path. For example, Cashu cross-mint transfers and Fedimint cross-federation payments use Lightning as an interoperability layer, with the source and destination mint or federation infrastructures participating in the movement of value [11,17]. Our goal is fundamentally different: the issuers remain outside the exchange protocol altogether. Instead, an independent liquidity provider already holding e-cash wallets from both issuer domains supplies the two inventories and directly facilitates exchange between users of the two currencies.

A public blockchain can coordinate such an issuer-independent exchange. However, a practical and secure blockchain-based implementation must overcome two key challenges:

Settlement security. Unlike ordinary on-chain tokens, e-cash cannot simply be escrowed and atomically transferred by a smart contract. Once one party releases a valid e-cash payment, the counterparty may attempt to withhold its own. The exchange must therefore ensure that an honest party either receives the promised counter-payment or is compensated for non-delivery.

Privacy. Public exchange introduces a privacy problem that does not arise in classical e-cash considered as isolated payments. The blockchain exposes structured information linking the two legs of a swap, while the issuers later observe the corresponding settlement records. Colluding issuers can combine these views to link public payments to settlements and potentially recover the payer’s identity, which remains hidden in classical e-cash. The exchange must therefore preserve privacy under this composed public-and-issuer view.

In this paper, we overcome these challenges by constructing an issuer-independent e-cash exchange from two complementary layers. The *blockchain layer* implements the exchange logic: an *LP* holding e-cash from both issuer domains supplies the two inventories, while a *user* exchanges one e-cash currency for the other by submitting swap requests to the pool. The blockchain records these requests, fixes their execution order and quoted outputs, verifies the submitted payment proofs, enforces completion deadlines, and manages collateral according to the settlement outcome. The *e-cash layer* introduces an exchange-compatible cryptographic e-cash construction that enables these operations without involving the issuers in individual swaps, while preserving user privacy under the stronger composed threats created by public exchange.

Our contributions are threefold.

- **Issuer-independent e-cash exchange.** We define the functionality required from an e-cash system to support public exchange and construct an AMM in which an independent LP supplies inventories of two separately issued e-cash currencies, while the issuers remain outside individual swaps. We develop two settlement modes with different concurrency-collateral tradeoffs. The *sequential* mode permits only one unresolved price-changing swap at a time, improving user-friendliness at the cost of lower concurrency. The *concurrent* mode admits multiple outstanding swaps by fixing their quotes at admission and using user-side collateral to cover losses that a failed earlier request may impose on later swaps. In both modes, the blockchain coordinates pricing, payment verification, deadlines, and settlement; if the LP fails to deliver, the user is compensated from LP-side collateral, while the concurrent mode additionally protects the LP against user aborts through user-side collateral.
- **New privacy notions for public e-cash exchange.** We formalize three complementary notions that arise in exchange-compatible e-cash protocols: *swap-settlement unlinkability* (SSU), which hides the link between a public payment and its later settlement at the issuer; *aggregate-source unlinkability* (ASU), which hides which source payments contribute to an aggregate reissue; and *linked-swap identity privacy* (LSIP), which protects the user’s registered identity even when the swap-settlement link is known. Our game-based definitions consider colluding semi-honest issuers and, in the strongest setting, additionally allow the semi-honest LP to collude.
- **Exchange-compatible compact e-cash.** We extend compact e-cash [22] with succinct publicly verifiable payment proofs and *aggregate anonymous reissue* (AAR), making it compatible with the public exchange setting. A receiver can aggregate arbitrary coins from one or more payments into a fresh compact wallet while hiding their source payments and partition. The construction preserves spent-state lookup and double-spender identification, and we prove the properties required for SSU, ASU, and LSIP.

2 Related Work

E-cash foundations. Chaum introduced blind signatures for untraceable payments [12], and Chaum–Fiat–Naor established the canonical offline double-spending pattern based on serial collisions [14]. Compact e-cash derives many spendable coins from a short wallet credential [7], while divisible constructions support efficient variable-value spending [9,10]. Rial and Piotrowska add threshold issuance and provide compact and divisible instantiations with detailed security analyses and implementations [22]; their compact construction is our starting point. Subsequent e-cash research has also explored transferability, refresh, and payee privacy. Transferable e-cash moves ownership cryptographically before redemption [4], while payee-private and refresh mechanisms allow spent value to be converted into freshly randomized value [13,20,16]. Our aggregate anonymous reissue (AAR) follows this latter high-level pattern, but allows coin-level subsets from multiple publicly verifiable compact-e-cash payments to be combined into a single fresh wallet while preserving later double-spend tracing.

Exchange, interoperability, and fair transfer. Existing cross-domain e-cash systems typically keep issuer or federation infrastructure in the conversion path: Cashu and Fedimint rely on Lightning-based interoperability [11,17], while Taler uses an exchange-centered architecture [16]. Our goal is instead to keep issuers outside individual swaps. We therefore build on fair-exchange ideas developed in coin ripping, optimistic fair exchange, endorsed e-cash, blockchain-based fair exchange, and atomic swaps [18,3,8,23,15]. The key challenge is that e-cash itself cannot be escrowed or atomically transferred by the blockchain, so the exchange must provide public coordination and economic protection against non-delivery without taking custody of either e-cash leg.

Decentralized exchanges and AMMs. Automated market makers (AMMs) are a central mechanism for decentralized exchanges, replacing order-book matching with algorithmic pricing against liquidity pools [19,24,5]. Constant-function market makers (CFMMs) capture a broad class of AMMs through reserve-dependent trading functions [2]; among them, constant-product market makers (CPMMs) are one of the most widely deployed designs, including in Uniswap [1]. Our exchange layer is formulated for a general AMM pricing, while the concurrent settlement and collateral analysis specializes to the CPMM.

3 Model, Interface, and Security Goals

This section formalizes the setting and security requirements of our exchange. We first describe the participating entities, their roles, and the system assumptions. We then define the exchange-compatible e-cash interface, including the operations required for public payment verification, aggregation, and settlement. Next, we formalize the privacy goals that arise when e-cash payments are exposed through a public exchange protocol. Finally, we summarize the cryptographic assumptions underlying our construction and security analysis.

3.1 Entities and exchange model.

We consider two independent issuers, B_0 and B_1 , issuing currencies E_0 and E_1 , respectively, with separate issuer state. A liquidity provider LP holds spendable wallets in both currencies and operates an exchange over a public blockchain $\mathcal{B}$ and a data-availability (DA) service $\mathcal{D}$, according to a specified pricing rule. Users hold e-cash issued by either bank and trade against LP according to the rule specified by the exchange: the user pays LP in one currency, and LP pays the user from its own wallet in the other. The blockchain verifies payment validity and enforces the exchange logic, while $\mathcal{D}$ ensures availability of the corresponding payment payloads for settlement. We make the following assumptions.

Blockchain and data availability. We assume deterministic contract execution, finality, and bounded liveness: a valid transaction submitted at time t is finalized by $t + \Delta_{\mathcal{B}}^{\text{live}}$, except with negligible probability. Larger payment payloads are

handled by the data-availability service $\mathcal{D}$. Publishing M returns a binding digest $d \leftarrow \text{Put}(M)$, and $M \leftarrow \text{Get}(d)$ retrieves the corresponding payload. Within $\Delta_{\mathcal{D}}^{\text{live}}$, $\text{VerifyDA}(d) = 1$ certifies that the payload is available for settlement. The contract verifies payment proofs and data availability separately.

Issuer-side settlement and funded double-spending recovery. Public verification establishes well-formedness, not freshness. We assume a *funded double-spending recovery functionality*: every cryptographically valid non-replay payment presented by an honest receiver is honored in full. First presentations consume fresh value; conflicting presentations trigger identification and funded replacement, while exact replays authorize no additional value. Recovery is backed by a pre-established payer deposit or equivalent issuer-enforced liability mechanism sufficient for all conflicting claims arising in executions covered by our settlement theorems. Identification alone does not fund a claim. The exchange itself does not guarantee solvency beyond this funded exposure bound.

3.2 Exchange-Compatible E-Cash

For each issuer B_b , we use a standard e-cash system in which the ordinary spending functionality is replaced by an exchange-specific interface:

$$\text{EC}_b^{\text{ex}} = (\text{Setup}_b, \text{Register}_b, \text{Issue}_b, \text{Exch}_b, \text{Settle}_b, \text{Identify}_b).$$

The operations Setup_b , Register_b , and Issue_b provide the usual issuer setup, holder registration, and wallet issuance functionality. Settle_b verifies that a receipt is eligible for settlement and either redeems its value to the receiver or reissues the accepted value as a fresh wallet. Identify_b recovers the registered payer from valid evidence of conflicting spends.

The exchange interface replaces ordinary Spend_b and is defined as

$$\text{Exch}_b = (\text{RKeyGen}, \text{SpendEx}_b, \text{PubVf}_b, \text{Aggregate}_b).$$

RKeyGen generates the exchange keys used to bind a payment to its designated receiver. SpendEx_b spends from an E_b wallet and produces an exchange payment receipt $\text{rcpt}_{\text{pay}} = (x_{\text{pay}}, M_{\text{pay}}, \Pi_{\text{pay}})$, where x_{pay} is the public statement, M_{pay} is the protected receiver-side settlement payload, and Π_{pay} proves consistency of the receipt. PubVf_b publicly verifies the payment statement and proof without accessing issuer settlement state. Finally, Aggregate_b allows a receiver to privately select coins from one or more valid payment receipts and produce a single aggregate receipt $\text{rcpt}_{\text{agg}} = (x_{\text{agg}}, M_{\text{agg}}, \Pi_{\text{agg}})$ whose total value equals that of the selected coins. The aggregate receipt can then be settled through anonymous reissue, replacing the selected received value with a fresh wallet of the same value.

Appendix A gives the complete syntax, execution roles, and semantics of the exchange-compatible e-cash system; in particular, Subsection A.2 specifies the four algorithms comprising Exch_b .

3.3 Privacy Requirements for Exchange-Compatible E-Cash

A central privacy goal of e-cash is that an issuer, although involved in registration and issuance, should not be able to link a later payment to the registered holder who made it. Exchange compatibility creates a new obstacle to this goal. To make an e-cash payment usable by a decentralized exchange, our interface exposes a publicly verifiable payment transcript and data-availability material on the blockchain, while the corresponding recovered coin records may later appear at an issuer during settlement. These two views create correlation channels that are absent from ordinary payer–merchant e-cash: an issuer may try to associate a settlement with a public swap, two issuers may combine their observations, and an LP may reveal recovery information that makes such an association explicit. We therefore introduce three complementary privacy notions for exchange-compatible e-cash: *swap–settlement unlinkability* (SSU), *aggregate-source unlinkability* (ASU), and *linked-swap identity privacy* (LSIP).

Threat model. Our privacy claims concern the cryptographic information exposed by the e-cash and exchange protocols. The public cryptographic blockchain transcript is visible to the adversary, whereas native blockchain-account identities, transaction timing and block position, network-layer metadata, and other external identity links are outside the model. For SSU and ASU, the two issuers may collude and combine their complete cryptographic views; for LSIP, the LP additionally joins this coalition and reveals its complete cryptographic state. Coalition members are *semi-honest*: the challenger executes their prescribed algorithms, including issuance, settlement, and the seed functionality $\mathcal{F}_{\text{seed}}$ of Appendix C.4, while the adversary receives all of their keys, randomness, and internal state. Our privacy results therefore do not cover actively malicious issuers or an actively malicious LP, for example ones that send malformed issuance or settlement responses, abort selectively, or use deviating keys. This is weaker than e-cash anonymity notions in which the adversary plays the bank, and extending our notions to active coalitions is left to future work. The experiments permit the ordinary adaptive queries of Appendix B before and after one protected challenge; challenge objects and their descendants cannot be submitted through state-changing ordinary oracles. For $X \in \{\text{SSU}, \text{ASU}, \text{LSIP}\}$, let $\text{Adv}_{\Pi, \mathcal{A}}^X(\lambda) = |\Pr[\text{Exp}_{\Pi, \mathcal{A}}^X(1^\lambda) = 1] - 1/2|$, where the experiment outputs 1 iff the adversary correctly guesses its hidden bit β .

Leakage. Each experiment compares two admissible challenge descriptions χ_0, χ_1 under equal explicit leakage. For a challenge description, let b denote the issuer receiving the anonymous reissue and V the value presented for reissue; for LSIP, let σ denote the public swap description fixed by the challenge. We define $\text{Leak}_{\text{SSU}}(\chi) = (b, V)$, $\text{Leak}_{\text{ASU}}(\chi) = (b, V)$, and $\text{Leak}_{\text{LSIP}}(\chi) = (\sigma, V)$. Accordingly, every experiment requires $\text{Leak}_X(\chi_0) = \text{Leak}_X(\chi_1)$. LSIP additionally reveals the actual association between the challenge swap and its issuer-side settlement.

Swap–settlement unlinkability. SSU protects the new link created when records originating from a public exchange payment later appear in an issuer-side anony-

mous settlement. Publishing the payment for exchange verification should not enable the issuers to recognize that same payment at settlement; otherwise, colluding issuers could correlate the public exchange transcript with the two issuer-side settlement legs and use the identities observed by the respective banks to deanonymize the exchange user. Thus, even after pooling their complete views, the issuers should be unable to determine which compatible public payment supplied a particular reissue.

Definition 1 (Swap–Settlement Unlinkability (SSU)). *Let χ_0, χ_1 be admissible SSU challenge descriptions satisfying $\text{Leak}_{\text{SSU}}(\chi_0) = \text{Leak}_{\text{SSU}}(\chi_1)$ and specifying two distinct honestly generated public payment receipts under the same issuer and receiver. After sampling $\beta \leftarrow \{0, 1\}$, the experiment aggregates the selected records prescribed by χ_β using Aggregate_b and submits the resulting rcpt_{agg} to Settle_b for anonymous reissue. The protocol satisfies SSU if $\text{Adv}_{\Pi, \mathcal{A}}^{\text{SSU}}(\lambda)$ is negligible for every PPT adversary given the public cryptographic transcript and the complete views of both issuers.*

Aggregate-source unlinkability. ASU extends this protection to settlements assembled from several public payments. Aggregation should hide not only a direct payment–settlement link but also the internal source organization of the settled value. Consequently, the issuer may learn the aggregate value being reissued, while remaining unable to determine which admissible selection of publicly visible source payments contributed that value.

Definition 2 (Aggregate-Source Unlinkability (ASU)). *Let χ_0, χ_1 be admissible ASU challenge descriptions satisfying $\text{Leak}_{\text{ASU}}(\chi_0) = \text{Leak}_{\text{ASU}}(\chi_1)$ and specifying distinct valid selections from a common family of honestly generated public payment receipts under the same issuer and receiver. After sampling $\beta \leftarrow \{0, 1\}$, the experiment applies Aggregate_b using the selection prescribed by χ_β and submits the resulting rcpt_{agg} to Settle_b for anonymous reissue. The two selections may contain different numbers of contributing source payments. The protocol satisfies ASU if $\text{Adv}_{\Pi, \mathcal{A}}^{\text{ASU}}(\lambda)$ is negligible for every PPT adversary given the public cryptographic transcript and the complete views of both issuers.*

Linked-swap identity privacy. When the LP joins the issuer coalition, its exchange secrets and recovered records may make the swap–settlement association directly available. In this case source unlinkability is no longer the relevant objective. LSIP asks whether the fundamental e-cash identity guarantee survives even under this known linkage: knowing exactly which settlement belongs to a swap should still not reveal which registered user participated in that swap.

Definition 3 (Linked-Swap Identity Privacy (LSIP)). *Let χ_0, χ_1 be admissible LSIP challenge descriptions satisfying $\text{Leak}_{\text{LSIP}}(\chi_0) = \text{Leak}_{\text{LSIP}}(\chi_1)$ and differing only in two distinct registered candidate users. After sampling $\beta \leftarrow \{0, 1\}$, the experiment executes the prescribed swap with the user specified by χ_β , anonymously reissues the prescribed received value, and reveals the resulting swap–settlement association. The protocol satisfies LSIP if $\text{Adv}_{\Pi, \mathcal{A}}^{\text{LSIP}}(\lambda)$ is negligible for*

every PPT adversary given that association, the public cryptographic transcript, the complete views of both issuers, and the LP’s complete cryptographic view.

SSU and ASU protect source association at two levels: SSU hides which compatible single payment supplied a settlement, whereas ASU hides which source selection supplied an aggregate settlement. LSIP addresses the complementary setting in which that association is already known and instead protects the registered identity behind the linked swap. The complete experiments, admissibility conditions, and oracle interfaces are given in Appendix B.

3.4 Cryptographic Assumptions

Our security analysis relies on the properties of the primitives and issuer functionalities specified below. The zero-knowledge proofs used for registration, issuance, payment, and aggregation provide completeness, zero knowledge, and extractability; the commitments used in issuance, reissue, and aggregation are hiding and binding; and the authenticated registration structure provides sound membership proofs. The blind-issued Pointcheval–Sanders (PS) credentials are unforgeable under the corruption model of the underlying e-cash scheme [22]. We further assume discrete-logarithm hardness in the groups used by the construction and XDH in the first pairing group. The privacy proofs additionally rely on pseudorandomness of the compact inverse-exponent outputs and of the DH-derived payment masks, together with collision-resistant hashing and domain separation. The aggregation-coefficient and tracing-challenge hashes are modeled as independent random oracles, and the public randomness and aggregation commitments used in the privacy experiments satisfy the required freshness and entropy conditions. Public parameters are honestly generated and proof-system trapdoors are unavailable to protocol participants. Wallet seeds are jointly generated from a hidden, previously committed holder contribution and a fresh issuer contribution. We explicitly assume the session-bound common-randomness functionality of Appendix C.4. The seed-separation proof uses commitment binding and extraction of completed proofs under Appendix C.2; it does not require extraction before the issuer challenge. These requirements do not follow from credential unforgeability alone. Appendix C gives the formal definitions, experiments, and reduction requirements.

4 Blockchain-Based Exchange Protocol

This section presents the blockchain execution layer of our e-cash exchange. The protocol is defined over a public pricing and state-transition rule and can therefore support different exchange mechanisms. We focus on AMMs and analyze the constant-product market maker (CPMM), where concurrent execution is particularly nontrivial because an unresolved trade can affect the prices quoted to later trades. We consider two modes: sequential execution permits only one unresolved price-changing swap at a time, while concurrent execution fixes each quote at admission, at the cost of requiring user collateral to cover losses caused by cancellation.

4.1 Pool Creation and AMM Pricing

The LP creates a pool by publishing

$$\mathcal{P} = (\text{pid}, E_0, E_1, \text{mode}, F, S^c, \text{PubVf}, \text{fee}, \Theta_{\text{tot}}, Y_{\text{LP}}, \rho_0, \rho_1, D_{\text{LP}}).$$

Here, pid uniquely identifies the pool and E_0, E_1 are the two e-cash currencies being exchanged. The parameter mode specifies whether swaps are processed sequentially or concurrently. The pair (F, S^c) describes the AMM: F is its pricing and state-transition rule and S^c is its current committed state; for a two-currency pool, $S^c = (x, y)$ contains the current reserve values of E_0 and E_1 . PubVf denotes the issuer-specific verifiers and keys; the contract selects the one fixed by the currency direction and payment leg. The parameter fee is the native-cryptocurrency fee charged by and paid to the LP for using the pool, and Y_{LP} is the LP's public exchange key. The parameter Θ_{tot} is the total time allowed for each payment step and is chosen to accommodate both submission and system-processing delays. Let Θ_{sub} denote the window within which a party must submit its payment. To account for DA and blockchain liveness, we require $\Theta_{\text{tot}} \geq \Theta_{\text{sub}} + \Delta_{\mathcal{D}}^{\text{live}} + \Delta_{\mathcal{B}}^{\text{live}}$. The LP also publishes rates ρ_0, ρ_1 , where ρ_b specifies the native-cryptocurrency value assigned to one unit of E_b and is used for both fractional settlement and collateral. The LP deposits a collateral balance D_{LP} of native cryptocurrency into the pool contract. These values are visible to users and may be updated during the pool's lifetime, provided that updates do not affect the rates or collateral already fixed for ongoing exchanges.

Although e-cash payments are discrete, F maintains continuous reserve values. For example, consider a zero-fee CPMM [1,25]. Let the current state be $S^c = (x, y)$ and $K = xy$. If a user provides an integer input a of E_0 , the AMM computes the output $b = y - K/(x + a)$. We write $b^{\text{ec}} = \lfloor b \rfloor$ for the e-cash component and $b^{\text{nat}} = \rho_1(b - \lfloor b \rfloor)$ for the native-currency component. The LP pays b^{ec} units of E_1 in e-cash and b^{nat} in native cryptocurrency, while the committed state is updated using the full output, $S^c \leftarrow (x + a, y - b)$. The reverse trading direction is defined symmetrically. Thus e-cash settlement remains discrete while the constant-product pricing rule is preserved without unit-level rounding in the AMM state.¹

4.2 Sequential mode

Sequential mode processes exchanges one at a time. Once a request is admitted and quoted against the current committed state S^c , no other swap is admitted until that exchange either completes or terminates. Thus the pricing state remains fixed throughout the exchange, so later trades are never priced using the outcome of an unresolved request. Consequently, the user does not need to post collateral

¹ In practice, reserve values are represented with sufficiently high fixed-point precision, introducing only a small bounded numerical error. Native settlement of the fractional remainder is especially useful when e-cash units have high denominations and only a few units are exchanged.

for cross-trade pricing risk. The LP, however, must reserve native collateral to protect the user against non-delivery after the user's payment has been accepted. The sequential protocol proceeds in three steps: request submission, user payment, and LP payment.

Step 1: Request. User U generates a fresh pair $(Y_i, \xi_i) \leftarrow \text{RKeyGen}(pp^x)$ and chooses a trading direction $\text{dir}_i \in \{0 \rightarrow 1, 1 \rightarrow 0\}$. If $\text{dir}_i = 0 \rightarrow 1$, then $E_{\text{in}} = E_0$ and $E_{\text{out}} = E_1$; for $\text{dir}_i = 1 \rightarrow 0$ the roles are reversed. The user specifies an exact integer input a_i of E_{in} , a minimum acceptable output $b_i^{\min}$ of E_{out} , and a minimum acceptable rate $\rho_{\text{out}}^{\min}$, and submits $\text{Req}_i = (\text{pid}, \text{dir}_i, a_i, b_i^{\min}, Y_i, \rho_{\text{out}}^{\min})$ together with the native-cryptocurrency fee.

Using the current committed state S^c , the contract evaluates F to obtain the output b_i for exactly a_i units of E_{in} and derives $b_i^{\text{ec}} = \lfloor b_i \rfloor$ and $b_i^{\text{nat}} = \rho_{\text{out}}(b_i - \lfloor b_i \rfloor)$. The request is admissible only if $b_i \geq b_i^{\min}$, $\rho_{\text{out}} \geq \rho_{\text{out}}^{\min}$, the required fee is supplied, and the LP has at least the required delivery collateral $D_i^{\text{del}} = \rho_{\text{out}} b_i$ available from its unreserved deposit. If all checks succeed, the contract admits the request, assigns a unique identifier rid_i , locks D_i^{del} from D_{LP} , and fixes b_i and ρ_{out} for this exchange. The locked collateral cannot be withdrawn or reused until the request terminates.

Step 2: User payment. The user has a submission window Θ_{sub} to invoke $\text{SpendEx}_{\text{in}}$ on its input-currency wallet, using payer secret ξ_i , receiver key Y_{LP} , and fresh public randomness ω_i^{U2LP} ,² and publish a payment of a_i units bound to $(\text{pid}, \text{rid}_i, \text{U2LP})$, together with its DA digest and public proof. An honest party submits by Θ_{sub} , leaving sufficient time for DA and blockchain processing before the total deadline Θ_{tot} . If the payment is confirmed available and passes PubVf by Θ_{tot} , the user's leg is accepted. Otherwise the request terminates: S^c remains unchanged, D_i^{del} is released to the LP, and the escrowed fee is paid to the LP.

Step 3: LP payment. Once the user's payment is accepted, the LP similarly has Θ_{sub} to invoke $\text{SpendEx}_{\text{out}}$ on its own output-currency wallet, using payer secret ξ_{LP} , receiver key Y_i , and fresh public randomness ω_i^{LP2U} , and publish a payment of b_i^{ec} units bound to $(\text{pid}, \text{rid}_i, \text{LP2U})$, together with its DA digest and public proof. If the payment is confirmed available and passes PubVf by the corresponding Θ_{tot} deadline, the exchange completes: the contract transfers b_i^{nat} from the locked delivery collateral to the user, updates $S^c \leftarrow F(S^c, \text{dir}_i, a_i, b_i)$, releases the remaining $D_i^{\text{del}} - b_i^{\text{nat}}$ to the LP, and transfers the escrowed fee to the LP. Otherwise, the contract transfers the full D_i^{del} to the user and refunds the escrowed fee.

Proposition 1 (Sequential settlement). *Consider a sequential trade i whose user payment has been accepted and for which the contract has reserved $D_i^{\text{del}} = \rho_{\text{out}} b_i$ from the LP's collateral. Assume honest payments are submitted within Θ_{sub} and $\Theta_{\text{tot}} \geq \Theta_{\text{sub}} + \Delta_{\mathcal{D}}^{\text{live}} + \Delta_{\mathcal{B}}^{\text{live}}$. Under sound public payment verification*

² The public randomness values ω_i^{U2LP} and ω_i^{LP2U} are sampled after the corresponding request context and exchange keys are fixed.

and the settlement functionality of Section 3.1, the contract reaches exactly one of the following terminal outcomes: (i) an acceptable LP payment of b_i^{ec} is accepted by Θ_{tot} and b_i^{nat} is transferred to the user, or (ii) no acceptable LP payment is accepted by Θ_{tot} and the full D_i^{del} is transferred to the user.

The proof of Proposition 1 is given in Appendix D.

4.3 Concurrent mode

Concurrent mode allows several exchanges to be active at the same time. A new request is priced assuming that earlier active exchanges will complete, allowing higher throughput but introducing cross-trade pricing risk.

Manipulation risk. Concurrency allows a user to submit one or more requests that change the pricing state and then submit later requests that benefit from the resulting prices. If the earlier requests are subsequently abandoned while the later ones complete, the LP may be forced to honor outputs determined from a state that never materializes. To prevent this attack, every cancellable user request locks concurrency collateral D_i^{con} sufficient to cover the maximum pricing loss its cancellation can impose on later admitted requests. This is separate from the LP's delivery collateral D_i^{del} , which protects the user against LP non-delivery.

Pricing states. Concurrent mode maintains a committed state S^c and a *virtual* pricing state S^v . S^c contains finalized exchange outcomes, while S^v additionally includes the expected flows of active requests in request order. Thus admitting a request changes S^v but not S^c . Once admitted, its full AMM output b_i is fixed. If an earlier request later cancels, S^v is repaired by replaying the remaining active requests under F ; let $\hat{b}_j$ denote the resulting reference output of request j . The contractual outputs b_j already fixed for admitted requests do not change.

This subsection presents the concurrent version of the exchange protocol, following the same three-step structure as the sequential mode. For the CPMM of Section 4.1, we derive the corresponding user-side concurrency-collateral bound.

Step 1: Request. User U samples $(Y_i, \xi_i) \leftarrow \text{RKeyGen}(pp^x)$ and submits $\text{Req}_i = (\text{pid}, \text{dir}_i, a_i, b_i^{\min}, Y_i, \rho_{\text{out}}^{\min}, \rho_{\text{in}}^{\max})$ together with fee and the required concurrency collateral. Using the current S^v , the contract evaluates F to obtain b_i and derives $b_i^{\text{ec}} = \lfloor b_i \rfloor$ and $b_i^{\text{nat}} = \rho_{\text{out}}(b_i - \lfloor b_i \rfloor)$. The request is accepted only if $b_i \geq b_i^{\min}$, $\rho_{\text{out}} \geq \rho_{\text{out}}^{\min}$, $\rho_{\text{in}} \leq \rho_{\text{in}}^{\max}$, the required fee and $D_i^{\text{con}} = \rho_{\text{in}}(a_i + \varepsilon_{\text{fp}})$ are supplied, and the LP has at least $D_i^{\text{del}} = \rho_{\text{out}}b_i$ of unreserved delivery collateral. On acceptance, the contract assigns rid_i , fixes $(a_i, b_i, \rho_{\text{in}}, \rho_{\text{out}})$, locks both collateral amounts, and adds the expected flow (a_i, b_i) to S^v .

Step 2: User payment. As in sequential mode, the user has Θ_{sub} to invoke $\text{SpendEx}_{\text{in}}$ on its input-currency wallet for a_i units, with payer secret ξ_i , receiver key Y_{LP} , public randomness ω_i^{U2LP} , and context $(\text{pid}, \text{rid}_i, \text{U2LP})$, and submit the DA digest and public proof, which must resolve by Θ_{tot} . If accepted, the user can

no longer cancel and D_i^{con} is returned. If the user fails to provide an acceptable payment, request i is cancelled, its expected flow is immediately removed from $S^\vee$, D_i^{del} is released, and the fee is paid to the LP. Its concurrency collateral D_i^{con} remains locked until the deferred compensation described below is resolved; then $\rho_{\text{in}} L_i$ is transferred to the LP and the remainder is returned to the user.

Step 3: LP payment. After the user's payment is accepted, the LP must provide the already fixed settlement by invoking $\text{SpendEx}_{\text{out}}$ on its own output-currency wallet for b_i^{ec} units, with payer secret ξ_{LP} , receiver key Y_i , public randomness ω_i^{LP2U} , and context $(\text{pid}, \text{rid}_i, \text{LP2U})$, with the same submission and resolution deadlines as in sequential mode. On success, the contract transfers b_i^{nat} from the locked delivery collateral to the user, records the successful settlement (a_i, b_i) , releases the remaining $D_i^{\text{del}} - b_i^{\text{nat}}$ to the LP, and pays the fee to the LP. If the LP fails to provide an acceptable payment, the full D_i^{del} is transferred to the user and the fee is refunded; the failed exchange does not modify S^c . Successful exchanges are incorporated into S^c in request order using their repaired reference flows $(a_j, \hat{b}_j)$. This recomputation affects only future quotes and never changes an already fixed contractual output b_j .

Aborted-request compensation. A request i is *aborted* if its user payment is not accepted by its total deadline after its expected flow has already been incorporated into $S^\vee$. Let $\mathcal{A}_i$ be the later requests admitted while i remained active; this set is fixed when i aborts. The LP incurs a loss from i 's abort only on requests in $\mathcal{A}_i$ whose quoted output it actually delivers. Compensation for i is therefore *deferred* until every request in $\mathcal{A}_i$ has reached a terminal outcome, which happens within the bounded deadlines of those requests. Let $\mathcal{C}_i \subseteq \mathcal{A}_i$ be the requests that *completed*, i.e., both of whose payments were accepted. Requests in $\mathcal{A}_i$ that were themselves aborted, or on which the LP defaulted, contribute no loss.

To measure the loss caused by aborting i , the contract replays the inputs of $\mathcal{A}_i$ in their original order, once with i 's expected flow included and once with it removed, keeping all other flows identical. If b_j^+ and b_j^- denote the resulting full AMM outputs, respectively, the loss attributable to i , measured in units of its input currency, is

$$L_i = \sum_{\substack{j \in \mathcal{C}_i \\ E_{\text{out},j} = E_{\text{in},i}}} [b_j^+ - b_j^-]_+.$$

If several requests abort, their compensations are resolved in the order of their aborts, and the compensation of i is resolved only after all of $\mathcal{A}_i$ has terminated and all earlier aborts have been resolved. For each aborted request i , the two replays start from the state incorporating all previously resolved aborts and differ only in whether i is included or removed. Thus L_i is the marginal loss caused by that removal on requests the LP actually honored, and a loss already attributed to an earlier abort is not attributed again. In particular, a request that shifts later prices and is then abandoned, for example by a party colluding with the LP, cannot inflate the compensation charged to another aborting user. Deferral does not delay pricing: i 's expected flow is removed from $S^\vee$ as soon as i aborts.

When i aborts, the LP receives $\rho_{\text{in}}L_i$ from its locked D_i^{con} , and the remainder $D_i^{\text{con}} - \rho_{\text{in}}L_i$ is returned to the user. This compensation rule is applicable whenever a sound admission-time upper bound on L_i can be established. Proposition 2 establishes such a bound for our constant-product AMM.

Proposition 2 (Constant-product abort-loss bound). *For the zero-fee CPMM, consider a request i with input a_i units of E_{in} that is incorporated into S^v before any later request is priced. Under exact arithmetic, for any sequence of later requests its aggregate cancellation loss satisfies $0 \leq L_i \leq a_i$, and the bound is tight. If the fixed-point implementation can increase the aggregate loss by at most ε_{fp} , then $L_i \leq a_i + \varepsilon_{\text{fp}}$. Hence $D_i^{\text{con}} = \rho_{\text{in}}(a_i + \varepsilon_{\text{fp}})$ is sufficient.*

Theorem 1 (Concurrent settlement safety). *Assume the same timing conditions and issuer-side settlement functionality as in sequential mode, sound public payment verification, and that each request i admits an upper bound $\bar{L}_i$ on its abort loss L_i , with $L_i \leq \bar{L}_i$ and $D_i^{\text{con}} \geq \rho_{\text{in}}\bar{L}_i$. Then every admitted request i reaches exactly one contract terminal outcome: (i) the user’s payment is not accepted, in which case i is aborted and, once its deferred compensation is resolved, D_i^{con} covers its loss L_i on the later requests the LP actually honored; (ii) both payments are accepted, in which case the user receives the fixed settlement $(b_i^{\text{ec}}, b_i^{\text{nat}})$; or (iii) the user’s payment is accepted but the LP’s payment is not, in which case the user receives D_i^{del} . Moreover, repairing S^v after an abort does not change the fixed output of any already admitted request.*

The proofs of Proposition 2 and Theorem 1 are given in Appendix D.

5 Exchange-Compatible Compact E-Cash

We instantiate our interface from the compact threshold e-cash construction of Rial and Piotrowska [22]. The construction preserves its compact serial/tag mechanism while adding receiver-bound exchange payments, public verification, and aggregate anonymous reissue (AAR). Zero-knowledge proofs establish consistency of the hidden wallet and payment data, while separate pairing checks verify the authenticity of the underlying Pointcheval–Sanders (PS) credentials [21]. Rerandomized credential presentations support unlinkable public payments and aggregation, while AAR allows selected records from multiple payments to be settled into a fresh wallet without revealing their source partition. Figure 1 summarizes the construction; the formal algorithms, proof relations, issuance procedure, and issuer-state transitions appear in Appendix E.

6 Security and Privacy Analysis

This section analyzes e-cash security and exchange privacy under the stated cryptographic assumptions and issuer-side functionality of Section 3.1.

Exchange-compatible e-cash with separate pairing verification.

- 1. Setup.** Publish PS verification elements $\tilde{\alpha}_b, \{\tilde{\beta}_{b,i}, \beta_{b,i}\}_{i=1}^3$, proof parameters, and the independent serial base $\delta = \text{HashToGroup}_{G_1}(\text{ecash/serial})$.
- 2. Register.** Holder X samples $sk_{X,b} \leftarrow \$ \mathbb{Z}_p$. Register $(id_X, P_{X,b} = g^{sk_{X,b}})$ in RegRoot_b ; retain the secret and membership witness.
- 3. Issue.** For authorized X and $1 \leq N \leq L_{\max}$, commit to $(sk_{X,b}, u, N)$ and prove $g^{sk_{X,b}} = P_{X,b}$ before fresh issuer $r \leftarrow \$ \mathbb{Z}_p$. Prove that the final request contains $(sk_{X,b}, v = u + r, N)$ with $v \in \mathcal{V}$; blindly issue its threshold PS credential and return $W_{X,b} = (\sigma, v, N, 0)$. Use the bounded seed-retry procedure of Appendix E.
- 4. Exchange keys.** RKeyGen returns $(Y = h^\xi, \xi)$ for $\xi \leftarrow \$ \mathbb{Z}_p$. The user uses a fresh pair per request; the LP may reuse its pool pair.
- 5. Exchange spending.** Check $0 \leq \ell \leq \ell + V \leq N$ and $0 \leq V \leq V_{\max}$; set $c = (\text{pid}, \text{rid}, \text{leg})$ and $K = H_{\text{kdf}}(Y_{\text{rcv}}^{\xi_{\text{snd}}}, c)$. For $0 \leq k < V$, compute

$$\mu_k = (v + \ell + k + 1)^{-1}, \quad R_k = H_{\text{ch}}(\omega, c, k) + H_d(K, c, k),$$

$$S_k = \delta^{\mu_k}, \quad T_k = g^{sk_{X,b}} g^{R_k \mu_k}, \quad \hat{S}_k = S_k g^{H_S(K, c, k)}, \quad \hat{T}_k = T_k g^{H_T(K, c, k)}.$$
 Sample $r_\sigma \leftarrow \$ \mathbb{Z}_p^*$ and independent $u_i \leftarrow \$ \mathbb{Z}_p$; set $h^* = h_{r_\sigma}^{r_\sigma}, s^* = s_{r_\sigma}^{r_\sigma}$ and

$$\Gamma_0 = h^*, \quad \Gamma_i = (h^*)^{m_i} g^{u_i} \quad (i = 1, 2, 3), \quad \Gamma_4 = (s^*)^{-1} \prod_{i=1}^3 \beta_{b,i}^{-u_i}.$$
 Set $M_{\text{pay}} = \{(\hat{S}_k, \hat{T}_k)\}_k$ and bind its digest, Γ , issuer, context, value, keys, and ω in x_{pay} . Π_{pay} binds Γ to the payment's hidden credential and messages and proves credential consistency, capacity, and these equations without pairings. Advance ℓ by V before publishing $(x_{\text{pay}}, M_{\text{pay}}, \Pi_{\text{pay}})$; never roll back.
- 6. Public verification.** Check context and digest, verify Π_{pay} , and separately check $\text{PairCred}_b(\Gamma) = 1$, defined by $\Gamma_0 \neq 1_{G_1}$ and

$$e(\Gamma_0, \tilde{\alpha}_b) \prod_{i=1}^3 e(\Gamma_i, \tilde{\beta}_{b,i}) e(\Gamma_4, \tilde{g}) = 1_{G_T}.$$
- 7. Receiver recovery and aggregation.** Using $K_j = H_{\text{kdf}}(Y_{\text{snd},j}^{\xi_{\text{rcv},j}}, c_j)$, recover and shuffle $1 \leq n \leq L_{\max}$ privately selected records with distinct serials into $\mathcal{C}$. For each contributing source $j \in \mathcal{J}$, privately form $\Gamma_j^\circ = \text{ReRand}_b(\Gamma_j)$. Commit to padded hidden sources, selection, and tuples before deriving nonzero transcript-bound weights a_j ; include each contributing source once:

$$\bar{\Gamma}_r = \prod_{j \in \mathcal{J}} (\Gamma_{j,r}^\circ)^{a_j} \quad (0 \leq r \leq 4), \quad \Gamma_{\text{agg}} = \text{ReRand}_b(\bar{\Gamma}).$$
 Restart if $\bar{\Gamma}_0 = 1_{G_1}$; use independent fresh final rerandomization. Commit to $(sk_{X,b}, u^*, n)$ in C_W^* , with $u^* \leftarrow \$ \mathbb{Z}_p$. Π_{agg} checks source ZK proofs and every $\Gamma_{j,0} \neq 1_{G_1}$; it proves receiver-secret knowledge, selection, distinctness, rerandomization, aggregation, and an opening of C_W^* to a registered secret, contribution u^* , and capacity n . No pairings occur in this proof. Publish $M_{\text{agg}} = \mathcal{C}$ and $x_{\text{agg}} = (B_b, n, H(\mathcal{C}), C_W^*, \text{RegRoot}_b, \nu_{\text{agg}}, \Gamma_{\text{agg}})$; keep sources, selection, and intermediate tuples hidden.
- 8. Settlement / reissue.** Check the digest and session, verify Π_{agg} , and separately check $\text{PairCred}_b(\Gamma_{\text{agg}}) = 1$. Reject exact replays; identify and fund conflicts through assumed issuer recovery. Atomically reserve the value. Supply fresh r^* and verify a final request bound to C_W^* with $v^* = u^* + r^* \in \mathcal{V}$; blindly issue $W_{X,b}^* = (\sigma^*, v^*, n, 0)$ once per reservation.
- 9. Identify.** For validated conflicts $(S, R, T), (S, R', T')$ with $R \neq R'$, recover $P_{X,b} = (T^{R'} / (T')^R)^{(R' - R)^{-1}}$ and resolve the payer in the registry.

Fig. 1. Exchange-compatible e-cash with separate ZK and native pairing verification.

6.1 Security Guarantees

We establish correctness and accountability of the exchange-compatible e-cash construction, including payment validity, value accounting under AAR, double-spender identification, exculpability, and receiver-exclusive settlement.

Theorem 2 (Exchange-compatible e-cash security). *Let Π be the construction of Appendix E. Under honest setup, proof extractability, commitment binding, authenticated-membership soundness, threshold blind-PS unforgeability, collision resistance, discrete-logarithm hardness in $\mathbb{G}_1$ and $\mathbb{H}$, the coefficient-oracle assumptions of Appendix C, joint seed generation as specified in Appendix C.4 and extraction of completed seed proofs as specified in Appendix C.2, serialized issuer issuance and settlement state, and the funded recovery functionality of Section 3.1, Π satisfies the following except with negligible probability: accepted payments are bound to authorized wallet credentials and capacity; every successful AAR accounts for exactly its selected distinct records, with first presentations consuming fresh value and conflicts using funded replacement value; exact replays authorize no additional value; every validated same-serial conflict with distinct challenges is attributable to its registered payer without framing an honest holder; and only the designated receiver can settle a receiver-bound payment. Honest protocol executions are accepted except for the negligible retry-exhaustion and collision events specified in the proof. Consequently, for every PPT adversary $\mathcal{A}$, $\text{Adv}_{\Pi, \mathcal{A}}^{\text{SEC}}(\lambda) = \text{negl}(\lambda)$.*

The proof is given in Appendix F.

6.2 Privacy Guarantees

Let Π denote the construction of Appendix E. The following results use the admissibility and exposure restrictions of their respective privacy experiments. Throughout, assume honest setup, perfect correctness of the blind-signature subprotocol, joint seed generation with the bounded retries of Appendix C.4, and the applicable hash, freshness, and parameter-growth conditions of Appendix C. Zero knowledge and hiding are required jointly under the issuer and LP auxiliary views permitted by each game, including the fixed-prefix simulation requirements of Appendix C.2.

Theorem 3 (Swap–Settlement Unlinkability). *Assume adaptive multi-theorem zero knowledge for holder-generated issuance proofs (including preliminary and final seed-consistency proofs), source-payment proofs Π_{pay} , and the tail of the aggregation proof Π_{agg} ; joint hiding of holder contributions u and seeds v in the preliminary and final issuance requests, and of padded source descriptions, coin selections, and rerandomized tuples Γ_j° in tr_{agg} ; joint pseudorandomness of the DH-derived offsets and masks and the paired inverse-exponent group outputs; and XDH. Then Π satisfies SSU as defined in Definition 1: for every PPT adversary $\mathcal{A}$, $\text{Adv}_{\Pi, \mathcal{A}}^{\text{SSU}}(\lambda) \leq \text{negl}(\lambda)$.*

Theorem 4 (Aggregate-Source Unlinkability). *Under the assumptions of Theorem 3, applied jointly to all source wallets and payments, and the fresh aggregation-prefix and coefficient-oracle conditions of Appendix C.7, Π satisfies ASU as defined in Definition 2, including selections with different numbers of contributing sources. For every PPT adversary $\mathcal{A}$, $\text{Adv}_{\Pi, \mathcal{A}}^{\text{ASU}}(\lambda) \leq \text{negl}(\lambda)$.*

Theorem 5 (Linked-Swap Identity Privacy). *Assume adaptive multi-theorem zero knowledge for the candidate users’ issuance and reissue seed-consistency proofs, the selected user’s payment proof, and the aggregation proof tail; joint hiding of source contributions and seeds in both issuance requests and of the reissued holder secret $sk_{U_\beta, \text{bout}}$ in C_W^* and $C_{W, \text{fin}}^*$; joint pseudorandomness of paired inverse-exponent group outputs; XDH; and the fresh, identity-independent context and beacon conditions of Appendix C.7. Then Π satisfies LSIP as defined in Definition 3, even given the complete issuer-LP view and the swap-settlement association: for every PPT adversary $\mathcal{A}$, $\text{Adv}_{\Pi, \mathcal{A}}^{\text{LSIP}}(\lambda) \leq \text{negl}(\lambda)$.*

The full proofs of Theorems 3, 4, and 5 are given in Appendices G.1, G.2, and G.3, respectively.

7 Conclusion and Future Work

We presented an issuer-independent e-cash exchange in which a public blockchain coordinates pricing, verifies receiver-bound payments, and enforces settlement and collateral, while issuers remain outside individual swaps. We extend compact threshold e-cash with publicly verifiable payments and aggregate anonymous reissue, and establish the accountability and privacy properties arising in exchange-compatible e-cash. Appendix H sketches a concrete instantiation based on recursion-friendly proof systems with transparent or universal public parameters. A main practical limitation is prover cost, since users may need to generate payment and aggregate-reissue proofs locally. Future work can investigate delegated proving, including LP-assisted proof generation or proof wrapping, as well as privacy-preserving outsourcing of aggregate-reissue proving without exposing private witnesses.

Acknowledgments

This work was supported by the Flemish Government’s Cybersecurity Research Program under grant number VOEWICS02.

References

1. Adams, H., Zinsmeister, N., Robinson, D.: Uniswap v2 core. Technical white paper (2020), <https://uniswap.org/whitepaper.pdf>

2. Angeris, G., Agrawal, A., Evans, A., Chitra, T., Boyd, S.: Constant function market makers: Multi-asset trades via convex optimization. In: Tran, D.A., Thai, M.T., Krishnamachari, B. (eds.) *Handbook on Blockchain, Springer Optimization and Its Applications*, vol. 194, pp. 415–444. Springer (2022). https://doi.org/10.1007/978-3-031-07535-3_13
3. Asokan, N., Shoup, V., Waidner, M.: Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications* **18**(4), 593–610 (2000). <https://doi.org/10.1109/49.839935>
4. Baldimtsi, F., Chase, M., Fuchsbauer, G., Kohlweiss, M.: Anonymous transferable e-cash. In: *Public-Key Cryptography – PKC 2015*. LNCS, vol. 9020, pp. 101–124. Springer (2015). https://doi.org/10.1007/978-3-662-46447-2_5
5. Bartoletti, M., Yu Chiang, J.H., Lluch-Lafuente, A.: A theory of automated market makers in DeFi. *Logical Methods in Computer Science* **18**(4), 12:1–12:46 (2022). [https://doi.org/10.46298/LMCS-18\(4:12\)2022](https://doi.org/10.46298/LMCS-18(4:12)2022)
6. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: *Proceedings of the 13th ACM Conference on Computer and Communications Security*. pp. 390–399. ACM (2006). <https://doi.org/10.1145/1180405.1180453>
7. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact e-cash. In: *Advances in Cryptology – EUROCRYPT 2005*. LNCS, vol. 3494, pp. 302–321. Springer (2005). https://doi.org/10.1007/11426639_18
8. Camenisch, J., Lysyanskaya, A., Meyerovich, M.: Endorsed e-cash. In: *2007 IEEE Symposium on Security and Privacy*. pp. 101–115. IEEE (2007). <https://doi.org/10.1109/SP.2007.15>
9. Canard, S., Gouget, A.: Divisible e-cash systems can be truly anonymous. In: *Advances in Cryptology – EUROCRYPT 2007*. LNCS, vol. 4515, pp. 482–497. Springer (2007). https://doi.org/10.1007/978-3-540-72540-4_28
10. Canard, S., Pointcheval, D., Sanders, O., Traoré, J.: Divisible e-cash made practical. In: *Public-Key Cryptography – PKC 2015*. LNCS, vol. 9020, pp. 77–100. Springer (2015). https://doi.org/10.1007/978-3-662-46447-2_4
11. Cashu Developers: Cashu protocol documentation and faq (2026), <https://docs.cashu.space/>, accessed 31 August 2026
12. Chaum, D.: Blind signatures for untraceable payments. In: *Advances in Cryptology – CRYPTO ’82*. pp. 199–203. Plenum Press (1983)
13. Chaum, D.: Privacy protected payments: Unconditional payer and/or payee untraceability. In: Chaum, D., Schaumüller-Bichl, I. (eds.) *Smart Card 2000*, pp. 69–93. North-Holland (1989)
14. Chaum, D., Fiat, A., Naor, M.: Untraceable electronic cash. In: *Advances in Cryptology – CRYPTO ’88*. LNCS, vol. 403, pp. 319–327. Springer (1990). https://doi.org/10.1007/0-387-34799-2_25
15. Chung, H., Masserova, E., Shi, E., Thyagarajan, S.A.K.: Rapidash: Atomic swaps secure under user–miner collusion. In: *Financial Cryptography and Data Security. Lecture Notes in Computer Science*, vol. 15751, pp. 249–266. Springer (2025). https://doi.org/10.1007/978-3-032-07024-1_15
16. Dold, F.: The GNU Taler System: Practical and Provably Secure Electronic Payments. Ph.D. thesis, University of Rennes 1 (2019), <https://taler.net/papers/thesis-dold-phd-2019.pdf>
17. Fedimint Developers: Fedimint documentation (2026), <https://fedimint.org/>, accessed 31 August 2026

18. Jakobsson, M.: Ripping coins for a fair exchange. In: Guillou, L.C., Quisquater, J.J. (eds.) *Advances in Cryptology – EUROCRYPT ’95*. Lecture Notes in Computer Science, vol. 921, pp. 220–230. Springer (1995). https://doi.org/10.1007/3-540-49264-X_18
19. Mohan, V.: Automated market makers and decentralized exchanges: A DeFi primer. *Financial Innovation* **8**(1), 20 (2022). <https://doi.org/10.1186/s40854-021-00314-5>
20. Ni, J., Au, M.H., Wu, W., Luo, X., Lin, X., Shen, X.: Dual-anonymous off-line electronic cash for mobile payment. *IEEE Transactions on Mobile Computing* **22**(6), 3303–3317 (2023). <https://doi.org/10.1109/TMC.2021.3135301>
21. Pointcheval, D., Sanders, O.: Short randomizable signatures. In: *Topics in Cryptology – CT-RSA 2016*. Lecture Notes in Computer Science, vol. 9610, pp. 111–126. Springer (2016). https://doi.org/10.1007/978-3-319-29485-8_7
22. Rial, A., Piotrowska, A.M.: Compact and divisible e-cash with threshold issuance. *Proceedings on Privacy Enhancing Technologies* **2023**(4), 381–415 (2023). <https://doi.org/10.56553/popets-2023-0116>
23. Tas, E.N., Seres, I.A., Zhang, Y., Melczer, M., Kelkar, M., Bonneau, J., Nikolaenko, V.: Atomic and fair data exchange via blockchain. In: *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*. pp. 3227–3241. ACM (2024). <https://doi.org/10.1145/3658644.3690248>
24. Xu, J., Paruch, K., Cousaert, S., Feng, Y.: SoK: Decentralized exchanges (DEX) with automated market maker (AMM) protocols. arXiv preprint arXiv:2103.12732 (2021), <https://arxiv.org/abs/2103.12732>
25. Zhang, Y., Chen, X., Park, D.: Formal specification of constant product market maker model and implementation. Tech. rep., Runtime Verification, Inc. (October 2018)

A Exchange-Compatible E-Cash Interface

This section introduces a general framework for exchange-compatible e-cash. Subsection A.1 defines the underlying e-cash functionality, while Subsection A.2 replaces ordinary spending with the functionality required by our privacy-preserving exchange protocol.

Notation. For $b \in \{0, 1\}$, issuer B_b issues currency E_b . A holder X is either a user or the LP, and λ is the security parameter. Issuer B_b has public parameters pp_b and private mutable state st_b , which contains its secret key material, registration records, and the state required for issuance, settlement, and double-spending checks; st'_b denotes the updated issuer state. Holder X retains private registration information

$$\text{reg}_{X,b} = (sk_{X,b}, \text{wit}_{X,b}^{\text{reg}}),$$

where $sk_{X,b}$ is its private registration secret and $\text{wit}_{X,b}^{\text{reg}}$ is its registration witness. A spendable wallet is denoted by $W_{X,b}$, its state after spending by $W'_{X,b}$, and a freshly issued wallet by $W^*_{X,b}$. An issued wallet has capacity N , V denotes a payment value, and failure is denoted by $\perp$.

For an interactive protocol $\mathcal{P}$ between holder X and issuer B_b , we write

$$\langle X(\text{in}_X), B_b(\text{in}_{B_b}) \rangle_{\mathcal{P}} \rightarrow (\text{out}_X; \text{out}_{B_b}),$$

where the arguments shown for each party are its local inputs. In particular, holder-private inputs are not revealed to the issuer unless explicitly stated.

A.1 Base E-Cash Interface

For issuer B_b , the underlying e-cash system is

$$\text{EC}_b = (\text{Setup}_b, \text{Register}_b, \text{Issue}_b, \text{Spend}_b, \text{Settle}_b, \text{Identify}_b).$$

- $\text{Setup}_b(1^\lambda) \rightarrow (pp_b, st_b)$ is run by issuer B_b and generates its public parameters and initial private state, using any independently generated common proof parameters required by the instantiation.
- $\langle X(\text{id}_X), B_b(st_b) \rangle_{\text{Register}_b} \rightarrow (\text{reg}_{X,b}; st'_b)$ is run interactively by X and B_b . During registration, X authenticates as id_X . The issuer records the association between id_X and the corresponding public tracing handle, while X retains $\text{reg}_{X,b} = (sk_{X,b}, \text{wit}_{X,b}^{\text{reg}})$ and keeps $sk_{X,b}$ private.
- $\langle X(\text{reg}_{X,b}, N), B_b(N, st_b) \rangle_{\text{Issue}_b} \rightarrow (W_{X,b}; st'_b)$ is run interactively by X and B_b . The holder proves the required relation to its registration without revealing $sk_{X,b}$ and obtains a wallet $W_{X,b}$ containing N spendable units of currency E_b .
- $\text{Spend}_b(W_{X,b}, \text{reg}_{X,b}, V, \text{payinfo}) \rightarrow (W'_{X,b}, \text{rcpt})$ is run locally by payer X . It spends V units, updates the wallet state, and produces a payment receipt rcpt bound to the intended receiver and application information payinfo .
- $\langle X(\text{rcpt}, \text{policy}, [\text{id}_X]_{\text{redeem}}), B_b(\text{policy}, st_b) \rangle_{\text{Settle}_b} \rightarrow (\text{out}_X; st'_b)$ is run interactively by receiver X and issuer B_b , where $\text{policy} \in \{\text{redeem}, \text{reissue}\}$. If $\text{policy} = \text{redeem}$, X authenticates as id_X during settlement and the accepted value is redeemed according to the issuer's redemption mechanism. If $\text{policy} = \text{reissue}$, no identity is disclosed and $\text{out}_X = W_{X,b}^*$ is a fresh wallet containing the accepted value. In either case, B_b verifies the receipt and checks the represented value against its settlement and double-spending state.
- $\text{Identify}_b(pp_b, \text{ev}, \text{ev}') \rightarrow \text{handle or } \perp$ is run by issuer B_b on scheme-specific evidence ev, ev' of conflicting spends of the same underlying coin and returns the corresponding registered tracing handle.

A.2 Exchange-Compatible Interface

For exchange use, the ordinary spending operation Spend_b is replaced by

$$\text{Exch}_b = (\text{RKeyGen}, \text{SpendEx}_b, \text{PubVf}_b, \text{Aggregate}_b).$$

The resulting exchange-compatible e-cash system is

$$\text{EC}_b^{\text{ex}} = (\text{Setup}_b, \text{Register}_b, \text{Issue}_b, \text{Exch}_b, \text{Settle}_b, \text{Identify}_b).$$

Let pp_{ex} denote the common public parameters of the exchange interface. For one exchange leg, let $c = (\text{pid}, \text{rid}, \text{leg})$, where pid is the pool identifier, rid is the request identifier, and $\text{leg} \in \{\text{U2LP}, \text{LP2U}\}$. The value sid identifies the

protocol deployment; it is common to all sessions of that deployment and never identifies an individual swap, request, or settlement. For that leg, $(Y_{\text{snd}}, \xi_{\text{snd}})$ and $(Y_{\text{rcv}}, \xi_{\text{rcv}})$ denote the sender and receiver exchange-key pairs, respectively. The value ω denotes the public post-request challenge fixed by the exchange protocol after the request context and exchange keys are fixed.

- $\text{RKeyGen}(pp_{\text{ex}}) \rightarrow (Y, \xi)$ is run locally by a holder and generates an exchange public key Y and corresponding secret ξ . For request i , the user generates a fresh pair (Y_i, ξ_i) , while the LP uses its pool pair $(Y_{\text{LP}}, \xi_{\text{LP}})$.
- $\text{SpendEx}_b(W_{X,b}, \text{reg}_{X,b}, V, \xi_{\text{snd}}, Y_{\text{rcv}}, c, \omega) \rightarrow (W'_{X,b}, \text{rcpt}_{\text{pay}})$ is run locally by payer X . Here Y_{snd} is the public key corresponding to ξ_{snd} . The algorithm spends V units according to the wallet-consumption semantics of Spend_b and produces $\text{rcpt}_{\text{pay}} = (x_{\text{pay}}, M_{\text{pay}}, \Pi_{\text{pay}})$. Here x_{pay} is the public payment statement, M_{pay} is the protected scheme-specific payload containing the receiver-side information required for settlement, such as protected coin records and double-spending or tracing tags, and Π_{pay} proves consistency of the receipt.
- $\text{PubVf}_b(pp_b, pp_{\text{ex}}, x_{\text{pay}}, \Pi_{\text{pay}}) \rightarrow \{0, 1\}$ is publicly executable and verifies a statement containing $(\text{sid}, B_b, \text{pid}, \text{rid}, \text{leg}, V, Y_{\text{snd}}, Y_{\text{rcv}}, \omega, d_{M_{\text{pay}}})$ together with any construction-specific public verification data. In Appendix E, these additional data are the credential presentation Γ ; verification checks both Π_{pay} and a separate native pairing equation. The digest $d_{M_{\text{pay}}}$ binds the protected payload M_{pay} . A party checking a full receipt additionally checks $H(M_{\text{pay}}) = d_{M_{\text{pay}}}$; the contract separately verifies availability of the payload bound to that digest. PubVf_b does not access issuer settlement or double-spending state.
- $\text{Aggregate}_b(\text{reg}_{X,b}, \{(\text{rcpt}_{\text{pay},j}, \xi_{\text{rcv},j})\}_{j=1}^m, \mathcal{I}) \rightarrow \text{rcpt}_{\text{agg}}$ is run locally by receiver X . For each $(j, k) \in \mathcal{I}$, unit k of $\text{rcpt}_{\text{pay},j}$ is selected, and $n = |\mathcal{I}|$ is the total selected value. The output is $\text{rcpt}_{\text{agg}} = (x_{\text{agg}}, M_{\text{agg}}, \Pi_{\text{agg}})$, where M_{agg} contains the selected scheme-specific settlement data. The statement and proof certify that exactly n distinct units are selected from valid receipts designated to X ; settlement verifies the proof and any separate native checks required by the instantiation. The receiver's registration witness and the partition of selected units among source receipts remain hidden. Aggregate_b does not access or modify st_b . The resulting rcpt_{agg} can be submitted to Settle_b under policy *reissue*.

B Full Privacy Experiments

These experiments formalize Definitions 1, 2, and 3 for the cryptographic protocol layer. They use one challenge episode and allow ordinary adaptive queries before and after the challenge. The hidden bit is β ; the currency index remains b . The adversary receives the complete cryptographic view prescribed by the corresponding coalition.

B.1 State, Views, and Object Classes

The challenger initializes the real system once, obtaining the public parameters, issuer states st_0, st_1 , and the pool state. Ordinary-query and challenge executions use these same states; in particular, the experiment maintains no separate challenge spent state or pool state.

For $b \in \{0, 1\}$, let view_{B_b} denote the complete cryptographic view of issuer B_b , including its secret state and randomness, registration and issuance transcripts, settlement inputs and outcomes, reservations, recovery authorizations and charges, and observable failures. Let pub denote the protocol-defined cryptographic projection of the public ledger transcript, including public payment statements, exchange keys, payloads and digests, proofs, verification outcomes, and public AMM data. For LSIP, view_{LP} additionally contains the LP's complete cryptographic state, including its wallets and credentials, exchange secrets, prover randomness, recovered records, and known payment-to-reissue associations.

When an instantiation uses independent proof setup, that setup is external to the issuers and the LP. Its private setup randomness and trapdoors are not part of their state or views; their own cryptographic views remain complete.

Native blockchain-account identities, transaction timing and block position, network-layer metadata, and other external identity information are outside these experiments.

Every wallet and every payment or aggregate receipt is assigned an immutable game label $\tau \in \{Q, C\}$, where Q denotes an ordinary-query object and C denotes a challenge object. These labels are used only for experiment bookkeeping and do not change the distribution or semantics of the underlying e-cash objects. Fresh wallets are generated independently, including wallets belonging to the same registered holder. Any object derived from a C -object is also labelled C .

State-changing ordinary oracles accept only Q -objects. Copying, re-encoding, or assigning a new handle to an object does not change its game label. Public algorithms, such as PubVf_b , may be evaluated on arbitrary inputs without changing issuer state.

The LP's persistent exchange key may be used in both ordinary and challenge executions. Such reuse does not change an object's game label. In LSIP, the corresponding secret key and recovery information are included in view_{LP} .

B.2 Ordinary Oracle Interfaces

The ordinary oracles model adaptive protocol activity before and after the single privacy challenge. They execute the real protocol on ordinary Q -labelled objects and use the same issuer and pool states as the challenge. Challenge objects are not addressable through these oracles. A failed handle, ownership, type, or range check returns $\perp$ without changing state; once a valid protocol execution begins, its actual outcome, including any failure, is retained in the prescribed view.

Game state and handles. The challenger assigns a unique handle to every wallet and receipt stored by the experiment. A handle is only an experiment-level

```

 $\mathcal{O}_{\text{Register}}(b, X)$ 
require  $b \in \{0, 1\}$ ;
if  $\text{Reg}[b, X] \neq \perp$  : return  $\text{pubReg}(b, X)$ ;
 $\langle X(\text{id}_X), B_b(st_b) \rangle_{\text{Register}_b} \rightarrow (\text{reg}_{X,b}; st'_b)$ ;
 $st_b \leftarrow st'_b, \quad \text{Reg}[b, X] \leftarrow \text{reg}_{X,b}$ ;
return  $(\text{pubReg}(b, X), \text{Emit}())$ .

```

Fig. 2. Registration oracle for ordinary protocol activity.

identifier used to address an object through the oracle interface; it is not part of the e-cash protocol.

For a wallet handle w , write $\text{Wal}[w] = (\tau, b, X, W)$, where $\tau \in \{\mathbf{Q}, \mathbf{C}\}$ is its game label, b identifies its issuer, X is its holder, and W is the underlying wallet.

For a payment-receipt handle r , write $\text{Rec}[r] = (\tau, b, X, \text{rcpt}_{\text{pay}}, \xi_{\text{rcv}})$, where X is the designated receiver and ξ_{rcv} is the receiver exchange secret required for later aggregation. For an aggregate-receipt handle r , write $\text{Rec}[r] = (\tau, b, X, \text{rcpt}_{\text{agg}})$. Registration information is stored separately as $\text{Reg}[b, X] = \text{reg}_{X,b}$.

The predicates $\mathbf{QWallet}$ and $\mathbf{QReceipt}$ check that the referenced object exists, has label $\mathbf{Q}$, and satisfies the required holder and issuer conditions. $\mathbf{QPayReceipts}$ additionally checks that a collection consists of payment receipts for the same receiver and issuer and that all selected positions are in range. These predicates do not perform issuer freshness, replay, or conflict checks.

StoreQW and StoreQR create handles only for successful ordinary-query outputs. Emit returns the newly generated part of the prescribed adversarial view, including observable failures. Let $\text{pubReg}(b, X)$ denote the public registration information associated with X under issuer B_b ; the private registration information remains in $\text{Reg}[b, X]$.

Revealed records holders whose complete private state has been returned by $\mathcal{O}_{\text{Corrupt}}$. At challenge admission, Protected is fixed to contain every holder whose private state must remain hidden in either challenge branch. A holder in Protected cannot subsequently be corrupted.

Registration and issuance. Figures 2 and 3 define the registration and issuance oracles, respectively.

Here $\text{IssueAuth}_b(X, N)$ denotes the ordinary identified issuance authorization for capacity N , bound to X 's registered handle $P_{X,b}$; the issuance proof must bind the wallet secret to this same handle.

Exchange spending. Figure 4 defines the exchange-spending oracle. The helper $\text{PayQ}(w, X_R, V, c)$ honestly generates the required exchange keys, obtains the prescribed blockchain randomness ω after the context and exchange keys are fixed, invokes SpendEx_b , updates the source wallet, runs the corresponding public verification, and stores a successful payment receipt for receiver X_R with label $\mathbf{Q}$.

```

 $\mathcal{O}_{\text{Issue}}(b, X, N)$ 
require  $\text{Reg}[b, X] \neq \perp, \quad 1 \leq N \leq L_{\max};$ 
require  $\text{IssueAuth}_b(X, N) = 1;$ 
 $\langle X(\text{Reg}[b, X], N), B_b(N, st_b) \rangle_{\text{Issue}_b} \rightarrow (W; st'_b);$ 
 $st_b \leftarrow st'_b;$ 
 $w \leftarrow \text{StoreQW}(b, X, W);$ 
return  $(w, \text{Emit}()).$ 
    
```

Fig. 3. Issuance oracle for ordinary wallets.

```

 $\mathcal{O}_{\text{SpendEx}}(w, X_R, V, c)$ 
require  $\text{QWallet}(w);$ 
 $(Q, b, X, W) \leftarrow \text{Wal}[w];$ 
require  $\text{Reg}[b, X_R] \neq \perp, \quad 0 \leq V \leq V_{\max};$ 
require  $W$  has at least  $V$  unused positions;
require  $c$  is a fresh valid exchange context;
 $r \leftarrow \text{PayQ}(w, X_R, V, c);$ 
return  $(r, \text{Emit}()).$ 
    
```

Fig. 4. Exchange-spending oracle for ordinary payments.

```

 $\mathcal{O}_{\text{Swap}}(w, X, \text{req})$ 
require  $\text{QWallet}(w, X);$ 
 $(b, a, a_{\text{out}}^{\text{ec}}, \text{ok}) \leftarrow \text{Admit}(\text{pool}, \text{req});$ 
require  $\text{ok} = 1, \quad \text{issuer}(w) = b, \quad \text{remaining}(w) \geq a;$ 
 $w_{\text{LP}} \leftarrow \text{NextLPQ}(B_{1-b}, a_{\text{out}}^{\text{ec}});$ 
require  $w_{\text{LP}} \neq \perp;$ 
 $(\text{rid}, r_{\text{LP}}, r_U) \leftarrow \text{RunSwapQ}(w, w_{\text{LP}}, \text{req});$ 
return  $(\text{rid}, r_{\text{LP}}, r_U, \text{Emit}()).$ 
    
```

Fig. 5. AMM-swap oracle for ordinary protocol activity.

AMM swaps. The swap oracle $\mathcal{O}_{\text{Swap}}$ in Figure 5 executes an ordinary honest swap using the real pool state. Let $\text{NextLPQ}(B, V)$ select an LP query wallet of issuer B with at least V unused units, or return $\perp$. The helper RunSwapQ executes the actual AMM protocol and stores both resulting payment receipts with label Q .

Admit evaluates the pool’s public admission conditions, while RunSwapQ performs the corresponding real pool transitions.

Aggregation and settlement. Figures 6 and 7 define the aggregation and settlement oracles, respectively.

```

 $\mathcal{O}_{\text{Aggregate}}(X, \{r_j\}_{j=1}^m, \mathcal{I})$ 
require QPayReceipts( $X, \{r_j\}_{j=1}^m, \mathcal{I}$ );
 $b \leftarrow \text{issuer}(\{r_j\}_{j=1}^m)$ ;
 $\{(\text{rcpt}_{\text{pay},j}, \xi_{\text{rcv},j})\}_{j=1}^m \leftarrow \text{Rec}[\{r_j\}_{j=1}^m]$ ;
 $\text{rcpt}_{\text{agg}} \leftarrow \text{Aggregate}_b(\text{Reg}[b, X], \{(\text{rcpt}_{\text{pay},j}, \xi_{\text{rcv},j})\}_{j=1}^m, \mathcal{I})$ ;
if  $\text{rcpt}_{\text{agg}} = \perp$  : return ( $\perp$ ,  $\text{Emit}()$ );
 $r_{\text{agg}} \leftarrow \text{StoreQR}(b, X, \text{rcpt}_{\text{agg}})$ ;
return ( $r_{\text{agg}}$ ,  $\text{Emit}()$ ).

```

Fig. 6. Aggregation oracle for ordinary payment receipts.

```

 $\mathcal{O}_{\text{Settle}}(X, r, \text{policy})$ 
require QReceipt( $r, X$ );
require  $\text{policy} \in \{\text{redeem}, \text{reissue}\}$ ;
 $b \leftarrow \text{issuer}(r)$ ,  $\text{rcpt} \leftarrow \text{receipt}(r)$ ;
if  $\text{policy} = \text{redeem}$  :
   $\langle X(\text{rcpt}, \text{redeem}, \text{id}_X), B_b(\text{redeem}, st_b) \rangle_{\text{Settle}_b} \rightarrow (\text{out}; st'_b)$ ;
   $st_b \leftarrow st'_b$ ;
  return ( $\text{out}$ ,  $\text{Emit}()$ );
else :
   $\langle X(\text{rcpt}, \text{reissue}), B_b(\text{reissue}, st_b) \rangle_{\text{Settle}_b} \rightarrow (W^*; st'_b)$ ;
   $st_b \leftarrow st'_b$ ;
  if  $W^* = \perp$  : return ( $\perp$ ,  $\text{Emit}()$ );
   $w^* \leftarrow \text{StoreQW}(b, X, W^*)$ ;
  return ( $w^*$ ,  $\text{Emit}()$ ).

```

Fig. 7. Settlement oracle for ordinary receipts.

```

 $\mathcal{O}_{\text{RevealWallet}}(w)$ 
require QWallet( $w$ );
 $(Q, b, X, W) \leftarrow \text{Wal}[w]$ ;
return  $W$ .

```

Fig. 8. Wallet-exposure oracle for ordinary wallets.

The settlement oracle does not pre-screen receipts for freshness, replay, or conflict. These checks are performed by the actual Settle_b execution, and its outcome remains observable.

Exposure oracles. The oracles in Figures 8 and 9 allow disclosure of ordinary wallets and corruption of holders unrelated to the protected challenge.

```

 $\mathcal{O}_{\text{Corrupt}}(X)$ 
require  $X \notin \text{Protected} \cup \{\text{LP}\};$ 
 $\text{Revealed} \leftarrow \text{Revealed} \cup \{X\};$ 
return  $\text{RetainedState}(X).$ 
    
```

Fig. 9. Holder-corruption oracle subject to challenge protection.

$\mathcal{O}_{\text{RevealWallet}}$ reveals only the named wallet and does not additionally reveal registration information, unrelated exchange secrets, receipt history, or prover randomness. $\mathcal{O}_{\text{Corrupt}}$ returns the holder’s complete retained private state. A holder already in **Revealed** cannot later be admitted to a protected challenge role.

No separate oracle is needed for **RKeyGen**, **PubVf_b**, or **Identify_b**: **RKeyGen** is run locally as part of honest oracle executions, **PubVf_b** is publicly executable, and **Identify_b** is invoked by the issuer when settlement encounters valid conflicting-spend evidence.

Challenge conventions. For each experiment, **SetProtected** places every holder whose private state is required by either challenge branch in **Protected** and ensures that every object created by either challenge description, and every descendant of such an object, is labelled **C**. **Emit_x** returns the increment of **View_x** generated by the challenge, excluding challenger bookkeeping and oracle handles.

An initial eligibility failure receives an independent fair score. After admission, actual issuance, proof, verification, settlement, replay, or conflict failures remain visible and are not resampled away. If a required intermediate object is not produced, dependent challenge steps are skipped rather than replaced by fabricated outputs. No ordinary oracle call is interleaved inside the challenge episode.

RunSourcesC honestly creates any fresh challenge wallets required by its source descriptions and executes the corresponding source swaps in a fixed order independent of β . It uses the real **Issue_b**, **SpendEx_b**, and **PubVf_b** algorithms and the real pool state, and retains the receiver exchange secrets required by **Aggregate_b**. Challenge exchange keys and blockchain randomness ω are generated with the same ordering as in ordinary executions.

IssueCall honestly issues the fresh challenge wallets prescribed for both candidate users and the LP, using independent issuance executions. **RunSwapC** executes one complete challenge swap using the specified wallets and the real pool state, and returns its public swap identifier $\text{swid} = (\text{pid}, \text{rid})$. This experiment-level identifier is distinct from the deployment identifier sid in payment and aggregation statements, and is never part of x_{agg} or ν_{agg} . All outputs of these helpers are labelled **C**.

B.3 Swap–Settlement Unlinkability

Swap–Settlement Unlinkability (SSU) requires that the issuer coalition cannot determine which of two compatible public swap payments generated a particular

```

Experiment  $\text{Exp}_{\mathcal{H}, \mathcal{A}}^{\text{SSU}}(1^\lambda)$ 
 $(pp, st_0, st_1, \text{pool}) \leftarrow \text{Init}(1^\lambda)$ 
 $\text{Protected} \leftarrow \emptyset, \text{Revealed} \leftarrow \emptyset$ 
 $(\chi_0, \chi_1, s) \leftarrow \mathcal{A}_{\text{pre}}^{\mathcal{O}}(pp, \text{View}_{\text{SSU}}())$ 
if  $\text{Check}_{\text{SSU}}(\chi_0, \chi_1) = 0$  :
     $z \leftarrow \{0, 1\}; \text{return } z$ 
 $\text{SetProtected}(\chi_0, \chi_1)$ 
 $\beta \leftarrow \{0, 1\}$ 
 $((\text{rcpt}_0, \xi_0), (\text{rcpt}_1, \xi_1), \text{ok}) \leftarrow \text{RunSourcesC}(S_0, S_1)$ 
if  $\text{ok} = 1$  :
     $\text{rcpt}_{\text{agg}} \leftarrow \text{Aggregate}_b(\text{Reg}[b, X], \{(\text{rcpt}_\beta, \xi_\beta)\}, \mathcal{I}_\beta)$ 
    if  $\text{rcpt}_{\text{agg}} \neq \perp$  :
         $\langle X(\text{rcpt}_{\text{agg}}, \text{reissue}), B_b(\text{reissue}, st_b) \rangle_{\text{Settle}_b} \rightarrow (W^*; st'_b)$ 
         $st_b \leftarrow st'_b$ 
     $y \leftarrow \text{Emit}_{\text{SSU}}()$ 
     $\beta' \leftarrow \mathcal{A}_{\text{post}}^{\mathcal{O}}(s, y)$ 
return  $[\beta' = \beta]$ 

```

Fig. 10. Swap-settlement unlinkability (SSU) experiment.

anonymous reissue. Both candidate swaps occur in the public transcript; the hidden information is only the association between one of those payments and the issuer-side settlement.

Let S_0, S_1 be two distinct admissible swap descriptions whose target payments are issued by the same B_b to the same registered receiver X . Let $\mathcal{I}_j$ be a valid nonempty selection from the target payment of S_j , and let $V(\mathcal{I}_j)$ denote its total selected value. Define

$$\text{Leak}_{\text{SSU}}(\chi_j) = (b, V(\mathcal{I}_j)), \quad \chi_j = (b, X, S_j, \mathcal{I}_j).$$

Challenge admission requires $\text{Leak}_{\text{SSU}}(\chi_0) = \text{Leak}_{\text{SSU}}(\chi_1)$. It additionally checks that both source descriptions and selections are valid, the required registrations exist, and every holder whose private state is required by either branch is outside **Revealed**.

The issuer-coalition view is

$$\text{View}_{\text{SSU}} = (\text{pub}, \text{view}_{B_0}, \text{view}_{B_1}).$$

Figure 10 gives the SSU experiment.

The SSU advantage is

$$\text{Adv}_{\mathcal{H}, \mathcal{A}}^{\text{SSU}} = \left| \Pr[\text{Exp}_{\mathcal{H}, \mathcal{A}}^{\text{SSU}}(1^\lambda) = 1] - \frac{1}{2} \right|.$$

The protocol satisfies SSU if this advantage is negligible for every PPT adversary.

```

Experiment  $\text{Exp}_{II, \mathcal{A}}^{\text{ASU}}(1^\lambda)$ 
 $(pp, st_0, st_1, \text{pool}) \leftarrow \text{Init}(1^\lambda)$ 
 $\text{Protected} \leftarrow \emptyset, \text{Revealed} \leftarrow \emptyset$ 
 $(\chi_0, \chi_1, s) \leftarrow \mathcal{A}_{\text{pre}}^{\mathcal{O}}(pp, \text{View}_{\text{ASU}}())$ 
if  $\text{Check}_{\text{ASU}}(\chi_0, \chi_1) = 0$  :
     $z \leftarrow \{0, 1\}; \text{ return } z$ 
 $\text{SetProtected}(\chi_0, \chi_1)$ 
 $\beta \leftarrow \{0, 1\}$ 
 $(\{\text{rcpt}_i, \xi_i\}_{i=1}^m, \text{ok}) \leftarrow \text{RunSourcesC}(\mathcal{S})$ 
if  $\text{ok} = 1$  :
     $\text{rcpt}_{\text{agg}} \leftarrow \text{Aggregate}_b(\text{Reg}[b, X], \{\text{rcpt}_i, \xi_i\}_{i=1}^m, \mathcal{I}_\beta)$ 
    if  $\text{rcpt}_{\text{agg}} \neq \perp$  :
         $\langle X(\text{rcpt}_{\text{agg}}, \text{reissue}), B_b(\text{reissue}, st_b) \rangle_{\text{Settle}_b} \rightarrow (W^*; st'_b)$ 
         $st_b \leftarrow st'_b$ 
     $y \leftarrow \text{Emit}_{\text{ASU}}()$ 
     $\beta' \leftarrow \mathcal{A}_{\text{post}}^{\mathcal{O}}(s, y)$ 
    return  $[\beta' = \beta]$ 
    
```

Fig. 11. Aggregate-source unlinkability (ASU) experiment.

B.4 Aggregate-Source Unlinkability

Aggregate-Source Unlinkability (ASU) extends source unlinkability to an aggregate reissue. The issuer learns the aggregate value but should not learn which selection of coins from the publicly visible source payments produced that aggregate. The two candidate selections may therefore involve different numbers of contributing source payments.

Let $\mathcal{S} = (S_1, \dots, S_m)$ be a common family of admissible public source swaps whose target payments are issued by the same B_b to the same registered receiver X . A candidate selection $\mathcal{I}_j$ is a valid nonempty set of source-position pairs (i, k) over this family. Define

$$\text{Leak}_{\text{ASU}}(\chi_j) = (b, V(\mathcal{I}_j)), \quad \chi_j = (b, X, \mathcal{S}, \mathcal{I}_j).$$

Challenge admission requires $\mathcal{I}_0 \neq \mathcal{I}_1$ and $\text{Leak}_{\text{ASU}}(\chi_0) = \text{Leak}_{\text{ASU}}(\chi_1)$. No equality requirement is imposed on the number of source payments containing selected coins. It additionally checks that all selected positions are valid, the required registration exists, and every holder whose private state is required by either branch is outside **Revealed**.

The issuer-coalition view is

$$\text{View}_{\text{ASU}} = (\text{pub}, \text{view}_{B_0}, \text{view}_{B_1}).$$

Figure 11 gives the ASU experiment.

```

Experiment  $\text{Exp}_{\Pi, \mathcal{A}}^{\text{LSIP}}(1^\lambda)$ 
 $(pp, st_0, st_1, \text{pool}) \leftarrow \text{Init}(1^\lambda)$ 
 $\text{Protected} \leftarrow \emptyset, \text{Revealed} \leftarrow \emptyset$ 
 $(\chi_0, \chi_1, s) \leftarrow \mathcal{A}_{\text{pre}}^\mathcal{O}(pp, \text{View}_{\text{LSIP}}())$ 
if  $\text{Check}_{\text{LSIP}}(\chi_0, \chi_1) = 0$  :
   $z \leftarrow \{0, 1\}$ ; return  $z$ 
 $\text{SetProtected}(\chi_0, \chi_1); \beta \leftarrow \{0, 1\}$ 
 $(W_0, W_1, W_{\text{LP}}, \text{ok}) \leftarrow \text{IssueCALL}(U_0, U_1, \text{req})$ 
if  $\text{ok} = 1$  :
   $(\text{swid}, \text{rcpt}_U, \xi_U, \text{ok}') \leftarrow \text{RunSwapC}(U_\beta, W_\beta, W_{\text{LP}}, \text{req})$ 
  if  $\text{ok}' = 1$  :
     $\text{rcpt}_{\text{agg}} \leftarrow \text{Aggregate}_{b_{\text{out}}}(\text{Reg}[b_{\text{out}}, U_\beta], \{(\text{rcpt}_U, \xi_U)\}, \mathcal{I})$ 
    if  $\text{rcpt}_{\text{agg}} \neq \perp$  :
       $\langle U_\beta(\text{rcpt}_{\text{agg}}, \text{reissue}), B_{b_{\text{out}}}(\text{reissue}, st_{b_{\text{out}}}) \rangle_{\text{Settle}_{b_{\text{out}}}} \rightarrow (W^*, st'_{b_{\text{out}}})$ 
       $st_{b_{\text{out}}} \leftarrow st'_{b_{\text{out}}}; \text{KnownLink} \leftarrow (\text{swid}, \text{rcpt}_{\text{agg}})$ 
   $y \leftarrow \text{Emit}_{\text{LSIP}}(); \beta' \leftarrow \mathcal{A}_{\text{post}}^\mathcal{O}(s, y)$ 
return  $[\beta' = \beta]$ 

```

Fig. 12. Linked-swap identity privacy (LSIP) experiment.

The ASU advantage is

$$\text{Adv}_{\Pi, \mathcal{A}}^{\text{ASU}} = \left| \Pr[\text{Exp}_{\Pi, \mathcal{A}}^{\text{ASU}}(1^\lambda) = 1] - \frac{1}{2} \right|.$$

The protocol satisfies ASU if this advantage is negligible for every PPT adversary.

B.5 Linked-Swap Identity Privacy

Linked-Swap Identity Privacy (LSIP) addresses the stronger coalition in which the LP and both issuers may collude and the swap-settlement association is assumed known. The property does not attempt to hide that association; it requires that the coalition still cannot determine which of two registered candidate users participated in the linked swap.

Figure 12 gives the LSIP experiment.

A challenge specifies two distinct registered candidates U_0, U_1 , one common admissible swap request req , and one fixed nonempty selection $\mathcal{I}$ from the payment received by the user. Both candidates have the required registrations and wallet capacities and neither belongs to **Revealed**. Let b be the input currency direction and let $b_{\text{out}} = 1 - b$. Define

$$\text{Leak}_{\text{LSIP}}(\chi_j) = (\text{req}, V(\mathcal{I})), \quad \chi_j = (U_j, \text{req}, \mathcal{I}).$$

Thus the public request and selected settlement value are identical in both branches.

Before the challenge, the coalition view is

$$\text{View}_{\text{LSIP}} = (\text{pub}, \text{view}_{B_0}, \text{view}_{B_1}, \text{view}_{\text{LP}}).$$

After a successful challenge settlement, $\text{Emit}_{\text{LSIP}}$ additionally reveals $\text{KnownLink} = (\text{swid}, \text{rcpt}_{\text{agg}})$, explicitly identifying the target public swap and its corresponding aggregate settlement.

The LSIP advantage is

$$\text{Adv}_{\Pi, \mathcal{A}}^{\text{LSIP}} = \left| \Pr[\text{Exp}_{\Pi, \mathcal{A}}^{\text{LSIP}}(1^\lambda) = 1] - \frac{1}{2} \right|.$$

The protocol satisfies LSIP if this advantage is negligible for every PPT adversary.

C Formal Cryptographic Assumptions

C.1 Groups, Setup, and Hardness Assumptions

Let gp describe honestly generated cyclic groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ and $\mathbb{H} = \langle h \rangle$ of prime order $p = p(\lambda)$, with generators $g \in \mathbb{G}_1$, $\tilde{g} \in \mathbb{G}_2$, and an efficiently computable, non-degenerate asymmetric pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. We require $1/p = \text{negl}(\lambda)$ and efficient group operations and membership checks. All group challenges below include the full description gp , including the pairing and the second source group.

Setup. The serial base δ is uniform in $\mathbb{G}_1 \setminus \{1_{\mathbb{G}_1}\}$ and independent of issuer keys and other setup material. We model its derivation $\text{HashToGroup}_{\mathbb{G}_1}(\text{ecash/serial})$ as an independent, domain-separated programmable random oracle into this set. Thus a reduction may install an independently distributed challenge base at that label, consistently with every subsequent oracle answer. No corresponding $\tilde{g}^{\log_g \delta}$ is published. Commitment parameters and proof parameters are honestly generated; real parties receive neither discrete-logarithm relations among independent commitment bases nor proof-setup trapdoors. Simulation and extraction trapdoors are confined to the reductions.

Discrete logarithm (DL). For $(\mathbb{K}, b) \in \{(\mathbb{G}_1, g), (\mathbb{H}, h)\}$ and every PPT algorithm $\mathcal{A}$, we assume

$$\text{Adv}_{\text{DL}, \mathcal{A}}^{\mathbb{K}}(\lambda) := \Pr[\mathcal{A}(gp, b, b^x) = x] \leq \text{negl}(\lambda), \quad x \leftarrow \$ \mathbb{Z}_p.$$

The probability includes group generation and the algorithm's randomness.

External Diffie–Hellman (XDH). We assume decisional Diffie–Hellman hardness in $\mathbb{G}_1$, even given the pairing parameters [22]: for every PPT $\mathcal{A}$,

$$\begin{aligned} \text{Adv}_{\text{XDH}, \mathcal{A}}(\lambda) := & \left| \Pr[\mathcal{A}(gp, g^a, g^u, g^{au}) = 1] \right. \\ & \left. - \Pr[\mathcal{A}(gp, g^a, g^u, g^z) = 1] \right| \leq \text{negl}(\lambda), \end{aligned}$$

where $a \leftarrow \$ \mathbb{Z}_p^*$ and $u, z \leftarrow \$ \mathbb{Z}_p$ are independent. Conditioning $a \neq 0$ gives the nonidentity base required by the setup; this is equivalent to the usual XDH formulation up to a negligible term. The privacy proofs' Adv_{XDH} includes the losses of all such reductions. DL hardness in $\mathbb{G}_1$ already follows from XDH; we name it separately for the accountability arguments.

C.2 Zero-Knowledge Proof Systems

For $t \in \{\text{reg}, \text{iss}, \text{seed}, \text{pay}, \text{agg}, \text{red}\}$, let $R_t(x, w)$ be the corresponding relation in Appendix E, with public input x and witness w . Let crs collect the fixed, domain-separated proof parameters and authorized verification keys. All algorithms share a consistent, domain-separated oracle interface, with programming permitted only by the specified proof-system model. We require the following joint properties of Prove_t and Verify_t .

Completeness. For honestly generated parameters and every (x, w) with $R_t(x, w) = 1$, $\text{Verify}_t(\text{crs}, x, \text{Prove}_t(\text{crs}, x, w)) = 1$ with probability one.

Adaptive multi-theorem zero knowledge. A simulated setup produces $(\widetilde{\text{crs}}, \tau)$, where τ contains the private simulation state. In experiments ZK_{real} and ZK_{sim} , a PPT adversary $\mathcal{A}$ receives the respective public parameters and makes polynomially many adaptive requests (t, x, w) . Both experiments reject $R_t(x, w) = 0$; otherwise they return a real proof or $\text{Sim}_t(\tau, x)$, respectively. The simulator receives no witness. The experiments output $\mathcal{A}$'s final bit, and we require

$$\text{Adv}_{\text{ZK}, \mathcal{A}}(\lambda) := \left| \Pr[\text{ZK}_{\text{real}}^{\mathcal{A}} = 1] - \Pr[\text{ZK}_{\text{sim}}^{\mathcal{A}} = 1] \right| \leq \text{negl}(\lambda).$$

This comparison includes setup, any proof-system random-oracle queries, and the issuer keys, coins, and LP state permitted by the privacy game. Trapdoors and the private coins of replaced provers remain unexposed; proofs with exposed prover coins remain real. The privacy proofs' Adv_{ZK} includes all real-to-simulated proof replacements and their verification-failure gaps.

Simulation extractability. For every PPT $\mathcal{A}$ given the simulated parameters and adaptive access to Sim , require an efficient extractor $\text{Ext}_{\mathcal{A}}$ with access to the adversary, extraction state, and oracle transcripts. If $\mathcal{A}$ outputs (t, x, π) and extraction returns w , then

$$\Pr[\text{Verify}_t(\widetilde{\text{crs}}, x, \pi) = 1 \wedge (t, x, \pi) \notin \mathcal{T}_{\text{sim}} \wedge R_t(x, w) = 0] \leq \text{negl}(\lambda),$$

where $\mathcal{T}_{\text{sim}}$ records the exact tagged simulated replies and failed extraction counts as $R_t(x, w) = 0$. Extraction state is generated jointly with the simulated setup and remains private. Copied simulated proofs are excluded from this guarantee. We also require this extraction guarantee under honest setup without simulation queries.

Extraction of completed seed proofs. The honest-setup extraction guarantee applies to the actual accepted public input of R_{seed} , yielding both commitment openings and the attributes satisfying the recorded challenge. No opening is required before that challenge is sampled. In the classical forking reduction of Lemma 3, extraction is performed after each continuation has finished and may rewind the prover. The reduction must be able to run these extractors with externally supplied, honestly distributed commitment parameters, without a commitment trapdoor. Oracle answers are consistent and any permitted proof-oracle programming remains confined to the proof-system domains. Extraction failures in the two continuations are charged explicitly in that lemma. This uses completed-proof extraction, not an online or straight-line extractor for the recursive aggregate proof.

For the batching argument of Lemma 1, additionally require honest-setup extraction for the actual accepted receipt, without changing that receipt or its coefficient-oracle answers. The adversary, verifier, and extractor must form a resettable PPT oracle computation using one consistent, unprogrammed H_{coef} table, including during extraction rewinds. Any proof-oracle programming is confined to separate proof-system domains. Extraction state must be generatable for externally supplied, honestly distributed commitment parameters without their trapdoor. This interface requirement must be established for the chosen proof system.

Simulation and composition in our protocol. For aggregation, the proof tail's public input is $(x_{\text{agg}}, \text{tr}_{\text{agg}})$; simulation preserves the supplied prefix. The simulator must run in polynomial time on every syntactically valid input occurring in the hybrids, including modified or false statements, with actual verification outcomes retained. Zero knowledge is invoked only on valid statements; it gives no acceptance or indistinguishability guarantee for false ones. When extracting an aggregate, the accountability argument must justify extraction from fresh embedded payment proofs under their fixed verification keys and account separately for copied proofs using their provenance. Outer extraction alone supplies neither inner witnesses nor soundness of the separate pairing check.

C.3 Commitments and Authenticated Membership

Let $\text{Com}_\theta(m; r)$ and $\text{VfCom}_\theta(C, m, r)$ denote commitment and opening verification for type $\theta \in \{\text{iss}, \text{agg}\}$, suppressing honestly generated parameters and public context. Issuance commitments include the complete structured final requests $C_W = (C, D_1, D_2, D_3)$ and $C_{W, \text{fin}}^*$, and their preliminary commitments C_W^{pre} and C_W^* ; aggregation commitments include the entire padded commitment vector. We require perfect correctness for every message in the commitment domain. Messages and public contexts have canonical encodings, and the proof statements bind the commitments to their prescribed contexts.

Adaptive joint hiding. In Hid_β , a PPT adversary $\mathcal{A}$ adaptively submits polynomially many requests (θ, m_0, m_1) , with both messages in the commitment domain

and of the same public padded shape and context. Each reply is $\text{Com}_\theta(m_\beta; r)$, using independently sampled fresh blinding; the same bit β is used throughout. The experiment outputs $\mathcal{A}$'s final bit. We require

$$\text{Adv}_{\text{Hid}, \mathcal{A}}(\lambda) := \left| \Pr[\text{Hid}_0^{\mathcal{A}} = 1] - \Pr[\text{Hid}_1^{\mathcal{A}} = 1] \right| \leq \text{negl}(\lambda).$$

The auxiliary view may contain issuer keys and coins, permitted LP state, and independently generated proof-simulation state, but not the fresh blinding randomness of replaced commitments. Commitments with exposed blinding remain real. The terms $\text{Adv}_{\text{Hid}}^{\text{iss}}$, $\text{Adv}_{\text{Hid}}^{\text{out}}$, and $\text{Adv}_{\text{Hid}}^{\text{agg}}$ cover all source-contribution and source-seed replacements in both request stages, output-holder replacements in both stages, and aggregation-prefix replacements, respectively. Each source-request hiding experiment includes the intervening public seed challenges and issuer state as auxiliary information; the holder's contribution, seed, and fresh commitment openings remain unexposed. They are distinct uses of this requirement, each counted once in the relevant proof bound.

Computational binding. For every PPT adversary outputting $(\theta, C, m, r, m', r')$ under one fixed public context, require

$$\Pr[m \neq m' \wedge \text{VfCom}_\theta(C, m, r) = \text{VfCom}_\theta(C, m', r') = 1] \leq \text{negl}(\lambda).$$

The probability includes honest setup and adversarial randomness, with the permitted auxiliary information and no commitment-setup trapdoor. Aggregation commitments and opening verification use fixed parameters and oracle domains disjoint from coefficient and proof-system oracles. Binding applies to the full message tuple. Conditional entropy of honest aggregation prefixes is a separate requirement in Appendix C.7.

Authenticated membership. For any accepted root RegRoot_b , let $\mathcal{S}_b$ be the set determined by its authorized registration history. Honest membership witnesses must verify, and no PPT adversary can produce an accepting membership witness for $P \notin \mathcal{S}_b$, except with negligible probability. Root updates must faithfully implement that history; update authorization is enforced by the registration protocol. The root is public and need not be hiding; the membership proofs reveal no information about the selected member or witness beyond their public statements, under Appendix C.2.

C.4 Threshold Credentials and Blind Issuance

For the attribute vector $(sk_{X,b}, v, N)$, we require the threshold blind PS issuance security used by Rial–Piotrowska [22, Appendices A.5 and D, Game 5], under its setup and random-oracle assumptions. With threshold t_b and $f_b < t_b$ statically corrupted authorities, every PPT adversary has negligible probability of producing a valid credential on a tuple for which fewer than $t_b - f_b$ distinct honest authorities issued shares; the chosen parameters must make the cited

threshold reduction loss preserve negligibility. The instantiation must retain the commitment-bound signature-base derivation and compatible share validation, unblinding, and combination, with all request components referring to the same three final attributes. The blind-signature subprotocol must complete correctly with probability one for well-formed authorized final requests and t_b distinct valid responses. This does not assert perfect correctness of the seed-generation wrapper, whose bounded retries may be exhausted. The proof and commitment requirements are given in Appendices C.2 and C.3. This corruption bound applies to accountability; the passive privacy experiments permit full issuer-key exposure. Authorization of aggregate reissue is analyzed separately in Appendix F.

Issuer common randomness and joint seeds. We assume an issuer-side functionality $\mathcal{F}_{\text{seed}}$: after an authorized, session-bound preliminary commitment to (sk, u, N) and its proof have been accepted, it records and returns a fresh uniform $r \leftarrow \$ \mathbb{Z}_p$. The contribution is independent of the committed opening and of the complete history before this challenge, including earlier challenges and the adversary’s state. All authorities use the same recorded $(\text{aid}, a, C^{\text{pre}}, r)$, where aid identifies one authorization and a is its attempt number. It is unavailable before the commitment is fixed; repeated retrieval returns the same value. Challenges are ordered and all attempts, including aborted attempts, count toward the experiment’s polynomial bound q_{seed} . Neither aborts nor retries authorize additional value, and at most one final request per authorization can receive shares. The final proof binds $v = u + r \pmod{p}$, the same sk, N , and both commitment openings to that recorded challenge.

This common-randomness functionality is an explicit model assumption under the threshold corruption bound, not a property supplied by PS unforgeability. A distributed implementation must realize it despite corrupt authorities, including agreement, unpredictability before commitment, and resistance to adaptive contribution cancellation; a holder-chosen public hash is not a substitute. A possible realization uses a threshold common coin in the random-oracle model, following the approach of Cachin–Kursawe–Shoup.³ For example, hash a unique threshold signature on a domain-separated encoding of $(B_b, \text{aid}, a, C^{\text{pre}})$ into $\mathbb{Z}_p$, with validated shares and honest shares released only after the authorization is fixed. Uniqueness gives agreement on a completed signature, but a realization must also prove pseudorandomness of its field output for adaptive sessions, account for early oracle queries, and specify setup and abort behavior. Its computational realization loss must be added to the security bounds; the present theorems retain $\mathcal{F}_{\text{seed}}$ as an explicit functionality assumption. The full issuer-key exposure allowed in the passive privacy games also requires a separate transcript argument, so signature unforgeability alone is not a replacement for that analysis. The commit-then-add pattern follows compact e-cash withdrawal [7, Section 4.1]; the threshold realization remains a separate requirement. In the passive privacy games, the functionality and issuers execute honestly even when their keys and coins are exposed.

³ <https://eprint.iacr.org/2000/034>

For honest holders, sample $u \leftarrow \$ \mathbb{Z}_p$ independently. If $u + r \notin \mathcal{V}$, keep the same preliminary commitment and obtain a new challenge under a new attempt number, up to $K_{\text{seed}} = \text{poly}(\lambda)$ attempts, with $K_{\text{seed}} \geq 1$. Each conditional rejection probability is L_{max}/p ; exhaustion for one honest authorization has probability at most $(L_{\text{max}}/p)^{K_{\text{seed}}}$. Accepted honest seeds are uniform on $\mathcal{V}$. This statement does not claim uniformity after malicious selective aborts: the security proof counts all q_{seed} attempts. Choosing $K_{\text{seed}} = 1$ gives the abort-on-invalid variant. Its visible rejection event still contributes to the source-privacy domain loss ε_{dom} in Appendix G; removing retries does not remove that event.

C.5 DH-Derived Pseudorandomness

For an honest exchange session s with context $c_s = (\text{pid}, \text{rid}, \text{leg})$, write

$$Z_s = Y_{\text{rcv},s}^{\xi_{\text{snd},s}} = Y_{\text{snd},s}^{\xi_{\text{rcv},s}}, \quad K_s = H_{\text{kdf}}(Z_s, c_s), \\ F_{\text{DH}}(Z_s, c_s, k) = (H_{\text{d}}(K_s, c_s, k), H_{\text{S}}(K_s, c_s, k), H_{\text{T}}(K_s, c_s, k)).$$

Define experiments DHReal and DHIdeal that generate keys as in RKeyGen , retain LP key reuse and fresh user keys, and give the adversary the public keys. For polynomially many adaptive target requests addressed by session and coin handles (s, k) , the real experiment returns $F_{\text{DH}}(Z_s, c_s, k)$. The ideal experiment instead samples an independent uniform triple in $\mathbb{Z}_p^3$ for each distinct (Z_s, c_s, k) and reuses it on repeated requests. This table is private; public hash evaluations and ordinary-session derivations remain real in both experiments. Opposite swap legs may share Z_s but have distinct contexts.

Admissible targets use endpoint key handles that remain unexposed; their contexts are unused by ordinary derivation queries, both before and after target selection. Ordinary exposure interfaces cannot request those keys or target-session Z_s , K_s , or real output triples. Public hash queries, including guesses of protected inputs, remain unrestricted. The auxiliary view includes independent issuer keys and coins, independently generated proof-simulation state, and the ordinary-session exposures permitted by SSU and ASU. For every PPT adversary $\mathcal{A}$, require

$$\text{Adv}_{\text{DH/KDF}, \mathcal{A}}(\lambda) := \left| \Pr[\text{DHReal}^{\mathcal{A}} = 1] - \Pr[\text{DHIdeal}^{\mathcal{A}} = 1] \right| \leq \text{negl}(\lambda),$$

where each experiment outputs $\mathcal{A}$'s final bit. The proof term $\text{Adv}_{\text{DH/KDF}}$ covers all replaced triples, including guessing and key-collision effects under the actual key-generation distribution. This is a requirement on the composed DH/KDF and offset/mask derivation; DL hardness alone does not imply it. LSIP retains the real values computable from the LP's exposed keys and does not invoke this assumption.

C.6 Inverse-Exponent Pseudorandomness

Let $L_{\text{max}} = \text{poly}(\lambda)$ with $1 \leq L_{\text{max}} < p$. Each protected target wallet w has an independent seed $v_w \leftarrow \$ \mathcal{V}$, where $\mathcal{V} = \mathbb{Z}_p \setminus \{-1, \dots, -L_{\text{max}}\}$. Under the full

pairing and serial-base setup of Appendix C.1, define

$$F_{v_w}(\ell) = (\delta^{1/(v_w+\ell+1)}, g^{1/(v_w+\ell+1)}), \quad 0 \leq \ell < L_{\max}.$$

In **PRFReal**, an adversary adaptively queries polynomially many wallet–position handles (w, ℓ) and receives $F_{v_w}(\ell)$. In **PRFIdeal**, each distinct pair instead receives $(\delta^{z_{w,\ell}}, g^{z_{w,\ell}})$ for independent $z_{w,\ell} \leftarrow \mathbb{Z}_p$, with the same output reused on repeated queries. Protected seeds and the inverse or ideal exponents remain unexposed. Non-target wallets, including ordinary-query wallets and the LP’s known wallets in LSIP, remain real with independent seeds.

The adversary receives the public parameters and may have issuer keys and coins, proof-simulation state, and the LP information permitted by the relevant privacy game. This auxiliary information must be efficiently generated without the protected seeds, except through the group-output oracle. In the privacy hybrids this includes the preliminary commitments, issuer seed challenges, final requests, and seed-consistency proofs after their hiding and simulation transformations; it does not assume that a real issuance transcript is independent of its seed. For every PPT adversary $\mathcal{A}$, we require

$$\text{Adv}_{\text{PRF}, \mathcal{A}}(\lambda) := \left| \Pr[\text{PRFReal}^{\mathcal{A}} = 1] - \Pr[\text{PRFIdeal}^{\mathcal{A}} = 1] \right| \leq \text{negl}(\lambda),$$

where the experiments output $\mathcal{A}$ ’s final bit. The proof term Adv_{PRF} covers all protected wallets and positions and the valid-seed distribution above; it is not charged separately per record.

The bounded-domain DDHI analysis of Rial–Piotrowska [22, Appendices A.6 and D, Game 13] provides the corresponding single-base approach; applying it here requires compatibility with the full setup and auxiliary information specified above. The ideal outputs still share one exponent. Their subsequent replacement by independent group elements uses XDH from Appendix C.1.

C.7 Hash Functions and Freshness Conditions

Hash model. All hash inputs use canonical, injective encodings and distinct domains. The payload hash H is collision resistant: for every PPT $\mathcal{A}$ outputting (M, M') from the public parameters, $\Pr[M \neq M' \wedge H(M) = H(M')] \leq \text{negl}(\lambda)$; the same requirement applies to hashes used for authenticated membership. Model $H_{\text{coef}} : \{0, 1\}^* \rightarrow \mathbb{Z}_p^*$ and $H_{\text{ch}} : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ as independent public random oracles, with independent uniform answers at distinct inputs and consistent repeated answers. Hash-to-group follows Appendix C.1. Additional proof-system and signature-base oracles must be specified by the instantiations in Appendices C.2 and C.4, under disjoint domains; the DH-derived hash interface follows Appendix C.5.

Honest aggregation prefixes. Every honest aggregation attempt, including retries, uses fresh independent commitment blinding. Let $\mathcal{H}$ contain the complete preceding oracle history, fixed public bindings, and permitted auxiliary information,

including issuer–LP state where exposed, and let m be the fixed committed messages. In the real executions and privacy hybrids invoking freshness, require, for every attainable $(\mathcal{H}, m)$,

$$\max_t \Pr[\text{tr}_{\text{agg}} = t \mid \mathcal{H}, m] \leq 2^{-\kappa(\lambda)}.$$

Prefix generation is independent of unused coefficient-oracle values; the prefix and committed messages are fixed before the honest prover queries its coefficients. This condition is additional to commitment hiding.

Post-request randomness. Sample ω after the exchange keys, context c , and offsets $d_k = H_d(K, c, k)$ are fixed. Let $\mathcal{H}_{\text{ch}}$ contain these data, the preceding oracle history, and all permitted issuer–LP information. For every attainable such history, require

$$\max_u \Pr[\omega = u \mid \mathcal{H}_{\text{ch}}] \leq 2^{-\kappa_{\text{ch}}(\lambda)},$$

and independence of unused H_{ch} values. For the same public request, exchange keys, and common history, context and beacon generation have the same law for both protected identities. These requirements apply in the relevant privacy hybrids; post-request ordering alone does not imply them.

Bounds and scope. All query, wallet, and record bounds are polynomial, with $\kappa, \kappa_{\text{ch}} = \omega(\log \lambda)$ and $1/p = \text{negl}(\lambda)$. Using the query counts and attempt cap $1 \leq L = \text{poly}(\lambda)$ of Appendices G.1, G.2, and G.3, the resulting honest-execution bounds are

$$\begin{aligned} \eta &= \frac{1}{p-1} + Q_{\text{coef}} 2^{-\kappa}, \quad \varepsilon_{\text{tr}} = Q_{\text{agg}} \eta^L, \\ \varepsilon_0 &= \frac{N_{\text{in}}}{p} + Q_{\text{ch}} 2^{-\kappa_{\text{ch}}}. \end{aligned}$$

These quantities are negligible. The prefix condition concerns honest generation; soundness against malicious aggregation requires a separate argument accounting for adversarial coefficient queries and selection of prefixes.

D Supplementary AMM Proofs

Proof (Proof of Proposition 1). When the user’s payment is accepted, the output b_i , its settlement components b_i^{ec} and b_i^{nat} , and the rate ρ_{out} are fixed, and the delivery collateral $D_i^{\text{del}} = \rho_{\text{out}} b_i$ is locked from the LP’s deposit.

A timely LP submission consists of publishing the e-cash payment through $\mathcal{D}$ and submitting its digest and public proof to the contract within Θ_{sub} . Since

$$\Theta_{\text{tot}} \geq \Theta_{\text{sub}} + \Delta_{\mathcal{D}}^{\text{live}} + \Delta_{\mathcal{B}}^{\text{live}},$$

an honest timely submission has sufficient time to obtain DA confirmation and on-chain verification before the total deadline.

If the LP's payment is accepted by the contract, **PubVf** soundness and receiver binding imply that it contains the fixed b_i^{ec} units of E_{out} for the user's exchange key and this request. The contract then transfers b_i^{nat} from the locked collateral to the user and releases the remaining $D_i^{\text{del}} - b_i^{\text{nat}}$ to the LP. Subsequent honoring of an honest receiver's valid non-replay receipt, including funded replacement for a conflicting spend, relies on Section 3.1.

Otherwise, if no acceptable LP payment is accepted by Θ_{tot} , the timeout transition transfers the full locked D_i^{del} to the user.

Finally, acceptance and timeout are mutually exclusive terminal transitions. Once either transition closes rid_i , the contract rejects any further transition for that request. Hence the contract reaches exactly one of the two terminal outcomes.

Proof (Proof of Proposition 2). Assume without loss of generality that request i inputs a_i units of E_0 . Let the CPMM state immediately before i be (x_0, y_0) with $K = x_0 y_0$. Consider the two pricing replays defining L_i : the $+$ execution includes i , while the $-$ execution omits it. Immediately after i ,

$$(x_0^+, y_0^+) = \left(x_0 + a_i, \frac{K}{x_0 + a_i} \right), \quad (x_0^-, y_0^-) = (x_0, y_0).$$

Each replayed request is evaluated using the full CPMM output, so both executions remain on the same constant-product curve,

$$x_t^+ y_t^+ = x_t^- y_t^- = K.$$

Define the E_0 -reserve gap

$$\delta_t = x_t^+ - x_t^-,$$

with $\delta_0 = a_i$. A later E_0 input increases both E_0 reserves equally, so $\delta_t = \delta_{t-1}$ and contributes nothing to L_i . For a later E_1 input c , the resulting E_0 reserve is

$$\phi_c(x) = \frac{Kx}{K + cx},$$

where

$$0 < \phi'_c(x) = \frac{K^2}{(K + cx)^2} < 1.$$

Hence

$$0 \leq \delta_t \leq \delta_{t-1}.$$

Moreover, the additional E_0 output in the $+$ replay is

$$\ell_t = (x_{t-1}^+ - x_t^+) - (x_{t-1}^- - x_t^-) = \delta_{t-1} - \delta_t.$$

Summing over all later requests therefore telescopes. Since every $\ell_t \geq 0$ and L_i sums ℓ_t only over the subset of these requests that complete,

$$0 \leq L_i \leq \sum_t \ell_t = \delta_0 - \delta_T \leq a_i.$$

The bound is tight in the supremum: for a single subsequent E_1 input $c \rightarrow \infty$, $\delta_T \rightarrow 0$ and hence $L_i \rightarrow a_i$.

If fixed-point arithmetic increases the aggregate loss by at most ε_{fp} , then

$$L_i \leq a_i + \varepsilon_{\text{fp}}.$$

Therefore

$$\rho_{\text{in}} L_i \leq \rho_{\text{in}}(a_i + \varepsilon_{\text{fp}}) = D_i^{\text{con}},$$

so the prescribed concurrency collateral is sufficient.

Proof (Proof of Theorem 1). Fix an admitted request i . At admission, its input a_i , output b_i , settlement components $(b_i^{\text{ec}}, b_i^{\text{nat}})$, rates, and collateral amounts are fixed, and its expected flow is incorporated into $S^\vee$ before any later request is priced.

Suppose first that the user's payment is not accepted by its deadline. The contract aborts i , removes its expected flow from $S^\vee$, and keeps D_i^{con} locked. Since $\mathcal{A}_i$ is fixed at the abort and each of its requests terminates within its bounded deadlines, the deferred compensation is resolved in bounded time. By assumption, its abort loss satisfies

$$L_i \leq \bar{L}_i, \quad D_i^{\text{con}} \geq \rho_{\text{in}} \bar{L}_i,$$

and therefore

$$\rho_{\text{in}} L_i \leq D_i^{\text{con}}.$$

Hence the collateral covers the loss attributed to the abort, and any remaining collateral is returned to the user. For multiple aborts, the same argument applies in resolution order to each marginal loss L_i computed after the previous aborts have been incorporated.

Now suppose that the user's payment is accepted. The abort transition is disabled and D_i^{con} is released. If the LP payment is accepted by its deadline, sound public verification ensures that it is the fixed b_i^{ec} -unit payment bound to Y_i and rid_i , while the contract transfers b_i^{nat} to the user. Thus the fixed settlement $(b_i^{\text{ec}}, b_i^{\text{nat}})$ is accepted; issuer-side honoring for an honest receiver relies on Section 3.1. If no acceptable LP payment is accepted by the deadline, the LP-default transition instead transfers the full D_i^{del} to the user.

These three cases are exhaustive and mutually exclusive, since each terminal transition closes rid_i . Hence every admitted request reaches exactly one of the stated terminal outcomes.

Finally, each output b_j is fixed at admission. Repairing $S^\vee$ after an abort changes only the pricing state used for future admissions and does not modify any already fixed request. Therefore the output of every previously admitted request remains unchanged.

E An Exchange-Compatible Compact E-Cash Construction

We instantiate the exchange-compatible e-cash interface of Section 3.2 from the compact threshold e-cash construction of Rial–Piotrowska [22]. For each issuer B_b ,

$$\begin{aligned} \text{EC}_b^{\text{ex}} &= (\text{Setup}_b, \text{Register}_b, \text{Issue}_b, \text{Exch}_b, \text{Settle}_b, \text{Identify}_b), \\ \text{Exch}_b &= (\text{RKeyGen}, \text{SpendEx}_b, \text{PubVf}_b, \text{Aggregate}_b). \end{aligned}$$

The construction retains the compact serial/tag algebra and augments it with an authenticated wallet capacity, receiver-bound exchange keys, masked public coin records, and anonymous aggregate reissue. Credential consistency is proved in zero knowledge, whereas PS credential authenticity is checked separately by a native pairing-product equation. We write $\text{reg}_{X,b}$ for the private registration state of holder X under issuer B_b , and

$$W_{X,b} = (\sigma, v, N, \ell), \quad \sigma = (h_\sigma, s_\sigma),$$

for a wallet whose PS credential authenticates $(sk_{X,b}, v, N)$ and whose next unused position is ℓ . Malformed inputs or failed authentication, proof, credential, or range checks return $\perp$; public verification returns 0.

$\text{Setup}_b(1^\lambda) \rightarrow (pp_b, st_b)$. Let pp_{ex} contain the common exchange parameters, including a cyclic DH group $\mathbb{H} = \langle h \rangle$ of prime order p , canonical encodings, and the domain-separated hash functions used below. Generate compatible asymmetric bilinear groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of order p with generators $g \in \mathbb{G}_1$ and $\tilde{g} \in \mathbb{G}_2$ and pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Fix $1 \leq L_{\max} < p$ with $L_{\max} = \text{poly}(\lambda)$, the supported payment bound $V_{\max}$, and a public aggregation bound $m_{\max}$ on the number of source payments. The group order is chosen so that $L_{\max}/p$ is negligible.

Issuer B_b generates threshold PS signing-key shares for three messages $(sk_{X,b}, v, N)$. Write the corresponding combined verification key as

$$\tilde{\alpha}_b = \tilde{g}^{x_b}, \quad \tilde{\beta}_{b,1} = \tilde{g}^{y_{b,1}}, \quad \tilde{\beta}_{b,2} = \tilde{g}^{y_{b,2}}, \quad \tilde{\beta}_{b,3} = \tilde{g}^{y_{b,3}},$$

and publish the matching $\mathbb{G}_1$ -elements

$$\beta_{b,1} = g^{y_{b,1}}, \quad \beta_{b,2} = g^{y_{b,2}}, \quad \beta_{b,3} = g^{y_{b,3}}.$$

The latter are used only to form and rerandomize public credential presentations.

For $\Gamma = (\Gamma_0, \Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4) \in \mathbb{G}_1^5$, define the native credential check

$$\text{PairCred}_b(\Gamma) = 1 \tag{1}$$

iff $\Gamma_0 \neq 1_{\mathbb{G}_1}$ and

$$e(\Gamma_0, \tilde{\alpha}_b) e(\Gamma_1, \tilde{\beta}_{b,1}) e(\Gamma_2, \tilde{\beta}_{b,2}) e(\Gamma_3, \tilde{\beta}_{b,3}) e(\Gamma_4, \tilde{g}) = 1_{\mathbb{G}_T}.$$

This pairing-product check is performed natively and is not part of the zero-knowledge circuit.

Generate the commitment parameters used by blind issuance and by the hiding and binding commitments in the aggregation transcript. Use transparent proof parameters or obtain the payment and aggregation reference strings from an independent proof setup; in the latter case, neither issuer nor the LP receives that setup's private randomness or trapdoors. Derive the compact serial base independently, with no party given its discrete logarithm relative to g , as

$$\delta = \text{HashToGroup}_{\mathbb{G}_1}(\text{ecash}/\text{serial}).$$

The public parameters pp_b contain the issuer verification material, $\{\beta_{b,i}, \tilde{\beta}_{b,i}\}_{i=1}^3$, $\tilde{\alpha}_b$, commitment and proof parameters, δ , and pp_{ex} . The issuer state st_b contains its threshold signing shares, an initially empty registration table and authenticated root RegRoot_b , initially empty honored-record and settlement-reservation state, a persistent table of seed authorizations, attempts, challenges, and finalized requests for $\mathcal{F}_{\text{seed}}$, and access to the funded recovery functionality of Section 3.1, including its funding and claim ledger.

Register_b. Registration is interactive:

$$\langle X(\text{id}_X), B_b(st_b) \rangle_{\text{Register}_b} \rightarrow (\text{reg}_{X,b}; st'_b).$$

The holder samples $sk_{X,b} \leftarrow \mathbb{Z}_p$, sets $P_{X,b} = g^{sk_{X,b}}$, authenticates as id_X , and proves knowledge of $sk_{X,b}$. The issuer stores the association $(\text{id}_X, P_{X,b})$ and updates RegRoot_b . The holder retains

$$\text{reg}_{X,b} = (sk_{X,b}, \text{wit}_{X,b}^{\text{reg}}).$$

The secret $sk_{X,b}$ is not disclosed to the issuer.

Joint seed finalization. An authorization aid fixes the issuer, mode, capacity, and one preliminary commitment C^{pre} to (sk, u, N) . Its accepted preliminary proof establishes knowledge of this opening. For ordinary identified issuance, the authorization also fixes X 's registered handle $P_{X,b}$ and the proof checks $g^{sk} = P_{X,b}$. For anonymous reissue, it instead checks membership of the hidden g^{sk} in RegRoot_b , without disclosing a handle. Bind the commitment and proof to (B_b, aid) and this stored authorization context; the holder cannot select a different mode. The issuer obtains r from $\mathcal{F}_{\text{seed}}$ only after this acceptance. The holder sets $v = u + r \pmod{p}$. If $v \notin \mathcal{V}$, it requests a fresh numbered attempt with the same preliminary commitment, up to K_{seed} attempts. An adversarial holder may also abandon an attempt, but every challenge is recorded and counts toward the cap; a new attempt invalidates the preceding one for finalization. Exhaustion returns $\perp$ without releasing shares or creating a new value authorization.

For $v \in \mathcal{V}$, form a freshly blinded structured PS request C^{fin} to (sk, v, N) and provide Π_{seed} for

$$\begin{aligned} R_{\text{seed}} : \quad C^{\text{pre}} &= \text{Com}_{\text{iss}}((sk, u, N); \rho_{\text{pre}}), \\ C^{\text{fin}} &= \text{Com}_{\text{iss}}((sk, v, N); \rho_{\text{fin}}), \\ v &= u + r \pmod{p}, \quad v \in \mathcal{V}. \end{aligned}$$

Here commitment notation includes verification of every structured request component and its context. The public input includes $B_b, \text{aid}, a, r, N, C^{\text{pre}}, C^{\text{fin}}$, the mode, and the accepted registration root. For identified issuance it also includes the stored $P_{X,b}$ and checks $g^{sk} = P_{X,b}$; anonymous reissue includes no public holder handle and checks hidden membership instead. The witness contains both openings, the secret, and the required registration witness. The proof also checks $1 \leq N \leq L_{\text{max}}$ and the same authorization context as the preliminary proof. Authorities check this proof and the stored current attempt, atomically bind the authorization to exactly this final request, and only then issue PS shares on (sk, v, N) . Resumption returns shares for that same request, never a second wallet. No preliminary contribution is itself signed.

Issue_b. Wallet issuance is interactive:

$$\langle X(\text{reg}_{X,b}, N), B_b(N, st_b) \rangle_{\text{Issue}_b} \rightarrow (W_{X,b}; st'_b),$$

for an authorized integer capacity $1 \leq N \leq L_{\text{max}}$, with $\mathcal{V} = \mathbb{Z}_p \setminus \{-1, \dots, -L_{\text{max}}\}$. The issuer allocates a fresh authorization identifier **aid** for **IssueAuth_b**(X, N), recording the authenticated X and its registered handle $P_{X,b}$. The holder samples $u \leftarrow \$ \mathbb{Z}_p$ and fresh blinding, commits to $(sk_{X,b}, u, N)$ in C_W^{pre} , and proves its opening, $g^{sk_{X,b}} = P_{X,b}$, and capacity, bound to this identified authorization. Membership of an arbitrary registered handle alone does not suffice. Apply joint seed finalization above with $C^{\text{pre}} = C_W^{\text{pre}}$ and $C^{\text{fin}} = C_W = (C, D_1, D_2, D_3)$. This yields $v = u + r \in \mathcal{V}$; only the accepted final request is signed.

The threshold authorities return blinded PS signature shares on $(sk_{X,b}, v, N)$. The holder validates the shares, retains a threshold set of distinct authorized responses, privately unblinds and combines them, and obtains

$$\sigma = (h_\sigma, s_\sigma).$$

The resulting credential satisfies

$$e(h_\sigma, \tilde{\alpha}_b \tilde{\beta}_{b,1}^{sk_{X,b}} \tilde{\beta}_{b,2}^v \tilde{\beta}_{b,3}^N) = e(s_\sigma, \tilde{g}).$$

The holder outputs

$$W_{X,b} = (\sigma, v, N, 0).$$

The issuer records the authorization and knows r and the public acceptance or retry outcomes. The values $sk_{X,b}, u, v$, the request openings, and the completed unblinded credential remain hidden, subject to these public outcomes. In particular, knowledge of r does not reveal $v = u + r$, because u remains hidden by the commitments and ZK proofs.

$\text{RKeyGen}(pp_{\text{ex}}) \rightarrow (Y, \xi)$. Sample $\xi \leftarrow \mathbb{Z}_p$ independently of all registration and wallet secrets and return

$$(Y, \xi) = (h^\xi, \xi).$$

A user generates a fresh pair for each exchange request; the LP may maintain a persistent pair $(Y_{\text{LP}}, \xi_{\text{LP}})$. On a payment leg, $(Y_{\text{snd}}, \xi_{\text{snd}})$ denotes the sender pair and Y_{rcv} the designated receiver key.

Public credential presentation. For a credential $\sigma = (h_\sigma, s_\sigma)$ on $(sk_{X,b}, v, N)$, sample

$$r_\sigma \leftarrow \mathbb{Z}_p^*, \quad u_1, u_2, u_3 \leftarrow \mathbb{Z}_p,$$

and set

$$h^* = h_\sigma^{r_\sigma}, \quad s^* = s_\sigma^{r_\sigma}.$$

Define

$$\begin{aligned} \Gamma_0 &= h^*, \\ \Gamma_1 &= (h^*)^{sk_{X,b}} g^{u_1}, \\ \Gamma_2 &= (h^*)^v g^{u_2}, \\ \Gamma_3 &= (h^*)^N g^{u_3}, \\ \Gamma_4 &= (s^*)^{-1} \beta_{b,1}^{-u_1} \beta_{b,2}^{-u_2} \beta_{b,3}^{-u_3}. \end{aligned} \tag{2}$$

For a valid PS credential, substitution into Eq. (1) cancels the randomizers u_1, u_2, u_3 and leaves exactly the PS verification relation. Thus Γ is a randomized public credential presentation; it is not itself a new credential.

We also use the following public rerandomization operation. For $\rho \leftarrow \mathbb{Z}_p^*$ and $t_1, t_2, t_3 \leftarrow \mathbb{Z}_p$, define

$$\text{ReRand}_b(\Gamma; \rho, t_1, t_2, t_3) = \Gamma', \tag{3}$$

where

$$\begin{aligned} \Gamma'_0 &= \Gamma_0^\rho, \\ \Gamma'_1 &= \Gamma_1^\rho g^{t_1}, \\ \Gamma'_2 &= \Gamma_2^\rho g^{t_2}, \\ \Gamma'_3 &= \Gamma_3^\rho g^{t_3}, \\ \Gamma'_4 &= \Gamma_4^\rho \beta_{b,1}^{-t_1} \beta_{b,2}^{-t_2} \beta_{b,3}^{-t_3}. \end{aligned}$$

The transformation preserves the credential relation:

$$\text{PairCred}_b(\Gamma) = 1 \implies \text{PairCred}_b(\Gamma') = 1.$$

SpendEx_b . The exchange-spending algorithm is

$$\text{SpendEx}_b(W_{X,b}, \text{reg}_{X,b}, V, \xi_{\text{snd}}, Y_{\text{rcv}}, c, \omega) \rightarrow (W'_{X,b}, \text{rcpt}_{\text{pay}}),$$

where $W_{X,b} = (\sigma, v, N, \ell)$ and $c = (\text{pid}, \text{rid}, \text{leg})$.

Require

$$0 \leq V \leq V_{\max}, \quad 0 \leq \ell \leq \ell + V \leq N.$$

Set

$$Y_{\text{snd}} = h^{\xi_{\text{snd}}}, \quad K = H_{\text{kdf}}(Y_{\text{rcv}}^{\xi_{\text{snd}}}, c).$$

For each $k \in \{0, \dots, V-1\}$, let

$$\ell_k = \ell + k, \quad \mu_k = (v + \ell_k + 1)^{-1},$$

and compute

$$q_k = H_{\text{ch}}(\omega, c, k), \quad d_k = H_{\text{d}}(K, c, k), \quad R_k = q_k + d_k \pmod{p}.$$

The compact serial and tracing tag are

$$S_k = \delta^{\mu_k}, \quad T_k = g^{sk_{X,b}}(g^{R_k})^{\mu_k}. \quad (4)$$

Derive

$$x_k^S = H_S(K, c, k), \quad x_k^T = H_T(K, c, k),$$

and mask the bank-facing record as

$$\hat{S}_k = S_k g^{x_k^S}, \quad \hat{T}_k = T_k g^{x_k^T}. \quad (5)$$

Independently sample fresh r_σ, u_1, u_2, u_3 and construct the payment credential presentation Γ according to Eq. (2). Define

$$M_{\text{pay}} = (\text{ver}, \{(\hat{S}_k, \hat{T}_k)\}_{k=0}^{V-1}), \quad d_{M_{\text{pay}}} = H(M_{\text{pay}}),$$

and

$$x_{\text{pay}} = (\text{sid}, B_b, \text{pid}, \text{rid}, \text{leg}, V, Y_{\text{snd}}, Y_{\text{rcv}}, \omega, d_{M_{\text{pay}}}, \Gamma).$$

Generate Π_{pay} proving knowledge of

$$\sigma, sk_{X,b}, v, N, \ell, \xi_{\text{snd}}, r_\sigma, u_1, u_2, u_3$$

and all derived values such that:

1. Γ is formed exactly according to Eq. (2) from the same hidden $(\sigma, sk_{X,b}, v, N)$ used by the payment;
2. the credential-consistency and wallet-capacity relations hold;
3. Eqs. (4) and (5) hold for every active coin;
4. the DH/KDF, challenge, masking, and payload-digest relations are correct and bound to the stated exchange context.

The relation does *not* evaluate the PS pairing equation. In particular, Π_{pay} proves consistency of the public Γ with the hidden credential witness, while authenticity of that credential is established separately by $\text{PairCred}_b(\Gamma)$.

Return

$$\text{rcpt}_{\text{pay}} = (x_{\text{pay}}, M_{\text{pay}}, \Pi_{\text{pay}}), \quad W'_{X,b} = (\sigma, v, N, \ell + V).$$

The payer adopts $W'_{X,b}$ before publication and does not restore consumed positions if the surrounding AMM execution later aborts.

PubVf_b . Public verification uses the interface

$$\text{PubVf}_b(pp_b, pp_{\text{ex}}, x_{\text{pay}}, \Pi_{\text{pay}}) \rightarrow \{0, 1\}.$$

Check the statement encoding, issuer and deployment identifiers, and exchange-key encodings, and parse Γ from x_{pay} . Return

$$\text{PubVf}_b(pp_b, pp_{\text{ex}}, x_{\text{pay}}, \Pi_{\text{pay}}) = 1 \quad (6)$$

iff these checks pass and

$$\text{VerifyZK}_b(x_{\text{pay}}, \Pi_{\text{pay}}) = 1 \quad \wedge \quad \text{PairCred}_b(\Gamma) = 1.$$

The first check establishes correct use of hidden wallet state and construction of the public payment; the second establishes issuer authenticity of the PS credential. Both refer to the same public Γ . Neither check consults or modifies issuer spent state.

A party verifying a full receipt $\text{rcpt}_{\text{pay}} = (x_{\text{pay}}, M_{\text{pay}}, \Pi_{\text{pay}})$ additionally checks $H(M_{\text{pay}}) = d_{M_{\text{pay}}}$. The AMM checks that the statement matches the admitted request and separately verifies availability of the payload bound to that digest.

Aggregate_b . Aggregation is a local receiver algorithm

$$\text{Aggregate}_b(\text{reg}_{X,b}, \{(\text{rcpt}_{\text{pay},j}, \xi_{\text{rcv},j})\}_{j=1}^m, \mathcal{I}) \rightarrow \text{rcpt}_{\text{agg}},$$

where X is the designated receiver of every selected source payment and $\mathcal{I}$ privately selects individual coin positions. Require $1 \leq m \leq m_{\text{max}}$. Source and selection witnesses are padded to the public bounds m_{max} and V_{max} ; the proof format and commitment prefix do not reveal the number of contributing sources.

For each source

$$\text{rcpt}_{\text{pay},j} = (x_{\text{pay},j}, M_{\text{pay},j}, \Pi_{\text{pay},j}),$$

parse its public credential presentation Γ_j , check the source statement and payload digest, verify its payment ZK proof, and verify that $\xi_{\text{rcv},j}$ opens the designated receiver key. Require $\Gamma_{j,0} \neq 1_{\mathbb{G}_1}$ for every contributing source. The aggregation proof recursively verifies the source ZK proofs and enforces this per-source nonidentity condition; it does not evaluate their PS pairings.

Derive

$$K_j = H_{\text{kdf}}(Y_{\text{snd},j}^{\xi_{\text{rcv},j}}, c_j)$$

and recover every selected (S, R, T) record by removing the masks and recomputing its hidden challenge offset. Let

$$\mathcal{J} = \{j : \exists k \text{ such that } (j, k) \in \mathcal{I}\}$$

be the set of *contributing source payments*. A source belongs to $\mathcal{J}$ exactly once, regardless of how many of its coins are selected.

Let $\mathcal{I}$ select n records in total, $1 \leq n \leq L_{\max}$. Reject repeated selected serials and uniformly shuffle the selected records into

$$\mathcal{C} = ((S_1, R_1, T_1), \dots, (S_n, R_n, T_n)).$$

To prepare reissue, choose a fresh reservation session rsid and let ν_{agg} encode $(\text{sid}, B_b, \text{rsid})$. Sample $u^* \leftarrow \mathbb{Z}_p$ and fresh blinding and form the structured preliminary commitment C_W^* to $(sk_{X,b}, u^*, n)$, bound to this context. Set $M_{\text{agg}} = \mathcal{C}$ and $d_{M_{\text{agg}}} = H(M_{\text{agg}})$. These values are fixed before deriving aggregation coefficients. No issuer interaction occurs during **Aggregate_b**; the final seed v^* is determined during **Settle_b**.

Before aggregating credential evidence, independently rerandomize every contributing source presentation. For each $j \in \mathcal{J}$, sample

$$\rho_j \leftarrow \mathbb{Z}_p^*, \quad t_{j,1}, t_{j,2}, t_{j,3} \leftarrow \mathbb{Z}_p,$$

and compute

$$\Gamma_j^\circ = \text{ReRand}_b(\Gamma_j; \rho_j, t_{j,1}, t_{j,2}, t_{j,3}).$$

The values Γ_j° are witness values of the aggregation proof and are not revealed.

Fix the padded source descriptions, selection, and rerandomized tuples using hiding and binding commitments before deriving aggregation coefficients. Let tr_{agg} be the public commitment prefix of Π_{agg} , containing these commitments and binding $(B_b, n, d_{M_{\text{agg}}}, C_W^*, \text{RegRoot}_b, \nu_{\text{agg}})$. Commitments and witness slots have a fixed padded shape. The remaining proof must establish that their openings are the same sources, selection, and rerandomized tuples used below. The fixed prefix determines nonzero coefficients

$$a_j \leftarrow \text{H}_{\text{coef}}(\text{tr}_{\text{agg}}, j) \in \mathbb{Z}_p^*, \quad j \in \mathcal{J},$$

using a domain-separated Fiat–Shamir challenge, with j the padded source-slot index. The prefix contains commitments to the hidden inputs, not the inputs themselves. Inactive slots do not contribute; each contributing source is included exactly once:

$$\bar{\Gamma}_r = \prod_{j \in \mathcal{J}} (\Gamma_{j,r}^\circ)^{a_j}, \quad r = 0, \dots, 4. \quad (7)$$

Finally, independently rerandomize the aggregate itself. Sample

$$\rho_{\text{agg}} \leftarrow \mathbb{Z}_p^*, \quad t_1^*, t_2^*, t_3^* \leftarrow \mathbb{Z}_p,$$

and set

$$\Gamma_{\text{agg}} = \text{ReRand}_b(\bar{\Gamma}; \rho_{\text{agg}}, t_1^*, t_2^*, t_3^*). \quad (8)$$

If the first component of $\bar{\Gamma}$ is the group identity, restart the aggregation with fresh rerandomization randomness. Only the final Γ_{agg} is exposed; neither the per-source Γ_j° nor $\bar{\Gamma}$ is published.

The public aggregation statement is

$$x_{\text{agg}} = (B_b, n, d_{M_{\text{agg}}}, C_W^*, \text{RegRoot}_b, \nu_{\text{agg}}, \Gamma_{\text{agg}}).$$

Generate Π_{agg} proving that:

1. every selected record is recovered from a valid source payment proof using the designated receiver secret;
2. $\mathcal{I}$ is applied correctly and M_{agg} is exactly the shuffled selected union;
3. all selected serials are distinct;
4. every source in $\mathcal{J}$ has $\Gamma_{j,0} \neq 1_{\mathbb{G}_1}$ and contributes its credential presentation exactly once, independently of the number of selected coins from that source;
5. every hidden Γ_j° is a correct rerandomization of the corresponding public source Γ_j according to Eq. (3), with $\rho_j \in \mathbb{Z}_p^*$;
6. the commitments in tr_{agg} open to the same padded source descriptions, selection, and rerandomized tuples; the coefficients a_j are the prescribed nonzero hashes of this fixed prefix, and $\bar{\Gamma}$ is formed according to Eq. (7);
7. $\bar{\Gamma}_0 \neq 1_{\mathbb{G}_1}$ and Γ_{agg} is formed by Eq. (8), with $\rho_{\text{agg}} \in \mathbb{Z}_p^*$;
8. C_W^* commits to the same registered secret $sk_{X,b}$, a contribution $u^* \in \mathbb{Z}_p$, and capacity n , in the reservation context, and $g^{sk_{X,b}}$ is a member of RegRoot_b . Uniform sampling of u^* is an honest-holder instruction, not a property asserted by this proof.

The proof performs no pairing evaluation. The source-payment partition, selection, receiver secrets, per-source rerandomization randomness, $\{\Gamma_j^\circ\}_{j \in \mathcal{J}}$, and the intermediate $\bar{\Gamma}$ remain hidden.

Return

$$\text{rcpt}_{\text{agg}} = (x_{\text{agg}}, M_{\text{agg}}, \Pi_{\text{agg}}).$$

Unselected coins remain available for later aggregation.

Settle_b. Settlement is interactive:

$$\langle X(\text{rcpt}, \text{policy}, [\text{id}_X]_{\text{redeem}}), B_b(\text{policy}, st_b) \rangle_{\text{Settle}_b} \rightarrow (\text{out}_X; st'_b),$$

where $\text{policy} \in \{\text{redeem}, \text{reissue}\}$. The anonymous reissue branch accepts

$$\text{rcpt}_{\text{agg}} = (x_{\text{agg}}, M_{\text{agg}}, \Pi_{\text{agg}}).$$

The issuer first checks the statement encoding, issuer identifier, current registration root, and reservation session, checks

$$H(M_{\text{agg}}) = d_{M_{\text{agg}}},$$

and verifies

$$\text{VerifyZK}_b(x_{\text{agg}}, \Pi_{\text{agg}}) = 1.$$

This establishes the source-recovery, selection, rerandomization, aggregation, registration, and preliminary-wallet relations, but does not establish PS credential authenticity.

The issuer then performs the separate native check

$$\text{PairCred}_b(\Gamma_{\text{agg}}) = 1. \tag{9}$$

Hence aggregate-reissue validity requires

$$\text{VerifyZK}_b(x_{\text{agg}}, \Pi_{\text{agg}}) = 1 \quad \wedge \quad \text{PairCred}_b(\Gamma_{\text{agg}}) = 1.$$

Because the credential relation is multiplicatively linear, Eq. (9) checks the weighted credential evidence of all contributing source payments using one native pairing-product relation, without evaluating source-credential pairings inside the recursive zero-knowledge proof.

After these cryptographic checks, the issuer processes every selected

$$C = (S, R, T) \in M_{\text{agg}}$$

against a tentative copy of its honored-record state. If no record is stored for S , classify C as fresh. If the exact pair (R, T) is already stored under S , classify it as a replay and abort without committing new state or authorizing additional value. If a different valid pair (R', T') is already stored under the same S , classify C as a conflict, invoke Identify_b on the validated records, and record the corresponding payer handle and replacement claim. A valid conflict remains admissible for the receiver's settlement value through the assumed funded recovery functionality. Obtain a one-use funded replacement authorization bound to this record and settlement reservation. Identification or an unfunded claim alone does not authorize value. Exact replays of previously honored conflicts are rejected by the same check.

Let n_{fresh} and n_{rec} count fresh records and admissible conflicts, respectively, so $n = n_{\text{fresh}} + n_{\text{rec}}$. If a required validation or funding authorization fails, abort without authorizing value; the functionality assumption guarantees funding for the valid non-replay payments of honest receivers. Otherwise atomically commit the new honored records, conflict claims, matching recovery-funding reservations or debits, and a settlement reservation authorizing exactly n units. The n_{rec} units are funded replacement value, not another fresh honoring of their serials. Under $\text{policy} = \text{reissue}$, use this reservation as aid, identified by the already bound reservation context ν_{agg} , and apply joint seed finalization with $C^{\text{pre}} = C_W^*$ and $N = n$; the accepted Π_{agg} supplies the preliminary opening and registration proof. The issuer records fresh r^* only now. The receiver forms $v^* = u^* + r^* \in \mathcal{V}$ and $C^{\text{fin}} = C_{W, \text{fin}}^*$ and proves R_{seed} . Only after accepting this proof and fixing the final request do the threshold authorities release blinded PS shares on $(sk_{X,b}, v^*, n)$. The receiver validates the responses, retains a threshold set of distinct valid shares, privately unblinds and combines them, verifies the resulting credential σ^* , and outputs

$$W_{X,b}^* = (\sigma^*, v^*, n, 0).$$

No authenticated receiver identity is disclosed in this reissue execution. An interrupted share-retrieval phase resumes from the existing reservation and its recovery authorizations, preserving the current seed attempt or the already finalized request, without issuing additional value or charging the recovery mechanism again. Seed retries and exhaustion leave the funded reservation recorded; they never make its source records spendable again. Retries are bounded, and exhaustion is the negligible honest failure specified in Appendix C.4.

Under `policy = redeem`, the receiver authenticates as id_X , and the issuer obtains its registered handle $P_{X,b} = g^{sk_{X,b}}$. The receiver additionally provides Π_{red} proving knowledge of $sk_{X,b}$ such that $P_{X,b} = g^{sk_{X,b}}$ and the same $sk_{X,b}$ opens the holder-secret component of C_W^* . The proof statement is bound to the current settlement session.

Identify_b. Given two issuer-validated conflicting records

$$C = (S, R, T), \quad C' = (S', R', T'),$$

require $S = S'$ and $R \neq R'$; otherwise return $\perp$. For a valid conflict, recover

$$P_{X,b} = g^{sk_{X,b}} = \left(\frac{TR'}{(T')^R} \right)^{(R'-R)^{-1}}.$$

The issuer resolves $P_{X,b}$ through its registration table to the corresponding registered payer. Bare triples alone do not constitute authenticated spend evidence; tracing is invoked only after the settlement checks above establish their validity.

F Security Analysis

This appendix proves Theorem 2. The experiment uses honestly generated public parameters and the issuer corruption bound permitted by the threshold-issuance assumption of Appendix C. Issuers execute the prescribed verification and serialized issuance and settlement state machines, including $\mathcal{F}_{\text{seed}}$ and the funded recovery functionality of Section 3.1. Its funding guarantee is assumed, not derived from the cryptographic reductions below. A PPT adversary may control arbitrary holders and exchange parties and may adaptively generate polynomially many payments and aggregate-settlement attempts. Any holder or receiver designated as honest in an exculpability or receiver-exclusivity challenge remains uncorrupted.

A *valid source receipt* is a source payment whose payment relation holds and whose credential presentation satisfies PairCred_b . Validity here is cryptographic and does not require that the source receipt have previously been settled.

Let Bad_{sec} be the union of the following events:

- (i) PubVf_b accepts a payment that is not bound to a valid, authorized wallet credential and in-range wallet positions;
- (ii) Settle_b accepts an anonymous reissue of capacity n that is not backed by exactly n distinct recovered records from valid source receipts, an exact previously honored record authorizes new value, or a conflict authorizes replacement value without a matching one-use funded recovery authorization;
- (iii) Identify_b identifies an honest registered holder who did not supply the repeated wallet position underlying the accusation;
- (iv) an accepted aggregate uses a source payment designated to an honest receiver although the adversary does not know that receiver's exchange secret; or

- (v) a redemption credits an authenticated holder whose registered handle is not bound to the holder secret committed in C_W^* ; or
- (vi) two validated records under one issuer have the same serial and distinct challenges, but Identify_b fails to return their common registered payer.

Define

$$\text{Adv}_{\Pi, \mathcal{A}}^{\text{SEC}}(\lambda) := \Pr[\text{Bad}_{\text{sec}}].$$

All primitive advantages below are taken with the polynomial query bounds of this experiment. Let ε_{ext} bound an accepted-proof extraction failure under Appendix C.2, and let q_{ext} and q_{agg} bound, respectively, the required extraction invocations and accepted aggregate-settlement attempts. These counts include extraction of completed final seed proofs. The separate two-continuation extraction loss is defined in Lemma 3. Let q_{seed} bound all issuer seed challenges, including those for abandoned, rejected, and reissued wallets; resuming the same recorded attempt is not a new challenge.

F.1 Aggregate-Credential Soundness

The aggregation proof establishes how the hidden source presentations are selected, rerandomized, and combined, but the final native pairing check is performed only on the aggregate presentation. We first show that this batching cannot hide an invalid contributing credential.

Lemma 1 (Aggregate-credential soundness). *Let an aggregate receipt pass both*

$$\text{VerifyZK}_b(x_{\text{agg}}, \Pi_{\text{agg}}) = 1 \quad \text{and} \quad \text{PairCred}_b(\Gamma_{\text{agg}}) = 1.$$

Under the honest-setup extraction interface of Appendix C.2, aggregation-commitment binding, and the coefficient random oracle of Appendix C.7, the probability that extraction fails or yields a contributing source j with $\text{PairCred}_b(\Gamma_j) = 0$ is at most

$$\varepsilon_{\text{batch}}(\lambda) := \varepsilon_{\text{ext}}(\lambda) + \frac{Q(\lambda)}{p-1} + \sqrt{Q(\lambda)\text{Adv}_{\text{Bind}}^{\text{agg}}(\lambda)} = \text{negl}(\lambda),$$

where Q polynomially bounds the distinct coefficient-oracle queries made by the verification/extraction computation.

Proof. For a presentation Γ , define its pairing residual

$$\mathcal{E}_b(\Gamma) := e(\Gamma_0, \tilde{\alpha}_b) \prod_{r=1}^3 e(\Gamma_r, \tilde{\beta}_{b,r}) e(\Gamma_4, \tilde{g}).$$

Successful extraction gives the committed active source set, the rerandomized presentations Γ_j° , and their openings. Since the relation enforces nonzero rerandomization exponents,

$$E_j := \mathcal{E}_b(\Gamma_j^\circ) = \mathcal{E}_b(\Gamma_j)^{\rho_j},$$

and

$$\mathcal{E}_b(\Gamma_{\text{agg}}) = \left(\prod_{j \in \mathcal{J}} E_j^{a_j} \right)^{\rho_{\text{agg}}}.$$

Hence aggregate acceptance implies

$$\prod_{j \in \mathcal{J}} E_j^{a_j} = 1_{\mathbb{G}_T}.$$

Assume some extracted source is invalid, so $E_j \neq 1_{\mathbb{G}_T}$. Fork the computation at the last first coefficient query corresponding to a nonidentity residual, keeping the setup, prefix, prior coefficient answers, and all non-coefficient oracles fixed. Bellare–Neven’s general forking lemma [6, Lemma 1] yields two accepting executions with a different answer to that query with probability at least

$$\alpha \left(\frac{\alpha}{Q} - \frac{1}{p-1} \right),$$

where α is the probability of acceptance, valid extraction, and at least one invalid residual.

The public aggregation prefix is identical in the two forks. If extraction opens one of its commitments differently, computational binding is violated. Otherwise the active set and all residuals coincide. Every other nonidentity residual has a coefficient fixed before the fork, so the two accepting equations imply

$$E_j^{a_j - a'_j} = 1_{\mathbb{G}_T},$$

contradicting $E_j \neq 1_{\mathbb{G}_T}$ and $a_j \neq a'_j$. Consequently

$$\alpha \leq \frac{Q}{p-1} + \sqrt{Q \text{Adv}_{\text{Bind}}^{\text{agg}}}.$$

Adding extraction failure gives the claimed bound.

F.2 Soundness of Accepted Objects

We next connect public and aggregate acceptance to the hidden wallet state from which the corresponding records originate.

Lemma 2 (Accepted-object soundness). *Except with probability*

$$\begin{aligned} \varepsilon_{\text{obj}}(\lambda) &:= q_{\text{ext}}(\lambda) \varepsilon_{\text{ext}}(\lambda) + q_{\text{agg}}(\lambda) \varepsilon_{\text{batch}}(\lambda) \\ &\quad + \text{Adv}_{\text{PS}}^{\text{uf}}(\lambda) + \text{Adv}_{\text{Bind}}^W(\lambda) + \text{Adv}_{\text{Mem}}(\lambda) + \text{Adv}_{\text{CR}}(\lambda), \end{aligned}$$

the following hold simultaneously for every accepted object in the security experiment:

(i) every payment accepted by PubVf_b has a witness

$$(\sigma, sk_X, v, N, \ell, \dots)$$

for its payment relation, where σ is an authorized PS credential on (sk_X, v, N) , g^{sk_X} is registered in RegRoot_b , and

$$0 \leq \ell \leq \ell + V \leq N;$$

(ii) every accepted anonymous reissue of capacity n contains exactly n pairwise-distinct recovered serial records from valid contributing source receipts, and its C_W^* is bound to a registered holder secret, a preliminary contribution u^* , and capacity exactly n . Its finalized request contains the same holder and capacity and the seed $v^* = u^* + r^* \in \mathcal{V}$ of its recorded attempt. Every authorized credential, including ordinary issuance, satisfies this same two-stage seed relation, and each authorization finalizes at most once.

Here $\text{Adv}_{\text{Bind}}^W$ denotes the total binding advantage for the preliminary and final issuance and output-wallet commitments, Adv_{Mem} denotes authenticated membership soundness, and Adv_{CR} covers collisions in the payload and aggregate digests.

Proof. Consider first an accepted payment. Honest-setup extraction of Π_{pay} gives the hidden credential, wallet attributes, position, and all values required by the payment relation. In particular, the relation binds the public Γ to the same (σ, sk_X, v, N) and proves the capacity constraint.

Since PubVf_b additionally checks

$$\text{PairCred}_b(\Gamma) = 1,$$

substituting the extracted relation for Γ yields the PS verification equation for σ on (sk_X, v, N) . A valid credential not originating from an authorized threshold issuance gives a threshold-PS forgery. By the issuance procedure, an authorized credential is issued only after the opening secret has been matched to the authenticated handle $P_{X,b}$ for ordinary issuance, or to the hidden registered holder authorized by the aggregate for anonymous reissue. Binding and the final seed proof preserve that secret in the signed request. Collision resistance binds the extracted payment records to the payload digest used by the public statement.

Now consider an accepted aggregate. Extraction of Π_{agg} gives the padded sources, selection, receiver openings, rerandomization witnesses, selected records, and the opening of C_W^* . The relation establishes that exactly n records are selected and that their serials are pairwise distinct.

For each contributing source proof, use the source-provenance rule of Appendix C.2: a copied proof retains the witness associated with its earlier accepted occurrence, while a fresh embedded source proof is extracted under its fixed verification key. Outer extraction alone is not used as a substitute for this source-proof extraction. Lemma 1 then guarantees that every contributing source presentation also satisfies its native pairing check.

Thus every selected record is recovered from a valid source payment. Finally, binding of C_W^* , authenticated membership, and the aggregate relation bind the preliminary reissue to the same registered secret and capacity n . Extracting R_{seed} binds the final seed to that preliminary opening and the recorded fresh issuer contribution. The same argument applies to ordinary issuance. Binding prevents a different preliminary opening after the challenge, and the one-use authorization state prevents a second finalized request. A union bound over the polynomially many extraction and settlement attempts gives ε_{obj} .

Lemma 3 (Separation of issued wallet positions). *Let $M = \binom{q_{\text{seed}}}{2}$ and $d = 2L_{\text{max}} - 1$. Under the joint seed-generation, binding, and completed-proof extraction requirements, except with probability $\varepsilon_{\text{obj}} + \varepsilon_{\text{seed}}$, no two distinct finalized wallet authorizations have overlapping sets $\{v + 1, \dots, v + N\} \subseteq \mathbb{Z}_p$, where*

$$\varepsilon_{\text{seed}}(\lambda) := M \sqrt{\frac{d}{p} + \text{Adv}_{\text{Bind}}^{\text{fork}}(\lambda) + \varepsilon_{\text{ext}}^{\text{fork}}(\lambda)}.$$

Here $\text{Adv}_{\text{Bind}}^{\text{fork}}$ bounds binding success of the reduction below, and $\varepsilon_{\text{ext}}^{\text{fork}}$ bounds failure of any of its at most $2q_{\text{seed}}$ completed-seed-proof extractions, with the reduction's polynomial resource bounds. Set $\varepsilon_{\text{seed}} = 0$ if $M = 0$.

Proof. Order all seed challenges, including rejected and abandoned attempts. Guess a pair of indices $i < j$ uniformly among the M possible pairs; a nonexistent attempt or two attempts of the same authorization cannot win. Fork the classical execution immediately before sampling r_j . Both preliminary commitments and r_i are already fixed, even if the earlier wallet has not yet finalized. Run two continuations with independent $r_j, r'_j \leftarrow \$ \mathbb{Z}_p$ and fresh subsequent coins. Preserve the prefix, commitment parameters, and consistent random-oracle functions; only the issuer functionality's challenge at the fork and subsequent fresh coins are resampled. After each continuation, extract the completed R_{seed} proofs for its finalized requests, and inspect the guessed pair in the resulting table. No extraction occurs before the fork.

Let ϵ be the probability of some cross-authorization overlap in this table of valid extracted seed witnesses, and let α be the conditional probability that the guessed pair exhibits such an overlap in one continuation. Conditioning also on the fixed oracle functions, the two continuations are independent. A union bound over pairs and Jensen's inequality give

$$\Pr[\text{both continuations exhibit the guessed overlap}] = \mathbb{E}[\alpha^2] \geq (\mathbb{E}[\alpha])^2 \geq (\epsilon/M)^2.$$

If a preliminary commitment has different extracted attribute tuples in the two continuations, output its two openings to violate binding. Otherwise the same u_i, u_j occur in both. Fixing the first continuation, an overlap in the second requires

$$r'_j \in u_i + r_i - u_j + \{-(L_{\text{max}} - 1), \dots, L_{\text{max}} - 1\},$$

a set of at most d field values. Since r'_j is fresh and independent of the first continuation, this event has probability at most d/p . Thus

$$(\epsilon/M)^2 \leq d/p + \text{Adv}_{\text{Bind}}^{\text{fork}} + \epsilon_{\text{ext}}^{\text{fork}},$$

which gives the stated bound. Binding reductions use the externally supplied commitment parameters without their trapdoor. Failures of accepted-object soundness in the original execution remain covered by ϵ_{obj} ; the fork terms cover the additional reduction runs. All attempts are counted before validity filtering or selective abort, so concurrent finalization and grinding do not evade the bound. The square-root and polynomial pair-guessing losses preserve negligibility. This proves separation, not uniformity of maliciously selected finalized seeds.

F.3 Proof of Exchange-Compatible E-Cash Security

Proof (Proof of Theorem 2). We first establish correctness. Honest threshold issuance produces a valid PS credential on (sk_X, v, N) . Proof-system completeness therefore makes honest payment and aggregation proofs verify, while rerandomization and aggregation of valid credential presentations preserve the native pairing relation. For an honest receiver's aggregate containing distinct valid non-replay records, settlement accepts and authorizes the selected capacity: fresh records are honored once, while conflicts receive funded replacement value under Section 3.1. The only additional honest-execution failures are the negligible retry-exhaustion and accidental serial-collision events covered by the polynomial bounds and freshness conditions of Appendix C. In particular, across q_{auth} honest authorizations, seed-retry exhaustion costs at most $q_{\text{auth}}(L_{\text{max}}/p)^{K_{\text{seed}}}$.

For accountability, exclude the accepted-object failures and cross-wallet overlaps of Lemma 3, at total cost at most $\epsilon_{\text{obj}} + \epsilon_{\text{seed}}$. All remaining arguments are on this good event.

Payment validity and capacity. Every accepted payment is now associated with an authorized credential on (sk_X, v, N) and with positions satisfying

$$0 \leq \ell \leq \ell + V \leq N.$$

Thus a credential can generate at most its authenticated set of wallet positions without reusing a position. Any adversarial payment outside this set would contradict extraction, authenticated membership, commitment binding, or threshold-PS unforgeability, all already charged in ϵ_{obj} .

Aggregate value accounting. For every successful anonymous reissue of capacity n , the aggregate relation and Lemma 2 give exactly n distinct recovered records from valid source receipts. The issuer evaluates those records against a serialized tentative copy of its honored-record state before authorizing reissue.

If no record with serial S has previously been honored, the record is committed as fresh. If the exact pair (R, T) is already stored under S , the receipt is a

replay and settlement aborts without authorizing additional value. A different validated pair under the same serial is processed only through the explicit conflict branch and is recorded as such. Any value authorized for such a valid conflict is replacement value provided by the assumed issuer-side settlement functionality of Section 3.1 and funded through its underlying double-spending recovery mechanism; it is not counted as a second fresh honoring of the same coin. Identification alone supplies no funds.

Partition the accepted batch into n_{fresh} first presentations and n_{rec} funded conflicts. Serialization and the atomic commit give

$$n = n_{\text{fresh}} + n_{\text{rec}}.$$

Exactly n_{rec} units of replacement value are matched by recovery funding reserved or debited. Every serial contributes fresh value at most once; every honored exact record, including a funded conflict, is stored and cannot authorize value again. Recovery authorizations are bound to their records and settlement reservation and cannot be reused. The issuer releases shares for capacity exactly n only after committing these records and matching funding charges. This accounting is conditional on the assumed funded recovery functionality; the exchange construction neither supplies its funds nor proves its solvency.

Double-spender identification. Consider any two validated records with the same serial under one issuer. Since δ has order p , equality of serials implies equality of the nonzero denominators $v + \ell + 1$. Lemma 3 rules out distinct wallet authorizations. Within one wallet the in-range positions are distinct modulo p , so the records use the same position and the same registered secret. They therefore share

$$\mu = (v + \ell + 1)^{-1}$$

and therefore

$$S = \delta^\mu, \quad T = g^{sk_X + R\mu}, \quad T' = g^{sk_X + R'\mu}.$$

For $R \neq R'$,

$$\frac{T^{R'}}{(T')^R} = g^{sk_X(R' - R)},$$

and hence

$$\left(\frac{T^{R'}}{(T')^R} \right)^{(R' - R)^{-1}} = g^{sk_X} = P_{X,b}.$$

Thus `Identifyb` returns the registered payer. If the repeated spend instead has the same challenge, then the same wallet position gives the same (S, R, T) and is an exact replay, which cannot authorize additional value.

Exculpability. It remains to rule out a same-serial conflict that falsely identifies an honest holder. On the good event fixed above, every record presented to Identify_b has an extracted compact-wallet witness and satisfies exactly the core serial/tag equations

$$S = \delta^{(v+\ell+1)^{-1}}, \quad T = g^{sk_X + R(v+\ell+1)^{-1}}.$$

Receiver masking, public verification, and AAR do not alter these core values before identification; the additional attribute N only restricts the permitted positions.

The separation lemma excludes conflicts assembled from distinct wallet authorizations with adversarially chosen related seeds. Thus the remaining exculpability case concerns the same authorized holder and wallet position. Every record supplied to Identify_b can be reduced, after extraction and removal of the exchange-layer masking, to the same (S, R, T) algebraic form used by the underlying compact e-cash tracing argument. Public verification and AAR introduce no additional degree of freedom in S or T and therefore do not create a new way to form conflicting tracing evidence outside these equations.

The DL-based exculpability argument of the compact construction of Rial-Piotrowska [22] therefore applies to these extracted records. In particular, a malicious record that collides with an honest user's serial without using that user's coin witness gives the honest-serial collision prohibited by the compact-wallet DL reduction. If both conflicting records are adversarially generated, their extracted wallet secret is the same on the good event. Identification of an honest handle would therefore reveal its discrete logarithm. The distinct-wallet, different-secret case is already charged to $\varepsilon_{\text{seed}}$ and is not justified by this DL argument. We denote the total loss of these compact-record DL reductions by $\text{Adv}_{\text{DL}}^{\mathbb{G}_1}(\lambda)$.

Receiver exclusivity. Finally, suppose an adversary causes a source payment addressed to an honest receiver key

$$Y_{\text{rcv}} = h^{\xi_{\text{rcv}}}$$

to contribute to an accepted aggregate without knowing ξ_{rcv} . Extraction of Π_{agg} yields a witness ξ satisfying

$$Y_{\text{rcv}} = h^{\xi},$$

and the relation uses this same witness to derive the DH/KDF value that recovers the selected source records. Thus the extractor outputs

$$\xi = \log_h Y_{\text{rcv}},$$

breaking discrete-logarithm hardness in $\mathbb{H}$. The probability of this event is therefore at most $\text{Adv}_{\text{DL}}^{\mathbb{H}}(\lambda)$. For redemption, an accepting redemption proof binds the authenticated holder's registered handle $P_{X,b}$ to the same $sk_{X,b}$ committed in C_W^* . Hence an adversary redirecting a copied aggregate receipt to a different registered identity would violate either the knowledge soundness of the redemption proof or the binding of C_W^* , except with negligible probability.

Combining the bad events gives

$$\begin{aligned}\text{Adv}_{\Pi, \mathcal{A}}^{\text{SEC}}(\lambda) &\leq \varepsilon_{\text{obj}}(\lambda) + \varepsilon_{\text{seed}}(\lambda) + \text{Adv}_{\text{DL}}^{\text{G}_1}(\lambda) + \text{Adv}_{\text{DL}}^{\text{H}}(\lambda) \\ &= \text{negl}(\lambda).\end{aligned}$$

This proves payment validity and capacity enforcement, aggregate value accounting, double-spender identification, exculpability, and receiver-exclusive settlement.

G Privacy Proofs

The following proofs are conditional on the applicable setup, privacy, honest correctness, and freshness requirements of Appendix C. Each primitive advantage and statistical loss uses the query and record bounds of its own experiment. The use of Appendix C.4 concerns perfect correctness of the blind-signature subprotocol and the stated joint seed-generation procedure, including its negligible retry-exhaustion probability; passive privacy still permits exposure of all issuer keys. Concrete instantiations must be shown to satisfy the requirements invoked below. All issuance and settlement views include the preliminary and final requests, seed challenges and attempt numbers, seed-consistency proofs, and retry or exhaustion outcomes. For each experiment let $q_{\text{seed}}^{\text{priv}}$ bound the seed attempts in protected source-wallet issuance and define

$$\varepsilon_{\text{dom}}(\lambda) := q_{\text{seed}}^{\text{priv}}(\lambda)L_{\text{max}}/p.$$

This bounds the probability of any invalid candidate seed in those honest executions. It is charged once when removing source-seed dependence; output reissuance retries are retained and coupled across branches.

G.1 Swap–Settlement Unlinkability

Fix admissible challenges $\chi_j = (b, X, S_j, \mathcal{I}_j)$, $j \in \{0, 1\}$, with common selected value n . Let $\mathbf{H}$ contain the common history through both source executions: public parameters, both issuers' states, keys and coins, registration and issuance transcripts, both public swap transcripts (including $(x_{\text{pay},j}, M_{\text{pay},j}, \Pi_{\text{pay},j})$), and pre-challenge query answers. The two adversarial views are

$$\text{View}^{(0)} = (\mathbf{H}, D_0, \mathbf{Q}_0), \quad \text{View}^{(1)} = (\mathbf{H}, D_1, \mathbf{Q}_1),$$

where the branch-dependent settlement data are

$$\begin{aligned}D_j &= (B_b, n, \mathcal{C}_j, H(\mathcal{C}_j), C_{W,j}^*, \text{RegRoot}_b, \nu_{\text{agg},j}, \\ &\quad \Gamma_{\text{agg},j}, \Pi_{\text{agg},j}, \tau_j, st_0^{(j)}, st_1^{(j)}), \\ \mathcal{C}_j &= \text{Shuffle}((S_{j,k}, R_{j,k}, T_{j,k})_{k \in \mathcal{I}_j}).\end{aligned}$$

Here $\mathcal{C}_j$ contains the unmasked records of Eq. (4); $C_{W,j}^*$ is the preliminary output-wallet commitment, $\Pi_{\text{agg},j}$ includes its public commitment prefix, τ_j contains the

issuer-visible settlement and reissue messages, coins and outcomes, and $st_0^{(j)}, st_1^{(j)}$ are the resulting issuer states. Q_j denotes the complete subsequent ordinary-query view. Thus $\beta = j$ selects the records from $\mathcal{I}_j$ for settlement; both source swaps appear in either view. On failure, unproduced fields are absent and the actual failure and resulting state remain in the view.

Our goal is to prove that $\text{View}^{(0)} \approx_c \text{View}^{(1)}$. Define G_0 as the real SSU experiment of Appendix B.3, and write $\text{View}_i^{(\beta)}$ for the complete adversarial view in game G_i when the challenge bit is β . We will transform G_0 through polynomially many games $G_1, \dots, G_t$, bounding the distinguishing advantage of each transition by a negligible function, until the two final views have identical distributions:

$$\text{View}_t^{(0)} \stackrel{d}{=} \text{View}_t^{(1)}.$$

The final view is therefore independent of β ; its components need not all be uniformly random. Transferring this indistinguishability back through the games establishes the desired negligible SSU advantage in G_0 .

Game G_1 . First cap every ordinary-query aggregation at $L \geq 1$ polynomially many attempts, returning failure on exhaustion. Let Q_{agg} bound these calls and Q_{coef} bound all honest and adversarial coefficient queries in capped executions. By Appendix C.7 and the retry argument in Appendix G.2, the truncation loss is at most $\varepsilon_{\text{tr}} = Q_{\text{agg}}\eta^L$, with η evaluated using these SSU query bounds. All subsequent SSU games retain the caps; they are proof modifications. The single-source challenge cannot cancel. Next, replace the holder-generated ZK proofs in challenge-wallet issuance (Issue_b : preliminary opening and equality to the authenticated handle, and final R_{seed} consistency and seed/capacity validity), the public payment proofs Π_{pay} of both candidate swaps, the challenge aggregation proof, and the reissue Π_{seed} verified by Settle_b with simulated proofs on their public inputs. Use the simulation setup where required, with its trapdoors outside the issuer views; ordinary-query and issuer-generated proofs remain honestly generated. For aggregation, retain the honestly generated commitment prefix tr_{agg} and simulate only the remaining ZK proof, taking $(x_{\text{agg}}, \text{tr}_{\text{agg}})$ as public input. Simulate source proofs before constructing the aggregation witness, and retain actual verification outcomes. The simulators require no private witnesses: the proofs disclose no additional information beyond their public inputs, up to computational indistinguishability. By the adaptive multi-theorem zero-knowledge requirement of Appendix C.2, with the issuers' complete views as auxiliary information, proof replacement costs at most $\text{Adv}_{\text{ZK}}(\lambda)$. Combining this with truncation and writing $G_i^A(1^\lambda) = 1$ for the game's winning event, we obtain

$$|\Pr[G_1^A(1^\lambda) = 1] - \Pr[G_0^A(1^\lambda) = 1]| \leq \text{Adv}_{\text{ZK}}(\lambda) + \varepsilon_{\text{tr}}(\lambda),$$

where Adv_{ZK} includes the joint loss for simulation setup and all replaced proofs, including any simulation-induced verification failures.

Game G_2 . Replace every valid public payment presentation Γ in both candidate swaps and the final settlement presentation Γ_{agg} by independent tuples

$(A_0, \dots, A_4)$, sampled for their respective issuer $B_{b'}$, $b' \in \{0, 1\}$, as

$$A_0 \leftarrow \mathbb{G}_1 \setminus \{1_{\mathbb{G}_1}\}, \quad A_1, A_2, A_3 \leftarrow \mathbb{G}_1, \quad A_4 = A_0^{-x_{b'}} \prod_{r=1}^3 A_r^{-y_{b',r}}.$$

Under the group setup of Appendix C.1, for a valid credential with $h_\sigma \neq 1_{\mathbb{G}_1}$, Eq. (2) has exactly this distribution: r_σ makes the first component uniform non-identity, the independent u_r make the next three components uniform, and the pairing relation uniquely determines the last component. Likewise, Eq. (3) gives this same distribution for every valid input tuple with $\Gamma_0 \neq 1_{\mathbb{G}_1}$, independently of that tuple and even given the issuer's signing keys. In SSU there is only one contributing source, whose nonzero coefficient and rerandomization exponent ensure $\bar{\Gamma}_0 \neq 1_{\mathbb{G}_1}$; hence the argument also applies to the fresh final rerandomization producing Γ_{agg} . Update dependent commitments and the simulated proofs from G_1 consistently, retaining all checks and failures. The resulting joint view has exactly the same distribution, so

$$|\Pr[G_2^A(1^\lambda) = 1] - \Pr[G_1^A(1^\lambda) = 1]| = 0.$$

Game G_3 . The commitments considered here reside in the public prefix tr_{agg} of the challenge proof Π_{agg} and commit to the padded source descriptions, coin selection, and privately rerandomized tuples Γ_j° . Replace them with fresh commitments to fixed dummy messages of the same padded lengths, preserving the prefix's public statement bindings, including C_W^* . Recompute $a_j = \text{H}_{\text{coef}}(\text{tr}_{\text{agg}}, j)$ and continue using the same total fixed-prefix simulator from G_1 on the updated public inputs, as required in Appendix C.2. Since G_2 already samples Γ_{agg} independently of these hidden openings and coefficients, the remaining affected computations, including verification outcomes, are efficient postprocessing of the commitments. For either fixed β , adaptive joint commitment hiding from Appendix C.3, with the issuers' complete views and the proof-simulation setup state as auxiliary information, makes this replacement computationally indistinguishable. Hence

$$|\Pr[G_3^A(1^\lambda) = 1] - \Pr[G_2^A(1^\lambda) = 1]| \leq \text{Adv}_{\text{Hid}}^{\text{agg}}(\lambda),$$

where $\text{Adv}_{\text{Hid}}^{\text{agg}}$ includes the loss for all replaced aggregation-prefix commitments.

Game G_4 . For every protected source-wallet issuance attempt, replace both the preliminary request to $(sk_{X,b}, u, N)$ and the final request to $(sk_{X,b}, v, N)$ by independently blinded requests to $(sk_{X,b}, 0, N)$. Retain the actual holder and capacity. The preliminary request still precedes the issuer's fresh challenge r ; the final request follows it. Simulate both proofs as in G_1 , including the seed-consistency proof, and run issuer responses and state updates with their actual keys and coins. Retain verification outcomes. Joint hiding applies adaptively to both stages and their intervening public challenge; the hidden relation $v = u + r$ is not asserted for the dummy openings. Simulation on these modified inputs is

efficient under Appendix C.2; the hiding comparison, rather than a ZK claim for false statements, bounds any resulting acceptance gap.

Keep the genuine candidate seed internally during these commitment replacements. The probability of any protected source attempt with $u + r \notin \mathcal{V}$ is at most ε_{dom} . Outside this event the first attempt suffices. After both requests and proofs have been replaced, its remaining private seed may equivalently be sampled as $v \leftarrow \$ \mathcal{V}$ independently of r : for uniform honest u , the pair $(u + r, r)$ conditioned on $u + r \in \mathcal{V}$ has exactly this law. Thus define the endpoint to use one attempt and such an independent private seed, at statistical cost at most ε_{dom} . Use actual issuer keys to generate the valid private credential on $(sk_{X,b}, v, N)$ needed for subsequent wallet checks. It is unexposed, and public presentations are already independent by G_2 . Signature responses to the final dummy request can be validated and unblinded using its matching opening; $0 \in \mathcal{V}$ and the blind-signature subprotocol is perfectly correct. Consequently, for either fixed β ,

$$|\Pr[G_4^A(1^\lambda) = 1] - \Pr[G_3^A(1^\lambda) = 1]| \leq \text{Adv}_{\text{Hid}}^{\text{iss}}(\lambda) + \varepsilon_{\text{dom}}(\lambda).$$

The hiding term covers both request stages jointly. The endpoint's public seed-generation transcript is now efficiently generated without the private source seed, as required by the later PRF hop.

Game G_5 . The offsets and mask exponents in each source payment are derived from $K = H_{\text{kdf}}(Z, c)$, where $Z = Y_{\text{rcv}}^{\xi_{\text{snd}}} = Y_{\text{snd}}^{\xi_{\text{rcv}}}$. For every active coin on both legs of both candidate swaps, replace

$$(d_k, x_k^S, x_k^T) = (H_d(K, c, k), H_S(K, c, k), H_T(K, c, k))$$

by three independent uniform elements of $\mathbb{Z}_p$. Store one sampled triple per distinct (Z, c, k) in a private table, reused only by the honest source and recovery computations. Public hash evaluations and ordinary-query derivations remain real; the table is not an adversarial oracle. Keep the public exchange keys and q_k unchanged; recompute $R_k = q_k + d_k \pmod{p}$, T_k , the masked records, and dependent public inputs. Continue using the proof simulators from G_1 , retaining actual checks, state updates, and failures. For either fixed β , joint DH/KDF-output pseudorandomness from Appendix C.5 makes this replacement indistinguishable, including LP key reuse across distinct leg contexts, issuer and simulation-setup auxiliary information, permitted ordinary-query exposures, and public-hash queries. Thus

$$|\Pr[G_5^A(1^\lambda) = 1] - \Pr[G_4^A(1^\lambda) = 1]| \leq \text{Adv}_{\text{DH/KDF}}(\lambda),$$

where $\text{Adv}_{\text{DH/KDF}}$ covers all replaced outputs. Protected exchange keys and contexts satisfy that assumption's exposure restrictions by challenge admission and the separation of ordinary-query objects. The uniform d_k hides the association between R_k and q_k , while the independent x_k^S, x_k^T make each masked pair uniform independently of its underlying record.

Remaining view. After G_5 , the targeted proofs are simulated, source commitments use dummy data, and credential presentations and payment masks are randomized. The bit β still selects the settlement's n unmasked records. For position ℓ of source wallet w , with seed v_w and payer public key P_w , they have the form

$$(S, R, T) = (\delta^{\mu_{w,\ell}}, R, P_w g^{R\mu_{w,\ell}}), \quad \mu_{w,\ell} = (v_w + \ell + 1)^{-1}.$$

Although R is uniform at generation, S and T remain correlated. Both public swaps, issuer views, reissue messages, actual checks and state updates, and subsequent ordinary-query answers remain visible.

Game G_6 . For every fresh protected source wallet w in either candidate execution and every used position $0 \leq \ell < L_{\max}$, replace the group-output pair

$$F_{v_w}(\ell) = (\delta^{\mu_{w,\ell}}, g^{\mu_{w,\ell}}) \quad \text{by} \quad (\delta^{z_{w,\ell}}, g^{z_{w,\ell}}), \quad z_{w,\ell} \leftarrow \mathbb{Z}_p.$$

Sample independently for distinct wallet–position pairs and reuse the same value at every repeated occurrence. Set $S = \delta^{z_{w,\ell}}$ and $T = P_w g^{Rz_{w,\ell}}$ throughout both candidate swaps and settlement, keeping the offsets and masks from G_5 . Update dependent public inputs and simulated proofs, and run all verification, replay, conflict, and ordinary-query checks with their actual outcomes. The scalars $z_{w,\ell}$ remain private. By G_1 – G_4 , other visible data can be generated without v_w ; Appendix C.4's perfect blind-signature correctness permits omitting the unexposed private-credential computations in the reduction. Under the joint pseudorandomness requirement of Appendix C.6 for $v_w \leftarrow \mathcal{V}$, the remaining view is efficient postprocessing of its outputs. Hence, for either fixed β ,

$$|\Pr[G_6^A(1^\lambda) = 1] - \Pr[G_5^A(1^\lambda) = 1]| \leq \text{Adv}_{\text{PRF}}(\lambda),$$

where Adv_{PRF} covers all challenge wallets and positions, the remaining auxiliary information, and the valid-seed restriction; no additional per-record or seed-conditioning term is charged.

Game G_7 . Since $R = 0$ makes $T = P_w$, modify offset generation for every active coin on both legs of both candidate swaps: sample $R_k \leftarrow \mathbb{Z}_p^*$ and set $d_k = R_k - q_k \pmod{p}$. Sample once per distinct (Z, c, k) input and reuse the result as in G_5 , retaining independent mask exponents. Recompute tags, masked records, dependent public inputs, and simulated proofs; all protocol checks and failures retain their actual outcomes. Let $N_{\text{rec}}(\lambda)$ be a fixed polynomial upper bound on the number of distinct source-record inputs generated in any execution of the two candidate swaps. Couple G_6 and G_7 identically until G_6 first samples $R_k = 0$. Each fresh R_k in G_6 is uniform given the preceding execution, so a union bound gives, for either fixed β ,

$$|\Pr[G_7^A(1^\lambda) = 1] - \Pr[G_6^A(1^\lambda) = 1]| \leq \frac{N_{\text{rec}}(\lambda)}{p}.$$

This statistical loss is negligible under the polynomial record bound and group-order requirement of Appendix C.7.

Game G_8 . For each used wallet–position pair (w, ℓ) in both candidate executions, replace the shared-exponent pair $(\delta^{z_{w,\ell}}, g^{z_{w,\ell}})$ with independent $U_{w,\ell}, V_{w,\ell} \leftarrow \mathbb{G}_1$, and set

$$S = U_{w,\ell}, \quad T = P_w V_{w,\ell}^R.$$

Reuse the same pair at every repeated occurrence, keep the offsets and masks from G_7 , and update dependent public inputs and simulated proofs. All verification, settlement, and ordinary-query checks retain their actual outcomes. Under XDH and the programmable serial-base setup of Appendix C.1, use a hybrid over wallet–position pairs [22]. Embed a DDH tuple $(g, A = g^a, B = g^u, C)$ with $a \in \mathbb{Z}_p^*$ by setting $\delta = A$ at the serial hash-to-group label and the target pair to (C, B) : $C = g^{au}$ gives the distribution in G_7 , whereas uniform $C \leftarrow \mathbb{G}_1$ gives the replacement in G_8 . Hence, for either fixed β ,

$$|\Pr[G_8^A(1^\lambda) = 1] - \Pr[G_7^A(1^\lambda) = 1]| \leq \text{Adv}_{\text{XDH}}(\lambda),$$

where Adv_{XDH} includes the loss from all wallet–position hybrids. Since $R \neq 0$, the map $V \mapsto P_w V^R$ is a bijection on $\mathbb{G}_1$. Thus, at generation, each distinct selected position gives independent uniform serial and tag values, even for fixed P_w and R .

Final view. Use the fresh protected challenge-wallet separation of the preceding games. Honest spending advances wallet positions, and an admissible selection contains n distinct positions. In G_8 , conditional on the common source history and pre-challenge ordinary-query state, their records are independent samples from $\mathbb{G}_1 \times \mathbb{Z}_p^* \times \mathbb{G}_1$ before the duplicate-serial check and shuffle. The private uniform masks hide their association with either public swap. Couple the selected records, output contribution, all issuer seed challenges, seed-retry decisions, preliminary and final output-request blinding, and reservation randomness, and aggregation presentation, dummy-prefix, and simulated-proof randomness across the two branches. Since b, X, n , the registration root, and the pre-settlement issuer state coincide, all verification, duplicate, replay, conflict, and reissue outcomes and state updates coincide as well. Source failures are independent of β , since both source executions precede selection. Subsequent ordinary queries use the coupled states and independent query wallets; challenge objects and their descendants cannot be queried through these oracles. Thus the complete views, including failures and future answers, with the same retained caps, satisfy

$$\text{View}_8^{(0)} \stackrel{d}{=} \text{View}_8^{(1)}, \quad \Pr[G_8^A(1^\lambda) = 1] = \frac{1}{2}.$$

The initial challenge-rejection case already returns an independent fair score.

Combining the bounds. Define the total hiding loss by

$$\text{Adv}_{\text{Hid}}(\lambda) := \text{Adv}_{\text{Hid}}^{\text{agg}}(\lambda) + \text{Adv}_{\text{Hid}}^{\text{iss}}(\lambda).$$

Writing $p_i = \Pr[G_i^A(1^\lambda) = 1]$, the triangle inequality over the eight transitions gives

$$\begin{aligned} \text{Adv}_{H,A}^{\text{SSU}}(\lambda) &= |p_0 - p_8| \leq \sum_{i=1}^8 |p_i - p_{i-1}| \\ &\leq \text{Adv}_{\text{ZK}}(\lambda) + \text{Adv}_{\text{Hid}}(\lambda) + \text{Adv}_{\text{DH/KDF}}(\lambda) \\ &\quad + \text{Adv}_{\text{PRF}}(\lambda) + \text{Adv}_{\text{XDH}}(\lambda) + \frac{N_{\text{rec}}(\lambda)}{p} + \varepsilon_{\text{tr}}(\lambda) + \varepsilon_{\text{dom}}(\lambda). \end{aligned}$$

Here $G_1 \rightarrow G_2$ contributes zero. Each primitive advantage includes all replacements in its hop, and the two hiding losses are counted once each as specified in Appendix C.3. Truncation is charged once in $G_0 \rightarrow G_1$, and ε_{dom} once in $G_3 \rightarrow G_4$; the single-source challenge adds no cancellation loss. Under the applicable requirements of Appendix C invoked above, including the growth and freshness conditions of Appendix C.7, the displayed sum is negligible. The per-hop view-indistinguishability arguments therefore make G_0 and G_8 computationally indistinguishable with advantage at most $\text{negl}(\lambda)$. Since the final view is independent of β , this establishes SSU under these requirements.

G.2 Aggregate-Source Unlinkability

Fix admissible ASU challenges $\chi_j = (b, X, \mathcal{S}, \mathcal{I}_j)$, $j \in \{0, 1\}$, over the common source family $\mathcal{S} = (S_1, \dots, S_m)$, with $V(\mathcal{I}_0) = V(\mathcal{I}_1) = n$. Let H be the common history defined in Appendix G.1, extended to all m source executions. For $\beta = j$, the adversarial view is

$$\begin{aligned} \text{View}_{\text{ASU}}^{(j)} &= (H, D_j, Q_j), \\ \mathcal{C}_j &= \text{Shuffle}((S_{i,k}, R_{i,k}, T_{i,k})_{(i,k) \in \mathcal{I}_j}), \end{aligned}$$

where D_j and Q_j contain the same settlement fields and subsequent ordinary-query view as in SSU, using this selected record batch $\mathcal{C}_j$. The same source family is executed in both branches; β selects which $\mathcal{I}_j$ is used for aggregation. SSU selects from a single source payment in each branch, while the ASU selections may overlap and involve different numbers of contributing payments, with the same issuer, receiver, and total value. On failure, unproduced fields are absent and actual outcomes and resulting states remain in the view.

Game G_0 . Let G_0 be the real ASU experiment of Appendix B.4. Our goal is to transform it into a game whose complete adversarial view is independent of β , bounding the change in winning probability at each transition.

Game G_1 . Apply the proof, presentation, request, and group-output transformations of SSU games $G_1, G_2, G_4, \dots, G_8$ in Appendix G.1, in that order as internal hybrids, to all fresh protected challenge wallets and both payment legs of all m sources. Account for truncation locally below, without importing

SSU's cap transition. Thus proofs and issuance requests are simulated, public presentations are sampled from the issuer-only distribution, and the DH/KDF outputs and serial/tag pairs are replaced as there, with nonzero R and consistent reuse at repeated inputs. Retain the real aggregation-prefix commitments, coefficient derivation, identity checks, and retries; sample the independent final presentation only after obtaining $\bar{T}_0 \neq 1_{G_1}$. This final sampling is exact even conditioned on the successful retry transcript. The retained prefix uses source receipts, the selection, and rerandomized public presentations, all constructible from the replacement outputs without the hidden wallet seeds. Update dependent inputs and simulated proofs, retaining actual checks, failures, states, and query answers. Under the Appendix C requirements cited in those SSU transitions, the triangle inequality over these internal hybrids, evaluated with the polynomial retry cutoff justified below using Appendix C.7, gives

$$\begin{aligned} & |\Pr[G_1^A(1^\lambda) = 1] - \Pr[G_0^A(1^\lambda) = 1]| \\ & \leq \text{Adv}_{\text{ZK}}(\lambda) + \text{Adv}_{\text{Hid}}^{\text{iss}}(\lambda) + \text{Adv}_{\text{DH/KDF}}(\lambda) \\ & \quad + \text{Adv}_{\text{PRF}}(\lambda) + \text{Adv}_{\text{XDH}}(\lambda) + \frac{N_{\text{rec}}^{\text{ASU}}(\lambda)}{p} + 2\varepsilon_{\text{tr}}(\lambda) + \varepsilon_{\text{dom}}(\lambda). \end{aligned}$$

Here each primitive advantage covers all replacements across the common source family, and $N_{\text{rec}}^{\text{ASU}}$ is a fixed polynomial bound on its distinct source-record inputs. The source-seed domain term ε_{dom} uses this experiment's protected issuance counts and is inherited once from SSU G_4 . The endpoint truncation loss $2\varepsilon_{\text{tr}}$ is bounded in G_2 below. Aggregation-prefix hiding is deferred and contributes no term yet.

Remaining view. Distinct fresh records now have the same distribution $\mathbb{G}_1 \times \mathbb{Z}_p^* \times \mathbb{G}_1$ at generation, before checks. The retained aggregation prefix and its retry condition can still depend on the selection. The next hops must bound cancellation and retries, justify efficient aggregation, and replace the padded prefix commitments with dummies at cost $\text{Adv}_{\text{Hid}}^{\text{agg}}(\lambda)$. The final argument must then couple the complete views for overlapping selections and different source counts, including failures, state updates, and subsequent queries.

Game G_2 . Keep the first challenge-aggregation prefix and its coefficients, but omit the test $\bar{T}_0 = 1_{G_1}$ and its retries. Sample the final presentation independently as in G_1 and use the same proof-tail simulator. Limit each ordinary-query aggregation to a polynomial number $L = L(\lambda)$ of attempts, returning failure on exhaustion; retain all other checks and outcomes. By Appendix C.7, H_{coef} is a domain-separated random oracle into $\mathbb{Z}_p^*$, and each honestly generated prefix has pointwise conditional min-entropy at least κ given the prior history and fixed committed messages, in the real game and the relevant internal hybrids. Let Q_{coef} bound all honest and adversarial coefficient queries in the capped executions. A previously queried prefix occurs with probability at most $Q_{\text{coef}}2^{-\kappa}$. Otherwise, the coefficients are independent uniform nonzero scalars: fixing all but one leaves at most one cancelling choice, since each contributing $\Gamma_{j,0}^\circ$ is nonidentity. For one

contributing source, cancellation is impossible. Comparing first with the capped version of G_1 , then coupling until the first challenge cancellation, gives

$$\eta(\lambda) := \frac{1}{p-1} + Q_{\text{coef}}(\lambda)2^{-\kappa(\lambda)},$$

$$|\Pr[G_2^A(1^\lambda) = 1] - \Pr[G_1^A(1^\lambda) = 1]| \leq \eta(\lambda) + \varepsilon_{\text{tr}}(\lambda).$$

On cancellation, the simulator may receive a false statement; this entire event is charged above, without invoking zero knowledge for false statements. Fresh blinding on every retry gives the same conditional bound, so the probability of exhausting L attempts in any of at most Q_{agg} aggregation calls, including ordinary queries, is at most

$$\varepsilon_{\text{tr}}(\lambda) := Q_{\text{agg}}(\lambda)\eta(\lambda)^{L(\lambda)}.$$

The capped internal hybrids are polynomial-time, and comparing their two endpoints with the original games accounts for the $2\varepsilon_{\text{tr}}$ term in G_1 . These caps are proof modifications; the protocol is unchanged. The growth conditions of Appendix C.7 make η and ε_{tr} negligible. The concrete prefix construction must satisfy its entropy requirement on every attempt; commitment hiding alone does not establish it.

Game G_3 . In the challenge prefix tr_{agg} , replace the commitments to padded source descriptions, the selection, and rerandomized tuples with fresh commitments to fixed dummy messages of the same padded lengths. The common public bounds determine the format, independently of the number of contributing sources. Preserve the public statement bindings, including C_W^* , and the independently sampled Γ_{agg} . Recompute the coefficients from the updated prefix and use the same fixed-prefix simulator of Appendix C.2 on $(x_{\text{agg}}, \text{tr}_{\text{agg}})$. Retain G_2 's single challenge attempt, ordinary-query caps, and all other checks and outcomes. Since no challenge retry or exposed presentation depends on the committed openings, the remaining affected computations are efficient postprocessing of the commitments. Adaptive joint hiding from Appendix C.3, as in SSU game G_3 , with both issuers' complete views, the proof-simulation setup state, and public coefficient-oracle access, therefore gives

$$|\Pr[G_3^A(1^\lambda) = 1] - \Pr[G_2^A(1^\lambda) = 1]| \leq \text{Adv}_{\text{Hid}}^{\text{agg}}(\lambda).$$

This term covers all replaced prefix commitments; the already simulated proof tail and retained query caps incur no additional transition loss.

Final view. Use the fresh protected challenge-wallet separation and private mask tables of the SSU transformations. Source failures are independent of β , since both branches execute the same source family. On reaching challenge aggregation in G_3 , conditional on the common source history and pre-challenge query state, either selection contains n distinct records distributed independently over $\mathbb{G}_1 \times \mathbb{Z}_p^* \times \mathbb{G}_1$, before checks and shuffling. Couple them by fixing positions in

$\mathcal{I}_0 \cap \mathcal{I}_1$ and swapping matched positions in $\mathcal{I}_0 \setminus \mathcal{I}_1$ and $\mathcal{I}_1 \setminus \mathcal{I}_0$; leave other positions fixed. Recouple private offsets and masks to preserve the common public source receipts. Couple the shuffle, output contribution, issuer seed challenges, retry decisions, both output-request blindings, reservation randomness, independent presentation, padded dummy commitments, proof-simulator coins, and public random-oracle functions. Since b, X, n , the registration root, and initial issuer states coincide, the resulting checks, duplicate and conflict outcomes, reissue messages, and state updates coincide as well. Subsequent ordinary queries use independent query wallets and these coupled states, with the same retry caps and failures; protected challenge objects remain inaccessible. Thus overlap and different contributing-source counts do not affect the complete views:

$$\text{View}_3^{(0)} \stackrel{d}{=} \text{View}_3^{(1)}, \quad \Pr[G_3^A(1^\lambda) = 1] = \frac{1}{2}.$$

The initial challenge-rejection case already returns an independent fair score.

Combining the bounds. As for SSU, define $\text{Adv}_{\text{Hid}} = \text{Adv}_{\text{Hid}}^{\text{iss}} + \text{Adv}_{\text{Hid}}^{\text{agg}}$, and write $p_i = \Pr[G_i^A(1^\lambda) = 1]$. The three transitions give

$$\begin{aligned} \text{Adv}_{\mathcal{H}, \mathcal{A}}^{\text{ASU}}(\lambda) &= |p_0 - p_3| \leq \sum_{i=1}^3 |p_i - p_{i-1}| \\ &\leq \text{Adv}_{\text{ZK}}(\lambda) + \text{Adv}_{\text{Hid}}(\lambda) + \text{Adv}_{\text{DH/KDF}}(\lambda) \\ &\quad + \text{Adv}_{\text{PRF}}(\lambda) + \text{Adv}_{\text{XDH}}(\lambda) + \frac{N_{\text{rec}}^{\text{ASU}}(\lambda)}{p} \\ &\quad + \eta(\lambda) + 3\varepsilon_{\text{tr}}(\lambda) + \varepsilon_{\text{dom}}(\lambda). \end{aligned}$$

Each primitive advantage includes all replacements across the ASU source family; the two hiding losses are counted once each. The truncation terms are $2\varepsilon_{\text{tr}}$ from the first transition and ε_{tr} from the second, evaluated with ASU's query bounds. Under the applicable requirements of Appendix C invoked above, including the entropy and growth conditions of Appendix C.7, the displayed sum is negligible. The concrete instantiation must satisfy those requirements. The per-hop view-indistinguishability arguments therefore make G_0 and G_3 computationally indistinguishable with advantage at most $\text{negl}(\lambda)$. Since the final view is independent of β , this establishes ASU under these requirements.

G.3 Linked-Swap Identity Privacy

Fix admissible challenges $\chi_j = (U_j, \text{req}, \mathcal{I})$, $j \in \{0, 1\}$, with input issuer B_b , output issuer $B_{b_{\text{out}}}$ for $b_{\text{out}} = 1 - b$, and selected value $n = V(\mathcal{I})$. Let $\mathbf{H}$ contain the common coalition history through issuance for both candidates and the LP. For $\beta = j$, the view is

$$\text{View}_{\text{LSIP}}^{(j)} = (\mathbf{H}, \text{Swap}_j, L_j, D_j, \text{KnownLink}_j, Q_j).$$

Here Swap_j contains both public payment receipts and the AMM transcript, L_j is the LP's complete challenge view, D_j contains the settlement fields defined for SSU with issuer $B_{b_{\text{out}}}$, and Q_j is the complete subsequent query view. The LP knows the DH-derived keys and masks on both legs. In particular, it recovers the user's input records, while settlement uses selected records from the LP's outgoing payment:

$$\begin{aligned}\mathcal{R}_j^{\text{in}} &= ((\delta^{\mu_{j,k}}, R_{j,k}, P_{U_{j,b}} g^{R_{j,k} \mu_{j,k}}))_{0 \leq k < V_{\text{in}}}, \\ \mathcal{C}_j &= \text{Shuffle}((S_{j,k}^{\text{LP}}, R_{j,k}^{\text{LP}}, T_{j,k}^{\text{LP}})_{k \in \mathcal{I}}).\end{aligned}$$

Here V_{in} is the common input value, v_j is the fresh input-wallet seed, $\mu_{j,k} = (v_j + k + 1)^{-1}$, and all challenges and masks follow the real derivations. The LP's view includes these records, its wallets, credentials, exchange secrets, and prover coins. The output request $C_{W,j}^*$ commits to $(sk_{U_{j,b_{\text{out}}}}, v_j^*, n)$, and $\text{KnownLink}_j = (\text{swid}_j, \text{rcpt}_{\text{agg},j})$ reveals the swap-settlement association. Thus β selects the user identity, while the request and selection are fixed. On failure, unproduced fields are absent and actual outcomes and resulting states remain visible.

Game G_0 . Let G_0 be the real LSIP experiment of Appendix B.5. Write $\text{View}_i^{(\beta)}$ for the complete coalition view in game G_i and $G_i^A(1^\lambda) = 1$ for its winning event. Our goal is to transform G_0 through indistinguishable games, bounding the change in winning probability at each transition, until the final view is independent of β :

$$\text{View}_t^{(0)} \stackrel{d}{=} \text{View}_t^{(1)}.$$

The adversary then wins with probability $1/2$. Transferring this bound back through the games will establish negligible LSIP advantage in G_0 , even with the LP's complete cryptographic view and the disclosed swap-settlement association.

Game G_1 . Reuse the proof, presentation, and request transformations of SSU games G_1, G_2, G_4 as internal hybrids, accounting for truncation locally below: simulate both candidates' preliminary and final issuance proofs, the selected user's payment proof, the aggregation proof tail, and the output seed-consistency proof; replace valid user-payment and final aggregate presentations with samples from the issuer-only distribution; and replace the seed messages in both stages of the candidates' issuance requests by 0, using SSU G_4 's one-attempt endpoint, independent private seeds, and private-credential handling. Keep the aggregation prefix and both output requests $C_W^*, C_{W,\text{fin}}^*$ real. LP-generated proofs and presentations remain honestly computed, as do all DH-derived keys, offsets, and masks. Update dependent inputs and retain actual checks, failures, and state transitions. Cap ordinary-query aggregation at $L \geq 1$ polynomially many attempts, returning failure on exhaustion, as in ASU. Apply zero knowledge from Appendix C.2, seed hiding from Appendix C.3, perfect blind-signature correctness and the source-seed domain bound from Appendix C.4, and the aggregation-retry bound of Appendix C.7, now with the full issuer-LP auxiliary view. The internal

hybrid argument gives

$$\begin{aligned} & |\Pr[G_1^A(1^\lambda) = 1] - \Pr[G_0^A(1^\lambda) = 1]| \\ & \leq \text{Adv}_{\text{ZK}}(\lambda) + \text{Adv}_{\text{Hid}}^{\text{iss}}(\lambda) + \varepsilon_{\text{tr}}(\lambda) + \varepsilon_{\text{dom}}(\lambda). \end{aligned}$$

The domain term uses LSIP's protected source-issuance counts and is charged once. The primitive advantages cover all replaced user proofs and issuance requests; presentation replacement has zero loss. Here $\varepsilon_{\text{tr}} = Q_{\text{agg}}\eta^L$ is the ASU truncation bound with LSIP query counts. Only the real endpoint is truncated, since G_1 retains the caps; no separate SSU truncation loss is imported.

Game G_2 . Replace the holder in both output requests: C_W^* now commits to $(sk_{U_0, b_{\text{out}}}, u^*, n)$ and $C_{W, \text{fin}}^*$ to $(sk_{U_0, b_{\text{out}}}, v^*, n)$, instead of the corresponding tuples for U_β , using fresh blinding. Retain u^* , the issuer challenges, $v^* = u^* + r^*$, capacity, and all seed-retry outcomes; the reference holder is registered by challenge admission. Update prefix bindings and coefficients and simulate both the aggregate tail and final seed proof on the updated inputs, as in G_1 . Run the issuers' responses with their actual keys and coins, using the replacement opening for private validation and unblinding. Under the perfect blind-signature correctness requirement of Appendix C.4 for either final request, private completion adds no attribute-dependent failure and can be omitted by a reduction without that opening; the output wallet and its descendants remain protected from ordinary queries. Retain the LP's computations, all other checks and outcomes, and the query caps. Joint hiding from Appendix C.3 for the complete structured requests' holder-secret attributes across both stages, with the full issuer-LP view and simulation setup as auxiliary information, gives

$$|\Pr[G_2^A(1^\lambda) = 1] - \Pr[G_1^A(1^\lambda) = 1]| \leq \text{Adv}_{\text{Hid}}^{\text{out}}(\lambda).$$

This is an additional use of request hiding beyond G_1 's seed hiding; the already simulated proof tail introduces no further ZK loss.

Game G_3 . For each used position $0 \leq k < V_{\text{in}}$ of the selected user's fresh input wallet, apply the group-output PRF replacement of SSU game G_6 :

$$(\delta^{\mu_{\beta, k}}, g^{\mu_{\beta, k}}) \mapsto (\delta^{z_k}, g^{z_k}), \quad z_k \leftarrow \mathbb{Z}_p.$$

Sample independently per position and reuse the pair on recovery; the scalars z_k remain private. Set $S_{\beta, k} = \delta^{z_k}$ and $T_{\beta, k} = P_{U_\beta, b}(g^{z_k})^{R_{\beta, k}}$. Keep the real DH-derived keys, $R_{\beta, k} = q_k + H_d(K, c, k)$, and masks. Recompute the masked input payload, the LP's recovered input records, and dependent public inputs and user-proof simulations. The LP's own wallet records and proofs remain honestly generated; retain all checks, failures, state updates, and query caps. By G_1 's seed hiding and presentation independence, the remaining view can be generated from these group outputs without v_β , omitting private credential computations by Appendix C.4's perfect blind-signature correctness. Joint pseudorandomness

from Appendix C.6, with the full issuer–LP auxiliary view and the LP’s known wallets kept real, therefore gives

$$|\Pr[G_3^A(1^\lambda) = 1] - \Pr[G_2^A(1^\lambda) = 1]| \leq \text{Adv}_{\text{PRF}}(\lambda).$$

The bound includes all used candidate-wallet positions and the valid-seed restriction. The serial and tag still share z_k ; their decorrelation is deferred.

Game G_4 . Before releasing the challenge user’s input payment, test its real values $R_{\beta,k} = H_{\text{ch}}(\omega, c, k) + H_d(K, c, k)$. If any is zero, terminate the experiment with an independent fair score; otherwise continue as in G_3 . This removes the event Bad_0 on which $T_{\beta,k} = P_{U_{\beta,b}}$, without resampling any DH-derived value or hash output. Invoke Appendix C.7: H_{ch} is an independent random oracle into $\mathbb{Z}_p$; after the context, keys, and offsets are fixed, ω has pointwise conditional min-entropy at least κ_{ch} given these data and the preceding oracle and issuer–LP history, independently of unused H_{ch} values. Let Q_{ch} bound all honest and adversarial H_{ch} queries before this sampling, and let N_{in} bound the number of user-input positions. Some target input (ω, c, k) was previously queried with probability at most $Q_{\text{ch}}2^{-\kappa_{\text{ch}}}$; otherwise each fresh uniform hash output cancels its fixed offset with probability $1/p$. Coupling the games until Bad_0 and taking a union bound gives

$$\begin{aligned} & |\Pr[G_4^A(1^\lambda) = 1] - \Pr[G_3^A(1^\lambda) = 1]| \\ & \leq \Pr_{G_3}[\text{Bad}_0] \leq \frac{N_{\text{in}}(\lambda)}{p} + Q_{\text{ch}}(\lambda)2^{-\kappa_{\text{ch}}(\lambda)} =: \varepsilon_0(\lambda). \end{aligned}$$

The growth conditions of Appendix C.7 make ε_0 negligible. The concrete beacon must satisfy the stated freshness requirements; post-request ordering alone does not imply them. All other checks, failures, and query caps remain as in G_3 .

Game G_5 . Retain G_4 ’s zero-challenge test and fair-score termination. In continuing executions, replace each user-input pair (δ^{z_k}, g^{z_k}) by independent $U_k, V_k \leftarrow \mathbb{G}_1$, and set

$$S_{\beta,k} = U_k, \quad T_{\beta,k} = P_{U_{\beta,b}} V_k^{R_{\beta,k}}.$$

Sample independently per position and reuse the pair on recovery. Keep the real DH-derived keys, challenges, and masks; recompute the masked input payload, recovered records, dependent public inputs, and simulated user proofs. The LP’s own wallet records and proofs remain honestly computed, with its visible state and coins consistent; retain all other checks, failures, state updates, and query caps. Under XDH and the programmable serial-base setup of Appendix C.1, apply the DDH embedding of SSU game G_8 to each input position. The remaining issuer–LP view is efficiently generated from these group outputs and the known keys, including the LP’s wallet seeds and prover coins. The hybrid argument therefore gives

$$|\Pr[G_5^A(1^\lambda) = 1] - \Pr[G_4^A(1^\lambda) = 1]| \leq \text{Adv}_{\text{XDH}}(\lambda),$$

where the bound includes all input-position hybrids. Since $R_{\beta,k} \neq 0$ on continuing executions, the map $V \mapsto P_{U_{\beta},b} V^{R_{\beta,k}}$ is a bijection on $\mathbb{G}_1$. Thus each serial-tag pair consists of independent uniform group elements at generation, even for fixed public key and real challenge, before the remaining protocol checks.

Final view. In G_5 , couple the common history H , issuer and pool states, public random-oracle functions, fresh user exchange key, and the LP's pre-swap state and coins across both choices of β . Use the identity-independent context and beacon law of Appendix C.7. For the same request and coupled exchange keys and public randomness, the contexts, real DH-derived values, and R values coincide, so G_4 's test has the same outcome. On continuing executions, couple the input serials and tags using their common uniform distribution from G_5 , and couple presentation and proof-simulation randomness. Both public payments and the LP's resulting swap view then coincide. The output request uses the same reference holder by G_2 . Couple its contribution u^* , issuer challenges r^* , seed-retry outcomes, and both commitment blindings, the shuffle and reservation randomness, and the aggregation rerandomization and commitment coins. Since the source is the same LP payment and $\mathcal{I}$ is fixed, the retained prefix, coefficients, settlement transcript, and **KnownLink** coincide. Anonymous reissue discloses no further receiver identity. All verification, duplicate, replay, conflict, and reissue checks therefore have identical inputs and coins, giving the same visible failures and shared-state updates. Subsequent ordinary queries use these coupled states and independent query wallets; both candidates are protected from corruption, and challenge objects and their descendants are inaccessible. Their answers consequently coincide as well. Taking a terminated execution's view to be its transcript up to termination, including any visible failure, we obtain

$$\text{View}_5^{(0)} \stackrel{d}{=} \text{View}_5^{(1)}, \quad \Pr[G_5^A(1^\lambda) = 1] = \frac{1}{2}.$$

The initial rejection and G_4 termination branches already return independent fair scores.

Combining the bounds. Write $p_i = \Pr[G_i^A(1^\lambda) = 1]$. Since $p_5 = \frac{1}{2}$, the five transitions give

$$\begin{aligned} \text{Adv}_{H,A}^{\text{LSIP}}(\lambda) &= |p_0 - p_5| \leq \sum_{i=1}^5 |p_i - p_{i-1}| \\ &\leq \text{Adv}_{\text{ZK}}(\lambda) + \text{Adv}_{\text{Hid}}^{\text{iss}}(\lambda) + \text{Adv}_{\text{Hid}}^{\text{out}}(\lambda) \\ &\quad + \text{Adv}_{\text{PRF}}(\lambda) + \text{Adv}_{\text{XDH}}(\lambda) + \varepsilon_{\text{tr}}(\lambda) + \varepsilon_0(\lambda) + \varepsilon_{\text{dom}}(\lambda). \end{aligned}$$

Each primitive advantage includes all replacements in its hop. The two hiding terms cover the source seeds and output holder secret once each; ε_{dom} is charged once in the first hop; ε_{tr} and ε_0 arise only in $G_0 \rightarrow G_1$ and $G_3 \rightarrow G_4$, respectively, using LSIP's query bounds. No DH/KDF idealization or aggregation-prefix hiding is used, so neither $\text{Adv}_{\text{DH/KDF}}$ nor $\text{Adv}_{\text{Hid}}^{\text{agg}}$ is charged. Under the setup,

zero-knowledge, hiding, blind-signature correctness, joint seed-generation, inverse-exponent, XDH, and freshness requirements cited above in Appendix C, the displayed sum is negligible; this includes Appendix C.7’s identity-independence and growth conditions. The concrete instantiation must satisfy those requirements. The per-hop view-indistinguishability arguments therefore make G_0 and G_5 computationally indistinguishable with advantage at most $\text{negl}(\lambda)$. Since the final view is independent of β , this establishes LSIP under these requirements.

H Practical Proof-System Architecture

We describe a concrete proof architecture for the relations of Appendix E. The main requirements are succinct public verification, efficient recursive verification of payment proofs during aggregate reissue, and proof parameters whose security does not depend on any issuer or LP knowing a setup trapdoor.

Proof-system choice. Proof systems requiring a circuit-specific trusted setup are unattractive in our setting. In particular, the LP must not generate such parameters, since knowledge of the corresponding setup trapdoor would invalidate proof soundness, while requiring issuer participation would introduce an unnecessary dependency on parties that should remain outside the exchange protocol. Groth16 is a prominent example of such a circuit-specific construction.

We therefore favor a recursion-friendly Plonkish proof system with either transparent polynomial commitments, such as IPA-based commitments with deterministically derived public generators, or a universal and updatable SRS. The former removes trusted setup entirely at the cost of somewhat larger proofs and verification, while the latter permits one independently generated public SRS to be reused across circuits. In either case, neither the LP nor the issuers control or know any setup trapdoor.

Seed-generation proofs. The preliminary identified-issuance relation proves an opening to (sk, u, N) and $g^{sk} = P_{X,b}$ for the authenticated authorization. For reissue, Π_{agg} instead proves hidden registered membership for C_W^* . A separate Π_{seed} in the interactive issuance or settlement phase binds both request openings, the recorded issuer challenge, $v = u+r$, and $v \in \mathcal{V}$. These checks require no pairing and add no issuer interaction to a swap or to Aggregate_b . The backend must realize the completed-proof extraction and composition requirements of Appendix C.2. Lemma 3 uses rewinding of completed executions and does not require a straight-line extractor for the recursive aggregate proof. The issuer common-randomness functionality must also be instantiated under the stated threshold corruption model, including the realization qualifications in Appendix C.4.

Payment proofs. The payment relation R_{pay} is compiled for a public bound V_{max} . Payments with $V < V_{\text{max}}$ use padded inactive slots, so one verification key supports every admissible payment value. The resulting Π_{pay} proves the hidden wallet, capacity, serial/tag, masking, receiver-binding, and payload-digest relations of Appendix E.

The receiver payload M_{pay} remains in the DA layer and only its digest is bound into x_{pay} . Hence blockchain verification consists of the succinct proof, the public statement, the DA certificate, and the separate native check $\text{PairCred}_b(I) = 1$. The PS pairing relation is kept outside the proof circuit, as in the construction.

Recursive aggregate reissue. The same recursion-friendly proof-system family is used for Π_{agg} . The aggregate circuit is compiled for at most m_{max} padded source slots and directly verifies the corresponding Π_{pay} proofs before enforcing receiver recovery, private coin selection, serial distinctness, presentation rerandomization, credential aggregation, and the preliminary output-wallet relation.

The commitment prefix tr_{agg} is fixed before deriving $a_j = H_{\text{coef}}(\text{tr}_{\text{agg}}, j) \in \mathbb{Z}_p^*$, and the proof binds its hidden openings to the same prefix. Settlement therefore requires one aggregate-proof verification followed by the native check $\text{PairCred}_b(I_{\text{agg}}) = 1$; the reissue branch additionally verifies the final seed-consistency proof before releasing signature shares. In particular, no pairing verification is performed recursively inside the aggregate circuit.

Curves and public parameters. The recursive backend should use a curve/field configuration for which verifier arithmetic is efficient inside the aggregate circuit, preferably a recursion-friendly curve cycle. For a transparent IPA-based backend, the required generator vectors can be derived publicly by domain-separated hash-to-curve and therefore contain no secret setup information. If a universal-SRS backend is used instead, the SRS can be taken from an existing independently generated and publicly auditable multiparty ceremony and reused across payment and aggregation circuits.

Cost profile. For fixed V_{max} , each payment has a succinct proof whose size is independent of V , while the $O(V)$ receiver payload remains in the DA layer. Transparent or universal-setup proofs may be larger than circuit-specific Groth16 proofs, but avoid introducing protocol-specific trusted setup. Aggregate proving requires at most m_{max} recursive payment-proof verifications together with $O(m_{\text{max}} V_{\text{max}})$ bounded recovery and selection work, while aggregate verification remains succinct.