Accountable Anonymous Cross-Chain Atomic Swaps via Threshold-Openable Linkable Ring Adaptor Signatures

YUANSHAO LIANG, Peking University Shenzhen Graduate School, China

HUI LI, Peking University Shenzhen Graduate School, China

Cross-chain atomic swaps enable decentralized asset exchange across independent ledgers, but their public transcripts may expose transaction flows, signer identities, and business relationships. Existing scriptless swap mechanisms based on adaptor signatures provide witness-coupled execution, yet the signer is public. Linkable ring adaptor signatures further hide the signer within a ring, but they do not support controlled opening when a completed swap becomes disputed. In this paper, we put forward ALRAS, an accountable anonymous cross-chain atomic-swap mechanism based on Pair-Consistent Threshold-Openable Linkable Ring Adaptor Signatures. We design a pair-consistent opening mechanism that binds a ring-adaptor pre-signature, an opening ciphertext, a scoped link tag, and the completed full signature to the same hidden signer. This mechanism prevents false-opening, ciphertext-substitution, and transcript-splicing attacks, while requiring threshold-governed and publicly verifiable opening in the dispute path. We construct a generic compiler from a NIZK-compatible, index-hiding LRAS using threshold ElGamal opening, consistency proofs, and public pair verification. We further formalize the signing-layer interface and security goals, and analyze signer anonymity, scoped linkability, witness extractability, non-frameability, and pair-consistent opening soundness. A reproducible Python artifact provides reference modules, attack-oriented sanity tests, and operation-count benchmarks. The results show the expected linear scaling with ring size and isolate the accountability overhead introduced by threshold opening and transcript binding.

CCS Concepts: • **Security and privacy → Cryptography**; **Security protocols**; *Privacy-preserving protocols.*

Additional Key Words and Phrases: Adaptor signatures, ring signatures, accountable anonymity, threshold opening, cross-chain atomic swaps, blockchain privacy

1 Introduction

Cross-chain atomic swaps allow two parties to exchange assets across independent ledgers without relying on a centralized intermediary [17]. Later work studies variants with improved space or local-time complexity for HTLC-based swap protocols [19]. Such swaps are useful for decentralized exchange, cross-chain liquidity, and inter-ledger settlement because neither party should be able to complete only one side of the exchange.

Adaptor and lockable signatures provide a scriptless way to realize this conditional execution: completing one signature reveals a witness that enables the counterparty to complete the other side [1, 16, 29]. However, when the swap transcript is posted to public or consortium ledgers, observers may correlate transaction flows, signer identities, and business relationships.

Ring-anonymous signing is desirable in normal executions because it hides which ring member authorized a swap. In regulated exchanges, institutional custody, or consortium-chain settlement, however, a completed swap may later become disputed due to fraud, double spending, money laundering, or contractual breach. In such cases, a judge or regulator may need to identify the responsible signer, but only through a threshold-governed and publicly verifiable procedure.

Existing cryptographic tools address only parts of this lifecycle. Adaptor signatures provide witness coupling for scriptless swaps, but they do not hide the signer. Linkable ring adaptor signatures add ring anonymity and scoped linkability, but they do not provide threshold-governed opening [30, 31]. Group signatures, accountable ring signatures,

Authors' Contact Information: Yuanshao Liang, Peking University Shenzhen Graduate School, Shenzhen, China, lys@stu.pku.edu.cn; Hui Li, Peking University Shenzhen Graduate School, Shenzhen, China, lih64@pkusz.edu.cn.

and deanonymizable ring signatures support controlled opening, but they are not designed for the pre-sign/adapt/extract lifecycle of atomic swaps [3, 6, 8, 14]. Therefore, these tools cannot be directly combined into an accountable anonymous swap protocol.

Unfortunately, simply attaching an opening ciphertext to an LRAS transcript is insufficient. The ciphertext may encrypt an unrelated ring index, be substituted from another transcript, or be spliced into a different completed signature. Even if the ciphertext can be opened, the judge still needs evidence that the opened public key is the same hidden signer who generated the adaptor pre-signature and whose completed full signature appears in the swap. A naive construction also risks giving one opener unilateral tracing power, which would weaken the anonymity guarantee in normal executions.

This paper asks: *How can an anonymous cross-chain swap preserve ring anonymity in normal executions while enabling threshold-governed, publicly verifiable opening only for disputed completed transcripts that are pair-consistent with their adaptor pre-signatures?*

Key observation. The accountable-opening problem is not merely about opening a ring signature; it is about opening the right signer for the right completed swap transcript. Thus, the opening ciphertext, LRAS pre-signature, scoped link tag, adaptor witness, and completed full signature must be bound to the same hidden ring index. This turns accountable anonymous swaps into a pair-consistent witness-coupling problem over a composite transcript.

We answer this question with ALRAS (Pair-Consistent Threshold-Openable Linkable Ring Adaptor Signatures), a signing layer for accountable anonymous atomic swaps. In the normal path, a signer generates an LRAS pre-signature together with a scoped link tag, an opening ciphertext, and a NIZK consistency proof. In the completion path, the adaptor witness turns the pre-signature into a full signature. A public PairVerify algorithm checks that the full signature corresponds to the original pre-signature, preventing transcript splicing. In the dispute path, a threshold of regulators publishes verifiable opening shares, and a judge accepts the opening only if the shares, the consistency proof, the pre/full-signature pair, and the opened public key all verify.

1.1 Contributions

We make the following contributions.

(1) **Accountable anonymous-swap model.** We formalize the accountable anonymous atomic-swap setting, including entities, transcript structure, leakage boundary, adversarial capabilities, and attack-driven requirements.

(2) ALRAS **construction.** We construct a pair-consistent threshold-openable LRAS layer from an index-hiding LRAS, threshold ElGamal opening, NIZK consistency proofs, public pair verification, and judge-verifiable threshold opening.

(3) **Security analysis.** We prove that ALRAS preserves signer anonymity in normal executions while providing pair-consistent opening soundness, non-frameability, scoped linkability, witness extractability, and threshold opening privacy under standard assumptions.

(4) **Evaluation artifact.** We provide a reference implementation, attack-oriented sanity tests, generated figures and tables, and operation-count benchmarks that isolate the transcript-binding, pair-verification, and dispute-opening overhead introduced by accountable opening.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 introduces the cryptographic building blocks. Section 4 presents the system and security model. Section 5 gives the ALRAS construction. Section 6

proves the security properties. Section 7 analyzes performance, including the communication, computation, and reference-evaluation costs of accountable opening. Section 8 concludes the paper.

2 Related Work

In this section, we review the works most relevant to accountable anonymous atomic swaps. Existing studies provide useful building blocks for conditional execution, signer anonymity, linkability, and controlled opening. However, these properties are usually studied separately. We divide the discussion into adaptor-signature-based atomic swaps, anonymous and linkable signing primitives, accountable opening mechanisms, and transaction-system accountability, and then summarize the functional comparison in Table 1.

Adaptor-signature-based atomic swaps. Cross-chain atomic swaps were formalized as a distributed coordination problem by Herlihy [17], with later work improving the complexity of HTLC-based swap protocols [19]. Adaptor signatures provide a scriptless technique for conditional payments and atomic swaps [1]. A signer first produces a pre-signature that is not yet a valid signature; once a witness satisfying a public statement is supplied, the pre-signature can be adapted into a full signature. The witness can then be extracted from the pre/full-signature pair. Identification-based adaptor signatures and later foundations formalize pre-signature unforgeability, adaptability, and witness extractability [12, 16]. Lockable signatures further generalize scriptless conditional execution for blockchain settings [29]. These constructions provide the witness-coupling mechanism required by atomic swaps. However, standard adaptor signatures are single-signer primitives: the signer identity is public, and there is no ring anonymity or signer-opening mechanism.

Anonymous and linkable signing primitives. Ring signatures allow a signer to sign on behalf of an ad-hoc set of public keys while hiding which member produced the signature [26]. Linkable ring signatures further support public detection of repeated signing within a scope, and have been used in privacy-preserving payment systems such as RingCT [21, 24]. More compact ring-signature constructions improve asymptotic signature size and verification cost [2]. These primitives provide anonymity and linkability, but they do not expose the pre-sign, adapt, and extract workflow needed for scriptless atomic swaps.

Linkable ring adaptor signatures combine ring anonymity with adaptor witness extraction. In an LRAS-based swap, the signer remains hidden within a ring while the counterparty can still recover the witness needed to complete the other side of the exchange. PayMo and later Monero-enabled payment-channel constructions develop LRAS-style techniques for privacy-preserving off-chain payments [30, 31]. Existing LRAS constructions are therefore close to the normal execution path required by ALRAS. However, they do not support threshold-governed opening, judge-verifiable accountability, or public verification that an opened identity corresponds to the signer of a completed pre/full-signature pair.

Accountable opening mechanisms. Group signatures initiated the controlled-opening model for anonymous signing [8], and later formal treatments clarify traceability and non-frameability requirements [3]. Traceable, accountable, and deanonymizable ring signatures add related opening or tracing mechanisms to ad-hoc anonymity sets [6, 14, 15]. Accountable tracing signatures and dynamic group-signature foundations further study how opening authority can be made accountable or flexible [5, 20]. These properties are related to the dispute path required by ALRAS. However, accountable opening primitives are not designed around adaptor-signature execution. They do not provide a pre-signature, an Adapt algorithm, an Extract algorithm, or a public relation between a pre-signature and a completed

Table 1. Functional comparison of ALRAS with related primitives. (● = supported, ○ = not supported.) "Pair-consistent opening" means that an opened identity is accepted only for a completed full signature publicly bound to the corresponding adaptor pre-signature.

Primitive	Ring anonymity	Witness extraction	Scoped linkability	Threshold opening	Public opening verify	Pair-consistent opening
Adaptor signature [1]	○	●	○	○	○	○
Linkable ring sig. [21]	●	○	●	○	○	○
Accountable ring sig. [6]	●	○	○	●	●	○
LRAS/LTRAS [30, 31]	●	●	●	○	○	○
ALRAS (this work)	●	●	●	●	●	●

signature. Directly replacing LRAS with accountable ring or group signatures would therefore remove the scriptless atomic-swap logic.

Transaction-system privacy and accountability. Blockchain privacy systems such as Zerocoin and Zerocash demonstrate how cryptographic proofs can hide transaction participants or values at the ledger layer [4, 22]. Surveys of blockchain privacy and payment-channel cryptography place these systems in a broader transaction-protocol context [9, 32]. Recent payment-channel and smart-contract studies also analyze system-level availability, congestion, or verification concerns [23, 25]. These works motivate the need to reason about cryptographic mechanisms together with protocol and governance boundaries. However, they usually operate at the transaction-system or smart-contract layer rather than defining a signing primitive that simultaneously supports witness extraction, ring anonymity, and threshold-governed opening. ALRAS is complementary to these systems: it supplies a cryptographic signing layer for accountable privacy in disputed anonymous swaps.

Positioning. In summary, adaptor signatures provide witness coupling but no signer anonymity; linkable ring signatures provide anonymity and linkability but no adaptor witness; LRAS provides both witness coupling and ring anonymity but no controlled opening; accountable ring signatures provide opening but no adaptor-signature lifecycle. None of these primitives simultaneously provides adaptor witness coupling, ring anonymity, scoped linkability, threshold-governed opening, public opening verification, and pair-consistent opening. Table 1 summarizes this positioning.

3 Preliminaries

This section introduces the notation and cryptographic building blocks used by ALRAS. Following the construction in Section 5, we need three tools: a linkable ring adaptor signature interface for anonymous witness coupling, a threshold encryption interface for controlled opening, and Fiat–Shamir-style zero-knowledge proofs for binding transcript components. We only define the abstract interfaces and security properties here; the ALRAS transcript objects and consistency relation are specified in Section 5.

3.1 Notation

Throughout, $\lambda \in \mathbb{N}$ denotes the security parameter, and $\mathsf{negl}(\lambda)$ denotes a negligible function. All algorithms are probabilistic polynomial time unless stated otherwise. Let $\mathbb{G}$ be a cyclic group of prime order q with generator G, and write group operations additively. The discrete-logarithm problem is assumed hard in $\mathbb{G}$. Let $\mathbb{Z}_q$ be the field of integers modulo q and let $\mathbb{Z}_q^* = \mathbb{Z}_q \setminus \{0\}$.

A ring is an ordered set $R = \{pk_1, \ldots, pk_n\}$ of well-formed and distinct public keys. The hidden signer has index $j \in [n]$, secret key sk_j, and public key $pk_j = sk_j G$. An adaptor statement is a group element $Y = wG$ for witness $w \in \mathbb{Z}_q$.

Table 2. Basic notation used throughout the paper.

Symbol	Meaning
λ	Security parameter
$\mathsf{negl}(\lambda)$	Negligible function
$\mathbb{G}, q, G$	Prime-order group, order, and generator
$\mathbb{Z}_q$	Scalar field modulo q
$R = \{pk_1, \ldots, pk_n\}$	Ring of candidate signing public keys
j, sk_j, pk_j	Hidden signer index, secret key, and public key
$Y = wG$	Adaptor statement and witness relation
sid	Scope identifier for linkability
H_{tag}	Hash-to-group function for scoped tags
Y_R	Regulator public key for threshold opening

A scope identifier $sid \in \{0,1\}^*$ defines the linkability domain. We use $H_{\text{tag}} : \{0,1\}^* \to \mathbb{G}$ as a hash-to-group function for scoped link tags and $H : \{0,1\}^* \to \{0,1\}^{\text{poly}(\lambda)}$ as a general random oracle. The regulator public key is denoted by Y_R.

3.2 Linkable Ring Adaptor Signatures

A linkable ring adaptor signature (LRAS) scheme combines the pre-sign/adapt/extract workflow of adaptor signatures with ring anonymity and scoped linkability. We use an abstract LRAS interface extracted from adaptor-signature and lockable-signature foundations and LRAS/LTRAS-style payment-channel constructions [16, 29–31]. It works as follows.

- $\mathsf{LRAS.Setup}(1^\lambda) \to pp_{\mathsf{LRAS}}$: On input a security parameter λ, it outputs the public parameters pp_{LRAS}.
- $\mathsf{LRAS.KeyGen}(pp_{\mathsf{LRAS}}) \to (pk, sk)$: On input pp_{LRAS}, it outputs a signing public/private key pair (pk, sk).
- $\mathsf{LRAS.PreSign}(sk, R, m, Y, sid; \rho) \to \hat{\sigma}_R$: On input a secret key sk, a ring R, a message m, an adaptor statement $Y = wG$, a scope identifier sid, and randomness ρ, it outputs a ring-adaptor pre-signature $\hat{\sigma}_R$.
- $\mathsf{LRAS.PreVerify}(R, m, Y, sid, \hat{\sigma}_R) \to b$: On input a ring R, message m, adaptor statement Y, scope sid, and pre-signature $\hat{\sigma}_R$, it outputs $b \in \{0,1\}$ indicating whether the pre-signature is valid.
- $\mathsf{LRAS.Adapt}(\hat{\sigma}_R, w) \to \sigma_R$: On input a valid pre-signature $\hat{\sigma}_R$ and a witness w satisfying $Y = wG$, it outputs a completed ring signature σ_R.
- $\mathsf{LRAS.Verify}(R, m, sid, \sigma_R) \to b$: On input a ring R, message m, scope sid, and completed signature σ_R, it outputs $b \in \{0,1\}$.
- $\mathsf{LRAS.Extract}(\hat{\sigma}_R, \sigma_R) \to w$ or $\perp$: On input a pre-signature and its completed signature, it outputs the adaptor witness w, or $\perp$ if extraction fails.
- $\mathsf{LRAS.Link}(\sigma_{R,1}, \sigma_{R,2}, sid) \to b$: On input two completed signatures and a scope identifier, it outputs $b \in \{0,1\}$ indicating whether the two signatures are linked within the same scope.

The interface needs to satisfy four properties. Correctness requires that honestly generated pre-signatures verify, adaptation with a valid witness produces a valid completed signature, and extraction recovers the witness. Signer anonymity requires that the transcript hides the signing index inside the ring. Witness extractability requires that an accepting pre/full pair yields w satisfying $Y = wG$. Scoped linkability requires public linking inside one scope and unlinkability across different scopes.

For the ALRAS compiler, we additionally require a public pair-consistency interface:

$$\mathsf{LRAS.PairVerify}(R, \hat{\sigma}_R, \sigma_R, Y) \to \{0, 1\}.$$

On input a ring, a pre-signature, a completed signature, and an adaptor statement, it outputs a bit indicating whether σ_R is a valid adaptation of $\hat{\sigma}_R$ under Y. This is an explicit requirement on the base primitive used by ALRAS; it is not claimed to be part of every prior LRAS construction.

3.3 Threshold ElGamal Opening

We use a (t, N) threshold encryption scheme Π_{TEnc} to encrypt a signing public key under a regulator public key. It works as follows.

- $\mathsf{TEnc.Setup}(1^\lambda) \to pp_{\mathsf{TEnc}}$: On input a security parameter λ, it outputs the public parameters pp_{TEnc}.
- $\mathsf{RegKeyGen}(pp_{\mathsf{TEnc}}, N, t) \to (Y_R, \{(sk_l, Y_l)\}_{l=1}^N)$: On input public parameters, the number of regulators N, and the threshold t, it outputs a regulator public key Y_R and N secret/public opening-share key pairs.
- $\mathsf{TEnc.Enc}(Y_R, P; r) \to C$: On input the regulator public key Y_R, a group element P, and randomness r, it outputs a ciphertext C encrypting P.
- $\mathsf{OpenShare}(pp, sk_l, Y_l, C) \to (l, D_l, \pi_l^{\mathrm{dec}})$: On input a regulator share sk_l, verification key Y_l, and ciphertext C, it outputs a partial decryption share D_l and a DLEQ proof π_l^{dec} showing that the share is well formed.
- $\mathsf{OpenCombine}(pp, C, \{(l, D_l, \pi_l^{\mathrm{dec}})\}_{l \in S}) \to P$ or $\perp$: On input a ciphertext and a set of decryption shares, it outputs the plaintext group element P if at least t valid shares are provided; otherwise it outputs $\perp$.

The interface needs to satisfy three properties. Correctness requires that any set of at least t valid shares recovers the encrypted group element. Below-threshold privacy requires that fewer than t corrupted regulators learn nothing about P from the ciphertext and their own shares. Share verifiability requires that invalid decryption shares are rejected before combination. In the concrete instantiation used later, we use ElGamal encryption, Shamir-style threshold reconstruction, and Chaum–Pedersen DLEQ proofs for share verifiability [7, 11, 28]. We do not model distributed key generation or proactive resharing; such regulator-key management can be supplied by threshold-cryptography mechanisms such as proactive secret sharing [18].

3.4 Sigma Protocols, NIZK, and DLEQ Proofs

Sigma protocols are standard tools for proving knowledge of witnesses in relations over groups. A sigma protocol is a three-move public-coin protocol between a prover and an honest verifier. First, the prover sends a commitment a. Second, the verifier samples a random challenge c. Finally, the prover returns a response z, and the verifier accepts if the transcript satisfies the verification predicate.

By applying the Fiat–Shamir transform in the random-oracle model, such protocols can be made non-interactive [13]. We use the notation

$$\mathsf{NIZK}\{x : y = xG\}$$

to denote a non-interactive zero-knowledge proof showing knowledge of a scalar x such that $y = xG$, without revealing x. More generally, for an efficiently decidable relation $\mathcal{R}$, a NIZK proof system provides the following interface.

- $\mathsf{NIZK.Setup}(1^\lambda, \mathcal{R}) \to crs$: On input a security parameter and a relation $\mathcal{R}$, it outputs a common reference string crs.

- NIZK.Prove$(crs, stmt, wit) \to \pi$: On input crs, a statement $stmt$, and a witness wit such that $(stmt, wit) \in \mathcal{R}$, it outputs a proof π.
- NIZK.Verify$(crs, stmt, \pi) \to b$: On input crs, a statement, and a proof, it outputs $b \in \{0, 1\}$.

The proof system needs to satisfy completeness, soundness, and zero knowledge. Stronger extractability requirements are stated explicitly in the security analysis when they are needed. The relations used by ALRAS consist of discrete-logarithm knowledge statements and linear group equations, which can be instantiated using Fiat–Shamir-transformed sigma protocols [10].

We also use Chaum–Pedersen DLEQ proofs to verify threshold-opening shares. A DLEQ proof shows equality of discrete logarithms with respect to two bases, for example

$$\mathsf{DLEQ}\{x : Y = xG \wedge D = xC_1\}.$$

In ALRAS, regulator l uses such a proof to convince the judge that its opening share is computed with the same secret share that defines its public verification key, without revealing the secret share itself.

4 System Architecture and Security Model

In this section, we define the accountable anonymous-swap model used by ALRAS. We first state the basic idea and modeling scope, then define the participating entities and trust boundary, and finally specify public transcript objects, adversarial capabilities, leakage, and security goals. The concrete normal-path and dispute-path workflow is given in Section 5.

4.1 Basic Idea and Modeling Scope

In an anonymous cross-chain atomic swap, a participant claims one side of the swap by releasing an adaptor witness, and the counterparty uses that witness to complete the other side. If the claim is authorized by an LRAS-style pre-signature, the signer remains hidden within a ring during normal execution while the witness-coupling property needed for atomicity is preserved.

Why naive opening fails. A completed swap may later become disputed. In that case, the system should identify the responsible signing public key only through a threshold-governed and publicly verifiable opening procedure. A naive approach is to attach an encryption of the signer identity to an LRAS transcript. This is insufficient because the ciphertext may encrypt an unrelated key, be substituted from another transcript, or be associated with a different completed signature.

ALRAS treats accountable opening as a pair-consistent transcript-binding problem. The augmented pre-signature contains a base LRAS pre-signature, an opening ciphertext, a scoped link tag, and a NIZK consistency proof. The proof binds the pre-signature, ciphertext, and tag to the same hidden ring member. After adaptation, PairVerify checks that the completed full signature is paired with the original pre-signature. A judge accepts an opening only when the transcript pair and the threshold opening certificate both verify.

Figure 1 shows the modeled roles and the trust boundary. The figure illustrates the normal anonymous swap path and the dispute opening path, but the algorithms and message flow are specified in Section 5; here we only define entities, public data, leakage, and security goals.

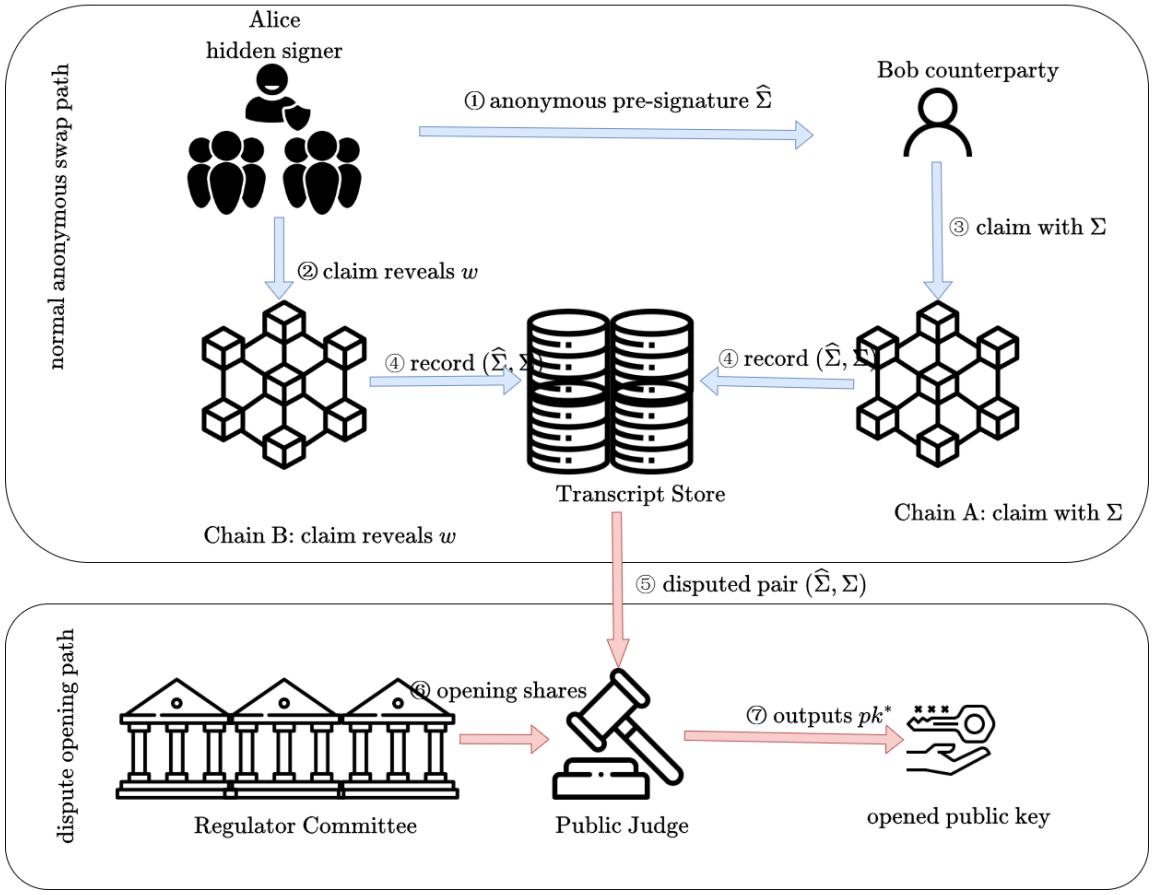

Fig. 1. System architecture and trust boundary. Alice and Bob are untrusted swap participants. ALRAS models the signing-layer transcript objects, transcript storage, threshold opening shares, and public judge verification. Opening authorization and public-key-to-identity mapping are outside the ALRAS cryptographic claim.

4.2 Entities and Trust Boundary

Signer. The signer is the privacy-seeking swap participant, denoted by Alice in the motivating workflow. She holds a signing secret key sk_j whose public key pk_j is one member of the ring $R = \{pk_1, \ldots, pk_n\}$. Alice generates an augmented pre-signature $\hat{\Sigma}$ for message m, adaptor statement $Y = wG$, scope sid, and regulator key Y_R. Alice is not trusted: she may choose malformed auxiliary data, encrypt a wrong opening key, or attempt to frame another ring member. The consistency proof forces the pre-signature, opening ciphertext, and scoped tag to refer to the same hidden signer.

Counterparty. The counterparty, denoted by Bob, receives $\hat{\Sigma}$ and verifies it before accepting the swap. After the adaptor witness w is revealed, Bob obtains a completed transcript Σ. Bob is not trusted: he may replay messages, delay completion, or attempt to splice a pre-signature from one execution with a full signature from another. The public PairVerify check prevents such spliced transcript pairs from being accepted by Judge.

Ledgers. The two underlying ledgers execute the asset-transfer logic of the atomic swap. They provide transaction ordering, validity checks, and timeout/refund enforcement after confirmation. ALRAS does not implement chain-level fairness; it supplies signature-level witness coupling and accountability data used by the swap protocol.

Transcript store. The transcript store is an on-chain or off-chain repository that records the augmented pre-signature $\hat{\Sigma}$, completed transcript Σ, and dispute metadata. It is not trusted for correctness: all retrieved objects must be verified before use. It is, however, an availability assumption. If a disputed transcript pair cannot be retrieved, Judge cannot run.

Regulator committee. The regulator committee consists of N members holding shares of a (t, N) threshold opening key. A regulator l can produce an opening share $(D_l, \pi_l^{\mathsf{dec}})$ for a disputed transcript. Fewer than t corrupted regulators should not learn the encrypted signer public key. A set of at least t valid shares enables public reconstruction of the opened public key, but only for the opening ciphertext included in the accepted transcript.

Public judge. The judge is a deterministic public verifier, not a trusted authority. Given $\hat{\Sigma}$, Σ, and an opening certificate τ_{open}, it checks transcript validity, pair consistency, opening-share correctness, and ring membership of the opened key. If all checks pass, it outputs an opened public key pk^*; otherwise it outputs $\perp$. The judge does not output a legal identity.

External governance. External governance includes opening authorization, legal process, consortium policy, regulator coordination policy, and KYC/PKI mapping from pk^* to a real-world identity. These functions are outside the ALRAS cryptographic claim. ALRAS verifies that an opening certificate is cryptographically consistent with a disputed transcript; it does not decide whether the opening is legally authorized.

4.3 Transcript Objects and Public Data

Ring and scope. A ring $R = \{pk_1, \ldots, pk_n\}$ is an ordered set of candidate signing public keys. The hidden signer has index $j \in [n]$. A scope identifier sid defines the linkability domain.

Adaptor statement. The adaptor statement is a group element $Y = wG$. The witness w is revealed when the swap is completed and is used to adapt the pre-signature into a full signature.

Pre-signature and completed transcript. The augmented pre-signature $\hat{\Sigma}$ contains a base LRAS pre-signature and accountability material. The completed transcript Σ contains the adapted full signature and the same accountability material. Their precise syntax and consistency relation are defined in Section 5; here we only require that both objects are available to the judge in a dispute.

Opening certificate. An opening certificate τ_{open} contains at least t verifiable regulator shares and an opened public key pk^*. The judge accepts it only together with a valid and pair-consistent transcript pair.

Public data. The public data includes R, m, Y, sid, Y_R, the transcript pair $(\hat{\Sigma}, \Sigma)$, and any opening certificate presented in a dispute. These values define the public statement checked by the verification and judging algorithms.

4.4 Adversarial Model and Leakage

Malicious users. Swap participants may be malicious. A malicious signer may generate malformed transcript fields, encrypt a wrong opening key, or attempt to frame another ring member. A malicious counterparty may replay off-chain

messages, delay completion, or combine transcript components from different executions. Security requires that such behavior cannot make Judge accept an opening inconsistent with the accepted transcript pair.

Corrupted regulators. For privacy before opening, the adversary may corrupt fewer than t regulators. Such an adversary should not learn the encrypted signer public key from C_{open}. For accountability, once at least t valid shares are presented, the opening must still be publicly verifiable and bound to the accepted transcript pair.

Public judge and verifiability. The judge is not trusted. Any party can run the judging algorithm on public inputs. Therefore, security does not rely on judge honesty; it relies on deterministic public verification of transcript validity, pair consistency, share correctness, and ring membership of the opened key.

Ledger and data availability assumptions. The ledgers are assumed to enforce transaction validity and timeout/refund rules after confirmation. If transcripts are stored off-chain, the transcript store must preserve availability for disputed transcripts. ALRAS does not protect against chain reorganizations, consensus failures, or denial-of-service against transcript availability.

Leakage. ALRAS intentionally reveals the ring R, message m, scope sid, adaptor statement Y, regulator public key Y_R, transcript sizes, same-scope linkability, and the opened public key after a valid threshold opening. It does not hide information that is already visible from the underlying ledgers or network layer.

Out-of-scope boundaries. ALRAS does not address device compromise, stolen signing keys, global traffic analysis, legal authorization policy, regulator governance, or KYC/PKI errors. It outputs an opened public key, not a legal identity.

4.5 Security Goals

ALRAS consists of setup and key-generation algorithms (Setup, KeyGen, RegKeyGen), signing and adaptation algorithms (PreSign, PreVerify, Adapt, Verify, Extract, Link, PairVerify), and threshold-opening algorithms (OpenShare, OpenCombine, Judge). We group the required properties into functionality, privacy, and accountability. Section 6 proves these goals through five modular theorem statements.

Functionality. Correctness requires that honestly generated transcripts verify, adaptation with a valid witness produces an accepted completed transcript, extraction recovers the adaptor witness, same-scope transcripts by the same signer link, and valid threshold opening shares make Judge output the hidden signer public key.

Privacy. Signer anonymity before opening requires that, before a valid threshold opening and against fewer than t corrupted regulators, the transcript hides the signer index among the ring members except for allowed leakage. Threshold opening privacy requires that fewer than t regulators cannot recover the encrypted signer public key. Cross-scope unlinkability requires that transcripts under different scopes cannot be linked to the same signer except with negligible probability.

Accountability. Pair-consistent opening soundness requires that any public key accepted by Judge corresponds to the hidden signer of the pre-signature that is pair-consistent with the completed signature. Non-frameability requires that malicious users and corrupted regulators cannot make Judge output the public key of an honest user who did not participate in the accepted transcript pair. Judge-verifiable opening requires that all accepted openings are publicly checkable from the transcript pair and opening certificate.

Building-block boundary. The base LRAS is assumed to provide index hiding, witness extractability, scoped linkability, and the pair-verification property required by the compiler. The explicit tag $\tau_{\mathsf{link}} = sk_j H_{\mathsf{tag}}(sid)$ also requires scoped-tag privacy: for a fresh scope, the tuple $(G, pk_j, H_{\mathsf{tag}}(sid), \tau_{\mathsf{link}})$ should not identify j inside R beyond allowed same-scope equality leakage. The public PairVerify interface is assumed pair-sound and index-hiding: it should accept only when σ_R is a valid adaptation of $\hat{\sigma}_R$ under some witness w satisfying $Y = wG$, and running the check should not reveal the hidden signer index. These are building-block assumptions, not separate ALRAS security claims.

5 Detailed Construction

In this section, we combine the building blocks introduced in Section 3 into ALRAS. Figure 2 gives the two-path protocol workflow used throughout this section. The upper panel shows the normal anonymous swap path: Alice sends an augmented pre-signature, the witness w is revealed on the claim path, and Bob adapts the transcript to complete the other side. The lower panel shows the dispute opening path: the stored transcript pair is opened by threshold regulators and verified by a public judge. We then define the transcript objects, the consistency relation, and the generic algorithms.

5.1 Construction Overview

The sequence in Figure 2 separates chain-level swap events from ALRAS signing-layer checks. The normal path explains how $\hat{\Sigma}$ becomes Σ. The dispute path explains how τ_{open} is checked and how pk^* is obtained.

5.2 Transcript Objects and Consistency Relation

The transcript format separates the base LRAS object from the accountability material added by ALRAS. Figure 3 shows that the opening ciphertext, scoped link tag, and consistency proof are created with the pre-signature and then copied unchanged into the completed transcript.

Augmented pre-signature. An ALRAS pre-signature is

$$\hat{\Sigma} = (\hat{\sigma}_R, C_{\mathsf{open}}, \tau_{\mathsf{link}}, \pi_{\mathsf{cons}}),$$

where $\hat{\sigma}_R$ is the base LRAS pre-signature, C_{open} is a threshold-encrypted opening ciphertext, τ_{link} is a scoped link tag, and π_{cons} proves that these values are bound to the same hidden signer.

Completed transcript. A completed transcript is

$$\Sigma = (\sigma_R, C_{\mathsf{open}}, \tau_{\mathsf{link}}, \pi_{\mathsf{cons}}),$$

where σ_R is the full LRAS signature obtained by adapting $\hat{\sigma}_R$ with witness w. The auxiliary fields are copied unchanged from $\hat{\Sigma}$.

Opening certificate. An opening certificate is

$$\tau_{\mathsf{open}} = (pk^*, S, \{(l, D_l, \pi_l^{\mathsf{dec}})\}_{l \in S}),$$

where $|S| \geq t$, each D_l is a decryption share, each π_l^{dec} proves share correctness, and pk^* is the opened public key.

Consistency relation. The opening ciphertext is not an auxiliary attachment. It must encrypt the same public key that generated the LRAS pre-signature and scoped link tag. For public statement

$$stmt = (R, m, Y, sid, Y_R, \hat{\sigma}_R, C_{\mathsf{open}}, \tau_{\mathsf{link}})$$

Table 3. ALRAS transcript objects.

Symbol	Meaning
$\hat{\sigma}_R$	Base LRAS pre-signature
σ_R	Base LRAS full signature
$\hat{\Sigma}$	ALRAS augmented pre-signature
Σ	ALRAS completed transcript
C_{open}	Threshold opening ciphertext
τ_{link}	Scoped link tag
π_{cons}	NIZK consistency proof
π_I^{dec}	DLEQ proof for regulator share
τ_{open}	Opening certificate

Table 4. Decomposition of the ALRAS consistency proof.

Subproof	Statement
π_{key}	prove $pk_j = sk_j G$ and $pk_j \in R$
π_{tag}	prove $\tau_{\mathsf{link}} = sk_j H_{\mathsf{tag}}(sid)$
π_{enc}	prove $C_{\mathsf{open}} = (rG, pk_j + rY_R)$
π_{pre}	prove the LRAS pre-signature relation for the same hidden index

and witness (j, sk_j, ρ, r), the relation $\mathcal{R}_{\mathsf{cons}}$ holds iff

$$\mathcal{R}_{\mathsf{cons}} = \mathcal{R}_{\mathsf{key}} \wedge \mathcal{R}_{\mathsf{tag}} \wedge \mathcal{R}_{\mathsf{enc}} \wedge \mathcal{R}^{\mathsf{pre}}_{\mathsf{LRAS}}.$$

$$\begin{aligned}
\mathcal{R}_{\mathsf{key}} : &\quad pk_j = sk_j G, \quad pk_j \in R, \\
\mathcal{R}_{\mathsf{tag}} : &\quad \tau_{\mathsf{link}} = sk_j H_{\mathsf{tag}}(sid), \\
\mathcal{R}_{\mathsf{enc}} : &\quad C_{\mathsf{open}} = (rG, pk_j + rY_R), \\
\mathcal{R}^{\mathsf{pre}}_{\mathsf{LRAS}} : &\quad (R, m, Y, sid, \hat{\sigma}_R; j, sk_j, \rho) \in \mathcal{R}^{\mathsf{LRAS}}_{\mathsf{pre}}.
\end{aligned}$$

The relation prevents false opening and ciphertext substitution: an adversary cannot replace the opening ciphertext, tag, or pre-signature without invalidating π_{cons}. Pair consistency with the completed signature is enforced separately by PairVerify.

5.3 Generic ALRAS Construction

The compiler ALRAS $\leftarrow$ Compile$(\Pi_{\mathsf{LRAS}}, \Pi_{\mathsf{TEnc}}, \Pi_{\mathsf{NIZK}})$ transforms the building blocks into a complete accountable signing layer. The compiler exposes the following interfaces:

Setup, KeyGen, RegKeyGen, PreSign, PreVerify, Adapt, Verify, Extract, Link,

PairVerify, OpenShare, OpenCombine, Judge.

These interfaces have different origins. KeyGen, PreSign, PreVerify, Adapt, Verify, Extract, and the background Link functionality are inherited from the base LRAS abstraction; PairVerify is an explicit compiler requirement on the base primitive; and RegKeyGen, the opening ciphertext, scoped tag, consistency proof, opening-share procedures, and Judge are added by the ALRAS accountability compiler. Table 5 summarizes the interface groups. We combine simple interfaces in shared algorithm boxes to avoid repetition.

Table 5. ALRAS interface summary.

Layer	Interfaces
Setup	Setup, KeyGen, and RegKeyGen generate public parameters, user keys, and regulator threshold keys.
Normal swap	PreSign, PreVerify, Adapt, Verify, and Extract create, validate, complete, and consume the adaptor transcript.
Privacy	Link compares scoped link tags inside one linkability domain.
Accountability	PairVerify, OpenShare, OpenCombine, and Judge bind a pre/full pair and verify threshold opening.

Algorithm 1 ALRAS.Setup, ALRAS.KeyGen, and ALRAS.RegKeyGen

Require: security parameter 1^λ, regulator number N, threshold t
Ensure: public parameters pp, user key pair (pk, sk), regulator keys $(Y_R, \{(sk_l, Y_l)\}_{l=1}^N)$

$pp_{\mathsf{LRAS}} \leftarrow \mathsf{LRAS.Setup}(1^\lambda)$
$pp_{\mathsf{TEnc}} \leftarrow \mathsf{TEnc.Setup}(1^\lambda)$
$crs_{\mathsf{cons}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda, \mathcal{R}_{\mathsf{cons}})$
$pp \leftarrow (pp_{\mathsf{LRAS}}, pp_{\mathsf{TEnc}}, crs_{\mathsf{cons}}, H, H_{\mathsf{tag}})$
$(pk, sk) \leftarrow \mathsf{LRAS.KeyGen}(pp_{\mathsf{LRAS}})$
$(Y_R, \{(sk_l, Y_l)\}_{l=1}^N) \leftarrow \mathsf{RegKeyGen}(pp_{\mathsf{TEnc}}, N, t)$
return $pp, (pk, sk), (Y_R, \{(sk_l, Y_l)\}_{l=1}^N)$

Algorithm 2 ALRAS.PreSign

Require: public parameters pp, secret key sk_j, index j, ring R, message m, statement Y, scope sid, regulator key Y_R
Ensure: augmented pre-signature $\hat{\Sigma}$

parse pp to obtain crs_{cons}
sample LRAS randomness $\rho \xleftarrow{\$} \mathbb{Z}_q^*$ and opening randomness $r \xleftarrow{\$} \mathbb{Z}_q^*$
$\hat{\sigma}_R \leftarrow \mathsf{LRAS.PreSign}(sk_j, R, m, Y, sid; \rho)$
$pk_j \leftarrow sk_j G$
$C_{\mathsf{open}} \leftarrow \mathsf{TEnc.Enc}(Y_R, pk_j; r)$
$\tau_{\mathsf{link}} \leftarrow sk_j H_{\mathsf{tag}}(sid)$
$stmt \leftarrow (R, m, Y, sid, Y_R, \hat{\sigma}_R, C_{\mathsf{open}}, \tau_{\mathsf{link}})$
$\pi_{\mathsf{cons}} \leftarrow \mathsf{NIZK.Prove}(crs_{\mathsf{cons}}, stmt, (j, sk_j, \rho, r))$
return $\hat{\Sigma} = (\hat{\sigma}_R, C_{\mathsf{open}}, \tau_{\mathsf{link}}, \pi_{\mathsf{cons}})$

Link. Although the base LRAS abstraction includes linkability, ALRAS exposes linking through the explicit scoped tag. On input two completed transcripts and a scope sid, Link parses the tags $\tau_{\mathsf{link},1}$ and $\tau_{\mathsf{link},2}$ and returns 1 iff the two transcripts are associated with the same scope and the tags are equal.

Correctness. For honestly generated parameters, keys, and transcripts, correctness follows by composition. The LRAS pre-signature verifies, the NIZK proof accepts for the honest witness (j, sk_j, ρ, r), adaptation produces a full LRAS signature paired with the pre-signature, and the auxiliary fields remain unchanged between $\hat{\Sigma}$ and Σ. Threshold ElGamal correctness and DLEQ share verification ensure that any t honest opening shares combine to the encrypted public key, so Judge outputs the hidden signer's public key.

Algorithm 3 ALRAS.PreVerify, ALRAS.Adapt, and ALRAS.Extract

procedure PreVerify($pp, R, m, Y, sid, Y_R, \hat{\Sigma}$)
  parse pp to obtain crs_{cons}
  parse $\hat{\Sigma} = (\hat{\sigma}_R, C_{\text{open}}, \tau_{\text{link}}, \pi_{\text{cons}})$
  if LRAS.PreVerify$(R, m, Y, sid, \hat{\sigma}_R) = 0$ **then**
    return 0
  end if
  $stmt \leftarrow (R, m, Y, sid, Y_R, \hat{\sigma}_R, C_{\text{open}}, \tau_{\text{link}})$
  if NIZK.Verify$(crs_{\text{cons}}, stmt, \pi_{\text{cons}}) = 0$ **then**
    return 0
  end if
  return 1
end procedure
procedure Adapt($pp, \hat{\Sigma}, w$)
  parse $\hat{\Sigma} = (\hat{\sigma}_R, C_{\text{open}}, \tau_{\text{link}}, \pi_{\text{cons}})$
  $\sigma_R \leftarrow$ LRAS.Adapt$(\hat{\sigma}_R, w)$
  return $\Sigma = (\sigma_R, C_{\text{open}}, \tau_{\text{link}}, \pi_{\text{cons}})$
end procedure
procedure Extract($pp, R, m, Y, sid, Y_R, \hat{\Sigma}, \Sigma$)
  if Verify$(pp, R, m, Y, sid, Y_R, \hat{\Sigma}, \Sigma) = 0$ **then**
    return $\perp$
  end if
  parse $\hat{\Sigma}$ and Σ to obtain $(\hat{\sigma}_R, \sigma_R)$
  return LRAS.Extract$(\hat{\sigma}_R, \sigma_R)$
end procedure

5.4 Accountable Anonymous Swap Workflow

The algorithms above define the signing layer. A cross-chain atomic swap uses this layer as follows; timeout and refund logic is enforced by the underlying chains, not by ALRAS.

Setup and pre-signature exchange. Alice and Bob agree on the two ledgers, assets, ring R, adaptor statement $Y = wG$, scope sid, and regulator public key Y_R. Alice runs PreSign to produce $\hat{\Sigma}$ and sends it to Bob. Bob accepts the pre-signature only if PreVerify returns 1.

Normal claim and adaptation. Bob locks or authorizes his side of the swap. Alice claims Bob's asset on the second chain, revealing w. Bob then runs Adapt to obtain Σ and uses the completed transcript to claim Alice's asset on the first chain.

Dispute opening. If the completed swap later becomes disputed, the pre-signature $\hat{\Sigma}$ and completed transcript Σ are retrieved. At least t regulators run OpenShare and publish DLEQ-verifiable shares. Anyone can run OpenCombine to form an opening certificate and then run Judge to obtain the opened public key pk^*.

5.5 Candidate Instantiation Blueprint

The generic construction above treats the base LRAS as an abstract pair-verifiable, index-hiding primitive. This is intentional: the accountability layer is independent of a particular LRAS implementation. The following blueprint demonstrates how the added opening and consistency components can be instantiated with standard group operations.

Algorithm 4 ALRAS.Verify with ALRAS.PairVerify

procedure PairVerify($pp, R, Y, \hat{\Sigma}, \Sigma$)
  parse $\hat{\Sigma} = (\hat{\sigma}_R, C, \tau, \pi)$ and $\Sigma = (\sigma_R, C', \tau', \pi')$
  if $(C, \tau, \pi) \neq (C', \tau', \pi')$ **then**
    return 0
  end if
  return LRAS.PairVerify$(R, \hat{\sigma}_R, \sigma_R, Y)$
end procedure
procedure Verify($pp, R, m, Y, sid, Y_R, \hat{\Sigma}, \Sigma$)
  parse $\hat{\Sigma} = (\hat{\sigma}_R, C, \tau, \pi)$ and $\Sigma = (\sigma_R, C', \tau', \pi')$
  if PreVerify$(pp, R, m, Y, sid, Y_R, \hat{\Sigma}) = 0$ **then**
    return 0
  end if
  if LRAS.Verify$(R, m, sid, \sigma_R) = 0$ **then**
    return 0
  end if
  if PairVerify$(pp, R, Y, \hat{\Sigma}, \Sigma) = 0$ **then**
    return 0
  end if
  return 1
end procedure

Algorithm 5 ALRAS.OpenShare and ALRAS.OpenCombine

procedure OpenShare(pp, sk_l, Y_l, Σ)
  parse Σ to obtain $C_{\mathsf{open}} = (C_1, C_2)$
  $D_l \leftarrow sk_l C_1$
  $\pi_l^{\mathsf{dec}} \leftarrow$ DLEQ.Prove$(Y_l, C_1, D_l; sk_l)$
  return $(l, D_l, \pi_l^{\mathsf{dec}})$
end procedure
procedure OpenCombine($pp, \Sigma, \{(l, D_l, \pi_l^{\mathsf{dec}})\}_{l \in S}$)
  parse Σ to obtain $C_{\mathsf{open}} = (C_1, C_2)$
  if $|S| < t$ **then**
    return $\perp$
  end if
  for all $l \in S$ **do**
    if DLEQ.Verify$(Y_l, C_1, D_l, \pi_l^{\mathsf{dec}}) = 0$ **then**
      return $\perp$
    end if
  end for
  compute interpolation coefficients $\{\lambda_l^S\}_{l \in S}$
  $pk^* \leftarrow C_2 - \sum_{l \in S} \lambda_l^S D_l$
  return $\tau_{\mathsf{open}} = (pk^*, S, \{(l, D_l, \pi_l^{\mathsf{dec}})\}_{l \in S})$
end procedure

Threshold opening. We instantiate Π_{TEnc} using threshold ElGamal over $\mathbb{G}$. The ciphertext encrypting pk_j is

$$C_{\mathsf{open}} = (C_1, C_2) = (rG, pk_j + rY_R).$$

Algorithm 6 ALRAS.Judge

Require: public inputs (pp, R, m, Y, sid, Y_R), pre-signature $\hat{\Sigma}$, completed transcript Σ, certificate τ_{open}
Ensure: opened public key pk^* or $\perp$
if $\mathsf{Verify}(pp, R, m, Y, sid, Y_R, \hat{\Sigma}, \Sigma) = 0$ **then**
  return $\perp$
end if
parse $\tau_{\mathsf{open}} = (pk^*, S, \{(l, D_l, \pi_l^{\mathrm{dec}})\}_{l \in S})$
if $|S| < t$ or $pk^* \notin R$ **then**
  return $\perp$
end if
parse Σ to obtain $C_{\mathsf{open}} = (C_1, C_2)$
for all $l \in S$ **do**
  if $\mathsf{DLEQ.Verify}(Y_l, C_1, D_l, \pi_l^{\mathrm{dec}}) = 0$ **then**
    return $\perp$
  end if
end for
compute interpolation coefficients λ_l^S for S
if $pk^* \neq C_2 - \sum_{l \in S} \lambda_l^S D_l$ **then**
  return $\perp$
end if
return pk^*

Each regulator l computes $D_l = sk_l C_1$ and proves by a Chaum–Pedersen DLEQ proof that $\log_G(Y_l) = \log_{C_1}(D_l)$. Given at least t valid shares, the combiner interpolates the partial decryptions and recovers pk_j.

NIZK consistency proof. The relation $\mathcal{R}_{\mathsf{cons}}$ is expressed as a conjunction of discrete-logarithm knowledge statements and linear relations over $\mathbb{G}$, all of which admit sigma protocols [10]. The Fiat–Shamir challenge hashes the full statement, including the opening ciphertext, link tag, LRAS pre-signature, adaptor statement, and scope, so that the consistency proof covers the whole pre-signature transcript.

Base LRAS boundary. Our compiler requires the base LRAS to provide index hiding, witness extractability, scoped linkability, and public pair verification. A concrete OR-proof-based LRAS may be used as a blueprint [10, 27], but proving a fully optimized production LRAS instantiation is orthogonal to the accountability compiler. The key requirement is that PairVerify accepts exactly when σ_R is a valid adaptation of $\hat{\sigma}_R$ under statement Y and does so without revealing the signer index.

Scope of the instantiation. The instantiation above demonstrates that the accountability additions of ALRAS can be realized using standard group operations, threshold ElGamal encryption, DLEQ proofs, and Fiat–Shamir-transformed sigma protocols. A production deployment should instantiate the base LRAS with a concrete pair-verifiable and index-hiding construction and prove the required LRAS properties for that implementation.

6 Security Analysis

This section proves the five core properties stated in Section 4.5: correctness, privacy before opening, pair-consistent accountability, non-frameability with witness coupling, and scoped linkability. The proofs establish the cryptographic claim of ALRAS: normal executions preserve LRAS-style anonymity and witness coupling, while disputed completed

transcript pairs can be opened only through a threshold-governed and publicly verifiable procedure. The corresponding cost of this accountability layer is analyzed separately in Section 7.

Proof roadmap. We prove the five core properties required by the model in Section 4.5. The proofs are modular proof sketches: each violation of an ALRAS property is mapped to a violation of one of the stated building-block assumptions. The analysis relies on LRAS correctness, signer anonymity, witness extractability, scoped-tag privacy, pair-soundness, threshold ElGamal privacy and correctness with DLEQ-verifiable shares, and NIZK completeness, zero knowledge, soundness, and simulation extractability where non-frameability is considered.

Theorem 6.1 (Correctness). *Let* $pp \leftarrow \mathsf{Setup}(1^\lambda)$, $(pk, sk) \leftarrow \mathsf{KeyGen}(pp)$, *and let* R *be any well-formed ring containing* pk. *For any message* m, *adaptor statement* $Y = wG$, *scope* sid, *and regulator key* $Y_R \leftarrow \mathsf{RegKeyGen}(pp, N, t)$, *let*

$$\hat{\Sigma} \leftarrow \mathsf{PreSign}(sk, R, m, Y, sid, Y_R), \qquad \Sigma \leftarrow \mathsf{Adapt}(pp, \hat{\Sigma}, w).$$

Then PreVerify *and* Verify *accept,* PairVerify *accepts the pair* $(\hat{\Sigma}, \Sigma)$, Extract *returns* w, *and any set of at least* t *honest opening shares yields an opening certificate* τ_{open} *such that*

$$\mathsf{Judge}(pp, R, m, Y, sid, Y_R, \hat{\Sigma}, \Sigma, \tau_{\mathsf{open}}) = pk.$$

Proof sketch. The proof follows the algorithms. PreVerify accepts because the base LRAS pre-signature is honestly generated and the NIZK proof is complete for the witness (j, sk_j, ρ, r). Adapt invokes the base LRAS adaptation algorithm with witness w, so the resulting full signature verifies. PairVerify accepts because the full signature is honestly adapted from the same pre-signature under Y. Extract returns w by the witness-extraction correctness of the base LRAS. Finally, each honest regulator share is accompanied by a valid DLEQ proof, and threshold ElGamal correctness reconstructs the plaintext of C_{open}. By the consistency relation, this plaintext is the same public key pk_j that generated the pre-signature and link tag; hence Judge outputs pk_j. □

Theorem 6.2 (Privacy Before Opening). *Assume that* Π_{LRAS} *satisfies signer anonymity, the exposed scoped tag satisfies scoped-tag privacy,* Π_{TEnc} *is IND-CPA secure against adversaries corrupting fewer than* t *regulators, and* Π_{NIZK} *is zero-knowledge. Then, before a valid threshold opening, any PPT adversary corrupting fewer than* t *regulators learns no more about the signer index than the allowed leakage of* R, m, Y, sid, Y_R, *transcript size, and same-scope linkability.*

Proof sketch. Consider the adversary's view of an augmented transcript $(\hat{\sigma}_R, C_{\mathsf{open}}, \tau_{\mathsf{link}}, \pi_{\mathsf{cons}})$. First replace π_{cons} with a simulated proof; zero knowledge guarantees indistinguishability. Next replace C_{open}, which encrypts the signer's public key, with an encryption of a dummy group element; this is indistinguishable by threshold ElGamal IND-CPA security because the adversary controls fewer than t regulators. Third, scoped-tag privacy ensures that τ_{link} reveals only equality within the same scope and does not identify the signer index inside R. The remaining view is the base LRAS transcript, so signer privacy reduces to LRAS signer anonymity. □

Theorem 6.3 (Pair-Consistent Accountability). *Assume that* Π_{NIZK} *is sound for* $\mathcal{R}_{\mathsf{cons}}$, *threshold opening is correct, threshold decryption shares are DLEQ-sound, and the base LRAS satisfies pair-soundness. If* Judge *accepts* $(\hat{\Sigma}, \Sigma, \tau_{\mathsf{open}})$ *and outputs* pk^*, *then* pk^* *is the public key bound by* $\mathcal{R}_{\mathsf{cons}}$ *to the accepted LRAS pre-signature, and* Σ *is pair-consistent with* $\hat{\Sigma}$ *under* Y.

Proof sketch. Suppose Judge accepts. Then Verify and PairVerify accept, all DLEQ proofs for the opening shares verify, and the interpolation check confirms that pk^* is the plaintext of C_{open}. Since PreVerify accepts, π_{cons} verifies

for the statement containing $(R, m, Y, sid, Y_R, \hat{\sigma}_R, C_{\mathsf{open}}, \tau_{\mathsf{link}})$. By NIZK soundness, this accepted statement belongs to $\mathcal{R}_{\mathsf{cons}}$. Hence C_{open} encrypts the same public key $pk_j = sk_jG$ whose secret key generated the link tag and the LRAS pre-signature. The accepted PairVerify check rules out replacing the completed signature with one adapted from another pre-signature. Therefore the opened key is bound to the same hidden signer of the accepted pre/full transcript pair. □

Theorem 6.4 (Non-Frameability and Witness Coupling). *Assume simulation extractability or knowledge soundness for the consistency proof when needed, LRAS unforgeability and witness extractability, LRAS pair-soundness, and DLEQ soundness. Even if the adversary obtains enough valid opening shares, it cannot make* Judge *output the public key of an honest user who did not participate in the accepted pre/full transcript pair. Moreover, every accepted pre/full pair yields a witness w satisfying $Y = wG$.*

Proof sketch. For non-frameability, suppose Judge outputs an honest public key pk_h for a transcript not produced by that honest user. This is different from unauthorized opening: if at least t regulators cooperate, they can open a real transcript, but the output is still the key encrypted and proved inside that transcript. A frame requires the accepted transcript pair to be produced under one signing key while opening to a different honest nonparticipant. Since Judge accepts, the transcript pair verifies, the opening shares verify, and the consistency proof binds C_{open}, τ_{link}, and the LRAS pre-signature to one hidden secret key. Outputting pk_h therefore requires using sk_h in the pre-signature relation, forging an LRAS transcript under pk_h, or making an invalid opening certificate verify. These possibilities contradict the stated assumptions. For witness coupling, Verify and PairVerify imply that $(\hat{\Sigma}, \Sigma)$ contains a valid LRAS pre/full pair under Y, so Extract returns a witness w with $Y = wG$. □

Theorem 6.5 (Scoped Linkability). *For a fixed scope sid, honestly generated transcripts by the same signer carry the same link tag. Transcripts by different signers collide only with negligible probability. For distinct scopes $sid_1 \neq sid_2$, the resulting tags are unlinkable beyond the allowed same-scope leakage under scoped-tag privacy and hash-to-group independence.*

Proof sketch. The link tag is $\tau_{\mathsf{link}} = sk_jH_{\mathsf{tag}}(sid)$. For a fixed scope, the same secret key deterministically produces the same tag. For different secret keys, tag equality implies a collision or nontrivial relation in the group except with negligible probability. For different scopes, $H_{\mathsf{tag}}(sid_1)$ and $H_{\mathsf{tag}}(sid_2)$ are modeled as independent group elements, and scoped-tag privacy prevents using the two tags to identify the same public key across scopes. □

Security boundary. The theorems above establish signature-level accountable privacy: witness coupling, anonymity before opening, pair-consistent threshold opening, and public verification of the opened key. They do not establish ledger-level fairness by themselves. Timeout and refund recovery are enforced by the two underlying chains. Opening authorization, transcript availability, regulator-key management, and public-key-to-identity mapping are deployment assumptions outside the ALRAS cryptographic claim.

7 Performance Analysis

This section analyzes the communication and computation costs of adding pair-consistent accountable opening to an LRAS-style anonymous adaptor-signature path. The purpose is to quantify incremental accountability overhead, not full-system throughput. ALRAS is therefore compared with a non-accountable LRAS/LTRAS-style lifecycle, which has the same pre-sign/adapt/extract/verify workflow but lacks opening ciphertexts, scoped tags, consistency binding, public pair verification, and threshold opening.

Table 6. Output sizes of ALRAS accountability components.

Component	Size	Path	Security role
C_{open}	$2\lvert\mathbb{G}\rvert$	normal/dispute	Enables threshold opening
τ_{link}	$\lvert\mathbb{G}\rvert$	normal	Provides scoped linkability
π_{cons}	$\lvert\pi_{\mathsf{cons}}(n)\rvert$	normal	Binds signer, ciphertext, tag, and pre-signature
D_l	$\lvert\mathbb{G}\rvert$ per share	dispute	Partial decryption share
π_l^{dec}	$\lvert\pi_{\mathsf{DLEQ}}\rvert$ per share	dispute	Makes shares publicly verifiable

Table 7. Transcript sizes in KB and ALRAS/LRAS overhead ratios for representative ring sizes.

n	LRAS	C_{open}	π_{cons}	τ_{link}	ALRAS	Overhead
8	0.94	0.06	1.50	0.03	2.53	2.70×
64	6.19	0.06	10.25	0.03	16.53	2.67×
512	48.19	0.06	80.25	0.03	128.53	2.67×

We do not use HTLC swaps, Zerocash-style ledgers, Zether-style systems, LedgerMaze, or payment-channel systems as runtime baselines because they operate at the transaction-system layer rather than the signing layer. The relevant question is instead how much extra transcript and verification cost is paid to obtain pair-consistent accountable opening while preserving the normal LRAS-style execution path.

7.1 Communication Cost

The communication overhead comes from two different paths. The normal path carries the base LRAS transcript plus accountability material that binds the hidden signer to the opening ciphertext and scoped link tag. The dispute path additionally carries regulator opening shares and their DLEQ proofs.

Let $n = |R|$ be the ring size, $s = |S|$ be the number of accepted opening shares, and $|\mathbb{G}|$ be the encoded size of one group element. If $|\hat{\sigma}_R(n)|$ and $|\sigma_R(n)|$ denote the base LRAS pre-signature and full-signature sizes, then

$$|\hat{\Sigma}| = |\hat{\sigma}_R(n)| + 2|\mathbb{G}| + |\mathbb{G}| + |\pi_{\mathsf{cons}}(n)|$$

and

$$|\Sigma| = |\sigma_R(n)| + 2|\mathbb{G}| + |\mathbb{G}| + |\pi_{\mathsf{cons}}(n)|.$$

The opening certificate used only in a dispute has size

$$|\tau_{\mathsf{open}}(s)| = |\mathbb{G}| + s\big(|\mathsf{id}_l| + |\mathbb{G}| + |\pi_{\mathsf{DLEQ}}|\big).$$

Thus C_{open} and τ_{link} are constant-size normal-path additions, $\pi_{\mathsf{cons}}(n)$ is the dominant transcript-binding term, and τ_{open} grows linearly with the number of accepted regulator shares.

Table 7 illustrates this decomposition for representative ring sizes generated by the reference artifact. In the reported reference settings, the overhead ratio remains relatively stable because both the base LRAS transcript and the modeled consistency proof scale with the ring size, while the opening ciphertext and scoped tag remain constant-size additions. This is the intended trade-off: ALRAS pays additional transcript size to make later opening meaningful, not merely to attach an unrelated encrypted identity.

Table 8. Computation profile of ALRAS relative to a non-accountable LRAS baseline.

Procedure	Added cost over LRAS	Driver	Path
PreSign	TEnc.Enc + HashToGroup + $P_{\mathsf{cons}}(n)$	proof relation over R	normal
PreVerify	$V_{\mathsf{cons}}(n)$	proof verification	normal
Adapt / Extract	none beyond base LRAS	base lifecycle	normal
Verify	$V_{\mathsf{cons}}(n)$	proof verification	normal
PairVerify	$C_{\mathsf{pair}}(n)$	pre/full pair relation	normal/dispute
OpenShare	$C_{\mathsf{share}} + P_{\mathsf{DLEQ}}$	one regulator share	dispute
OpenCombine	$sV_{\mathsf{DLEQ}} + I(s)$	accepted shares	dispute
Judge	Verify+PairVerify+OpenCombine+C_{mem}	n and s	dispute

Table 9. Representative computation costs from the shared primitive calibration. Times are median milliseconds.

Procedure	Setting	Time (ms)	Path	Role
LRAS PreSign	$n = 64$	572.5	normal	non-accountable baseline
ALRAS PreSign	$n = 64$	1338.0	normal	adds ciphertext, tag, and consistency proof
ALRAS Verify	$n = 64$	949.4	normal	checks transcript binding
PairVerify	$n = 64$	205.0	normal/dispute	checks pre/full pair consistency
OpenShare	$(N, t) = (7, 4)$	93.1	dispute	one regulator share
OpenCombine	$(N, t) = (7, 4)$	148.2	dispute	combines accepted shares
Judge	$(N, t) = (7, 4)$	266.7	dispute	checks transcript and opening certificate

7.2 Computation Cost

The computation cost separates into normal-path signing checks and dispute-path opening checks. Let $P_{\mathsf{cons}}(n)$ and $V_{\mathsf{cons}}(n)$ be the consistency-proof generation and verification costs, $C_{\mathsf{pair}}(n)$ be the public pair-verification cost, P_{DLEQ} and V_{DLEQ} be DLEQ proof generation and verification costs, and $I(s)$ be interpolation over s shares.

Table 8 gives the main cost profile. Ordinary swaps do not execute OpenShare, OpenCombine, or the threshold-opening part of Judge. They pay only the normal-path accountability layer: encryption of the opening target, scoped-tag computation, consistency-proof generation or verification, and public pair verification. The dispute path pays the threshold-opening work, which is independent of the ring size per regulator and linear in the number of accepted shares during combination.

Table 9 instantiates the symbolic model at a representative setting. At $n = 64$, ALRAS pre-signing is larger than LRAS pre-signing because it includes the opening ciphertext, scoped tag, and modeled consistency proof. PairVerify is separated from Verify to show the explicit price of pre/full transcript binding. For $(N, t) = (7, 4)$, OpenShare is a per-regulator dispute operation, while OpenCombine and Judge are paid only when a disputed transcript is opened.

Each added operation corresponds to a specific attack surface. Without C_{open}, no public opening target exists. Without π_{cons}, the ciphertext or tag may be unrelated to the hidden signer. Without PairVerify, a completed signature from another execution can be spliced into the disputed pair. Without DLEQ-verified shares, a judge cannot publicly reject malformed opening shares. The extra computation is therefore not an accidental implementation artifact; it is the cost of pair-consistent accountable opening.

Table 10. Same-layer baselines and accountability ablations.

Variant	Normal-path payload	Dispute opening	Interpretation
B0: LRAS/LTRAS baseline	base pre/full signatures	none	no accountable opening
B1: LRAS + C_{open}	base transcript plus ciphertext	decrypts a candidate key	ciphertext may be unrelated to signer
B2: bound transcript without PairVerify	ciphertext, tag, consistency proof	opens the pre-signature target	pre/full splicing remains possible
B3: full ALRAS	ciphertext, tag, proof, PairVerify	DLEQ-verified threshold opening	pair-consistent accountable opening

Table 11. End-to-end sanity benchmarks. Times validate workflow wiring rather than optimized deployment throughput.

Scenario	(N, t)	n	PreSign	Adapt/Extract	OpenShare	OpenCombine+Judge	End-to-end	95% CI
small reference	(5,3)	8	209.4	0.0	164.5	860.0	1559.1	65.9
medium deployment	(7,4)	64	196.7	0.0	164.9	1090.1	1946.5	24.2
large anonymity	(7,4)	128	202.9	0.0	169.5	1133.6	2014.8	86.8

7.3 Reference Evaluation

We implement a Python reference artifact to calibrate the symbolic cost model and validate the transcript lifecycle. The artifact evaluates ring sizes $n \in \{8, 16, 32, 64, 128, 256, 512\}$ and regulator thresholds $(N, t) \in \{(5, 3), (7, 4), (10, 7)\}$. The measurements are calibrated reference results rather than optimized deployment throughput; implementation details and reproduction commands are given in Appendix A.

Table 10 gives the same-layer comparison that is meaningful for ALRAS. A non-accountable LRAS baseline is the right normal-path reference because it already has ring anonymity and the adaptor lifecycle. Adding only an opening ciphertext gives a decryptable value but does not prove that the value belongs to the signer. Binding the ciphertext and tag to the pre-signature removes ciphertext substitution, but without PairVerify a full signature from another execution may still be spliced into the disputed pair. Full ALRAS pays for all four pieces because accountable opening is useful only when the opened key is bound to the accepted pre/full transcript pair.

The two scaling parameters are the ring size n and the number of accepted opening shares s. A larger n improves signer anonymity but increases the LRAS and consistency-proof costs. A larger threshold strengthens opening governance but increases dispute-path share verification and interpolation. Figure 4 shows that these costs are separated: n-dependent costs appear in the normal signing path, whereas s-dependent threshold costs are activated only after a dispute trigger. The end-to-end panel is an integration sanity check for the two-path workflow, not the main throughput claim.

Table 11 reports representative workflow sanity timings. These measurements exercise pre-signature generation, adaptation or extraction, regulator share generation, opening combination, and public judging. They are useful for checking that the two-path workflow is executable, but they do not change the main performance claim: ordinary swaps pay transcript-binding and pair-check costs; regulator-side opening is activated only after a dispute trigger.

Finally, we run functional sanity checks to ensure that the executable transcript lifecycle matches the security motivation. These checks do not replace the proofs in Section 6; they confirm that the reference implementation rejects the concrete failure modes discussed in the model.

Overall, the performance results support the intended design trade-off. ALRAS does not aim to reduce the cost of LRAS-style anonymous adaptor signing. Instead, it adds a constant-size opening ciphertext, a scoped tag, a consistency proof, public pair verification, and dispute-only opening shares to obtain pair-consistent accountable opening. The dominant normal-path overhead is the consistency proof, while threshold opening is activated only after a dispute.

Table 12. Functional sanity checks for the executable transcript lifecycle.

Check	Expected outcome
honest execution	accept
ciphertext substitution	reject
transcript splicing	reject
invalid DLEQ share	reject
below-threshold opening	reject
same-scope linking	link

This separation explains why ordinary swaps preserve the LRAS-style scaling, whereas disputed executions pay the additional cost required for judge-verifiable accountability.

8 Conclusion

We presented an accountable privacy mechanism for anonymous cross-chain atomic swaps. Its normal path preserves ring anonymity and adaptor witness coupling, while its dispute path permits threshold-governed, publicly verifiable opening only for a completed and pair-consistent transcript. Four motivating attacks shaped the protocol requirements, and the workflow makes the separation between cryptographic accountability and chain-level recovery explicit.

The mechanism is built around ALRAS, a signing layer rather than a turnkey compliance system. We specified its interface and security goals, gave a generic compiler from a NIZK-compatible index-hiding LRAS, and described a candidate instantiation blueprint. The security analysis proves the intended anonymity, opening privacy, pair-consistent accountability, non-frameability, witness extractability, and scoped-linkability properties under the stated assumptions. The performance analysis shows that the normal path keeps the LRAS-style dependence on the ring size, while the threshold-opening work is paid only on the dispute path; the added transcript and verification costs are the explicit price of pair-consistent accountable opening.

The remaining work is concrete. A production deployment should select and prove a fully concrete LRAS instantiation, benchmark an optimized proof system, formalize ledger-level fairness together with timeout and refund rules, and define governance for opening authorization, transcript retention, regulator-key rotation, and public-key-to-identity mapping. These boundaries do not weaken the accountable privacy goal; they identify precisely which guarantees come from cryptography and which must be supplied by the surrounding system.

References

[1] Lukas Aumayr, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostakova, Matteo Maffei, Pedro Moreno-Sanchez, and Siavash Riahi. 2021. Generalized Channels from Limited Blockchain Scripts and Adaptor Signatures. In *Advances in Cryptology – ASIACRYPT 2021*. Springer. doi:10.1007/978-3-030-92075-3_22

[2] Michael Backes, Nico Döttling, Lucjan Hanzlik, Kamil Kluczniak, and Jonas Schneider. 2019. Ring Signatures: Logarithmic-Size, No Setup—from Standard Assumptions. In *Advances in Cryptology – EUROCRYPT 2019*. Springer. doi:10.1007/978-3-030-17659-4_10

[3] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. 2003. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. In *Advances in Cryptology – EUROCRYPT 2003*. Springer, 614–629.

[4] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. 2014. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *2014 IEEE Symposium on Security and Privacy*. IEEE, 459–474. doi:10.1109/SP.2014.36

[5] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. 2021. Foundations of Fully Dynamic Group Signatures. *Journal of Cryptology* 34 (2021). doi:10.1007/s00145-021-09386-7

[6] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. 2016. Short Accountable Ring Signatures Based on DDH. In *European Symposium on Research in Computer Security – ESORICS 2016*. Springer. doi:10.1007/978-3-319-45744-4_13

[7] David Chaum and Torben P. Pedersen. 1993. Wallet Databases with Observers. In *Advances in Cryptology – CRYPTO 1992*. Springer, 89–105. doi:10.1007/3-540-48071-4_7
[8] David Chaum and Eugène van Heyst. 1991. Group Signatures. In *Advances in Cryptology – EUROCRYPT 1991*. Springer, 257–265.
[9] Xiuyuan Chen, Junzuo Lai, Chao Lin, Xinyi Huang, and Debiao He. 2025. Cryptographic Primitives in Script-based and Scriptless Payment Channel Networks: A Survey. *Comput. Surveys* 57, 10, Article 264 (2025), 264:1–264:34 pages. doi:10.1145/3725846
[10] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. 1994. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In *Advances in Cryptology – CRYPTO 1994*. Springer. doi:10.1007/3-540-48658-5_19
[11] Taher ElGamal. 1985. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory* 31, 4 (1985), 469–472. doi:10.1109/TIT.1985.1057074
[12] Andreas Erwig, Sebastian Faust, Kristina Hostakova, Monosij Maitra, and Siavash Riahi. 2021. Two-Party Adaptor Signatures from Identification Schemes. In *Public-Key Cryptography – PKC 2021*. Springer. doi:10.1007/978-3-030-75245-3_17
[13] Amos Fiat and Adi Shamir. 1987. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology – CRYPTO 1986*. Springer, 186–194. doi:10.1007/3-540-47721-7_12
[14] Ashley Fraser and Elizabeth A. Quaglia. 2019. Protecting the Privacy of Voters: New Definitions and Constructions for Demonymizable Ring Signatures. In *Australasian Conference on Information Security and Privacy – ACISP 2019*. Springer. doi:10.1007/978-3-030-21548-4_17
[15] Eiichiro Fujisaki and Koutarou Suzuki. 2007. Traceable Ring Signature. In *Public Key Cryptography – PKC 2007*. Springer. doi:10.1007/978-3-540-71677-8_13
[16] Paul Gerhart, Dominique Schröder, Pratik Soni, and Sri Aravinda Krishnan Thyagarajan. 2024. Foundations of Adaptor Signatures. In *Advances in Cryptology – EUROCRYPT 2024*. Springer. doi:10.1007/978-3-031-58723-8_6
[17] Maurice Herlihy. 2018. Atomic Cross-Chain Swaps. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing (PODC)*. ACM, 245–254. doi:10.1145/3212734.3212736
[18] Amir Herzberg, Stanisław Jarecki, Hugo Krawczyk, and Moti Yung. 1995. Proactive Secret Sharing Or: How to Cope With Perpetual Leakage. In *Advances in Cryptology – CRYPTO 1995*. Springer. doi:10.1007/3-540-44750-4_27
[19] Soichiro Imoto, Yuichi Sudo, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. 2019. Atomic Cross-Chain Swaps with Improved Space and Local Time Complexity. In *International Conference on Networked Systems*. Springer.
[20] Markulf Kohlweiss and Ian Miers. 2015. Accountable Tracing Signatures. *Proceedings on Privacy Enhancing Technologies* 2015, 1 (2015), 228–247.
[21] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. 2004. Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups. In *Information Security and Privacy – ACISP 2004*. Springer. doi:10.1007/978-3-540-27800-9_28
[22] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. 2013. Zerocoin: Anonymous Distributed E-Cash from Bitcoin. In *2013 IEEE Symposium on Security and Privacy*. IEEE, 397–411. doi:10.1109/SP.2013.34
[23] Susil Kumar Mohanty and Somanath Tripathy. 2024. Flexichain: Flexible Payment Channel Network to Defend Against Channel Exhaustion Attack. *ACM Transactions on Privacy and Security* 27, 4, Article 30 (2024), 30:1–30:26 pages. doi:10.1145/3687476
[24] Shen Noether, Adam Mackenzie, and the Monero Research Lab. 2016. Ring Confidential Transactions. *Ledger* 1 (2016), 1–18. doi:10.5195/ledger.2016.34
[25] Rodrigo Otoni, Matteo Marescotti, Leonardo Alt, Patrick Eugster, Antti E. J. Hyvärinen, and Natasha Sharygina. 2023. A Solicitous Approach to Smart Contract Verification. *ACM Transactions on Privacy and Security* 26, 2, Article 15 (2023), 15:1–15:28 pages. doi:10.1145/3564699
[26] Ronald L. Rivest, Adi Shamir, and Yael Tauman. 2001. How to Leak a Secret. In *Advances in Cryptology – ASIACRYPT 2001*. Springer. doi:10.1007/3-540-45682-1_32
[27] Claus-Peter Schnorr. 1991. Efficient Signature Generation by Smart Cards. *Journal of Cryptology* 4, 3 (1991), 161–174. doi:10.1007/BF00196725
[28] Adi Shamir. 1979. How to Share a Secret. *Commun. ACM* 22, 11 (1979), 612–613. doi:10.1145/359168.359176
[29] Sri Aravinda Krishnan Thyagarajan and Giulio Malavolta. 2021. Lockable Signatures for Blockchains: Scriptless Scripts for All Signatures. In *2021 IEEE Symposium on Security and Privacy*. 937–954. doi:10.1109/SP40001.2021.00065
[30] Sri Aravinda Krishnan Thyagarajan, Giulio Malavolta, Fritz Schmid, and Dominique Schröder. 2022. Verifiable Timed Linkable Ring Signatures for Scalable Payments for Monero. In *Computer Security – ESORICS 2022*. Springer, 467–486. doi:10.1007/978-3-031-17146-8_23
[31] Xiaohu Wang, Chao Lin, Xinyi Huang, and Debiao He. 2024. Anonymity-Enhancing Multi-Hop Locks for Monero-Enabled Payment Channel Networks. *IEEE Transactions on Information Forensics and Security* 19 (2024), 2438–2453. doi:10.1109/TIFS.2023.3346177
[32] Rui Zhang, Rui Xue, and Ling Liu. 2019. Security and Privacy on Blockchain. *Comput. Surveys* 52, 3, Article 51 (2019), 51:1–51:34 pages. doi:10.1145/3316481

A Artifact and Reproducibility

The evaluation artifact is a self-contained Python repository under `artifact/`. It includes reference modules for group arithmetic, hashing, serialization, Schnorr/adaptor signatures, an LRAS/LTRAS-style structural baseline, threshold ElGamal, DLEQ share proofs, the NIZK consistency-proof model, and the ALRAS interface. It also includes benchmarks, plotting scripts, generated figures, processed tables, and tests.

Environment. The reference run used Python 3.12 on Windows 11 with an Intel64 desktop CPU, a fixed seed `20260611`, and pure-Python affine secp256k1 arithmetic. Fast mode uses 10 scalar-multiplication trials for primitive calibration and larger repetition counts for hashing and serialization. These settings are intended for reproducible trend checking rather than optimized deployment throughput.

Quick start. From the artifact directory, run:

`python run_all.py -mode fast`.

This command generates raw CSV files, processed summaries, paper figure sources, generated PDF figures, Tables 7, 11, and 12, and the sanity-test report. A longer primitive-calibration run is available with `python run_all.py -mode full -seed 20260611`. The pipeline runs primitive calibration and operation-count benchmarks, writes raw CSV files under `artifact/raw-results/`, aggregates summaries under `artifact/processed-results/`, regenerates the paper figure and tables, runs correctness and attack-oriented tests, and writes `processed-results/test_report.json`.

Generated outputs. The main generated outputs are:

- `primitive_costs.csv`, which records scalar multiplication, point addition, SHA-256, and serialization timing statistics;
- `runtime_by_ring.csv`, which records LRAS pre-signing, ALRAS pre-signing, ALRAS verification, and PairVerify costs across ring sizes;
- `opening_by_threshold.csv`, which records OpenShare, OpenCombine, and Judge costs for the evaluated regulator thresholds;
- `transcript_size_by_ring.csv`, which records base LRAS transcript size and the additional ALRAS ciphertext, consistency-proof, and link-tag components;
- `end_to_end.csv`, which records representative complete swap/dispute paths used as integration sanity benchmarks;
- `attack_tests_summary.csv`, which records the attack-test summary;
- generated PDF/SVG/TIFF/PNG evaluation summary figure, together with artifact-only runtime, opening-cost, and transcript-size figure sources;
- generated LaTeX tables for transcript size, end-to-end cost, and attack-oriented tests.

Tests and limitations. The artifact tests honest correctness, pre-signature/full-signature splicing rejection, ciphertext substitution rejection, invalid DLEQ share rejection, threshold boundary behavior, and scoped linkability. The artifact should be read as a reference and reproducibility artifact. It is written in Python, uses a structural LRAS/LTRAS-style lifecycle baseline, and models the NIZK layer through a statement-binding proof profile. Consequently, it validates transcript wiring, attack rejection, threshold-opening behavior, and relative overhead trends, but it does not provide constant-time cryptographic code, a production zero-knowledge backend, or optimized production throughput.

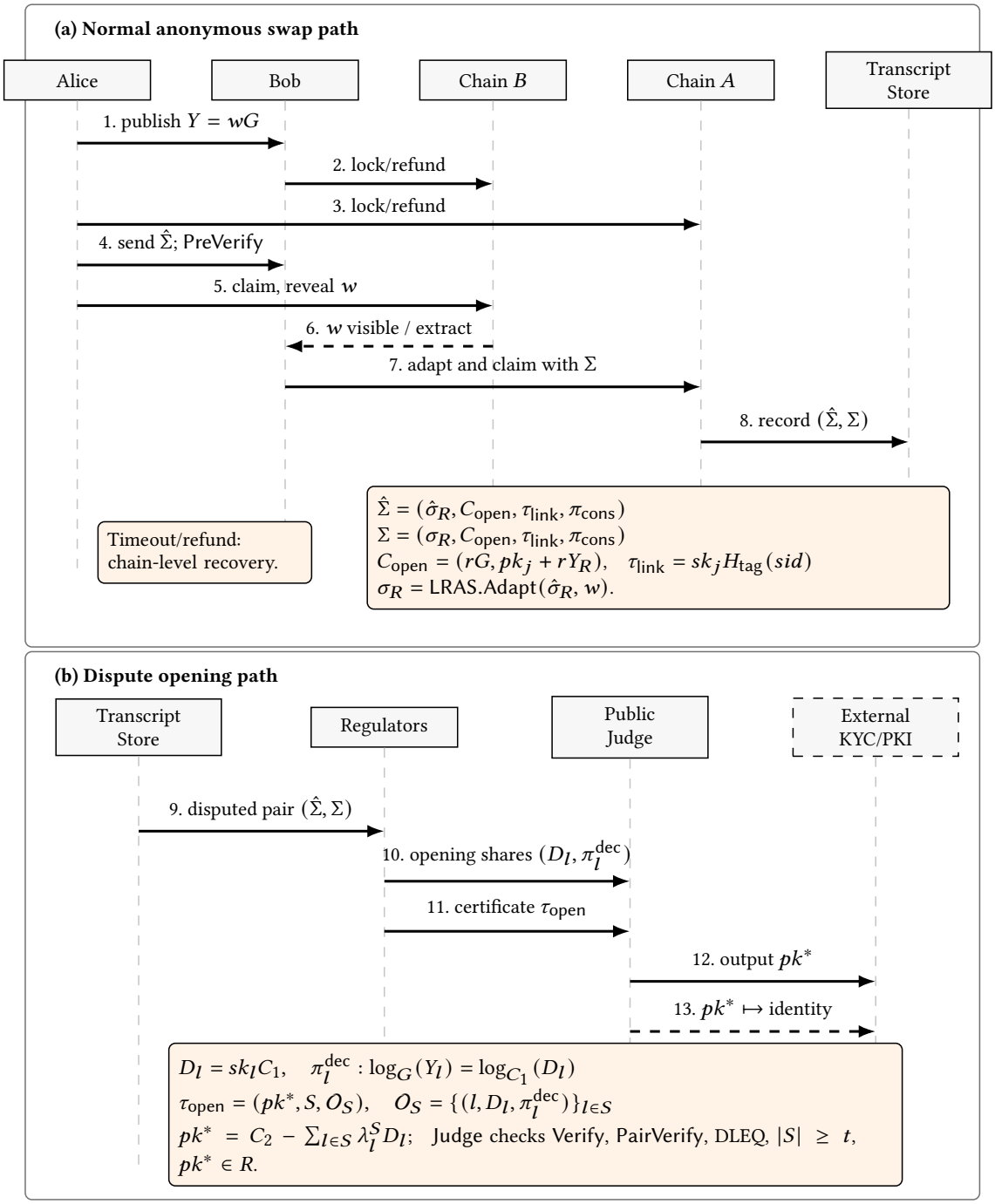

Fig. 2. Two-path protocol workflow of ALRAS. The normal path realizes anonymous witness-coupled swapping, while the dispute path opens a stored pair-consistent transcript using t-of-N verifiable opening shares. Chain-level refunds and external identity mapping are outside the ALRAS cryptographic claim.

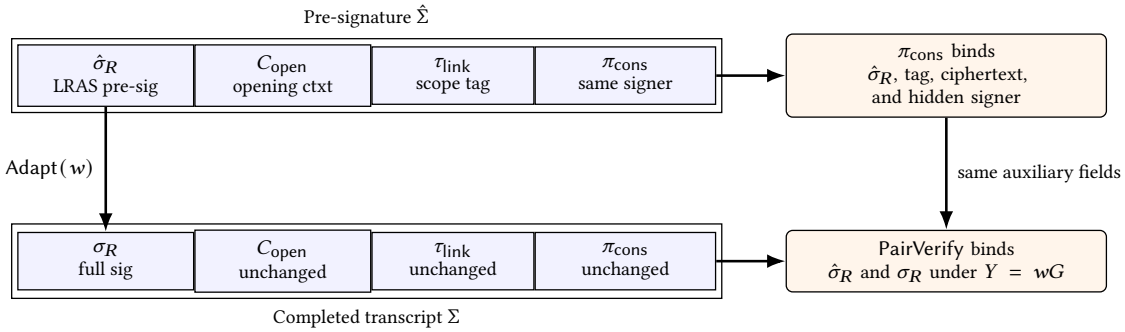

Fig. 3. Augmented transcript structure. The opening ciphertext, link tag, and consistency proof remain identical between $\hat{\Sigma}$ and Σ, while PairVerify checks the adaptor relation between $\hat{\sigma}_R$ and σ_R.

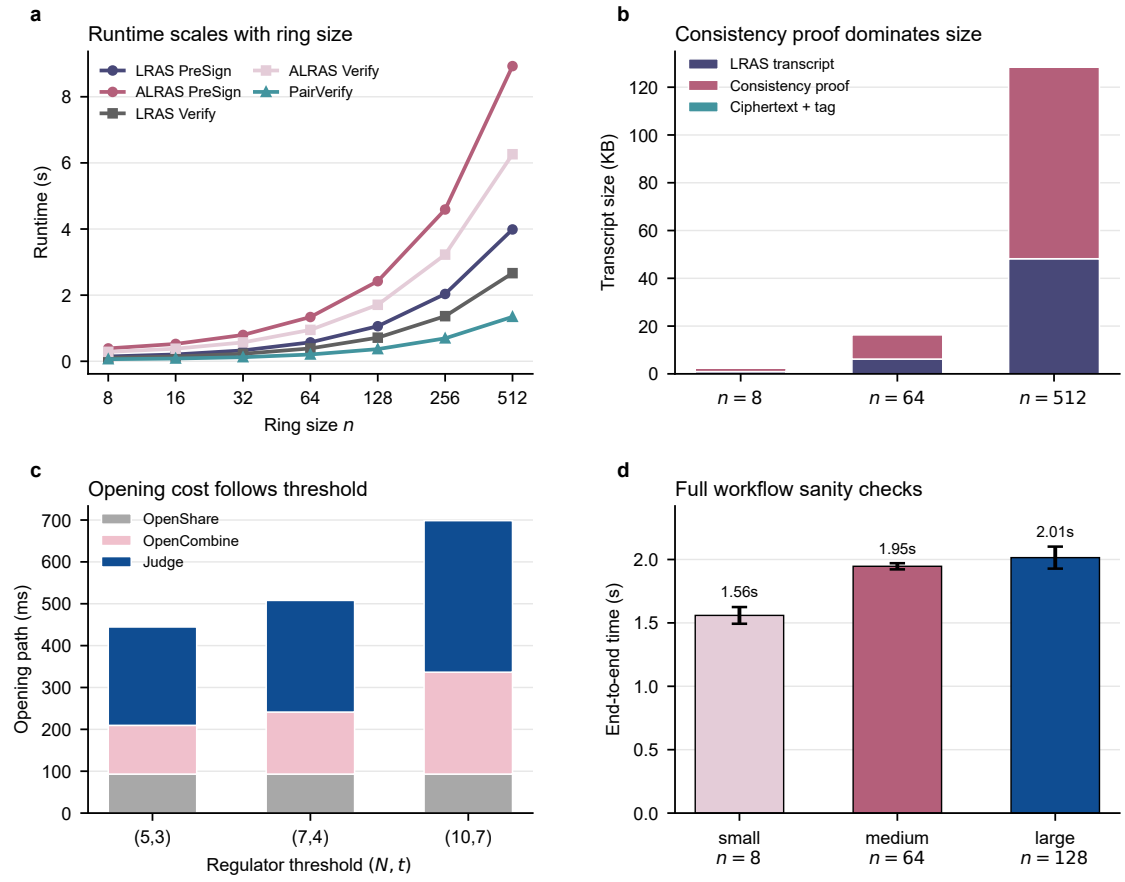

Fig. 4. Reference evaluation summary. (a) Normal-path runtime of the LRAS baseline and ALRAS procedures as the ring size grows. (b) Transcript-size decomposition. (c) Dispute-path threshold-opening cost across regulator thresholds. (d) End-to-end swap/dispute sanity checks with 95% confidence intervals.