Salvium One White Paper

A Privacy-Preserving Proof-of-Work Blockchain with Regulatory-Compliant Two-Way Transactions

The Salvium Team

Version 1.0 | September 2025

Contents

Executive Summary

Privacy-preserving cryptocurrencies represent a $15 billion market segment but face an existential challenge: every existing protocol — from Monero to Zcash — lacks the ability to comply with new regulatory mandates without undermining privacy. This has led to exchange delistings, shrinking liquidity, and uncertainty about the long-term viability of privacy coins under frameworks such as the EU's Markets in Crypto-Assets (MiCA).

Salvium One is the first and only privacy-preserving blockchain to solve this problem. Through a novel construction called SPARC (Spend Proofs and Anonymized Returns for CARROT), Salvium introduces a capability absent from all other projects: privacy-preserving two-way transactions that enable compliant fund returns and selective auditability without identity disclosure. This cryptographic breakthrough positions Salvium as the only privacy protocol capable of remaining both private by default and viable for regulated exchange listings.

The core innovation, SPARC (Spend Proofs and Anonymized Returns for CARROT), has undergone a security audit by Cypher Stack and operates on mainnet with 320,000+ blocks validated as of September 2025. The protocol provides the following.

- Anonymized return mechanisms allowing transaction rejection while preserving ring signature anonymity (Section 7.1).
- Selective transparency through enhanced view keys, allowing voluntary compliance without protocol-level surveillance.
- Native staking implementation distributing 20% of block rewards to participants (21,600-block lockup period).
- Compatibility layer built on proven CARROT addressing with T-CLSAG signature adaptations.

By enabling regulatory compliance without sacrificing default privacy, Salvium One provides exchanges with a viable privacy coin listing while preserving fundamental privacy guarantees for users. The protocol bridges existing privacy technologies with evolving regulatory requirements through cryptographic primitives rather than administrative controls.

Technical documentation and implementation details available at https://salvium.io.

1. Introduction

1.1. The Privacy-Compliance Challenge

Privacy is a fundamental human right and essential for financial sovereignty. Yet, the cryptocurrency sector grapples with balancing user anonymity against regulatory oversight. As detailed in the Executive Summary, traditional privacy coins like Monero and Zcash prioritize anonymity through features like ring signatures but lack tools for selective disclosure, resulting in delistings from exchanges amid regulations like MiCA.

In contrast, transparent blockchains such as Bitcoin and Ethereum meet compliance needs but expose transaction details, posing risks to users. This tension highlights a key industry gap: protocols must enable oversight without inherent surveillance.

Key regulatory context: The EU's Markets in Crypto-Assets (MiCA) regulation, Article 76(3), requires crypto-asset service providers (CASPs)—such as exchanges—to identify holders and transaction histories for privacy coins, shifting the compliance burden to service providers rather than protocols.

Glossary: CASPs refer to regulated entities like exchanges; ring signatures mix transactions for unlinkability.

1.2. Salvium's Solution: Compliance Without Compromise

Salvium One adopts privacy by default while offering optional compliance tools. Built as a protocol (not a CASP), it equips exchanges with cryptographic mechanisms to fulfill obligations, preserving anonymity for non-regulated users.

Core enablers:

- Selective transparency: Users prove ownership of outputs without exposing private keys.
- View-only subaccounts: Scoped monitoring for designated wallets.
- Anonymous refunds: Two-way transactions via SPARC (detailed in Section 3).
- Zero-knowledge proofs: For audits and compliance verification.

This approach resolves the challenge cryptographically, without backdoors or mandatory surveillance.

1.3. Target Use Cases

Salvium serves diverse stakeholders:

Users: Default private transactions, optional compliance for exchanges, staking rewards (20% of block rewards), DeFi participation, and surveillance protection.

Exchanges: Compliant listings, reporting tools, risk mitigation, and privacy safeguards for customers.

Developers: DeFi primitives with privacy, smart contract support, variable privacy levels, and cross-chain features.

2. System Overview and Architecture

Salvium One is a proof-of-work blockchain integrating CryptoNote privacy with compliance innovations. Below is a high-level architecture diagram.

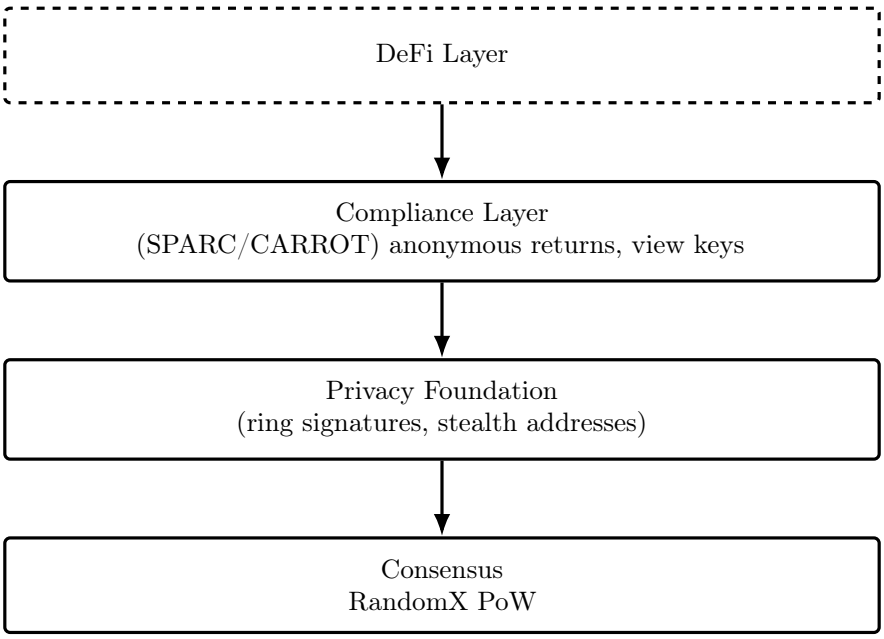

Figure 1: Layered Architecture of Salvium Blockchain

2.1. Consensus and Monetary Policy

Salvium employs RandomX, a CPU-focused PoW algorithm for ASIC resistance and decentralization.

- **Block Parameters**: 120-second intervals, dynamic difficulty, initial reward 123.825 SAL.
- **Emission Schedule**: Total supply 184.4 million SAL; modified Monero curve with 5x per-block emission; tail emission 3 SAL/block post-main phase.
- **Pre-mine**: 12.01% locked in governance wallet (24-month vesting for operations/development).
- **Native Staking**: Phase 1: 20% of rewards to stakers (21,600-block lockup, ∼30 days); future phases shift to fee-based.

2.2. Privacy Foundation

Built on CryptoNote:

- Ring Signatures: Unlinkable transactions.
- Stealth Addresses: Recipient anonymity.
- Confidential Transactions: Hidden amounts.
- Dynamic Ring Sizes: Larger anonymity sets can offer better obfuscation.

2.3. Compliance Layer

Layered atop privacy (no default compromise):

- Refundable Transactions: Returns without sender addresses (via SPARC; see Section 3).
- View Key Extensions: Selective history disclosure (via CARROT; Appendix C).
- Audit Proofs: Zero-knowledge validations for validity and spend authority (see Section 5 for regulatory ties).

2.4. Monero's Cryptographic Pillars: Foundations for Privacy

Salvium One builds directly on Monero's privacy-focused architecture, which relies on CryptoNote principles to ensure transaction anonymity. To understand Salvium's innovations (such as SPARC and T-CLSAG), it's essential to grasp Monero's four core zero-knowledge "pillars": ownership, unspentness, linkability, and balance. These must be proven in every transaction without revealing sensitive details, using cryptographic tools to maintain privacy while preventing fraud. Below, this section explains what each pillar is, why it's necessary, and how Monero implements it. Salvium preserves these pillars but augments them for compliance features, as detailed in later sections.

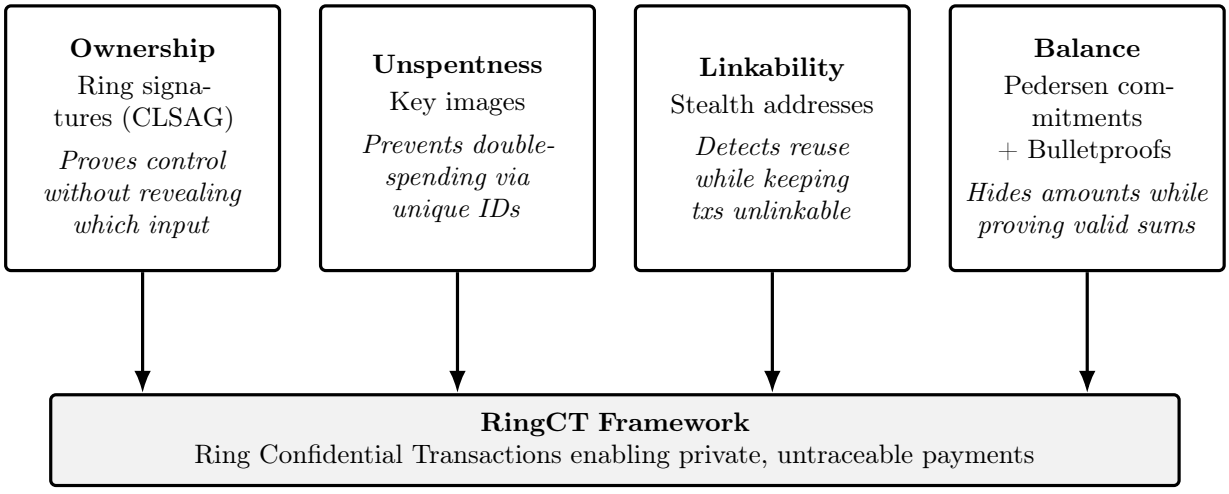

Figure 2: Monero's four cryptographic pillars that ensure transaction validity while preserving privacy.

2.4.1. The Four Pillars in Detail

2.4.2. Ownership

What it is:
A zero-knowledge proof that the sender controls (owns) the transaction inputs without revealing which specific inputs they are or exposing private keys.

Why it's needed:
In privacy coins, transactions mix real inputs with decoys to hide the true sender. Without ownership proofs, anyone could claim control over others' funds, leading to theft or invalid transactions.

How Monero implements it:
Uses ring signatures, specifically CLSAG (Concise Linkable Spontaneous Anonymous Group) since 2019. CLSAG proves knowledge of a private key for one output in a "ring" of decoy outputs, ensuring signer ambiguity (anonymity set). For example, in a ring of 11 outputs, the signature hides which one is real while proving the signer knows the scalar (private key) for it. This is done via elliptic curve operations on Ed25519, with the proof size optimized for efficiency.

Salvium's augmentation:
Salvium has designed and implemented T-CLSAG, which adapts CLSAG for CARROT's dual-scalar (x, y) outputs, enabling compatibility with compliance tools like view keys (see Section 4).

2.4.3. Unspentness

What it is:
A zero-knowledge proof that the inputs have not been spent before, preventing double-spending.

Why it's needed:
Blockchains must ensure each output is used only once to maintain the integrity of the money supply and transaction validity.

How Monero implements it:
Employs key images, unique identifiers derived from the private key (e.g., $K = kH_p(K)$, where k is the private key and $H_p(K)$ is a hash-to-point function). The network checks for duplicate key images in the blockchain; if one appears twice, it's a double-spend attempt. This is integrated into ring signatures (e.g., MLSAG predecessors to CLSAG) without revealing the output's identity.

Salvium's augmentation:
SPARC extends this with spend authority proofs, allowing anonymous returns while verifying unspentness in compliance scenarios (see Section 3.3).

2.4.4. Linkability

What it is:
A zero-knowledge mechanism to link transactions if the same input is reused (e.g., in a double-spend), without compromising privacy otherwise.

Why it's needed:
It detects and prevents fraud like double-spending by making reuses detectable, while keeping unrelated transactions unlinkable.

How Monero implements it:
Key images (from unspentness) serve dual duty here—if the same private key is used twice,

the same key image is generated, linking the attempts. Ring signatures ensure unlinkability for honest users by mixing decoys, and stealth addresses (one-time public keys) prevent address reuse. Subaddresses and integrated addresses further enhance this by deriving multiple receive addresses from a single wallet without linkable patterns.

Salvium's augmentation:
CARROT introduces rerandomizable outputs for forward secrecy, reducing linkability risks in compliant refunds (see Appendix C).

2.4.5. Balance

What it is:
A zero-knowledge proof that the transaction inputs equal outputs plus fees, with all amounts non-negative, without revealing the actual values.

Why it's needed:
Prevents creation of money out of thin air (inflation) or negative spends, ensuring economic soundness while hiding amounts for privacy.

How Monero implements it:
Relies on Pedersen commitments (additively homomorphic: $C = vH + rG$, where v is the amount, r a blinding factor, H and G generators) to hide amounts. Bulletproofs (since 2018) provide range proofs, verifying amounts are in $[0, 2^{64} - 1]$ without revealing them. The transaction verifies sum of input commitments = sum of output commitments + fee, using pseudo-output commitments to offset blinding factors. **Salvium's augmentation:**
Maintains this but integrates with SPARC for zero-knowledge spend proofs in returns, ensuring balance in two-way transactions (see Section 3.3). Salvium also implements a "commitment to a difference" scheme in the balance checks, reducing information leakage.

These pillars work together in Monero's RingCT (Ring Confidential Transactions) framework, enabling private, untraceable payments. However, they limit compliance (e.g., no selective disclosure or refunds), which Salvium addresses cryptographically. For mathematical details, see Monero's "Zero to Monero" guide. The following sections detail how Salvium evolves these.

Monero Transaction Flow

The following diagram illustrates the complete transaction flow inherited from Monero, showing how the four cryptographic pillars combine to create a valid private transaction. Each step represents a security checkpoint that maintains validity while preserving anonymity. The annotations indicate where Salvium augments these operations with T-CLSAG for dual-scalar support, SPARC for compliant returns, and enhanced balance verification—building regulatory compatibility atop this proven cryptographic foundation.

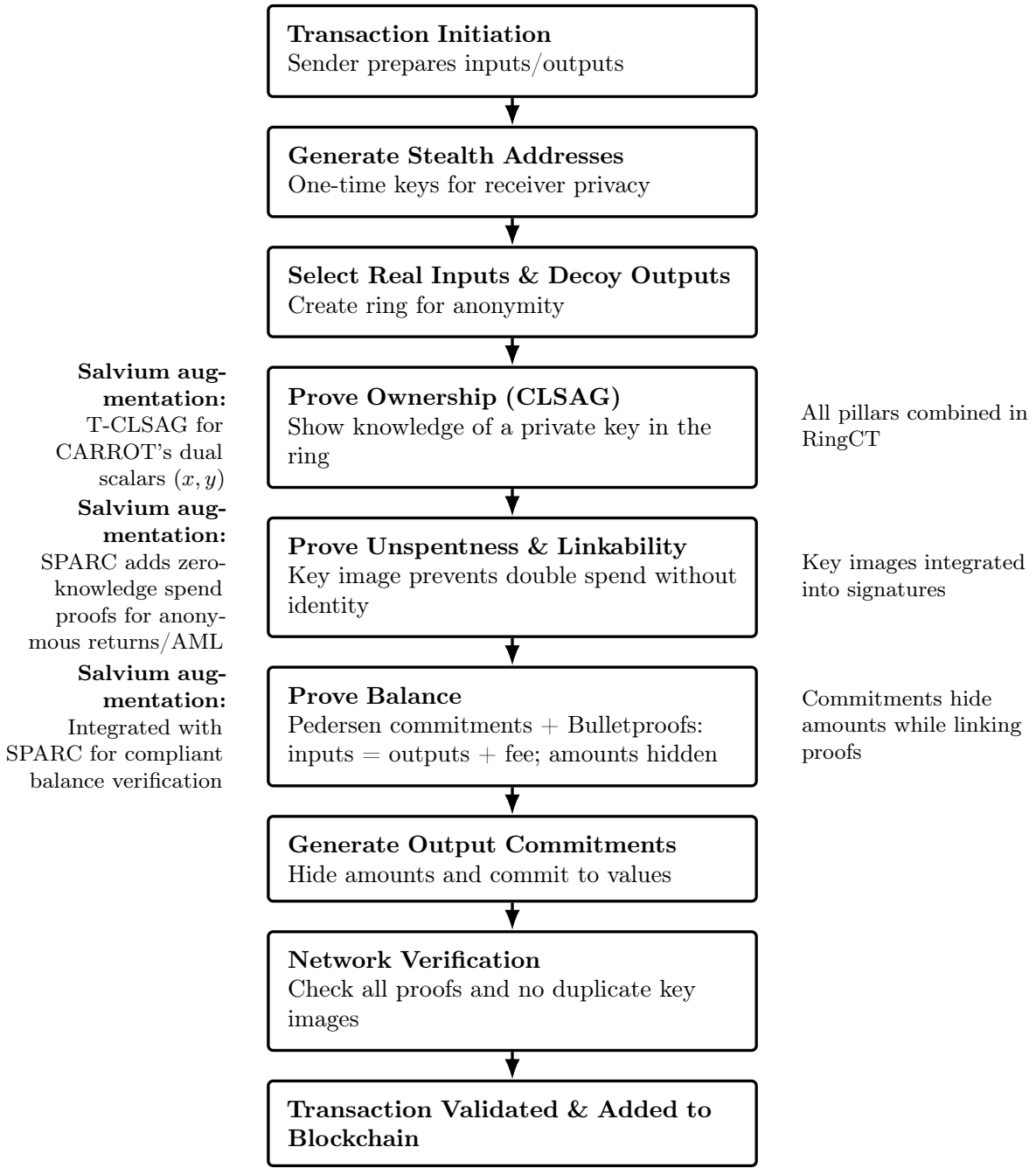

Figure 3: Detailed Monero Transaction Flow with Salvium Augmentations

3. CARROT: Foundation for Compliance

CARROT (CryptoNote Address on Rerandomizable RingCT Output Transactions) is a Monero-originated addressing scheme adopted by Salvium for enhanced compliance. Not Salvium's creation, it enables SPARC (Section 3). The Monero community funded two proposals that led to CARROT's development and implementation, and it is proposed for Monero's FCMP++ upgrade. Salvium is the first to implement CARROT in a live environment.

3.1. Why CARROT Matters for Salvium

Traditional CryptoNote limits compliance:

- No refunds without sender input.
- All-or-nothing viewing.
- Lacking forward secrecy.

CARROT addresses this via:

- Dual-key addresses ($K_o = xG + yT$) for advanced protocols.
- Tiered view keys for selective transparency.
- Rerandomizable outputs for forward secrecy.

CARROT is proposed for Monero's FCMP++ upgrade and is first implemented live in Salvium. See the specification at https://github.com/jeffro256/carrot.

4. SPARC: Core Innovation for Privacy-Preserving Compliance

4.1. Overview

SPARC (Spend Proofs and Anonymized Returns for CARROT) enables rejection of unauthorized transactions without privacy loss, solving a key regulatory hurdle for privacy coins. Unlike Monero's one-way transactions, SPARC uses cryptography for two-way flows, avoiding surveillance.

4.2. Anonymized Returns Protocol

SPARC enables recipients (e.g., exchanges) to return funds anonymously, addressing regulatory needs like MiCA's transaction rejection requirement (see Section 5.1). Built on 2019 Monero proposals, SPARC enhances CARROT compatibility and mitigates quantum vulnerabilities. When Alice sends funds to Bob, she embeds an encrypted return path ensuring:

- Only Alice and Bob decode the path.
- No identity exposure.
- Alice recognizes returned funds.
- Anti-laundering safeguards via spend proofs (Section 3.3).

See Appendix A.1 for technical derivations. This meets MiCA Article 76(3) for transaction rejection (see Section 5).

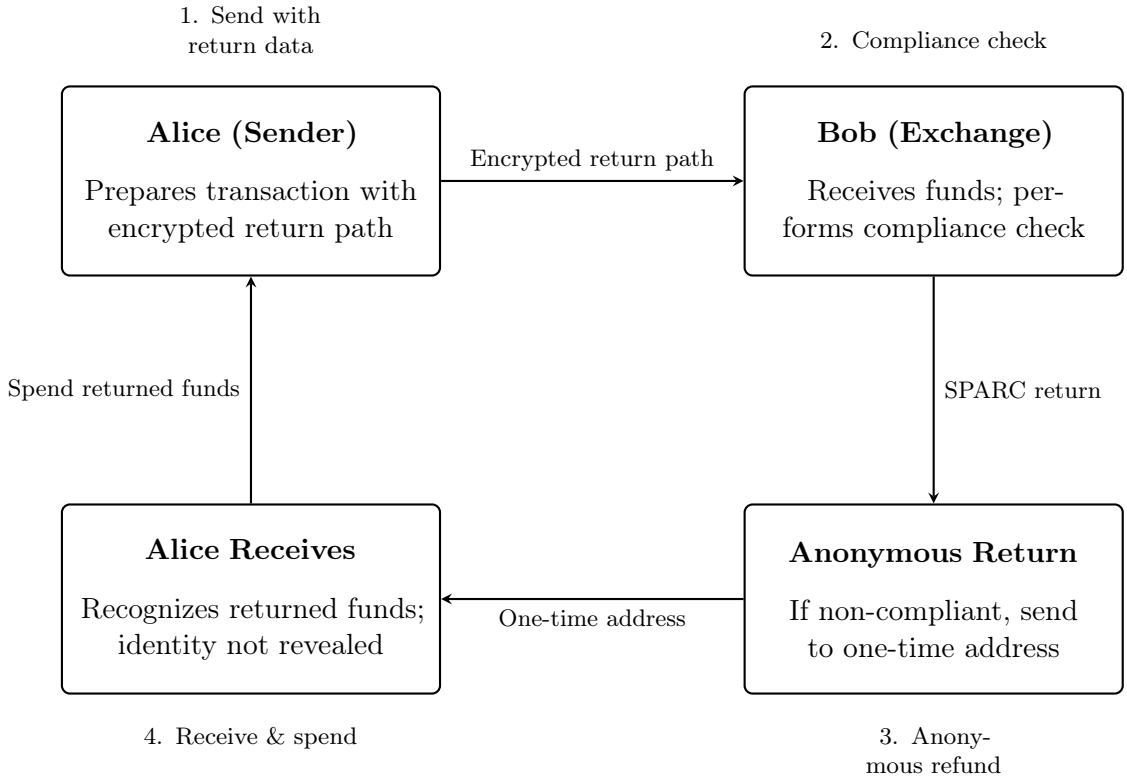

Figure 4: SPARC Workflow: Conceptual Overview

4.3. Zero-Knowledge Spend Proofs

Proves sender control over outputs without secrets, preventing misuse. Using Schnorr/Fiat-Shamir:

- Demonstrates auditable custody.
- Properties: Unforgeability (verified by audits), zero-knowledge, binding, soundness.

4.4. Technical Innovation and Security Validation

SPARC extends CARROT with:

- Evolved key derivation for quantum resistance.
- Cascade vulnerability fixes.
- Secure challenges and domain separation.

Cypher Stack audit confirmed no vulnerabilities; see Appendix A for constructions.

4.5. Implications for Adoption

For exchanges: MiCA compliance, non-KYC rejection, liability protection. For users: Default privacy, automatic returns. Positions Salvium for regulated environments.

5. T-CLSAG: Adapted Ring Signature Scheme

5.1. Background

CLSAG (used in Monero since 2020) proves single-scalar knowledge but is incompatible with CARROT's dual-scalar secrets. Since Salvium One requires CARROT to support MiCA compliance for viewing the full transaction history of a wallet, the Salvium team had to replace CLSAG. The result was Two-scalar Concise Linkable Spontaneous Anonymous Group (T-CLSAG), which adapts the core mathematical principles from CLSAG and extends those principles to dual-scalar operation to support the CARROT address scheme.

5.2. Overview

T-CLSAG proves knowledge of the dual scalars (x, y) for outputs, reworking CLSAG calculations while retaining:

- Anonymity in rings.
- Compatible key images.
- Compact transactions.
- Provable security.

Unlike CLSAG which proves knowledge of a single secret scalar for ring inputs, CARROT requires proving both x and y scalars since both are needed to compute the output's one-time address ($K_o = xG + yT$). T-CLSAG modifies the signature calculations to handle this dual-scalar requirement.

The core CLSAG principle remains: the scheme computes commitments and responses that verify correctly only when the sender knows the secret scalars for the actual input being spent, while maintaining privacy about which ring member is real.

T-CLSAG underwent comprehensive security audits by Cypher Stack in 2025, with both implementation and mathematical reviews confirming correctness without vulnerabilities.[1] See Appendix B for complete mathematical specifications.

[1] Implementation security audit (August 2025): https://github.com/salvium/salvium_library/blob/main/audits/T_CLSAG_Implementation_Review.pdf; Mathematical proof review (July 2025): https://github.com/salvium/salvium_library/blob/main/audits/T_CLSAG_Review-Final.pdf

6. Regulatory Compliance Integration

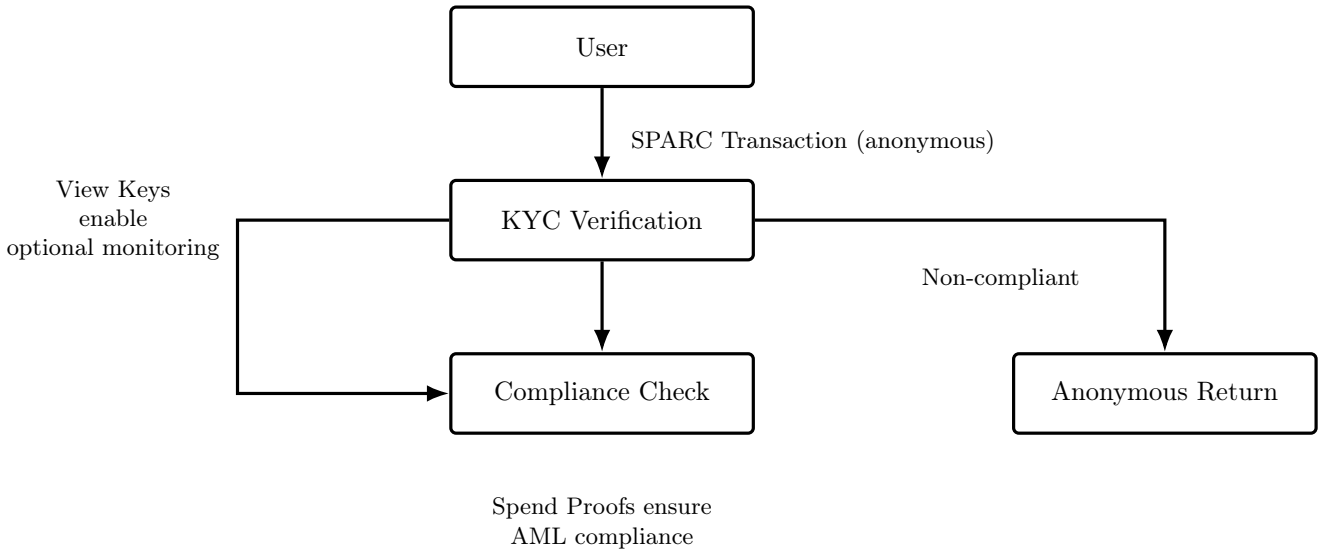

Figure 5: Regulatory compliance workflow under MiCA

6.1. MiCA and Global Frameworks

MiCA (Regulation (EU) 2023/1114) requires CASPs to enable transaction rejection and monitoring for privacy coins under Article 76(3), prohibiting listings unless holders and histories are identifiable. SPARC enables returns for non-compliant funds, while CARROT view keys support traceability without default surveillance.

Broader obligations under Article 66 include acting in clients' best interests, clear disclosures, risk warnings, and robust AML/KYC: customer due diligence, ongoing transaction monitoring, suspicious activity reporting (SARs), and sanctions screening. Salvium's selective transparency (view-only subaccounts) facilitates this, ensuring CASPs meet reporting requirements (e.g., 5-year record-keeping) without compromising user privacy.

6.2. Salvium's Mechanisms

Salvium equips CASPs with:

- **Anonymous Returns**: SPARC enables non-KYC fund returns (Section 3.2).
- **View Key Extensions**: CARROT provides view-only wallets for transaction history (Appendix C).
- **Spend Proofs**: Zero-knowledge proofs ensure AML compliance (Section 3.3).

6.3. Compliance Principles

- Default Privacy: Features inactive without request.
- User Consent: Explicit authorization required.
- Selective Disclosure: User-controlled revelations.
- No Backdoors: No access to funds/data.

7. Implementation and Deployment

7.1. Development Phases

- **Phase 1 (Live)**: RandomX mining (120s blocks), 20% staking, basic refunds, CryptoNote privacy.
- **Phase 2 (Live)**: CARROT, SPARC, MiCA tools, advanced views.
- **Phase 3 (H1 2026)**: Anonymous payment channels, smart contracts, conditional payments, bridges.
- **Phase 4 (2027+)**: Layer-2 scaling, DeFi suite, institutional tools.

7.2. Audits and Security

Cypher Stack audited SPARC and T-CLSAG security:

- Correctness verified.
- Zero-knowledge validated.
- Quantum improvements confirmed.
- No vulnerabilities.

Reports: https://github.com/salvium/salvium_library/tree/main/audits.

8. Economic Model and Tokenomics

8.1. Token Distribution

Total Supply: 184.4M SAL.

Table 1: Token Distribution

Category	Percentage	Allocation
Block Rewards	87.99%	Miners (80%) and stakers (20%)
Development Fund	3.53%	Contributors/incentives
Operations Fund	8.48%	Locked governance (24-month vesting)

8.2. Emission Schedule

The emission schedule modifies Monero's curve to accelerate distribution and liquidity. With 120-second blocks maintained, per-block emission increases 5x while the initial supply phase doubles in duration. This yields 184.4 million total supply with a broader, flatter emission curve than Monero's concentrated distribution.

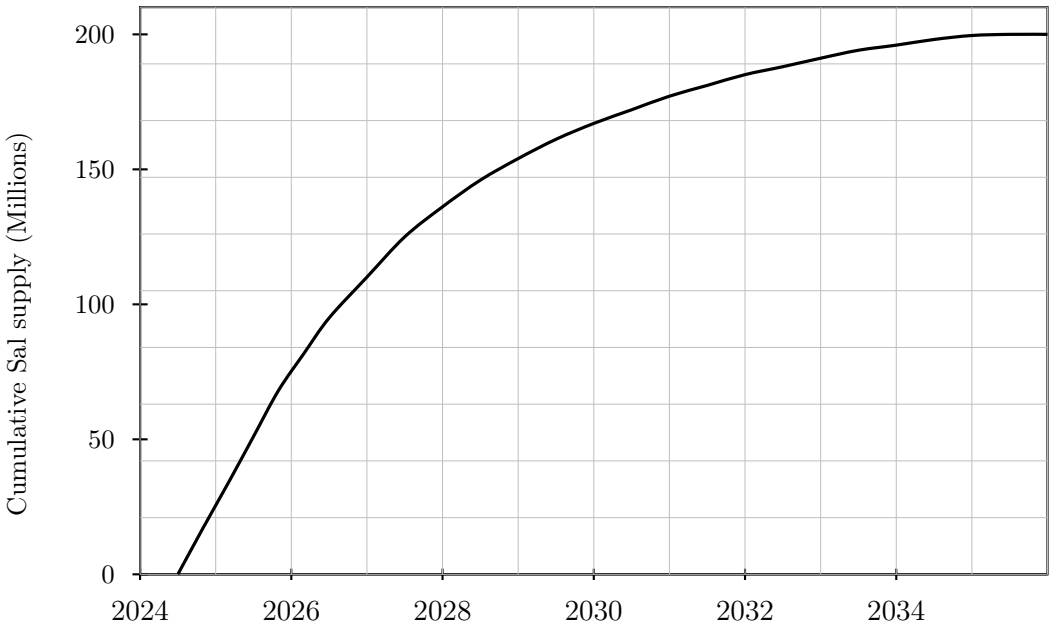

Figure 6: Salvium Token Emission Schedule

8.3. Staking Economics

- **Current (Phases 1-2)**: Stakers 20% rewards, miners 80%; 21,600-block stake (∼30 days).
- **Future (Phase 3+)**: Stakers earn DeFi fees; miners 100% rewards.

9. Comparison with Existing Privacy Coins

Privacy-preserving cryptocurrencies employ distinct architectural approaches with corresponding trade-offs. This section compares Salvium's technical capabilities against established privacy protocols to contextualize its contributions to the field.

The comparison matrix below evaluates seven key features across four representative blockchain architectures: Bitcoin (transparent baseline), Monero (mandatory privacy), Zcash (optional privacy), and Salvium One (privacy with selective disclosure). The analysis focuses on technical capabilities rather than implementation quality or adoption metrics.

Salvium's distinguishing characteristic lies in its cryptographic resolution of previously incompatible requirements: maintaining ring signature anonymity while enabling transaction rejection and selective auditability.

9.1. Feature Comparison

Table 2: Comparison of Privacy Coin Features

Feature	**Bitcoin**	**Monero**	**Zcash**	**Salvium One**
Default Privacy	No	Yes	Optional	Yes
Anonymous Refunds	No	No	No	Yes
Regulatory Compliance	Yes	No	Limited	Yes
Native Staking	No	No	No	Yes
Smart Contracts	No	No	No	Phase 3
View Proofs	No	Limited	Yes	Yes
Programmable Privacy	No	No	Limited	Yes

Accurate as of September 2025. Monero lacks refund mechanisms; Zcash offers opt-in compliance; Salvium's SPARC enables compliance and staking now.

10. Future Research and Development

10.1. Cryptographic Enhancements

- **Quantum Resistance**: Post-quantum signatures, safe addresses, migration plans.
- **Advanced Privacy**: Set membership proofs to replace rings; improved confidential transactions; cross-chain privacy.

10.2. Scalability and DeFi Roadmap

Salvium's foundations—Transactional Imbalances (mint/burn) and Asynchronous Transactions (`protocol_tx`)—enable private staking and future DeFi (live since 2024; see https://docs.salvium.io/THE%20PROTOCOL). Planned features by end of H1 2026 include:

- Private DEXs (no front-running).
- Hidden lending positions.
- Anonymous payment channels via SPARC.

Layer-2 solutions (rollups, state channels) and cross-chain interoperability are targeted for 2027, enhancing scalability while preserving privacy.

Unlike Ethereum (visible transactions) or Monero (no DeFi), Salvium modifies state privately. See docs.salvium.io for details on `protocol_tx` and Asynchronous Transactions.

11. Conclusion

Salvium One advances privacy-preserving blockchain technology by demonstrating that cryptographic innovation can resolve the tension between privacy and regulatory compliance. The protocol's original contributions—SPARC (Spend Proofs and Anonymized Returns for CARROT) and T-CLSAG (Two-scalar Concise Linkable Spontaneous Anonymous Group signatures)—enable transaction rejection and selective auditability without compromising default privacy.

Key achievements include:

- **SPARC**: Salvium-developed anonymous return mechanism enabling MiCA-compliant transaction rejection
- **T-CLSAG**: Original adaptation of CLSAG for dual-scalar outputs required by CARROT
- **First CARROT implementation**: Live deployment of Monero's addressing scheme
- **Regulatory compliance**: Meeting Article 76(3) requirements without backdoors
- **DeFi foundations**: Transactional Imbalances and Asynchronous Transactions for private smart contracts

Through SPARC and T-CLSAG, Salvium proves that privacy coins need not choose between delisting and surveillance. As cryptocurrency regulation evolves globally, protocols that balance individual privacy with institutional requirements through cryptographic primitives rather than administrative controls may define the future of digital finance.

As crypto matures, Salvium balances needs for a regulated future. The protocol evolves toward DeFi while upholding standards. Salvium represents more than a technical framework—it is a vision for digital finance that respects both individual rights and systemic integrity. By blending innovation with responsibility, Salvium aims to set the tone for the next wave of privacy-preserving DeFi. Join the community to contribute, shape the protocol's direction, and be part of building a financial system that values privacy without sacrificing trust. Technical documentation and community resources are available at `https://salvium.io`.

A. SPARC - Complete Technical Specification

A.1. Anonymized Returns

A.1.1. Overview

A "return address scheme" was first proposed for Monero and its derivations by knaccc in 2019. Salvium Zero implemented the mechanism, along with some innovation in order to support the Salvium "protocol_tx" functionality (a necessary precursor to staking and accrued yield in Salvium).

However, Salvium One development has revealed that the original scheme is vulnerable to a quantum adversary capable of solving the discrete logarithm problem, and therefore needed to evolve. Anonymized Returns is the next generation return-address scheme. Fully CARROT-compatible, it addresses the "cascade" vulnerability present in the original design, and provides increased resistance against a quantum adversary.

A Monero developer suggested a new scheme, which addressed the cascade vulnerability in Salvium Zero's approach. The Salvium development team subsequently extended the new scheme to conform better to the practices already provided by CARROT.

A.1.2. Scheme Derivations

Intermediate Values

$$m_{return} = \text{SecretDerive}(\text{"SPARC return pubkey encryption mask"} \| s_{sr} \| \text{input\_context} \| K_o) \tag{1}$$

$$K_{change} = \text{output onetime address of change enote} \tag{2}$$

Component Values

$$k_{return} = \text{ScalarDerive}(\text{"SPARC return address scalar"} \| s_{vb} \| K_o) \tag{3}$$

$$k_{idx} = \text{ScalarDerive}(\text{"SPARC return address scalar"} \| K_o \| idx) \tag{4}$$

$$K_{return} = k_{return} G + k_{idx} T \tag{5}$$

$$K_r = K_{change} + K_{return} \tag{6}$$

A.1.3. Sending Process

When sending funds to Bob, Alice must construct 2 enotes (enote_{out}, enote_{change}), by doing the following:

1. construct enote_{out} $\{K_o, C_a, a_{enc}, v_t, \text{anchor}_{enc}, \text{pid}_{enc}\}$ normally
2. compute k_{return}
3. compute k_{idx}
4. compute $K_{return} = k_{return} G + k_{idx} T$

5. compute m_{return} = SecretDerive("SPARC return pubkey encryption mask"$\|s_{sr}\|$input_context$\|K_o$)
6. compute $\text{return}_{enc} = K_{return} \oplus m_{return}$
7. embed return_{enc} in enote_{out}, giving $\{K_{o,out}, C_a, a_{enc}, v_t, \text{anchor}_{enc}, \text{pid}_{enc}, \text{return}_{enc}\}$
8. construct enote_{change} normally
9. compute $K_r = K_{change} + K_{return}$
10. add $K_r \rightarrow \langle \text{enote}_{out}, \text{enote}_{change}, idx \rangle$ to the ledger of "allowable return addresses"

A.1.4. Return Process - Sender

If Bob wishes to return the payment to Alice, he must first recover the value of K_{return}:

1. compute s_{sr}
2. compute m_{return}
3. compute $K_{return} = \text{return}_{enc} \oplus m_{return}$

Bob then constructs a special "return" enote enote_{return}, by performing the following steps:

4. compute d_e in the normal manner, but substituting K_{change} in place of $K_{s,j}$
5. recompute $s_{sr} = K_{return} \cdot d_e$
6. compute $K_o = K_{return} + K_{change}$, and embed in enote_{return}
7. compute the remaining enote_{return} values $(C_a, a_{enc}, v_t, \text{anchor}_{enc}, \text{pid}_{enc})$ normally

Bob then sends the completed enote_{return} to the one-time address K_o.

A.1.5. Return Process - Receiver

Prior to performing the existing enote scan process as defined by Carrot, Alice checks her ledger to see if the value K_o is present in the private hashmap of known K_r values. If the value K_o is found, then Alice proceeds to process the return directly, using enote_{out}, enote_{change}, and enote_{return}.

1. compute k_{return}
2. compute K_{return}
3. compute K_{change}
4. if $K_o \neq K_{return} + K_{change}$, then ABORT
5. Let $s_{sr} = k_{return} \cdot D_e$
6. Let v_t' = SecretDerive("Carrot view tag"$\|s_{sr}\|$input_context$\|K_o$)[:3]

7. if $v'_t \neq v_t$, then ABORT
8. Let $m_a = \text{SecretDerive}(\text{"Carrot encryption mask a"} \| s_{sr}^{ctx} \| K_o)[:8]$
9. Let $a' = \text{BytesToInt}(a_{enc} \oplus m_a)$
10. Let $k'_a = \text{ScalarDerive}(\text{"Carrot commitment mask"} \| s_{sr}^{ctx} \| a' \| K_{change} \| \text{enote\_type})$
11. Let $C'_a = k'_a G + a' H$
12. if $C'_a \neq C_a$, then ABORT
13. Let $m_{pid} = \text{SecretDerive}(\text{"Carrot encryption mask a"} \| s_{sr}^{ctx} \| K_o)[:16]$
14. Set $\text{pid}' = \text{pid}_{enc} \oplus m_{pid}$
15. Let $m_{anchor} = \text{SecretDerive}(\text{"Carrot encryption mask anchor"} \| s_{sr}^{ctx} \| K_o)[:16]$
16. Set $\text{anchor}' = \text{anchor}_{enc} \oplus m_{anchor}$
17. Let $d'_e = \text{ScalarDerive}(\text{"Carrot sending key normal"} \| \text{anchor}' \| \text{input\_context} \| K_{change} \| \text{pid}')$
18. Let $D'_e = d'_e \cdot \text{ConvertPointE}(K_{change})$
19. if $D'_e \neq D_e$, then ABORT
20. Return successfully

NOTE: the above process is largely identical to the existing Carrot enote scan process, with the exception of the following replacements:

- $k_v \rightarrow k_{return}$
- $K'_{s,j} \rightarrow K_{change}$

If the value K_o is not found by Alice in her ledger, the existing enote scan process is performed instead. Note also that once Alice has received a first "return" enote for a given "sent" enote, it is necessary for her to recompute the hashmap entry, because the value of k_{idx} will have changed, and therefore so will the value of K_{return} and thus K_r. The steps to be performed are as follows:

21. Obtain $K_o \rightarrow \langle \text{enote}_{out}, \text{enote}_{change}, idx \rangle$ from the ledger of allowable "return_payment" onetime addresses
22. Remove K_o entry from the ledger
23. Increment idx value
24. Compute new k_{idx} value

A.1.6. Determining spendability and computing key images

In order to determine spendability of the returned output, it is necessary to verify the one can derive the public spend key (in this case, K_{change}).

The key image required to spend the returned output is normally calculated as:

$$L = (k_{g,i} \cdot k_{\text{subscal},j} + k_{g,o})H_p(K_o)$$

However, $k_{g,o}$ has not been used, because $K_{return} = k_{return}G$ was used instead of the $k_{g,o}G + k_{t,o}T$ terms, and because K_{change} was used in place of $K_{s,j}$. Therefore the key image will be calculated as:

$$L = (k_{g,i} \cdot k_{\text{subscal},j} + k_{return} + k_{g,o\_change})H_p(K_o)$$

A.2. Spend Authority Proof

A.2.1. Overview

The purpose of the "spend authority proof" is to establish, in zero knowledge, that the prover knows the secret values x, y for a given key K_o, such that $K_o = xG + yT$. This requirement comes from the fact that knowledge of the x, y values permits an individual to be able to spend the output with the one-time address output key K_o.

The proving scheme relies on a variation of a Schnorr zero-knowledge proof, and uses the Fiat-Shamir heuristic to make the proof non-interactive. The sender assumes the role of prover in this scenario, with the verifier being the recipient of any transaction output. The goal of the proof is to allow the verifier to know whether returned funds would be received by the prover. It is expected that $K_o = K_c$ (the one-time address output key for the transaction).

A.2.2. Proof Structure

The proof structure includes:

- Commitments: $R = r_x G + r_y T$, where r_x and r_y are random scalars.
- Responses: $z_x = r_x + cx$ and $z_y = r_y + cy$ where $c = H_s(R \| K_o)$

A.2.3. Verification Process

The verifier performs the following checks:

1. Recomputes the challenge $c' = H_s(R \| K_o)$.
2. Validates the commitments by testing the validity of the following calculation:

$$z_x G + z_y T = cK_o + R$$

Which expands as:

$$\begin{aligned} z_xG + z_yT - cK_o &= r_xG + r_yT \\ r_xG + cxG + r_yT + cyT - cK_o &= r_xG + r_yT \\ cxG + cyT &= cK_o \end{aligned}$$

Thus, the proof is valid if $z_xG + z_yT - cK_o$ matches R.

A.2.4. Security Analysis

To ensure the prover cannot manipulate the proof:

Unforgeability

- The challenge c is a cryptographic hash of the commitment and K_o. As a result:
  - The prover cannot predict or manipulate c because cryptographic hashes are resistant to pre-image attacks.
  - Any tampering with r_x, r_y, or K_o changes c, making it impossible to construct valid responses z_x and z_y without knowing x and y.

Binding of Responses

- The responses $z_x = r_x + cx$ and $z_y = r_y + cy$ bind c to the secrets x and y:
  - If x or y are incorrect, the responses will not satisfy the verification equation $z_xG + z_yT - cK_o = R = r_xG + r_yT$.
  - The responses inherently depend on the random scalars r_x, r_y and the challenge c, ensuring there is no way to "fake" them.

Zero-knowledge

- The commitment $R = r_xG + r_yT$ is independent of the secrets x and y thanks to the random values r_x and r_y.
- The verifier learns nothing about x or y because:
  - The challenge c is deterministic but unpredictable, derived from a cryptographic hash.
  - The responses z_x and z_y are randomised by r_x and r_y, ensuring no direct leakage of x or y.

Soundness

- The verification equation ensures soundness because:

$$z_xG + z_yT - cK_o = r_xG + r_yT$$

  If the prover does not know x and y, they cannot produce responses z_x and z_y.

A.2.5. Assumptions

1. The hash function H_s used to calculate c is cryptographically secure and resistant to collisions.
2. The random scalars r_x and r_y are truly random and kept secret.
3. The scalar multiplication operations r_xG, r_yT, and others are performed correctly in the elliptic curve group.

B. T-CLSAG Signature Scheme

B.1. Overview

This specification defines a secure, quantum-forward-secret (QFS) adaptation of the CLSAG signature scheme, which we call "Two-scalar Concise Linkable Spontaneous Anonymous Group" (or "T-CLSAG" for short), to support the Carrot addressing scheme, where each output is defined as $K_o = xG + yT$. The signer must prove knowledge of both secret scalars $\{x, y\}$ while maintaining anonymity within a ring of decoys.

B.2. Signing Process

Let $s \in [1, n]$ be the index of the real input in the ring. The signer knows x_s, y_s for the output $K_{o,s}$ being spent. They perform the following steps:

1. Compute the key image (the CLSAG "linking tag")

$$I = x_s H_p(K_{o,s})$$

2. Choose random scalars

$$\alpha_x, \alpha_y \xleftarrow{\$} \mathbb{Z}_\ell$$

3. Compute the commitments

$$L_0 = \alpha_x G + \alpha_y T \tag{7}$$
$$R_0 = \alpha_x H_p(K_{o,s}) \tag{8}$$

4. Compute aggregate public keys and key images

   (a) Aggregate Public Key for ring member i:

$$W_i = H_n(\text{msg}, I, K_{o,1}, \ldots, K_{o,n}) \cdot K_{o,i}$$

   (b) Aggregate Key Image:

$$\widetilde{W} = H_n(\text{msg}, I, K_{o,1}, \ldots, K_{o,n}) \cdot I$$

5. Compute the initial challenge

$$c_{s+1} = H(\text{msg}, I, K_{o,1}, \ldots, K_{o,n}, L_0, R_0)$$

6. For $i = s+1, s+2, \ldots, n, 1, 2, \ldots, s-1$

(a) Compute c_{i+1} (replacing $n+1 \to 1$)

$$L_i = r_{x,i}G + r_{y,i}T + c_i W_i \tag{9}$$
$$R_i = r_{x,i}H_p(K_{o,i}) + c_i\widetilde{W} \tag{10}$$
$$c_{i+1} = H(\text{msg}, I, K_{o,1}, \ldots, K_{o,n}, L_i, R_i) \tag{11}$$

7. Compute Real Signer's Responses ($r_{x,s}$ and $r_{y,s}$)

$$r_{x,s} = \alpha_x - c_s \cdot H_n(\text{msg}, I, K_{o,1}, \ldots, K_{o,n}) \cdot x_s \tag{12}$$
$$r_{y,s} = \alpha_y - c_s \cdot H_n(\text{msg}, I, K_{o,1}, \ldots, K_{o,n}) \cdot y_s \tag{13}$$

8. Output Signature

$$\text{Signature} = (c_1, I, \{K_{o,i}\}_{i=1}^n, \{r_{x,i}, r_{y,i}\}_{i=1}^n)$$

B.3. Verification Process

Given a signature $(c_1, I, \{K_{o,i}\}, \{r_{x,i}, r_{y,i}\})$, a verifier performs the following steps:

1. Loop over all ring members ($i \in [1, n]$)

   (a) Recompute

$$L_i = r_{x,i}G + r_{y,i}T + c_i W_i \tag{14}$$
$$R_i = r_{x,i}H_p(K_{o,i}) + c_i\widetilde{W} \tag{15}$$
$$c_{i+1} = H(\text{msg}, I, K_{o,1}, \ldots, K_{o,n}, L_i, R_i) \tag{16}$$

2. Test if $c_{n+1} = c_1$. If true, the signature is valid, that is, is, the signer has proven knowledgeboth openings $\{x, y\}$ for one of the ring members.

C. Cryptographic Primitives and Security Model

Salvium's security draws from established primitives, ensuring privacy and integrity.

C.1. Cryptographic Foundations

- **Elliptic Curves**: Curve25519 (Montgomery, for Diffie-Hellman with view keys); Ed25519 (Edwards, for other operations). These are birationally equivalent for compatibility.
- **Hash Functions**: Blake2b (sequential mode); Keccak-256 ($r = 1088, c = 512, d = 256$).
- **Commitments**: Pedersen schemes for binding/hiding amounts.
- **Ring Signatures**: T-CLSAG, a bespoke dual-scalar CLSAG variant (using $m = 1$ modality; see Section 4).

C.2. Security Assumptions

The security of Salvium's cryptographic protocols rests on several well-established mathematical assumptions that underpin most modern cryptographic systems. These are:

- Discrete Logarithm Problem (DLP) hardness in elliptic groups.
- Hash collision resistance and random oracle model.
- Ring signature unforgeability/anonymity.
- Commitment binding/hiding.

These assumptions are standard; quantum threats are mitigated in SPARC (see Section 3.4) and planned for future upgrades (Section 9.1).

D. Salvium Glossary

A

Anonymized Returns – Salvium's mechanism allowing recipients to return funds to senders without revealing either party's identity, maintaining ring signature privacy while enabling regulatory compliance.

Asynchronous Transactions (AT) – Salvium's innovation allowing transaction input and output amounts to be decoupled, such that the inputs are burnt, and, at a later date, the coins are minted and returned in a different transaction. This enables features like staking and DeFi.

C

CARROT – Addressing scheme (originally from Monero research) first implemented live by Salvium, enabling dual-key addresses ($K_o = xG + yT$) for compliance features.

Cascade Vulnerability – Security weakness in the original return address scheme that Salvium One addresses through evolved key derivation methods.

Commitment to Difference – Salvium's balance verification scheme that reduces information leakage compared to standard implementations.

D

Dual-scalar Outputs – CARROT's innovation using two secret values (x, y) instead of one, enabling advanced compliance features.

E

Exchange Mode – Salvium's compliance feature allowing exchanges to operate with selective transparency for regulatory requirements.

K

Key Image – Unique identifier preventing double-spending in Salvium, computed as $L = (k_{g,i} \cdot k_{\text{subscal},j} + k_{return} + k_{g,o\_change})H_p(K_o)$.

P

Programmable Privacy – Salvium's approach allowing DApp developers to implement variable privacy levels rather than all-or-nothing anonymity.

Protocol_tx – Salvium's mechanism for protocol-level transactions enabling staking and yield generation on the base layer.

R

Refundable Payments – Salvium's feature allowing exact transaction amounts to be returned without requesting sender addresses.

Return Address Scheme – Original 2019 proposal by knacc, evolved by Salvium to support anonymous refunds.

S

SAL – Salvium's native token (from Latin *salvus*: safe, secure, salvation).

Salvium One/Zero – Version nomenclature where Zero was the initial implementation and One includes SPARC and compliance features.

SPARC – Spend Proofs and Anonymized Returns for CARROT. Salvium's core innovation enabling two-way privacy-preserving transactions.

Spend Authority Proof – Zero-knowledge proof demonstrating control over outputs without revealing private keys, preventing unauthorized returns.

Subaddress View Keys – Salvium's tiered viewing mechanism allowing selective transaction history disclosure for specific wallets.

T

T-CLSAG – Two-scalar Concise Linkable Spontaneous Anonymous Group signatures. Salvium's adaptation of Monero's CLSAG for CARROT compatibility.

Tail Emission – Salvium's perpetual block reward of 3 SAL per block after main emission completes, ensuring long-term mining incentives.

Transactional Imbalances (TI) – Salvium's mechanism allowing controlled supply modifications for staking rewards and DeFi operations.

V

View-only Subaccounts – Compliance feature allowing exchanges to monitor specific wallet activity without accessing funds.

View Tag – Three-byte identifier in CARROT transactions for efficient scanning, computed as $v_t = \text{SecretDerive}(\text{"Carrot view tag"} \| s_{sr} \| \text{input\_context} \| K_o)[:3]$.

Numeric Terms

21,600 blocks – Salvium's staking lockup period, approximately 30 days at 120-second block times.

184.4 million SAL – Total supply cap before tail emissions begin.

12.01% pre-mine – Initial allocation split between development (3.53%) and operations (8.48%, vesting over 24 months).

References

[1] Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. https://bitcoin.org/bitcoin.pdf

[2] van Saberhagen, N. (2013). *CryptoNote v 2.0*. https://cryptonote.org/whitepaper.pdf

[3] Noether, S. (2015). *Ring Signature Confidential Transactions for Monero*. https://eprint.iacr.org/2015/1098

[4] Bünz, B., et al. (2018). *Bulletproofs: Short Proofs for Confidential Transactions and More*. IEEE S&P. https://doi.org/10.1109/SP.2018.00020

[5] Goodell, B., et al. (2019). *Concise Linkable Ring Signatures and Forgery Against Adversarial Keys*. https://eprint.iacr.org/2019/654

[6] European Parliament. (2023). *Regulation on Markets in Crypto-Assets (MiCA)*. https://eur-lex.europa.eu/eli/reg/2023/1114/oj

[7] Financial Action Task Force. (2021). *Updated Guidance for a Risk-Based Approach to Virtual Assets*. https://www.fatf-gafi.org/en/publications/Fatfrecommendations/Guidance-RBA-virtual-assets-2021.html

Resources

Technical

- **GitHub**: https://github.com/salvium
- **Documentation**: https://docs.salvium.io
- **Audit Reports**: https://github.com/salvium/salvium_library/
- **Community**: https://salvium.io/community.html

Community

- **Discord**: https://discord.com/invite/P3rrAjkyYs
- **Telegram**: https://t.me/salvium_official
- **Twitter**: https://x.com/salvium_io

Legal Disclaimer

This white paper is informational and does not constitute investment, financial, or legal advice. The Salvium team does not recommend buying, selling, or holding SAL or any cryptocurrency. Perform your own due diligence and consult with advisors before making decisions.

Information may change without notice; this is not a commitment of the team. It is not a prospectus or a securities offer in any jurisdiction.

Risk Disclosures

Cryptocurrency investments involve substantial risk, including a possible total loss. Salvium faces regulatory uncertainty, technical vulnerabilities, market volatility, and competition. The protocol is experimental and may contain undiscovered bugs. Regulatory environments may change adversely. No guarantees are made about exchange listings, price performance, or technical functionality.