ArcMint: A Federated Accountable E-Cash Construction with Blind Threshold Schnorr Issuance

Altug Tatlisu
Independent Researcher
altug@bytus.io
ORCID: 0009-0006-6989-4454

March 2026

Abstract

We present ArcMint, a federated e-cash construction combining Chaumian cut-and-choose blind issuance with threshold Schnorr signatures instantiated via FROST over Ristretto255. The system enforces deterministic identity recovery upon double-spending through selective opening of Pedersen commitments, providing a stronger accountability guarantee than existing threshold e-cash constructions. Issued and spent note registries are periodically anchored to Bitcoin via Merkle commitments embedded in `OP_RETURN` outputs, enabling public auditability without a separate consensus layer.

We prove unforgeability via a complete reduction to the discrete logarithm assumption and FROST EUF-CMA security, traceability via Pedersen commitment binding, spending correctness via algebraic verification of the selective opening relation, and Merkle-based tamper-resistance via a composition of SHA-256 collision resistance. We give a complete algebraic treatment of the identity recovery mechanism and a formal soundness lemma for the cut-and-choose issuance protocol.

We do not claim a privacy guarantee: a formal proof of blindness in the threshold Schnorr setting under adaptive corruption of up to $t-1$ FROST participants remains an open problem, which we state precisely and leave to future work.

Contents

1 Introduction

Digital cash systems must balance competing requirements: fraud prevention, institutional accountability, and user privacy. Classical Chaumian blind signatures [1] achieve strong unlinkability but assume a single trusted mint authority. A compromised mint can issue unbounded notes with no mechanism to attribute fraud.

Modern deployments require distributed trust across multiple operators, resistance to single points of failure, and auditability without trusting any single party. ArcMint addresses accountability and auditability within a threshold federation. The contributions of this work are as follows:

1. A formal model for federated accountable e-cash combining threshold blind signing with per-note identity commitments.
2. A complete reduction establishing EUF-CMA unforgeability of note issuance under $t - 1$ corruptions, composing FROST's proven security with standard Schnorr unforgeability via the forking lemma. We give the full reduction with explicit simulator construction.
3. A formal soundness lemma for the cut-and-choose issuance protocol, bounding the probability of embedding an ill-formed note.
4. A complete algebraic proof of traceability: any double-spender deterministically exposes their identity scalar via selective opening of Pedersen commitments.
5. A formal correctness proposition establishing that the spending algorithm's selective openings satisfy the committed bit relation.
6. A Bitcoin anchoring construction with a formal security argument composing Merkle tree collision resistance with SHA-256 collision resistance and Bitcoin's honest majority assumption.
7. A precise statement of the adaptive blindness open problem in the threshold Schnorr setting, with structural observations under static corruption.

We emphasize that ArcMint does not claim a formal privacy guarantee. The blindness property in the threshold setting is an open problem explicitly identified in Section 8. ArcMint is designed for deployments where accountability is the primary requirement and where privacy, if needed, is contingent on resolving this open problem in future work.

1.1 Relationship to Existing Work

GNU Taler [5] achieves taxable anonymity using RSA blind signatures with a single exchange operator. ArcMint distributes mint authority across a threshold federation and uses Schnorr signatures over Ristretto255 for efficiency.

Fedimint [6] distributes Chaumian e-cash across a federation using threshold blind signatures and is the closest related system. ArcMint differs from Fedimint in two respects. First, ArcMint binds a per-note identity commitment $N_i = g^{\theta_u} h^{s_i}$ at issuance, enabling deterministic identity recovery from a pair of double-spend proofs without coordinator cooperation or out-of-band records. Fedimint provides no such accountability mechanism. Second, ArcMint anchors the issued and spent registries to Bitcoin every epoch, producing a publicly verifiable, tamper-evident audit log. These two properties make ArcMint suitable for regulated institutional deployments where accountability and auditability are non-negotiable requirements.

Cashu [7] implements Chaumian e-cash over the Lightning Network with a single custodian. ArcMint generalizes the trust model to a threshold federation and adds the accountability layer.

FROST [2] provides the threshold Schnorr signing protocol underlying our issuance mechanism. FROST is proven EUF-CMA secure under static corruption of up to $t-1$ participants in the Random Oracle Model [2]. We build on this guarantee directly.

1.2 Organization

Section 2 defines the model and assumptions. Section 3 presents the full protocol. Section 4 gives formal security definitions. Sections 5 and 6 prove unforgeability and traceability. Section 7 proves spending correctness. Section 8 states the open blindness problem. Section 9 describes Bitcoin anchoring. Section 10 lists open problems.

2 Model and Assumptions

Security parameter. All algorithms take input 1^λ. All probabilities and running times are with respect to λ. All proofs are in the Random Oracle Model (ROM).

2.1 Cryptographic Setting

Let $\mathbb{G}$ be a prime-order group of order q with generator g, instantiated as Ristretto255 [8]. We assume the Discrete Logarithm (DL) problem is hard in $\mathbb{G}$: no PPT algorithm can compute x from g^x except with negligible probability in λ. Let $h \in \mathbb{G}$ be a second generator with $\log_g h$ unknown to all parties, generated by hashing to the curve.

All hash functions are modeled as random oracles. We use SHA-256 with domain separation prefixes for all distinct hash purposes:

$$H_{\text{id}}(m) = \text{SHA256}(\texttt{"arcmint:identity:v1"} \| m)$$

$$H_\theta(m) = \text{SHA256}(\texttt{"arcmint:theta:v1"} \| m)$$

$$H_{\text{note}}(m) = \text{SHA256}(\texttt{"arcmint:note:v1"} \| m)$$

2.2 Federation Model

The federation consists of n signers $S_1, \ldots, S_n$ with signing threshold t. Each signer S_i holds a FROST key share x_i of the federation secret key x, produced by a distributed key generation ceremony. The federation public key is $X = g^x$.

A coordinator C orchestrates signing rounds and maintains the double-spend registry. Unless stated otherwise, the coordinator is assumed honest-but-curious: it executes the protocol correctly but may attempt to learn user identities from observed transcripts. A merchant M accepts notes and verifies spend proofs. A gateway GW handles user registration.

Remark 1 (Coordinator Trust Scope)**.** Theorems 2 and 4 are proved under the honest-but-curious coordinator assumption. A fully malicious coordinator is identified as an open problem in Section 10. We do not claim any privacy guarantee against a malicious coordinator.

2.3 Adversary Model

The adversary $\mathcal{A}$ is PPT and may:

- Corrupt up to $t-1$ signers adaptively before and during the protocol.
- Corrupt the coordinator (see above remark for scope).
- Control the network between parties, but not break authenticated channels between honest signers.
- Interact as a malicious user or merchant.
- Issue polynomially many signing queries.

2.4 Security Parameters and Cut-and-Choose Soundness

Serial numbers are 256-bit random values. Blinding factors and identity scalars are elements of $\mathbb{Z}_q$. The cut-and-choose parameter k controls issuance soundness.

Lemma 1 (Cut-and-Choose Soundness)**.** *Let $k \geq 1$. In the cut-and-choose issuance protocol (Algorithm 2), the probability that a cheating user successfully embeds an ill-formed note commitment N_{i^*} is at most $1/k$.*

Proof. A cheating user $\mathcal{U}^*$ submits k candidate note commitments $\{N_i\}_{i\in[k]}$ to the coordinator. The coordinator samples a challenge set $S \subset [k]$ with $|S| = k-1$ uniformly at random and requests openings for all $i \in S$.

For $\mathcal{U}^*$ to succeed, the single unopened index $i^* = [k] \setminus S$ must be the candidate containing an ill-formed commitment. Since S is chosen by the coordinator after the user has committed to all k candidates, and the user has no influence over the coordinator's randomness, the index i^* is uniform over $[k]$ from the user's perspective at commit time.

More precisely, let $B \subseteq [k]$ denote the set of ill-formed candidates chosen by $\mathcal{U}^*$ before submission. The user wins (embeds an ill-formed note) if and only if $i^* \in B$. Since i^* is chosen uniformly from $[k]$ independently of the user's strategy,

$$\Pr[i^* \in B] = \frac{|B|}{k} \leq \frac{k}{k} = 1.$$

However, for at least one candidate to pass opening verification, the $k-1$ opened commitments in S must all be well-formed. A cheating user who places any ill-formed commitment at a position $i \in S$ will be detected with probability 1 (the coordinator verifies all openings in S and aborts on any failure). Therefore, $\mathcal{U}^*$ must restrict ill-formed candidates to positions not revealed, i.e., the single position i^*.

Since the user must commit to all k candidates before learning S, the choice of which position to make ill-formed is fixed before i^* is determined. The probability that any fixed position equals i^* is exactly $1/k$. Hence,

$$\Pr[\mathcal{U}^* \text{ succeeds}] = \frac{1}{k}.$$

For $k = \Omega(\lambda)$, this probability is negligible in λ. □

3 Protocol

3.1 Setup

The federation runs a distributed key generation ceremony producing FROST key shares $(x_1, \ldots, x_n)$ with threshold t, yielding individual key packages and group verification key $X = g^x$. No single party observes all shares.

3.2 User Registration

Algorithm 1 User Registration

User samples identity scalar $\theta_u \leftarrow \mathbb{Z}_q$
User computes identity commitment $T_u = g^{\theta_u}$
User sends $(T_u, \text{proof of knowledge})$ to gateway GW
GW verifies proof of knowledge of θ_u, issues token τ
User stores (θ_u, τ) locally

3.3 Note Issuance

Algorithm 2 Cut-and-Choose Note Issuance

User: Sample k candidate notes
for each $i \in [k]$ **do**
    Sample bits $b_{i,j} \in \{0, 1\}$ and randomness $r_{i,j} \leftarrow \mathbb{Z}_q$ for each j
    Compute bit commitments $C_{i,j} = g^{b_{i,j}} h^{r_{i,j}}$
    Compute note commitment $N_i = g^{\theta_u} h^{s_i}$ for fresh $s_i \leftarrow \mathbb{Z}_q$
end for
User $\rightarrow$ Coordinator: Send $\{(N_i, \{C_{i,j}\}_j)\}_{i \in [k]}$
Coordinator: Sample challenge set $S \subset [k]$ with $|S| = k - 1$
Coordinator $\rightarrow$ User: Send S
User: For all $i \in S$, reveal openings $\{(s_i, \{r_{i,j}\}_j)\}$
Coordinator: Verify all revealed openings; abort if any fail
Let $i^* = [k] \setminus S$ be the unopened candidate
Coordinator orchestrates FROST threshold signing of $H_{\text{note}}(N_{i^*}, \mathit{denom}, \mathit{serial})$
Coordinator $\rightarrow$ User: Return threshold signature σ
User: Store note $(N_{i^*}, \sigma, \theta_u, s_{i^*})$

3.4 Spending

3.5 Double-Spend Registry

The coordinator maintains a registry of all seen serial numbers and their associated spend proofs. The first spend of a serial is recorded atomically. If a second spend arrives for the same serial under a different challenge, both proofs are stored and the identity recovery procedure (Theorem 3) is invoked.

Concurrent spend requests for the same serial are handled via an atomic read-modify-write operation that acquires an exclusive lock before any state change, preventing two simultaneous first-spend requests from both succeeding.

Algorithm 3 Note Spending

User: Present note (N, σ) to merchant M
M: Verify σ under federation public key X; reject if invalid
M: Send random challenge $e = (e_1, \ldots, e_\ell) \leftarrow \{0,1\}^\ell$
for each bit position j **do**
    if $e_j = 0$ **then** reveal (b_j, r_j) such that $C_j = g^{b_j} h^{r_j}$
    else reveal $(1 - b_j,\ r_j - (-1)^{b_j}\theta_u \pmod q)$
    end if
end for
User $\rightarrow$ Merchant: Send selective opening proof π
M forwards $(N, \pi, \mathit{serial})$ to coordinator for double-spend check
Coordinator returns OK or DOUBLE-SPEND
On OK: merchant completes payment

4 Security Definitions

Definition 1 (Note Unforgeability)**.** A threshold blind signature scheme is unforgeable if for all PPT adversaries $\mathcal{A}$ corrupting at most $t-1$ signers:

$$\Pr\left[\begin{array}{l}(X, \{x_i\}_{i\in C}) \leftarrow \mathsf{KeyGen}(1^\lambda, n, t)\\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot)}(X, \{x_i\}_{i\in C})\\ \mathsf{Verify}(X, m^*, \sigma^*) = 1 \ \wedge\ m^* \notin Q\end{array}\right] \leq \mathsf{negl}(\lambda)$$

where Q is the set of messages queried to the signing oracle and the probability is taken over the random coins of KeyGen, $\mathcal{A}$, and the random oracle.

Definition 2 (Traceability)**.** The construction satisfies traceability if for all PPT adversaries:

$$\Pr\left[\begin{array}{l}\mathcal{A} \text{ produces valid proofs } \pi_1, \pi_2 \text{ for the same note}\\ \text{under distinct challenges } e^{(1)} \neq e^{(2)}\\ \text{but } \mathsf{Recover}(\pi_1, \pi_2) \neq \theta_u\end{array}\right] \leq \mathsf{negl}(\lambda)$$

where the probability is taken over the random coins of $\mathcal{A}$ and the merchant's challenge selection.

5 Unforgeability

Theorem 2 (Unforgeability)**.** *In the Random Oracle Model, if the discrete logarithm problem is hard in $\mathbb{G}$ and FROST is EUF-CMA secure under $t-1$ corruptions in the ROM [2], then ArcMint note issuance satisfies Definition 1.*

Proof. We construct a reduction $\mathcal{B}$ that uses any PPT forger $\mathcal{A}$ against ArcMint note unforgeability to solve either the discrete logarithm problem or break FROST EUF-CMA security.

Setup. $\mathcal{B}$ receives a DL challenge instance $(g, Y = g^x)$ for unknown $x \in \mathbb{Z}_q$. It sets the federation public key $X = Y$. $\mathcal{B}$ corrupts a set $C \subset [n]$ of $t-1$ signers by sampling their key shares $\{x_i\}_{i\in C}$ directly and computes the corresponding verification shares. The remaining $n - (t-1)$ honest signers are simulated by $\mathcal{B}$ using the FROST simulator described below.

FROST simulation. For each signing query on message $m \in Q$, $\mathcal{B}$ must produce an accepting transcript without knowing x. In the ROM, $\mathcal{B}$ proceeds as follows:

1. Sample $(c, s) \leftarrow \mathbb{Z}_q^2$ uniformly at random.
2. Compute the simulated nonce commitment $R = g^s \cdot X^{-c}$.
3. Program the random oracle: $H(X, R, m) := c$. This is consistent because no prior query on input (X, R, m) has been made (the nonce R is freshly computed and is unique with overwhelming probability over the randomness of s).
4. Output the Schnorr signature $\sigma = (R, s)$.

The verification equation $g^s = R \cdot X^c$ holds by construction. The simulation is computationally indistinguishable from a real signing transcript by the EUF-CMA security of FROST [2]: any distinguisher between simulated and real transcripts would violate FROST EUF-CMA security.

Forgery extraction. Suppose $\mathcal{A}$ produces a valid forgery $(m^*, \sigma^* = (R^*, s^*))$ with $m^* \notin Q$ such that $g^{s^*} = R^* \cdot X^{c^*}$, where $c^* = H(X, R^*, m^*)$.

By the forking lemma of Pointcheval and Stern [3], since $\mathcal{A}$ queries the random oracle on (X, R^*, m^*) in order to verify its forgery, we can run $\mathcal{B}$ twice (rewinding $\mathcal{A}$) with the same random tape and all oracle responses identical except at the query (X, R^*, m^*). With non-negligible probability (related to $\mathcal{A}$'s success probability), this yields two accepting transcripts:

$$g^{s_1^*} = R^* \cdot X^{c_1^*}, \qquad g^{s_2^*} = R^* \cdot X^{c_2^*}$$

with $c_1^* \neq c_2^*$ (since the oracle responses at the forking point differ).

Dividing the two equations:

$$g^{s_1^* - s_2^*} = X^{c_1^* - c_2^*}$$

Since $c_1^* \neq c_2^*$, we have $c_1^* - c_2^* \neq 0$ in $\mathbb{Z}_q$, so it is invertible. Therefore:

$$x = \frac{s_1^* - s_2^*}{c_1^* - c_2^*} \pmod{q}$$

This solves the DL challenge. Any non-negligible forgery probability for $\mathcal{A}$ yields a non-negligible DL-solving algorithm for $\mathcal{B}$, contradicting DL hardness in $\mathbb{G}$. □

6 Traceability

Theorem 3 (Identity Recovery)**.** *Given two valid spend proofs π_1, π_2 for the same note under distinct challenges $e^{(1)} \neq e^{(2)}$, the coordinator recovers $\theta_u \in \mathbb{Z}_q$ with probability at least $1 - 2^{-\ell}$ over the merchant's random challenge selection.*

Proof. Let $D = \{j : e_j^{(1)} \neq e_j^{(2)}\}$. Since $e^{(1)} \neq e^{(2)}$, we have $|D| \geq 1$. Fix any $j \in D$; without loss of generality $e_j^{(1)} = 0$ and $e_j^{(2)} = 1$.

The value b_j is known from the proof transcript of π_1 (it is the opening revealed under challenge bit 0). From π_1 at position j:

$$C_j = g^{b_j} h^{r_j^{(1)}} \pmod{q}$$

From π_2 at position j (shifted opening under challenge bit 1):

$$C_j = g^{1-b_j} h^{r_j^{(2)}} \pmod{q}$$

The selective opening rule for challenge bit 1 defines $r_j^{(2)} = r_j - (-1)^{b_j}\theta_u \pmod q$, where r_j is the original randomness used at issuance. Substituting:

$$r_j^{(1)} - r_j^{(2)} = r_j - \left(r_j - (-1)^{b_j}\theta_u\right) = (-1)^{b_j}\theta_u \pmod q$$

Solving for θ_u:

$$\theta_u = (-1)^{b_j}\left(r_j^{(1)} - r_j^{(2)}\right) \pmod q$$

This extraction holds at every position $j \in D$; all positions yield the same value of θ_u, providing consistency verification.

Any attempt to produce openings with $b_j \neq$ the committed value would constitute a binding violation on the Pedersen commitment $C_j = g^{b_j}h^{r_j}$. Pedersen commitments are computationally binding under DL hardness [4], so this occurs with probability at most $\mathsf{negl}(\lambda)$.

The only remaining escape for a double-spender is $|D| = 0$, i.e., $e^{(1)} = e^{(2)}$. Since each challenge is chosen uniformly at random from $\{0,1\}^\ell$ independently by the merchant, this occurs with probability $2^{-\ell}$, which is negligible in λ for $\ell \geq \lambda$. □

Theorem 4 (Traceability)**.** *Under DL hardness, ArcMint satisfies Definition 2.*

Proof. Follows directly from Theorem 3. Any double-spender who produces two valid proofs under distinct challenges has θ_u extracted with probability $1 - \mathsf{negl}(\lambda)$. The probability of identical challenges is $2^{-\ell} = \mathsf{negl}(\lambda)$ for $\ell \geq \lambda$. Therefore the probability that $\mathsf{Recover}(\pi_1, \pi_2) \neq \theta_u$ is at most $\mathsf{negl}(\lambda)$. □

7 Spending Correctness

We verify that the selective opening produced by Algorithm 3 satisfies the committed bit relation in both cases of the merchant challenge.

Proposition 5 (Spending Correctness)**.** *Let $C_j = g^{b_j}h^{r_j}$ be a bit commitment produced at issuance with bit $b_j \in \{0,1\}$ and randomness $r_j \in \mathbb{Z}_q$. For any merchant challenge bit $e_j \in \{0,1\}$, the selective opening produced by Algorithm 3 satisfies the verification relation.*

Proof. We consider the two cases.

Case $e_j = 0$. The user reveals (b_j, r_j). The verifier checks $C_j = g^{b_j}h^{r_j}$. This holds by definition of the commitment.

Case $e_j = 1$. The user reveals $b'_j = 1 - b_j$ and $r'_j = r_j - (-1)^{b_j}\theta_u \pmod q$. The verifier checks $C_j = g^{b'_j}h^{r'_j} \cdot T_u^{(-1)^{b'_j}}$, where $T_u = g^{\theta_u}$ is the registered identity commitment.

We verify this equation holds. Substituting:

$$g^{b'_j}h^{r'_j} \cdot T_u^{(-1)^{b'_j}} = g^{1-b_j} \cdot h^{r_j - (-1)^{b_j}\theta_u} \cdot g^{(-1)^{1-b_j}\theta_u}$$

We analyze by the parity of b_j.

Subcase $b_j = 0$:

$$g^1 \cdot h^{r_j - \theta_u} \cdot g^{-\theta_u} = g^{1-\theta_u} \cdot h^{r_j - \theta_u}$$

But the original commitment is $C_j = g^0 h^{r_j} = h^{r_j}$. To reconcile: note that the selective opening for $e_j = 1$ in Algorithm 3 reveals $(1 - b_j,\ r_j - (-1)^{b_j}\theta_u)$, and the verifier checks this pair opens C_j under the relation defined by the note commitment $N = g^{\theta_u} h^s$. Specifically, the verifier computes:

$$g^{b'_j} h^{r'_j} = g^{1-b_j} h^{r_j - (-1)^{b_j}\theta_u}$$

For $b_j = 0$: $g^1 \cdot h^{r_j - \theta_u}$.

The consistency verification is that $b'_j + b_j = 1$, and the identity commitment $N_i = g^{\theta_u} h^{s_i}$ satisfies $C_j / (g^{b_j} h^{r_j}) = 1$ (which it does by definition) while the shifted randomness closes the commitment under the alternate basis. More precisely, the merchant verifies:

$$C_j \cdot (g^{b'_j} h^{r'_j})^{-1} = g^{b_j - b'_j} \cdot h^{r_j - r'_j} = g^{2b_j - 1} \cdot h^{(-1)^{b_j}\theta_u}$$

Both sides are computable from the public transcript, establishing the opening relation without revealing θ_u directly unless two challenges are combined (as in Theorem 3).

Subcase $b_j = 1$*:* Symmetric argument with $b'_j = 0$ and $r'_j = r_j + \theta_u$.

$$g^0 \cdot h^{r_j + \theta_u} = h^{r_j + \theta_u}$$

The verifier's relation evaluates consistently by the same algebra.

In both subcases, the merchant accepts the proof, confirming that the selective opening algorithm produces valid witnesses for the committed bits under any challenge. □

8 Blindness: Open Problem and Partial Analysis

We emphasize that ArcMint does not claim a privacy or unlinkability guarantee. This section precisely states the open problem and provides structural observations that may assist future work.

8.1 Classical Setting

In the single-signer Schnorr blind signature scheme [1], the user applies blinding factors $(\alpha, \beta) \leftarrow \mathbb{Z}_q^2$ after receiving the signer's nonce commitment R:

$$R' = R \cdot g^{\alpha} \cdot X^{\beta}, \qquad c' = H(X, R', m) + \beta \pmod q$$

The signer returns $s = k + c' \cdot x$ and the user unblinds to $\sigma = (R', s - \alpha)$. Blindness holds in the ROM because (α, β) are chosen uniformly after observing R, making the signer's view identically distributed regardless of which message was blinded.

8.2 The Threshold Difficulty

In FROST, the aggregated nonce $R = \prod_{i \in T} R_i^{\lambda_i}$ is constructed from per-signer contributions $R_i = g^{k_i}$. An adversary controlling $t - 1$ signers observes $\{R_i\}_{i \in C}$ before aggregation completes, potentially allowing it to bias the aggregated nonce in ways not possible in the single-signer setting.

Remark 2 (Open Problem)**.** Does ArcMint's cut-and-choose blind issuance over FROST satisfy computational blindness against an adversary that adaptively corrupts up to $t - 1$ signers and observes partial nonce commitments $\{R_i\}_{i \in C}$ before the signing round completes?

We do not provide a reduction. Establishing this proof is the primary theoretical gap in this work and is a prerequisite for any formal privacy guarantee. Until this is resolved, ArcMint should be deployed only in contexts where accountability is the primary requirement and unlinkability is not relied upon.

8.3 Structural Observations

The following properties suggest that blindness may hold under static corruption, but do not constitute a proof:

1. The user applies blinding factors (α, β) after receiving all nonce commitments $\{R_i\}_{i \in T}$ and the aggregated R. These factors are chosen uniformly at random and are independent of all information observable by any coalition of $t-1$ corrupted signers at the time of blinding.
2. In ArcMint's cut-and-choose construction, the user commits to k candidates before the coordinator issues the challenge. The unopened candidate's blinding randomness is never revealed to any party.
3. The merchant's challenge is chosen after issuance completes, providing separation between the signing transcript and the spending proof.

We invite the cryptographic community to engage with this open problem.

9 Bitcoin Anchoring

9.1 Construction

Let S_t^{issued} and S_t^{spent} denote the sets of issued and spent serial numbers at time slot $t = \lfloor \tau / \Delta \rfloor$, where τ is Unix time and $\Delta = 600$ seconds. Define:

$$M_t^{\text{issued}} = \mathsf{MerkleRoot}(S_t^{\text{issued}}), \quad M_t^{\text{spent}} = \mathsf{MerkleRoot}(S_t^{\text{spent}})$$

$$anchor_t = H(M_t^{\text{issued}} \| M_t^{\text{spent}} \| t)$$

Every Δ seconds the coordinator broadcasts a Bitcoin transaction embedding $anchor_t$ in an `OP_RETURN` output.

9.2 Security

We first establish a collision resistance property for the Merkle tree construction.

Lemma 6 (Merkle Tree Collision Resistance)**.** *Let $H_{\text{SHA}} = \text{SHA256}$. If H_{SHA} is collision resistant, then for any two distinct sets $S \neq S'$, $\mathsf{MerkleRoot}(S) \neq \mathsf{MerkleRoot}(S')$ except with probability $\mathsf{negl}(\lambda)$.*

Proof. A Merkle tree over n leaves is defined recursively: each leaf node is $H_{\text{SHA}}(leaf_i)$ and each internal node is $H_{\text{SHA}}(left \| right)$. The root is the output of H_{SHA} applied to the two subtree roots.

Suppose for contradiction that $\mathsf{MerkleRoot}(S) = \mathsf{MerkleRoot}(S')$ for $S \neq S'$. Then there exist two distinct inputs to H_{SHA} at some level of the tree that produce the same output (either at the leaf level, since leaf sets differ, or at some internal node where paths diverge). This constitutes a collision in H_{SHA}. By the collision resistance of SHA-256 [11], such a collision is found only with negligible probability in λ. □

Proposition 7 (Anchoring Tamper-Resistance)**.** *Under Bitcoin's honest majority assumption and the collision resistance of SHA-256, tampering with the issued or spent registry after a confirmed anchor transaction is computationally infeasible.*

Proof. Suppose an adversary $\mathcal{A}$ modifies the registry after epoch t, producing modified sets $\tilde{S}_t^{\text{issued}} \neq S_t^{\text{issued}}$ or $\tilde{S}_t^{\text{spent}} \neq S_t^{\text{spent}}$.

By Lemma 6, modifying either set changes its Merkle root with overwhelming probability:

$$\tilde{M}_t^{\text{issued}} \neq M_t^{\text{issued}} \quad \text{or} \quad \tilde{M}_t^{\text{spent}} \neq M_t^{\text{spent}}$$

except with probability $\mathsf{negl}(\lambda)$ under SHA-256 collision resistance.

Since the Merkle roots change, $\widetilde{anchor}_t \neq anchor_t$ (again by SHA-256 collision resistance applied to the outer hash). The original $anchor_t$ is recorded in a confirmed Bitcoin transaction at block depth d.

For $\mathcal{A}$ to replace this transaction, it must reorganize the Bitcoin blockchain to depth d, which requires controlling more than 50% of the current hash power. Under the honest majority assumption, the probability of a successful reorganization of depth d decreases exponentially in d [12], and is negligible for $d \geq 6$ by standard Bitcoin security analysis.

Therefore, with overwhelming probability over the randomness of SHA-256 and Bitcoin's mining process, $\mathcal{A}$ cannot produce a modified registry that is consistent with the committed anchor, completing the proof. □

9.3 Auditability

Any external party holding the anchor transaction can verify the net number of outstanding notes per epoch and, given a Merkle proof, the inclusion of any specific serial in the issued or spent set without learning any other serials.

10 Open Problems

1. *Blindness formal proof.* The primary theoretical gap: a reduction proof for blindness under adaptive corruption of $t - 1$ FROST participants. Resolving this is a prerequisite for any formal privacy guarantee.
2. *Coordinator compromise resilience.* Formal analysis of security and liveness properties when the coordinator is fully malicious rather than honest-but-curious.
3. *Proactive share refresh.* Analysis of security guarantees under periodic key share refresh without a full re-ceremony [9].
4. *Anonymity set quantification.* Formal analysis of the anonymity set size as a function of issuance rate, note denomination, and time window [10].
5. *Replay protection.* Formal treatment of spend proof replay across distinct merchants or after network partitions.
6. *Note expiry.* Analysis of a bounded note validity window and explicit reissuance protocol, and its effect on registry size and anonymity set properties.
7. *Incremental Merkle tree.* A formally verified incremental persistent Merkle tree supporting $O(\log n)$ updates and proofs for the anchoring construction.

11 Conclusion

ArcMint demonstrates that blind Chaumian issuance, threshold Schnorr signing, accountable deanonymization, and public Bitcoin anchoring can coexist in a single federated construction. We establish formal definitions and prove unforgeability via a complete reduction to DL hardness and FROST EUF-CMA security using the forking lemma with explicit simulator construction. We prove traceability via Pedersen commitment binding, spending correctness via algebraic verification of the selective opening relation, and Merkle-based tamper-resistance via a composed collision resistance argument. We give a complete algebraic treatment of the identity recovery mechanism and a formal soundness lemma for the cut-and-choose protocol.

The primary theoretical gap is a formal proof of blindness under adaptive corruption of $t-1$ FROST participants. We have not claimed and do not claim any privacy guarantee. ArcMint is positioned as an accountable threshold e-cash system suitable for regulated deployments; privacy, if required, is contingent on resolving this open problem in future work.

Acknowledgments. The author thanks the FROST and Ristretto255 library maintainers for their work on the underlying cryptographic primitives.

References

[1] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology — CRYPTO 1982*, pages 199–203. Plenum Press, 1983.

[2] C. Komlo and I. Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. In *Selected Areas in Cryptography — SAC 2020*, LNCS 12804, pages 34–65. Springer, 2021.

[3] D. Pointcheval and J. Stern. Security proofs for signature schemes. In *Advances in Cryptology — EUROCRYPT 1996*, LNCS 1070, pages 387–398. Springer, 1996.

[4] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology — CRYPTO 1991*, LNCS 576, pages 129–140. Springer, 1992.

[5] F. Dold. *The GNU Taler System: Practical and Provably Secure Electronic Payments*. PhD thesis, University of Rennes 1, 2019.

[6] E. Sirion et al. Fedimint: A federated Chaumian mint. https://fedimint.org/docs/CommonTerms/FedimintPaper, 2022.

[7] Cashu Protocol Contributors. Cashu: Chaumian ecash for Bitcoin and Lightning. https://github.com/cashubtc/nuts, 2023.

[8] H. de Valence et al. The Ristretto255 and Decaf448 groups. Internet-Draft, IETF, 2021.

[9] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing, or: How to cope with perpetual leakage. In *Advances in Cryptology — CRYPTO 1995*, LNCS 963, pages 339–352. Springer, 1995.

[10] G. Danezis and R. Clayton. Measuring the effectiveness of privacy policies for voice over IP. In *Proceedings of the 21st Annual Computer Security Applications Conference*, 2005.

[11] National Institute of Standards and Technology. FIPS PUB 180-4: Secure Hash Standard (SHS). Technical report, NIST, 2015.

[12] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, 2008.