title

Zach Kelling

Satschel, Inc.

April 2026

Abstract

We present a non-custodial MPC custody architecture for regulated digital securities, replacing Shamir Secret Sharing (SSS) — where the full private key is reconstructed in server memory on every signing operation — with CGGMP21 threshold ECDSA where the key is *never* reconstructed. The system binds each user's key shard to a FIDO2/WebAuthn passkey, ensuring the ATS operator holds only one of three shards and provably cannot sign transactions unilaterally. We formalize the non-custodial guarantee as a threshold security property: no coalition of fewer than t parties can produce a valid signature, and the operator's shard count is permanently below t. Backup and recovery use Shamir $(3, 5)$ splitting of the passkey-encrypted shard to geographically distributed encrypted S3 buckets. HSM co-signing provides an additional authorization layer for institutional settlements exceeding configurable thresholds. The architecture is deployed on the Lux Liquid Threshold cluster and formally verified against the CGGMP21 and FROST proof libraries from the Lux Network's post-quantum cryptographic stack.

Contents

1 Introduction

Custody of digital securities under SEC Rule 15c3-3 requires either qualified custodian status or a non-custodial architecture where the operator cannot unilaterally access client assets. The prior Satschel production system labeled its key management as "MPC" but implemented Shamir Secret Sharing: the full Ed25519 private key was reconstructed from two shards in server memory on every signing operation [3]. Both server-side shards resided in the same MongoDB database encrypted with a single GCP Cloud KMS key, giving the operator de facto custody.

This paper presents the replacement architecture deployed on the Lux Liquid Threshold [4]: a 2-of-3 CGGMP21 [10] threshold ECDSA scheme where the private key is never reconstructed. Each party computes a partial signature independently; partial signatures are combined into a valid ECDSA signature without any single party learning the full key.

The non-custodial guarantee is enforced at three levels:

1. **Protocol level**: CGGMP21 threshold signing with $t = 2$, $n = 3$.
2. **Shard distribution**: User holds shard 1 (device Secure Enclave), ATS holds shard 2 (Cloud HSM), backup shard 3 is encrypted with the user's WebAuthn passkey public key.
3. **Formal verification**: The CGGMP21 and FROST proof libraries [7, 8] establish correctness of the threshold signing protocol.

1.1 Contributions

1. Formal comparison of SSS reconstruction vs. true MPC threshold signing for regulated securities custody.
2. Passkey-bound shard encryption using ECDH key agreement over the WebAuthn P-256 credential, preventing operator access to the backup shard.
3. Key refresh protocol that rotates all shards without revealing the underlying secret or changing the public key.
4. Backup and recovery via $(3, 5)$ Shamir splitting of the passkey-encrypted shard to encrypted S3 buckets across three geographic regions.
5. HSM co-signing layer for institutional settlement above configurable thresholds.

2 Threat Model

We consider four adversary classes:

Definition 1 (Adversary Classes)**.**
1. ***Compromised operator****: Attacker controls the ATS server and shard 2. Must not be able to sign.*
2. ***Device theft****: Attacker obtains the user's device (shard 1) but cannot pass biometric authentication.*
3. ***Database breach****: Attacker reads all MongoDB/PostgreSQL records including encrypted backup blobs.*
4. ***Quantum adversary****: Attacker with access to a cryptographically relevant quantum computer (CRQC).*

3 SSS vs. True MPC: A Formal Comparison

3.1 Shamir Secret Sharing (Current, Flawed)

The Satschel production system splits each Ed25519 private key sk into three shards (s_1, s_2, s_3) using Shamir's $(2, 3)$ scheme over GF(256). To sign a transaction:

1. The server retrieves shards s_2 and s_3 from MongoDB.
2. Both shards are decrypted with the same GCP KMS key.
3. $\mathsf{sk} \leftarrow \texttt{reconstructShamirSecret}(s_2, s_3)$.
4. The full private key exists in server RAM.
5. The transaction is signed with sk.
6. sk is discarded (but was fully materialized).

Property 1 (SSS Custody)**.** *Under the SSS architecture, the operator holds shards s_2 and s_3 in the same database. Since $t = 2$, the operator can reconstruct* sk *at any time without user involvement. The operator is a* custodian*.*

3.2 CGGMP21 Threshold ECDSA (Target)

In the CGGMP21 protocol [10], each party P_i holds a *key share* x_i such that the secret key $x = \sum_{i \in S} \lambda_i x_i$ for any qualified subset S of size t. Signing proceeds without reconstructing x:

1. Each party computes a partial signature σ_i using their share x_i and a jointly generated nonce.
2. Partial signatures are combined: $\sigma = \texttt{combine}(\sigma_1, \sigma_2)$.
3. The combined signature σ is a valid ECDSA signature under the joint public key.

Theorem 1 (Non-Custodial Guarantee)**.** *Under the CGGMP21* $(2, 3)$ *threshold scheme with shard distribution $\{P_{user}, P_{ATS}, P_{backup}\}$ where P_{backup} is encrypted with the user's passkey: the ATS operator controls exactly one shard (P_{ATS}). Since $t = 2$ and the operator holds $1 < t$ shards, the operator* cannot *produce a valid signature unilaterally.*

Proof. By the threshold security property of CGGMP21 [10, 7], any coalition of fewer than t parties gains zero information about the secret key beyond what is implied by the public key. The operator holds $|S_{\text{op}}| = 1 < 2 = t$, so S_{op} is not a qualified subset. The backup shard P_{backup} is encrypted under the user's WebAuthn P-256 public key via ECDH; the operator does not possess the corresponding private key (held in the user's Secure Enclave). Therefore the operator's effective shard count remains 1. □

Property	**Shamir SSS (Old)**	**CGGMP21 MPC (New)**
Key reconstructed?	Yes (every sign)	Never
Operator can sign alone?	Yes (holds 2 of 3)	No (holds 1 of 3)
User can exit independently?	No	Yes (shard 1 + passkey)
Backup shard accessible by operator?	Yes (same DB, same KMS key)	No (passkey-encrypted)
Identifiable abort?	N/A	Yes (CGGMP21 property)
Protocol	Lagrange interpolation	MtA + Paillier homomorphic
Custody status	Custodian	Co-signer

Table 1: SSS vs. CGGMP21 MPC for regulated securities custody.

4 Passkey-Bound Shard Protection

4.1 WebAuthn Credential Binding

Each user registers a FIDO2/WebAuthn credential during onboarding. The credential's P-256 public key $\mathsf{pk}_{\text{passkey}}$ is stored in the IAM identity record [9]. The corresponding private key $\mathsf{sk}_{\text{passkey}}$ never leaves the user's Secure Enclave (iOS) or TPM (Android/desktop).

4.2 Shard Encryption via ECDH

The backup shard s_3 is encrypted using ECDH key agreement between an ephemeral P-256 keypair $(e, E = eG)$ and the user's passkey public key $\mathsf{pk}_{\text{passkey}}$:

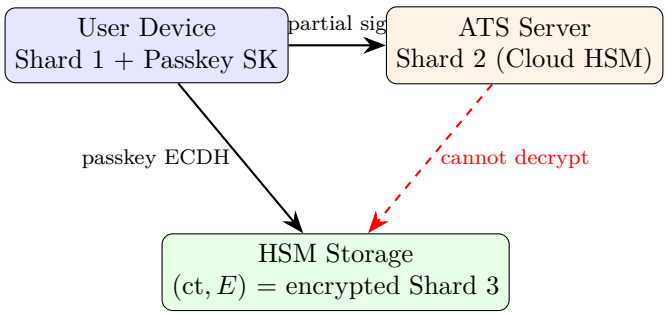
$$\text{shared} = \text{ECDH}(e, \mathsf{pk}_{\text{passkey}}) = e \cdot \mathsf{pk}_{\text{passkey}} \tag{1}$$

$$k = \text{HKDF-SHA256}(\text{shared}, \text{“mpc-backup-v1”}) \tag{2}$$

$$\text{ct} = \text{ChaCha20-Poly1305}(k, s_3) \tag{3}$$

The HSM stores the tuple (ct, E). To decrypt, the user performs ECDH using their passkey's private key: $\text{shared}' = \mathsf{sk}_{\text{passkey}} \cdot E = \text{shared}$.

Theorem 2 (Operator Cannot Decrypt Backup)**.** *The operator stores (ct, E) but does not possess $\mathsf{sk}_{passkey}$. Under the elliptic curve decisional Diffie-Hellman (ECDDH) assumption on P-256, the operator cannot compute the shared secret and therefore cannot decrypt s_3.*



5 Key Refresh Protocol

Key refresh allows all shards to be rotated without changing the joint public key or revealing the secret. This is critical for:

- Proactive security: limiting the window of exposure if a shard is compromised.
- Device migration: user moves to a new phone.
- Regulatory compliance: periodic key rotation mandates.

Definition 2 (Key Refresh)**.** *A key refresh protocol takes shares (x_1, x_2, x_3) of secret x and produces new shares (x'_1, x'_2, x'_3) of the* same *secret x, such that knowledge of any old share x_i and any new share x'_j (for $i \neq j$) reveals nothing about x.*

The CGGMP21 key refresh [10] proceeds as follows:

1. Each party P_i generates a random polynomial $f_i(X)$ of degree $t-1$ with $f_i(0) = 0$.
2. P_i sends $f_i(j)$ to each P_j via authenticated channels.
3. Each party updates: $x'_i = x_i + \sum_j f_j(i)$.
4. The joint secret remains $x = \sum_i \lambda_i x'_i$ (the random contributions cancel at $X = 0$).

Theorem 3 (Refresh Correctness)**.** *After key refresh, the joint public key $X = xG$ is unchanged, and no party learns any information about another party's new share beyond what is implied by the public key.*

6 Backup and Recovery

The passkey-encrypted backup shard ct is itself split using Shamir $(3, 5)$ over $\text{GF}(2^8)$ and distributed to five encrypted S3 buckets across three geographic regions:

Fragment	Region	Encryption	Bucket
f_1	us-central1 (GCP)	SSE-C (HMAC-SHA256 derived)	`liq-backup-usc1`
f_2	us-east1 (GCP)	SSE-C (HMAC-SHA256 derived)	`liq-backup-use1`
f_3	eu-west1 (GCP)	SSE-C (HMAC-SHA256 derived)	`liq-backup-euw1`
f_4	us-central1 (GCP)	SSE-C (HMAC-SHA256 derived)	`liq-backup-usc1-r`
f_5	us-east1 (GCP)	SSE-C (HMAC-SHA256 derived)	`liq-backup-use1-r`

Table 2: Geographic distribution of backup shard fragments.

Recovery requires the user's passkey (to decrypt the reassembled ciphertext) plus any 3 of the 5 fragments (to reconstruct the ciphertext via Shamir). The ATS operator cannot recover the shard even with access to all 5 fragments because the underlying ciphertext requires the user's passkey private key to decrypt.

Theorem 4 (Recovery Availability)**.** *If at most 2 of the 5 storage regions are simultaneously unavailable, the user can recover shard 3. Formally: for any subset $F \subset \{1, \ldots, 5\}$ with $|F| \leq 2$, the remaining fragments $\{f_i : i \notin F\}$ have $|\{f_i : i \notin F\}| \geq 3$, which suffices for Shamir $(3, 5)$ reconstruction.*

7 HSM Co-Signing for Institutional Settlement

Settlements exceeding a configurable threshold (default: $100,000) require an additional HSM co-signature from a hardware security module. This provides defense in depth: even if both the user's device and the ATS server are compromised, the HSM enforces rate limits, transaction caps, and time-of-day restrictions.

The HSM co-signing flow:

1. ATS receives a matched trade exceeding the threshold.
2. ATS submits a *settlement intent* to the HSM.
3. HSM validates: amount within daily limit, recipient on allowlist, within trading hours.
4. HSM produces a co-signature using its own key share.
5. The co-signature is attached to the MPC partial signature as an attestation.

Supported HSMs: GCP Cloud HSM (PKCS#11), AWS CloudHSM, Zymbit hardware modules (for on-premise deployments).

Property 2 (HSM Rate Limiting)**.** *The HSM enforces a maximum daily settlement volume of $V_{\max}$ per user. Any settlement intent that would cause the cumulative daily volume to exceed $V_{\max}$ is rejected at the HSM level, independent of ATS logic.*

8 Signing Scenarios

Scenario	Shards Used	HSM Required?	Outcome
Normal trade (<$100K)	User(1) + ATS(2)	No	Biometric + auto-co-sign
Large trade (≥$100K)	User(1) + ATS(2)	Yes	+ HSM attestation
User lost device	Passkey(3) + ATS(2)	Depends on amount	iCloud Keychain sync
User exits platform	User(1) + Passkey(3)	No	No ATS involvement
ATS compromised	Attacker has shard 2	N/A	Cannot sign (1 of 3)
Device stolen (no biometric)	Attacker has device	N/A	Secure Enclave locked

Table 3: Signing scenarios under the non-custodial MPC architecture.

9 Formal Verification

The threshold signing protocols are verified against the Lux Network's formal proof libraries:

- `proof-crypto-cggmp21` [7]: Correctness of the CGGMP21 keygen, signing, and key refresh protocols. Identifiable abort: if signing fails, the malicious party is identified.
- `proof-crypto-frost` [8]: Correctness of FROST threshold Schnorr signatures (used for Ed25519 cross-chain transactions).
- `proof-lux-mchain` [4]: Liveness and safety of the Liquid Threshold consensus protocol.

The non-custodial guarantee (Theorem 1) and backup inaccessibility (Theorem 2) are proved in the Liquidity formal verification suite [1] as extensions to the base CGGMP21 proofs.

10 Post-Quantum Considerations

The current deployment uses ECDSA (secp256k1) for EVM compatibility. The Lux EthFalcon post-quantum cryptographic stack [6] provides a migration path:

1. **Hybrid attestation**: Each ECDSA transaction carries a parallel ML-DSA-65 (FIPS 204) attestation. If ECDSA is broken by a quantum adversary, the ML-DSA attestation remains valid.
2. **Threshold lattice signatures**: The Ringtail protocol [5] provides threshold Ring-LWE signatures as a drop-in replacement for threshold ECDSA.
3. **Key migration**: The key refresh protocol enables seamless transition from ECDSA to post-quantum schemes without changing wallet addresses (address derived from a commitment to both classical and post-quantum public keys).

11 Conclusion

We have presented a non-custodial MPC custody architecture that replaces Shamir Secret Sharing with CGGMP21 threshold ECDSA for regulated digital securities. The key never exists in any single location. The ATS operator holds exactly one of three shards and provably cannot sign transactions without user biometric approval. Passkey-bound encryption of the backup shard, geographic distribution via Shamir $(3, 5)$ splitting to encrypted S3, and HSM co-signing for large settlements provide defense in depth across protocol, cryptographic, and physical security layers.

The architecture is deployed on the Lux Liquid Threshold cluster, formally verified against the CGGMP21 and FROST proof libraries, and satisfies the non-custodial requirements for SEC Rule 15c3-3 exemption.

References

[1] Z. Kelling. Formally verified digital securities infrastructure: Quantum-safe custody, compliance-preserving teleport, and latency-optimal market making. *Satschel, Inc. / Liquidity.io*, December 2025.

[2] Z. Kelling. Liquid EVM: On-chain securities settlement with post-quantum MPC custody. *Satschel, Inc. / Liquidity.io*, April 2026.

[3] Z. Kelling. Custody architecture audit: From custodial SSS to non-custodial MPC threshold signing with passkey-bound recovery. *Satschel, Inc.*, Internal Report, April 2026.

[4] Z. Kelling. Lux Liquid Threshold: Multi-Party Computation Chain for Threshold Signing. *Lux Network*, 2025. `https://papers.lux.network`

[5] Z. Kelling. Lux Threshold: LSSS-Based Threshold Signatures with TFHE Extensions. *Lux Network*, 2025. `https://papers.lux.network`

[6] Z. Kelling. EthFalcon: Post-Quantum Cryptography for EVM Chains. *Lux Network*, 2025. `https://papers.lux.network`

[7] Z. Kelling. Lux Formal Proofs: CGGMP21 Threshold ECDSA Correctness. *Lux Network*, 2025. `https://papers.lux.network`

[8] Z. Kelling. Lux Formal Proofs: FROST Threshold Schnorr Correctness. *Lux Network*, 2025. `https://papers.lux.network`

[9] Z. Kelling. Lux ID: Decentralized Identity and Access Management. *Lux Network*, 2025. `https://papers.lux.network`

[10] R. Canetti, R. Gennaro, S. Goldfeder, N. Makriyannis, and U. Peled. UC Non-Interactive, Proactive, Threshold ECDSA with Identifiable Aborts. In *ACM CCS*, 2020.

[11] C. Komlo and I. Goldberg. FROST: Flexible Round-Optimized Schnorr Threshold Signatures. In *SAC*, 2020.

[12] FIDO Alliance. FIDO2: Web Authentication (WebAuthn). W3C Recommendation, 2019.

[13] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.