Thunderbolt: A Formally Verified Protocol for Off-Chain Bitcoin Transfers

Hongbo Wen
hongbowen@ucsb.edu
Nubit
University of California, Santa Barbara

Hanzhi Liu
hanzhi@ucsb.edu
Nubit
University of California, Santa Barbara

Jingyu Ke
windocotber@sjtu.edu.cn
Nubit
Shanghai Jiao Tong University

Yanju Chen
yanju@ucsb.edu
University of California, Santa Barbara

Dahlia Malkhi
dahliamalkhi@ucsb.edu
University of California, Santa Barbara

Yu Feng
yufeng@cs.ucsb.edu
Nubit
University of California, Santa Barbara

Abstract

We present **Bitcoin Thunderbolt**, a novel off-chain protocol for asynchronous, secure transfer of Bitcoin UTXOs between uncoordinated users. Unlike prior solutions such as payment channels or the Lightning Network, Bitcoin Thunderbolt requires no prior trust, direct interaction, or continuous connectivity between sender and receiver. At its core, Bitcoin Thunderbolt employs a Byzantine fault-tolerant committee to manage threshold Schnorr signatures, enabling secure ownership delegation and on-chain finalization.

Our design supports recursive, off-chain UTXO transfers using tweakable, verifiable signature components. The protocol tolerates up to f malicious nodes in a $3f + 1$ committee and ensures correctness, consistency, and one-time spendability under asynchronous network conditions.

We formally verify Bitcoin Thunderbolt's key security properties, namely, unforgeability, ownership soundness, and liveness—using the Tamarin prover. Our results demonstrate that Thunderbolt provides robust, scalable, and non-interactive off-chain Bitcoin transfers, significantly expanding the practical utility of Bitcoin for decentralized applications.

Keywords

Formal Verification, Blockchains, Consensus Protocols

1 Introduction

Bitcoin has emerged as the most widely deployed cryptocurrency, renowned for its decentralized consensus, strong security guarantees, and censorship resistance [16]. However, these properties come at the cost of latency: Bitcoin transactions require approximately ten minutes for confirmation and multiple blocks for high assurance. This delay renders it impractical for latency-sensitive applications such as point-of-sale payments, online gaming, or financial systems requiring instant settlement.

To mitigate this, various off-chain solutions have been proposed, most notably payment channels [9, 10] and layer-two networks like the Lightning Network [10, 21]. These protocols offer lower-latency payment capabilities by avoiding immediate on-chain finality. However, they typically require interactive communication, pre-established channels, and active monitoring, making them ill-suited for non-custodial, asynchronous, or casual transfer scenarios. Moreover, limitations such as routing complexity [14], channel liquidity constraints [13, 17], and failure recovery have hindered widespread usability.

In this paper, we present **Thunderbolt**, a novel off-chain protocol that enables *asynchronous, secure, and instant UTXO transfers* on Bitcoin between mutually distrusting users without requiring prior coordination or live connectivity. Unlike payment channels, Thunderbolt does not rely on bilateral agreements or route-based liquidity. Instead, it introduces a **semi-trusted** threshold-signing committee to mediate ownership, ensuring cryptographically verifiable transfers while remaining fully compatible with Bitcoin's native transaction model. The committee is trusted only to be available and to not collude with a sender once a transfer completes.

Thunderbolt builds on the recently adopted Schnorr signature standard for Bitcoin (BIP340) [24], whose algebraic structure enables linearity and efficient aggregation. In particular, at the core of Thunderbolt is a cryptographic protocol that enables the committee and the sender to collaboratively construct a transferable signature, with the signing capability securely re-assigned to a new recipient. Transfers are achieved using a tweakable multi-signature structure, where each participant adjusts their signing contribution using shared secrets. This approach is inspired by recent work on secure delegation and key shifting in Schnorr-based constructions [18, 19, 22]. However, Thunderbolt is uniquely tailored for non-interactive, recursive delegation under adversarial conditions.

Designing such a system raises several challenges. Threshold signing in asynchronous networks is notoriously difficult, as message delays and node failures can break liveness and soundness guarantees [8]. Moreover, signature generation must be robust to equivocation and enforce one-time use of each spend path. Thunderbolt addresses these issues by incorporating a consensus-backed committee using Byzantine fault-tolerant replication [5, 25]. As long as $2f + 1$ out of $3f + 1$ committee members behave honestly, the protocol ensures correctness, consistency, and liveness despite partial faults or denial-of-service attacks.

Given the security-critical nature of Bitcoin transactions, we formally verify Thunderbolt's key properties using the Tamarin

prover [6, 15]. Our model captures realistic adversarial behavior, including adaptive corruption, message delay, and protocol deviation. We prove properties including unforgeability, one-time spendability, ownership soundness, and committee accountability. This rigorous analysis ensures that Thunderbolt is secure by construction and suitable for deployment in high-assurance systems.

Summary of Contributions. This work makes the following contributions:

(1) **The Thunderbolt Protocol:** A novel protocol for asynchronous, off-chain Bitcoin UTXO transfers between uncoordinated parties, requiring no prior trust or interaction.
(2) **Robust Asynchronous Threshold Signatures:** A resilient, tweakable Schnorr threshold signature system that supports key re-assignment and is robust to Byzantine committee faults.
(3) **Formal Security Proofs:** A machine-checked formalization of Thunderbolt's safety, correctness, and liveness guarantees using the Tamarin prover, accounting for adversarial network and committee behavior.

Together, these contributions provide a practical and formally verified foundation for scalable, user-friendly, and secure off-chain Bitcoin transactions.

2 Background and Related Work

Bitcoin employs a UTXO-based transaction model and achieves consensus through proof-of-work mining and longest-chain rule. While this approach provides strong consistency and censorship resistance, it suffers from limited throughput and long confirmation delays, typically requiring at least one block interval (about 10 minutes) to finalize a transaction. These constraints significantly hinder Bitcoin's use in interactive or real-time applications.

2.1 Off-Chain Transaction Mechanisms

To overcome latency and scalability limitations, the Bitcoin ecosystem has explored various off-chain techniques. Notably, the Lightning Network [20] introduces bidirectional payment channels where participants can execute multiple micro-transactions off-chain. However, Lightning requires the receiver to be online during the transaction and necessitates prior channel establishment, limiting usability in spontaneous or anonymous transfers.

Other constructions, such as state channels and sidechains [1], similarly aim to reduce on-chain load but often trade off trust assumptions, setup complexity, or user interactivity requirements. These models typically assume synchronized communication between senders and receivers or rely on ongoing connectivity, which is unsuitable for asynchronous or one-shot transfers.

2.2 Threshold Schnorr Signatures

Schnorr signatures, recently adopted in Bitcoin via BIP340 [24], are particularly well-suited to threshold constructions due to their linearity and efficient aggregation properties. These algebraic features enable multiple signers to jointly produce a single compact signature by aggregating commitments and key shares. This makes Schnorr a natural fit for multi-party signing and distributed custody, where correctness, efficiency, and accountability are paramount.

Threshold signatures allow a subset of parties to collaboratively generate a valid digital signature on behalf of a group. In a (t, n) threshold scheme, any t out of n participants can jointly sign, while fewer than t cannot forge a valid signature. Such schemes provide robustness and decentralization in blockchain applications, especially those requiring fault tolerance and key management across multiple entities.

Prior work has explored threshold Schnorr signatures [3, 12, 22] and multi-party key generation [18]. However, many of these designs are tailored for synchronous settings [3, 7], assume honest-majority participation [7, 12], or are vulnerable to Byzantine faults in asynchronous networks. These assumptions limit their applicability in decentralized environments where message delays, node failures, or adversarial coordination may occur.

Thunderbolt departs from these models by enabling resilient, asynchronous threshold signing within a Byzantine fault-tolerant framework. It supports dynamic, recursive delegation of signing power, allowing secure off-chain asset transfers without assuming synchronized interaction or honest coordination among signers.

2.3 Consensus-Based Committees

Distributed consensus protocols, such as PBFT [5], allow a set of $n = 3f + 1$ replicas to reach agreement despite up to f Byzantine faults. These protocols have been widely used in permissioned blockchains and fault-tolerant services. When applied to threshold signature generation, committee-based approaches can enforce collective behavior, accountability, and consistency in signature state and UTXO ownership records.

To maintain off-chain ownership state in a decentralized yet reliable manner, Thunderbolt leverages a consensus-based committee to execute signing operations and record transfer events. The committee is trusted only for availability and robustness, not for custody or correctness, as the security of each transfer depends on cryptographic guarantees rather than behavioral assumptions.

2.4 Adapter Signatures and Scriptless Contracts

Adapter signatures [19] allow embedding hidden data within partially completed signatures. These constructs enable conditional payments and atomic swaps off-chain. However, adapter signatures generally require interactive communication and a trusted counterparty during setup, limiting their usability for non-custodial or interaction-free use-cases.

Scriptless contracts and multi-party ECDSA constructions [11] have explored similar directions but often entail complex zero-knowledge proofs or costly interactive rounds. In contrast, Thunderbolt simplifies secure delegation by combining Schnorr homomorphism with carefully designed signature tweaking that enables asynchronous and uncoordinated off-chain ownership transfer.

2.5 Formal Verification of Crypto Protocols

Formal verification tools such as the Tamarin prover [23] and ProVerif [4] allow symbolic reasoning about security properties of cryptographic protocols under adversarial models. These tools have been successfully applied to analyze key exchange, secure

messaging, and consensus protocols. In high-stakes financial systems, such as Bitcoin, formal methods provide strong guarantees of protocol correctness, soundness, and safety.

In this work, we formally model the Thunderbolt protocol in Tamarin and verify key properties including integrity, authentication, and spendability, ensuring its resilience against adversarial behaviors in realistic asynchronous environments.

2.6 Threat Model and Assumptions

The Thunderbolt protocol is designed to operate securely in adversarial environments, including partially trusted off-chain committees and asynchronous communication networks. This section defines the security assumptions under which the protocol operates, the capabilities of adversaries it is intended to resist, and the scope of protection it aims to provide.

Thunderbolt assumes a model in which participants include honest users, a committee of replicated nodes, and a network adversary. The user, such as Alice or Bob, seeks to either create, transfer, or redeem a Bitcoin UTXO via off-chain delegation. The committee comprises $n = 3f + 1$ nodes responsible for maintaining off-chain state and generating threshold Schnorr signatures. Importantly, the committee is not trusted with funds—it holds no unilateral signing authority. Its sole role is to remain available for signing and to refrain from collusion with the sender after a transfer completes.

We assume a powerful Dolev-Yao adversary who has full control over the communication network. The adversary may delay, drop, replay, or reorder any message. Moreover, the adversary may statically compromise up to f committee members, granting them full access to their local state, including secret shares and vote histories. Users are assumed to maintain the confidentiality of their own secret values during transfer, and cryptographic primitives such as Schnorr signatures and hash functions are assumed to be secure against standard attacks.

Thunderbolt does not assume synchronous communication. Nodes may be offline, delayed, or receive messages in an arbitrary order. However, for liveness to hold, it is assumed that at least $2f+1$ honest committee members will eventually respond to a transfer or finalization request. Under this assumption, Thunderbolt guarantees that either the UTXO will be delegated to the intended recipient, or a valid finalization signature will be produced for on-chain redemption.

The protocol does not aim to protect against denial-of-service attacks targeting individual committee members or users. Similarly, it does not address the case where a user's local secrets are compromised prior to transfer, nor does it attempt to enforce privacy or unlinkability of off-chain ownership transfers. These goals are orthogonal to the protocol's primary objective: secure and asynchronous delegation of Bitcoin ownership with minimal trust assumptions.

3 Thunderbolt Protocol Design

The Thunderbolt protocol enables asynchronous, off-chain transfers of Bitcoin UTXOs between mutually mistrusting parties, without requiring interactive coordination or synchronous communication. It achieves this by combining threshold Schnorr signatures with a consensus-based committee that securely manages off-chain ownership. Once a UTXO is locked on-chain, Thunderbolt allows it to be transferred any number of times off-chain, with each ownership change cryptographically enforced and verifiable by the recipient alone.

To support robustness against faults and adversarial behavior, Thunderbolt assumes a committee of $n = 3f + 1$ participants, tolerating up to f Byzantine members. As long as at least $2f + 1$ nodes are honest, the protocol ensures signature correctness, consistency, and one-time spendability. The design adapts recent advances in threshold Schnorr signature protocols to a new setting—dynamic and recursive asset transfer—where ownership delegation must remain safe and non-interactive.

This section presents the core technical design of Thunderbolt. We begin by describing the system's participants and cryptographic foundations, and then provide a detailed walkthrough of the protocol's four main phases. All notation is introduced in context to ensure clarity and self-containment.

3.1 Protocol Overview

Thunderbolt is a Bitcoin-native protocol that enables secure, asynchronous, and off-chain transfer of UTXO ownership across multiple recipients—without requiring direct interaction, online availability, or prior coordination between sender and receiver. At its core, Thunderbolt leverages tweakable Schnorr signatures and a replicated Byzantine committee to track off-chain ownership and ensure single-spend finality. This design preserves Bitcoin's trust-minimized execution model while enabling flexible delegation chains that terminate in on-chain redemption.

Protocol Intuition. As illustrated in Figure 1, a Thunderbolt transfer begins when Alice locks a UTXO on-chain under a joint Schnorr key controlled by both herself and a decentralized committee. Off-chain, Alice transfers ownership to Bob by applying a cryptographic tweak to her half of the signature; the committee concurrently tweaks its half using a secret known only to Bob. These tweaks ensure that only Bob can invert the transformation and reconstruct the final signature. The committee verifies and logs the transfer in its replicated ledger, and marks Bob as the current off-chain owner.

This process generalizes recursively. Bob can reassign ownership to Carol using fresh tweaks, and Carol can further delegate to Zenni. Each hop updates the ledger without touching the Bitcoin blockchain. At any point, the current owner can trigger finalization, requesting the committee's release of its final signature share. Once both shares are assembled and the tweaks are canceled, the owner can produce a valid Schnorr signature and redeem the UTXO on-chain.

Entities. The protocol involves the following participants:

- **Sender (Alice):** The original on-chain UTXO owner.
- **Receivers (Bob, Carol, Zenni):** Off-chain delegates who receive and optionally forward UTXO ownership.
- **Committee:** A replicated, $n = 3f + 1$ node set that assists in threshold signing, verifies off-chain transfers, and maintains a consistent ledger of ownership. The system tolerates up to f Byzantine faults.
- **Bitcoin Network:** The public blockchain on which UTXOs are locked and ultimately redeemed.

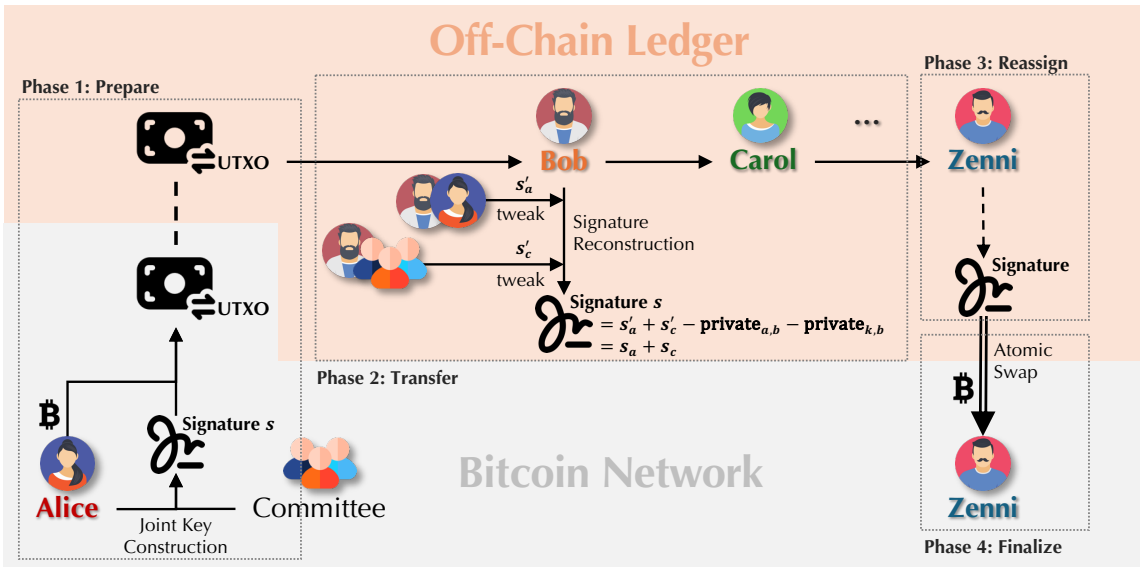

Figure 1: Illustrative example of Thunderbolt.

Protocol Phases. Thunderbolt operates through four conceptual phases:

(1) **Prepare.** Alice locks a UTXO on Bitcoin using a 2-of-2 Taproot-style key shared with the committee. Once confirmed, the committee records Alice as the initial off-chain owner.
(2) **Transfer.** To delegate ownership to Bob, Alice tweaks her partial signature using a secret known to Bob. The committee performs a complementary tweak and records Bob as the new owner. After this step, Alice can no longer spend the UTXO, as her version of the signature is no longer valid.
(3) **Reassign (Optional).** Bob may re-delegate the UTXO to a downstream user (e.g., Carol or Zenni) by initiating another round of tweaks and ledger updates. This phase can repeat recursively.
(4) **Finalize (Optional).** The current owner requests the committee's final signature share. Once received, they combine it with their own share and subtract the cumulative tweaks to produce a valid Schnorr signature over the original Taproot spend script. This signature is used to redeem the UTXO on-chain. The committee then irrevocably marks the UTXO as spent.

Ledger and Consensus. The committee uses a Byzantine fault-tolerant consensus protocol (e.g., PBFT) to maintain a replicated ledger of off-chain ownership. This ensures:

- Transfers are only accepted if they conform to the tweak structure and current ledger state,
- Each UTXO has a single, globally consistent owner at any time,
- Finalized UTXOs are never reused or re-signed,
- Liveness and correctness are preserved despite up to f faulty committee nodes.

Summary. Thunderbolt decouples asset control from network synchrony and direct interactivity. Through the combination of tweakable Schnorr signatures and fault-tolerant consensus, the protocol enables fast, flexible, and trust-minimized Bitcoin ownership transfers—extending Bitcoin's utility into asynchronous and programmable off-chain environments.

3.2 Crypto Primitives: Schnorr Signatures

The Thunderbolt protocol builds on the Schnorr signature scheme, which has been adopted as the standard signature primitive in Bitcoin (via BIP340 [24]). Schnorr signatures are not only efficient and compact but also enjoy strong algebraic properties that are essential to Thunderbolt's off-chain ownership delegation and multi-party signing.

A standard Schnorr signature on a message m is constructed as follows. The signer generates a secret nonce $r \in \mathbb{Z}_q$ and computes its public commitment $R = r \cdot G$. [1] Given a private key p and its corresponding public key $P = p \cdot G$, the signer computes a challenge scalar $c = H(R \parallel P \parallel m)$, where H is a cryptographic hash function. The final signature consists of the scalar

$$s = r + c \cdot p,$$

and the pair (R, s) is valid if it satisfies the verification equation:

$$s \cdot G = R + c \cdot P.$$

A key feature of Schnorr signatures is their *additive homomorphism*. Given multiple signers with keys $p_1, p_2, \ldots, p_n$ and nonces

[1] Elliptic curve cryptography (ECC) relies on the abelian group formed by the points on an elliptic curve defined over a finite field. The fixed generating point G is chosen for its large prime order, ensuring that every point in the subgroup can be produced by scalar multiplication, while making it computationally infeasible for an adversary to recover the scalar r from the public point R = r · G.

$r_1, r_2, \ldots, r_n$, their individual contributions can be linearly aggregated:

$$s = \sum_{i=1}^{n} (r_i + c \cdot p_i) = \left(\sum r_i\right) + c \cdot \left(\sum p_i\right).$$

This property enables decentralized threshold signing and key shifting, making it ideal for off-chain transfer schemes where ownership of an asset must be dynamically reassigned.

Participant Secrets and Commitments. Thunderbolt relies on the secure management and transformation of signature shares. Each participant in the protocol maintains private scalars and corresponding public commitments. We briefly describe the key material held by each role:

Alice (the initial UTXO owner) holds a private signing key p_a, a nonce r_a, and a tweak secret t_a used for computing pairwise shared secrets. Her public commitments include $p_a \cdot G$, $r_a \cdot G$, and $t_a \cdot G$.

Bob (the intended new owner) does not participate in the original signing but in order to receive transfers, publishes public commitments to two tweak secrets: $t_b \cdot G$ and $t2_b \cdot G$. These are used to derive shared secrets with both Alice and the committee.

The committee collectively manages a threshold private key p_k, a nonce r_k, and one tweak scalar per transfer, $t_{k,1}$, $t_{k,2}$, ... Their corresponding public commitments include $p_k \cdot G$, $r_k \cdot G$, and $t_{k,1} \cdot G$, and additional $t_{k,i} \cdot G$ after each transfer.

These values form the foundation of Thunderbolt's cryptographic shifting protocol. Through pairwise multiplications (e.g., $t_a \cdot t_b \cdot G$ or $t_{k,1} \cdot t2_b \cdot G$), parties can construct masked partial signatures that preserve soundness and unforgeability while transferring signing power to the next owner. This design enables asynchronous, liveness-free ownership transfers that remain secure under threshold adversaries.

3.3 Phase 1: Prepare

The Thunderbolt protocol begins with the creation of an on-chain UTXO that is jointly controlled by the sender (Alice) and a decentralized committee. This UTXO forms the initial anchor in Bitcoin, and its associated locking script ensures that it can only be unlocked with a valid Schnorr signature co-produced by Alice and the committee.

Joint Key Construction. To enable joint control of the UTXO, Alice and the committee collaboratively construct a composite public key by aggregating their respective signing materials:

- p_a: Alice's private key
- p_k: The committee's aggregate private key
- $P = p_a \cdot G + p_k \cdot G$: The combined public key

They also independently generate nonce values:

- r_a: Alice's nonce for randomness
- r_k: The committee's nonce
- $R = r_a \cdot G + r_k \cdot G$: The combined nonce commitment

The Schnorr challenge is computed as:

$$c = H(R \parallel P \parallel m)$$

where m is the Bitcoin transaction message (i.e., the transaction digest to be signed), and H is a cryptographic hash function.

The final signature is constructed as:

$$s = r_a + r_k + c \cdot (p_a + p_k)$$

which satisfies the Schnorr verification equation:

$$s \cdot G = R + c \cdot P$$

Bitcoin UTXO Creation. Alice constructs and broadcasts a Bitcoin transaction, denoted Tx_0, that locks funds under the composite public key P. This transaction includes a Taproot output (or any compatible script) that requires the above signature (R, s) to unlock.

Importantly, this transaction appears on-chain as a standard single-signer transaction, preserving privacy and indistinguishability from typical Taproot outputs.

Committee Ledger Update. Once the transaction is confirmed on-chain, the committee observes the blockchain and verifies that the UTXO was created correctly with the expected public key P. Upon confirmation, the committee records Alice as the initial off-chain owner of the UTXO in its local replicated state machine (ledger). This off-chain state acts as the authoritative record for future ownership transfers, ensuring that all signature generation and delegation actions correspond to legitimate and valid UTXO holders.

Security Responsibility. From this point forward, the committee is responsible for:

- Ensuring that the UTXO exists and remains unspent.
- Refusing to sign with p_k unless protocol rules are followed.
- Tracking ownership transitions and preventing double-spending.

This step anchors the UTXO in Bitcoin while establishing the cryptographic foundation for future asynchronous transfers. It marks the beginning of Thunderbolt's off-chain execution lifecycle, enabling trust-minimized, liveness-free value transfer.

3.4 Phase 2: Transfer

After the initial UTXO is created and recorded as belonging to Alice, she may transfer ownership off-chain to another user, Bob. The core idea of this phase is to securely reassign the signing capability from Alice to Bob without requiring synchronous communication or a new on-chain transaction. This is achieved by leveraging the homomorphic properties of Schnorr signatures and jointly computed tweak secrets.

At a high level, the process involves two signature tweaks—one performed by Alice, and the other by the committee. Each tweak is constructed using a shared secret between the signer (Alice or the committee) and Bob. These secrets are created using Diffie-Hellman-style constructions from pre-committed public values.

Step 1: Alice's Signature Tweak. Alice first computes a secret offset known only to herself and Bob:

$$\text{private}_{a,b} = f(t_a \cdot t_b \cdot G)$$

This value is derived from the product of generator G, Alice's and Bob's tweak secrets. It is an indistinguishable scalar from a random value to any third party. Alice then uses this to compute a tweaked signature component:

$$s'_a = s_a + \text{private}_{a,b} = r_a + c \cdot p_a + f(t_a \cdot t_b \cdot G)$$

Here:

- r_a is Alice's original nonce,
- $c = H(R \parallel P \parallel m)$ is the Schnorr challenge as defined in Phase 1,
- p_a is Alice's private key.
- f is a cryptographic hash function for mapping a curve point to a uniformly distributed scalar.

To convince the committee about the correctness of s'_a, Alice must prove:

$$F = (s'_a - s_a) \cdot G =? f(t_a \cdot t_b \cdot G) \cdot G$$

Since F is publicly known to the committee, verification can be reformulated as follows: Given the public witnesses F, G, and two commitments $T_a = t_a \cdot G$ and $T_b = t_b \cdot G$, Alice is required to demonstrate that F was calculated correctly as $F = f(t_a \cdot t_b \cdot G) \cdot G$. This correctness can be proven by generating a Non-Interactive Zero-Knowledge (NIZK) proof. [2] A cheaper alternative is requiring Bob to be online during the transfer, since Bob knows t_b and could verify the correctness of F directly.

If s'_a and the proof are valid, the committee records this tweaked signature component in its off-chain ledger (through a replicated Byzantine committee) as Alice's final contribution to the transfer.

Step 2: Committee's Signature Tweak. Next, the committee performs a similar tweak to its own signature component using another shared secret with Bob:

$$\text{private}_{k,b} = f(t_{k,1} \cdot t2_b \cdot G)$$

This secret scalar is derived from the generator, committee's tweak secret $t_{k,1}$ and Bob's second tweak value $t2_b$. The committee then updates its portion of the signature s_k:

$$s'_k = s_k + \text{private}_{k,b} = r_k + c \cdot p_k + f(t_{k,1} \cdot t2_b \cdot G)$$

Where r_k and p_k are the committee's nonce and private key, respectively.

To allow Bob and other committee members to verify the tweak, the committee provides the following relation with another NIZK proof:

$$s'_k \cdot G = r_k \cdot G + c \cdot p_k \cdot G + f(t_{k,1} \cdot t2_b \cdot G) \cdot G$$

After this update, the committee revokes access to the original secret r_k and commits to a new secret $t_{k,2}$ for future transfers. Reuse of $t_{k,1}$ is forbidden to forward secrecy.

Final Reconstruction by Bob. By receiving both s'_a and (later) s'_k, Bob can recover the original valid Schnorr signature when needed. Because only Bob knows both tweak values t_b and $t2_b$, only he can subtract the combined offset ($\text{private}_{a,b} + \text{private}_{k,b}$) and reconstruct:

$$s = s'_a + s'_k - \text{private}_{a,b} - \text{private}_{k,b} = s_a + s_k$$

This allows Bob to either spend the UTXO on-chain (in Phase 3) or transfer it again off-chain (in Phase 4), completing the ownership transition securely and asynchronously.

[2] The zero-knowledge proof involves three standard, efficiently provable computations: two elliptic curve scalar multiplications $t = t_a \cdot T_B$ and $F = h \cdot G$, and one evaluation of the cryptographic hash function $h = f(t)$ with T_B, F, G publicly known and t_a, t, h are private inputs. These computations correspond to standard, well-established arithmetic circuits (or gates) commonly used in zero-knowledge proof systems, making proof generation straightforward and uncontroversial.

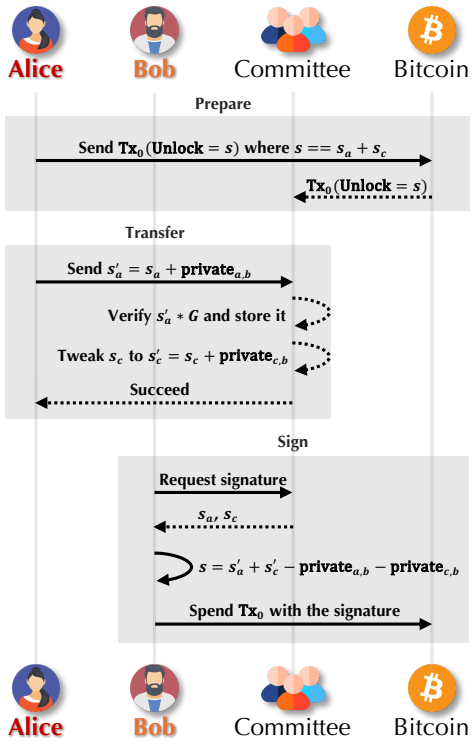

Figure 2: Thunderbolt Phase 3: Signing the UTXO using shifted secrets.

3.5 Phase 3: Reassign

After receiving ownership of a UTXO from Alice, Bob may decide to transfer it off-chain to a third party, Zenni, instead of redeeming it on-chain. Thunderbolt supports such recursive off-chain ownership transitions through a repeatable cryptographic tweak mechanism that ensures only the next recipient can unlock the final valid signature.

This phase mirrors the structure of Phase 2: Bob and the committee both adjust their existing (already tweaked) signature parts so that only Zenni—who holds new, secret tweak values—can reconstruct the valid full signature required to spend the UTXO. The design preserves forward secrecy: previous owners (e.g., Alice or Bob) can no longer derive the signature after the transfer is completed. Figure 3 visualizes the process. It highlights the interaction between Bob, Zenni, and the committee, and shows how the signature state is securely updated for the next ownership handoff.

Zenni's Setup. In order to receive a transfer, Zenni generates and publishes two public tweak commitments, $t_z \cdot G, t2_z \cdot G$. These values serve the same purpose as Bob's tweak secrets in Phase 2. They are used to derive shared secrets with both Bob and the committee for the next round of tweaking.

Committee's Signature Update. Recall that the current committee partial signature is:

$$s'_k = s_k + \text{private}_{\text{k,b}} = r_k + c \cdot p_k + f(t_{k,1} \cdot t2_b \cdot G)$$

To transfer ownership to Zenni, the committee performs a new tweak using its next round secret $t_{k,2}$ and Zenni's $t2_c$:

$$\begin{aligned}\text{private}_{\text{k,z}} &= f(t_{k,2} \cdot t2_z \cdot G)\\ s_k'' &= s_k' - \text{private}_{\text{k, b}} + \text{private}_{\text{k, z}}\\ &= s_k' - f(t_{k,1} \cdot t2_b \cdot G) + f(t_{k,2} \cdot t2_z \cdot G)\end{aligned}$$

After performing this update, the committee revokes and refuses to release s_k' or any past secrets, ensuring one-time use of each signature state.

Bob's Signature Update. After generating a joint secret $private_{b,z}$ with Zenni, Bob updates his previously tweaked signature s_a' (from Alice) by subtracting the joint secret with Alice and then re-tweaking for Zenni:

$$s_a'' = s_a' - \text{private}_{a,b} + \text{private}_{b,z}$$

Substituting in the known values:

$$s_a'' = s_a' - f(t_a \cdot t_b \cdot G) + f(t_b \cdot t_z \cdot G)$$

This operation cancels the previous secret influence, then adds a new one that only Zenni can resolve using her private tweak t_z.

Zenni's Reconstruction. With the updated values s_a'' and s_k'', Zenni (and only Zenni) can compute:

$$s = s_a'' + s_k'' - f(t_b \cdot t_z \cdot G) - f(t_{k,2} \cdot t2_z \cdot G)$$

This cancels the newly added secrets and yields:

$$s = s_a + s_k = r_a + r_k + c \cdot (p_a + p_k)$$

Zenni now possesses the valid signature on the original UTXO and is the only party capable of spending it. She may either repeat this process to transfer to another recipient or redeem it on-chain using Phase 4.

Forward Secrecy and Security. As in all previous phases, once the transfer is completed:

- Bob loses access to the valid signature and cannot spend the UTXO.
- The committee commits to fresh secrets $t_{k,3}$ for any further use.
- The old secret $t_{k,2}$ is retired and not reused.

This ensures that each ownership transfer introduces fresh entropy and guarantees that only the current owner (Zenni) has the ability to derive the spendable signature.

3.6 Phase 4: Finalize

This phase completes the Thunderbolt transfer cycle by enabling the final off-chain owner—Zenni—to redeem the original UTXO on the Bitcoin network. Upon receiving ownership (via Phase 3), Zenni may either continue transferring the UTXO off-chain or finalize it on-chain. We focus on the latter case, which involves two layers: (1) a cryptographic finalization step, where Zenni reconstructs the full Schnorr signature originally committed by Alice and the committee, and (2) an on-chain realization via an atomic swap, since Zenni cannot directly spend the output due to the fixed structure of the original transaction. Together, these two components ensure that Thunderbolt transfers can be securely and asynchronously finalized without requiring prior coordination or script introspection.

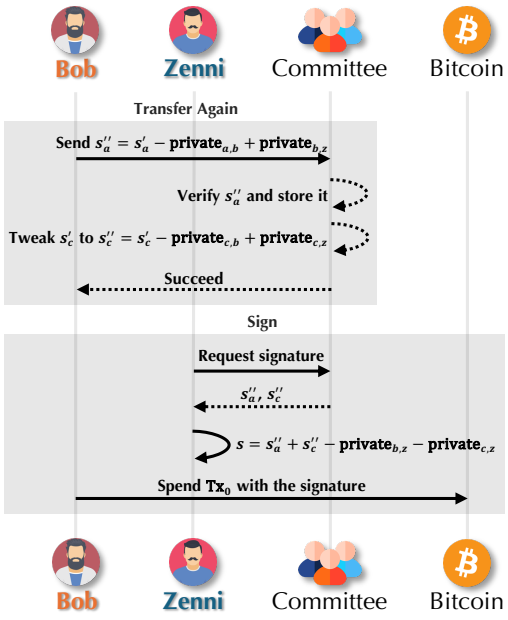

Figure 3: Thunderbolt Phase 4: Signature reconstruction and optional finalization by the recipient.

Requesting the Final Signature. To finalize the UTXO, Zenni must reconstruct the original Schnorr signature associated with the Taproot output locked by Alice and the committee. Alice's (tweaked) partial signature share s_a'' was published earlier in the delegation chain; Zenni now requests the committee's corresponding tweaked share s_k'', which is released only once for security and liveness. The committee checks its ledger to confirm Zenni's current ownership and replies with:

$$s_k'' = r_k + f(t_{k,2} \cdot t2_z \cdot G) + c \cdot p_k$$

Zenni produces the full signature as described in Section 3, under "Zenni's Reconstruction".

This yields a valid Schnorr signature (R, s) where:

$$R = R_a + R_k, \quad P = P_a + P_k, \quad c = H(R \parallel P \parallel m)$$

Recipient Authorization. Although both s_a'' and s_k'' may be publicly visible, only Zenni—who knows both tweak scalars t_z and $t2_z$—can cancel the delegated components and recover the correct signature. Thus, control of the UTXO is cryptographically tied to knowledge of these secrets, not to secrecy of the signature shares themselves.

3.6.1 Phase 4.1: Atomic Swap Realization. While Zenni can reconstruct a valid signature for the original UTXO (as described above), he cannot directly redeem the funds from the original output, as the transaction T_1 that spends T_0 was prepared and partially signed at the time of the initial transfer (e.g., from Alice to Bob). Consequently, Zenni's address could not have been included in the output script of T_1, and he is not authorized to spend the output directly. To resolve this, Thunderbolt uses an *Atomic Swap* procedure in which the committee temporarily fronts the funds and ensures settlement through a sequence of verifiable Bitcoin transactions.

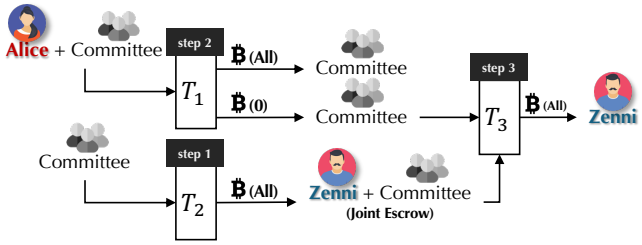

Figure 4: Overview of the atomic swap procedure.

Swap Construction. As shown in Figure 4, the atomic swap involves three Bitcoin transactions, each fulfilling a specific role in the handoff between the original UTXO and Zenni's final receipt.

- T_1**:** A transaction from Alice and the committee to the committee This transaction spends the original UTXO T_0. Its input requires the original full signature from Alice and the committee. The transaction has two outputs:
  - Output[0]: Transfers the full monetary value to the committee's wallet.
  - Output[1]: A dummy output sending zero BTC to the committee, used only as a placeholder to encode dependency in subsequent transactions.

  Including this dummy output allows the committee to verify that T_1 has been posted and confirmed on-chain, as it becomes an input in the later transaction T_3 which fronts money from the Committee to Zenni.
- T_2**:** A transaction from the committee to a joint Zenni+Committee escrow address. This transaction locks funds from the committee's own wallet into a 2-of-2 multisig output shared with Zenni. This serves as a temporary escrow that ensures the funds will only move if both Zenni and the committee cooperate. It does not rely on any prior Thunderbolt logic and can be prepared independently on Bitcoin.
- T_3**:** A transaction from the Zenni+Committee escrow to Zenni. This transaction completes the atomic swap. It has two inputs:
  - Input[0]: Spends T_1's dummy output, requiring a signature by the committee. This acts as a proof that T_1 has been finalized on-chain.
  - Input[1]: Spends the escrow output from T_2, requiring both Zenni and the committee to sign.

  The sole output sends the full value to Zenni, completing the swap securely.

Swap Protocol. The atomic swap proceeds in the following steps, illustrated in Figure 4:

(1) **Escrow initialization.** The committee first posts T_2 on the Bitcoin network, creating a joint 2-of-2 escrow between itself and Zenni. This ensures the funds are temporarily reserved for Zenni pending the completion of T_1.
(2) **Partial signature distribution.** The committee provides Zenni with its partial signature on the input of T_1. Now Zenni alone has the power to unlock T_1 and release the funds to the Committee.
(3) **Settlement witness preparation.** The committee provides Zenni with its signature on the first (dummy) input of T_3, authorizing "spending" of input[0], using `SIGHASH_ALL` to bind the signature to the exact structure of T_3. This signature will allow Zenni to later prove on-chain that T_1 was indeed settled before claiming the funds from escrow.[3]
(4) **Triggering T_1 finalization.** Zenni sends the committee his partial signature on the input of T_1. With both parts now available, the committee can complete T_1 and claim the funds from the original UTXO T0.
(5) **Final redemption.** Once T_1 is confirmed on-chain, Zenni completes and broadcasts T_3 using his knowledge of the multisig escrow (from T_2) and the proof of settlement (from T_1's dummy output). As a result, Zenni receives through the escrow the funds initially committed in T0.

This protocol ensures that neither party can gain unfair advantage. Zenni only signs T_1 after the committee commits its liquidity to the escrow (via T_2), and the committee only finalizes T_1 once Zenni commits to T_3. The dummy output in T_1 serves as a cryptographic anchor, proving that the transaction chain progressed as expected. While the committee could technically spend T_1's dummy output prematurely, doing so would break the atomicity of the protocol and provide no financial gain. Hence, a rational or honest committee has no incentive to deviate.

3.7 Security Considerations

Thunderbolt relies on a consensus-based committee to mediate off-chain ownership and partial signature generation. To ensure safety and consistency across all phases of the protocol, the committee must uphold a set of integrity-preserving rules. Below, we outline the key safeness properties, explain why they are necessary, and demonstrate how they can be enforced under a Byzantine fault-tolerant (BFT) consensus framework such as PBFT.

Safety Properties. Each phase of the protocol imposes specific obligations on the committee to preserve correctness, soundness, and non-repudiation. The committee must enforce the following:

(1) **Prepare Phase:** Accept only UTXOs that are valid and confirmed on the Bitcoin blockchain.
(2) **Transfer Phase (Sender):** Accept and record the sender's partial signature s'_a only if it passes cryptographic verification.
(3) **Transfer Phase (Committee):** Provide a correct and verifiable partial signature s'_k consistent with the committee's ledger.
(4) **Sign Phase:** Once the full signature is generated and handed out, mark the corresponding UTXO as *spent*.

These checks ensure that: (1) only confirmed UTXOs are accepted, (2) ownership is transferred based on valid cryptographic input, (3) the committee cannot produce inconsistent or conflicting signatures, and (4) each UTXO is spent exactly once.

Enforcing Safeness via PBFT.. To maintain a consistent and tamper-resistant off-chain ledger, Thunderbolt uses a PBFT-style consensus protocol to replicate state across committee nodes. This ensures

[3]Requiring the committee's signature on the dummy output prevents frivolously "spending" this output and blocking Zenni from redeeming her funds through T_3.

deterministic agreement even when up to f out of $3f+1$ members are faulty.

Property 1 (UTXO Confirmation). In the prepare phase, committee nodes independently validate that the referenced UTXO exists on the Bitcoin chain and is unspent. This check is straightforward when all nodes follow the longest-chain rule and run trusted Bitcoin full nodes. Once $2f+1$ members confirm the UTXO's presence, the committee can safely record Alice as the initial owner.

Property 2 (Verification of s'_a). During a transfer, Alice provides a tweaked partial signature s'_a. Each committee member verifies:

$$s'_a \cdot G =? r_a \cdot G + c \cdot p_a \cdot G + f(t_a \cdot t_b \cdot G)$$

This is a standard Schnorr verification adjusted for the protocol's use of tweaks. Verification is local and deterministic, so honest nodes will all agree whether the signature is valid.

Property 3 (Marking UTXO as Spent). Once Bob invokes the sign phase to redeem the UTXO on-chain, the committee generates and releases s'_k. At that point, all honest nodes update their local state to mark the UTXO as spent. The PBFT protocol guarantees that all non-faulty nodes reach this conclusion consistently.

The Challenge: Ensuring Signature Correctness. Property 3—ensuring that the committee always produces the correct s'_k—is the most subtle and security-critical requirement. Thunderbolt uses threshold Schnorr signatures, where the private key p_k and nonce r_k are distributed among committee members. This means signature generation must be both *robust* (tolerating partial faults) and *sound* (always yielding a valid signature).

Recent asynchronous Schnorr threshold signing protocols, such as ROAST [3, 22], have demonstrated how to achieve these goals. In such schemes, nodes commit to nonce values, compute a shared challenge c, and then locally generate responses that are linearly aggregated. As long as $2f+1$ out of $3f+1$ nodes follow the protocol, the aggregated signature will be valid and verifiable.

In Thunderbolt, the committee executes such a protocol internally during transfer and sign phases. It ensures that:

- All partial signatures are consistent with the agreed public key and challenge.
- Any malicious or incorrect shares can be detected and excluded.
- Each resulting s'_k is consistent with the committee's state and ledger.

Summary. Together, the use of a BFT consensus protocol and a threshold Schnorr signing scheme enables Thunderbolt to maintain:

- **Correctness:** Every signature is valid and attributable.
- **Non-repudiation:** Ownership transitions are cryptographically verifiable.
- **Robustness:** The system tolerates faults while ensuring liveness.
- **One-time spendability:** Each UTXO is tracked and cannot be spent twice.

These properties are central to Thunderbolt's ability to provide secure, asynchronous off-chain Bitcoin transfers with auditability and minimal trust.

4 Formal Verification of Thunderbolt Protocol

To formally validate the correctness and security guarantees of Thunderbolt, we model the protocol using the **Tamarin Prover** [15], a symbolic analysis tool specialized for reasoning about cryptographic protocols under adversarial conditions. Tamarin has been successfully applied to verify protocols like the Signal messaging protocol [**?**], TLS 1.3 [**?**], and the Messaging Layer Security (MLS) group protocol [2]. It supports reasoning about stateful protocol logic, algebraic properties (e.g., for Diffie-Hellman and Schnorr signatures), and unbounded adversaries within the Dolev-Yao model.

We develop a rigorous formal model of Thunderbolt in Tamarin, encompassing its key mechanisms: off-chain Schnorr signature tweaking, asynchronous delegation, committee-based authorization, and recursive ownership updates. This model enables us to prove key security properties—such as signature unforgeability, single-spend soundness, and protocol liveness—even in the presence of network asynchrony and malicious committee members.

4.1 Tamarin Prover in a Nutshell

Tamarin encodes protocols as multiset rewriting systems where each *state* is a multiset of **facts**, and each *transition* is defined by a **rule** that consumes and produces facts. Protocol executions are sequences of rule applications called **traces**. Properties—such as authentication or secrecy—are specified as temporal logic formulas over all possible traces.

Facts. Facts represent pieces of state. They may be persistent (e.g., `Honest(c)` indicating that committee member c is honest) or linear (e.g., `Locked(utxo)`, which must be consumed exactly once).

Rules. A rule defines a protocol step. For instance, below is a simplified rule where a user locks a UTXO:

```
rule LockUTXO: [Fr(r), Fr(p)] --> [Locked(utxo), Nonce(r), Key(p)]
```

Listing 1: Toy rule: UTXO locking

Here, `Fr(x)` means x is freshly generated, and the rule outputs facts indicating the UTXO is now locked under (r, p).

Traces. A trace is a sequence of rule applications. For example, one trace might be:

$$\texttt{LockUTXO} \rightarrow \texttt{Transfer} \rightarrow \texttt{Finalize}$$

Each application transforms the system state by updating the multiset of facts.

Security Properties. Properties are written as first-order temporal logic lemmas. For example, the following lemma enforces that any revealed secret nonce must have been explicitly released:

```
lemma NoImplicitLeak:
  "All r #i. Out(r)@i ==> Exists #j. Leak(r)@j"
```

Listing 2: Example lemma: no silent leakage

This states that for any trace where the adversary learns r (`Out(r)`), there must exist an earlier or concurrent point where the protocol intentionally released it (`Leak(r)`).

Algebra and Adversary Model. Tamarin operates in the Dolev-Yao model: the adversary controls the network and can intercept, replay, delay, or reorder messages. Tamarin also supports reasoning over algebraic structures like cyclic groups, modular exponentiation, and homomorphic operations—critical for modeling Schnorr signatures and key tweaks.

Summary. The Tamarin Prover allows us to symbolically simulate the Thunderbolt protocol, apply adversarial actions, and verify whether our protocol guarantees continue to hold across all possible traces. The following sections describe how we encode Thunderbolt's protocol logic into Tamarin and how we formally verify its key correctness properties.

4.2 Symbolic Modeling of Thunderbolt

We formalize Thunderbolt using Tamarin's multiset rewriting rules and algebraic term system. Our model captures users, the threshold committee, off-chain ownership, Schnorr signatures, and tweakable signature shifts.

4.2.1 *Entities and Objects.*

- **Users:** Alice, Bob, Carol, etc., participate in UTXO transfer.
- **Committee:** A set $\{c_1, \ldots, c_n\}$ with $n = 3f + 1$ signers.
- **Adversary:** A Dolev-Yao attacker who can delay, replay, and reorder messages, and statically corrupt up to f committee nodes.

UTXO Identifiers. Unique symbolic terms for each off-chain UTXO.

Schnorr Signature.

$$s = r_a + r_k + c \cdot (p_a + p_k), \quad c = H(R \parallel P \parallel m)$$

Tweaked Signatures.

$$s' = s + T_1 + T_2, \quad T_1 = t_a \cdot t_b \cdot G, \quad T_2 = t_k \cdot t_{b2} \cdot G$$

4.2.2 *Rules and State Transitions.*

Prepare. Before Alice transfers a UTXO, she must first lock it on-chain under a 2-of-2 signature with the committee. The `PrepareUTXO` rule (Listing 3) emits all the secret values and registers the initial Locked(utxo) state.

```
rule PrepareUTXO: [ Fr(id), Fr(sender) ]
--> [ Locked(id), !OwnerState(sender, id, 'Pending'), !UTXO(id) ]
```

Listing 3: Tamarin rule for preparing a UTXO

Transfer. To transfer ownership from Alice to Bob off-chain, Thunderbolt shifts the signature using two tweaks known only to Bob. The `Transfer` rule (Listing 4) rule updates the owner and records the new tweaked signature state.

```
rule Transfer:
  [ !Committee(committee), !OwnerState(sender, id, 'Approved'),
    !TweakFor(sender, receiver, t_sr),
    !TweakFor(committee, receiver, t_cr), Fr(sig), !UTXO(id) ]
  -->
  [ !OwnerState(receiver, id, 'Pending'),
    TweakedSig(receiver, sig), ValidSignature(sig),
    HasTweakSecret(receiver),
    TransferEvent(committee, sender, receiver, id) ]
```

Listing 4: Tamarin rule for off-chain ownership transfer

Finalize. Once Bob wants to spend the UTXO on-chain, he requests the committee's signature. The `Finalize` rule (Listing 5) rule consumes the locked state and emits the finalized signature.

```
rule Finalize: [!OwnerState(receiver, id, 'Approved'), Locked(id)]
  --> [Finalized(id), TransferFinalized(receiver, id), Spent(id)]
```

Listing 5: Tamarin rule for finalizing a UTXO

Reassign. Bob can recursively transfer the UTXO to Carol by applying a new tweak. The `Reassign` rule (Listing 6) rule shifts ownership again and updates the ledger.

```
rule Reassign:
  [ !Committee(committee), !OwnerState(sender, id, 'Approved'),
    !TweakFor(sender, receiver, t_sr),
    !TweakFor(committee, receiver, t_cr), Fr(sig), !UTXO(id) ]
  -->
  [ !OwnerState(receiver, id, 'Pending'),
    TweakedSig(receiver, sig),
    ValidSignature(sig), HasTweakSecret(receiver),
    TransferEvent(committee, sender, receiver, id) ]
```

Listing 6: Tamarin rule for reassignment to next user

4.3 Verified Properties and Logical Definitions

We now present the five core security and liveness properties that Thunderbolt enforces via its Tamarin model. Table 1 summarizes each property and its corresponding logical form.

Ownership Soundness. This property ensures that only the intended recipient (e.g., Bob) can reconstruct a valid Schnorr signature from the tweaked signature share s'. The adversary cannot guess or forge tweak values to impersonate recipients. This guarantees secure, exclusive ownership.

```
lemma OwnershipSoundness: all-traces "All sig #i.
  ValidSignature(sig)@#i ==> Exists u #j. HasTweakSecret(u)@#j"
```

Listing 7: Lemma for ownership soundness

Unforgeability. This lemma shows that no party can produce a valid Schnorr signature unless a quorum of the committee has contributed honestly. It captures the essential threshold assumption: with fewer than $2f + 1$ honest nodes, signature generation cannot succeed.

```
lemma Unforgeability: all-traces "All sig #i.
  ValidSignature(sig)@#i ==>
  Exists committee sender receiver id #j #k.
    TweakedSig(receiver, sig)@#j & HasTweakSecret(receiver)@#j &
    CommitteeApproved(committee, id, sender)@#k & #k<#j & #j=#i"
```

Listing 8: Lemma for unforgeability with committee approval

One-Time Spendability. A UTXO can only be finalized once. This lemma prevents double-spending in the finalize phase. Tamarin enforces this via linear logic: once a fact like `Finalized(utxo)` can only occur once for any given UTXO.

```
lemma OneTimeSpendability: all-traces "All utxo #i #j.
  Finalized(utxo)@i & Finalized(utxo)@j ==> i = j"
```

Listing 9: Lemma for one-time spendability

Table 1: Summary of verified Thunderbolt protocol properties and their verification steps in Tamarin.

Property	Description	# of verification steps
Ownership Soundness	Only the recipient can reconstruct the signature from shared tweaks	2
Unforgeability	No valid signature can be generated without $2f + 1$ committee shares	2
One-Time Spendability	A UTXO may only be finalized once	12
Liveness (transfer)	Any honest user with a responsive quorum can complete transfer	43
Liveness (finalize)	Any honest user with a responsive quorum can complete finalization	225

Liveness. Provided the user is honest and at least $2f+1$ committee members respond, either a transfer or finalization will eventually succeed. We split this into three concrete lemmas capturing increasingly minimal paths to finality:

```
lemma CanFinalize: exists-trace
  "Exists id committee sender receiver #i #ij #j #k #l.
     Prepared(id)@#i &
     CommitteeApproved(committee, id, sender)@#ij &
     TransferEvent(committee, sender, receiver, id)@#j &
     CommitteeApproved(committee, id, receiver)@#k &
     Finalized(id)@#l & #i < #ij & #ij < #j & #j < #k & #k < #l"
```

Listing 10: Liveness: full finalize path with approvals

```
lemma TransferCompletionPossible: exists-trace
  "Exists id committee sender receiver #i #j #k #l.
     Prepared(id)@#i &
     CommitteeApproved(committee, id, sender)@#j &
     TransferEvent(committee, sender, receiver, id)@#k &
     Finalized(id)@#l & #i < #j & #j < #k & #k < #l"
```

Listing 11: Liveness: transfer completion path

```
lemma FinalizationSuccessPossible: exists-trace
  "Exists id committee sender receiver #i #j #k #l.
     Prepared(id)@#i &
     TransferEvent(committee, sender, receiver, id)@#j &
     CommitteeApproved(committee, id, receiver)@#k &
     Finalized(id)@#l & #i < #j & #j < #k & #k < #l"
```

Listing 12: Liveness: receiver finalized after approval

Adversarial Trace Simulation and Security Intuition. To validate that the model rejects unsafe executions, we simulate adversarial traces where an attacker attempts to double-spend a UTXO by issuing multiple `Finalize` actions. Due to the linear consumption of the `Locked(utxo)` fact in Rule 5, Tamarin correctly blocks any such replays, confirming the one-time spendability property (Lemma 9). We further simulate equivocation attacks, vote replays, and off-path tweak applications, all of which are rejected by our symbolic constraints and consistency lemmas.

Beyond formal proofs, Thunderbolt's defenses also hold under intuitive adversarial scenarios. As shown in Table 2, a malicious committee cannot front-run the recipient, since the final Schnorr signature includes tweak terms $T_1 = t_a \cdot t_b \cdot G$ and $T_2 = t_k \cdot t_{b2} \cdot G$, known only to the recipient. Similarly, vote reuse or signature equivocation is blocked by quorum tracking per-UTXO, enforced both in the replicated ledger and in the symbolic model. The protocol remains live under partial committee failures thanks to its asynchronous threshold signature design, which tolerates up to f faulty nodes. Finally, forgery attempts fail cryptographically: without knowledge of the tweak scalars, adversaries cannot derive a valid signature due to the hardness of the discrete logarithm problem. These intuitive defenses align directly with our formal properties, reinforcing Thunderbolt's robustness in real-world adversarial settings.

4.4 Results and Summary

We formally verified the Thunderbolt protocol using the Tamarin Prover, modeling all key protocol transitions and security properties at the symbolic level. Our formalization includes 513 lines of Tamarin code, capturing the core behavioral rules of Thunderbolt—initial setup, signature tweaking, off-chain transfers, committee authorization, and on-chain finalization.

Across this model, we verify five safety properties and three liveness properties, covering both universal and existential guarantees. The partial verification results are summarized in Table 1, with complexity measured by the number of symbolic proof steps required.

As expected, safety properties such as one-time spendability and unforgeability are efficiently discharged by Tamarin's built-in SMT solver. These properties encode local invariants and ledger consistency checks that are straightforward to verify in the multiset rewriting model.

In contrast, liveness properties—particularly those involving finalization under partial quorum responsiveness—require significantly more effort. These lemmas involve reasoning about asynchronous traces, committee coordination, and progress under eventual message delivery. For example, the `finalizationSuccessPossible` lemma (lemma 12) requires 225 symbolic steps, including interactive guidance to construct viable execution paths and instantiate existential variables.

These results confirm that Thunderbolt achieves strong symbolic correctness guarantees across adversarial conditions. Even in the presence of delayed communication, reordering, and Byzantine committee behavior, the protocol provably ensures that delegated UTXOs are non-replayable, unforgeable, and eventually redeemable on-chain. Future extensions may incorporate finer-grained cryptographic modeling (e.g., explicit signature composition and tweak arithmetic), though this would increase both symbolic complexity and proof effort.

5 Deployment

We conclude with a brief discussion of the practical steps required to deploy Thunderbolt on Bitcoin main-net.

5.1 Committee and Incentives

A Thunderbolt committee must (i) hold long-lived threshold key material, (ii) run a Byzantine-fault-tolerant replication layer (e.g. PBFT), and (iii) expose two authenticated API calls—Transfer and Finalize. Three deployment models are immediately available:

Table 2: Adversarial strategies and corresponding Thunderbolt defenses.

Threat Scenario	Defense Mechanism
Double-spend or replay	Linear UTXO state and consensus ledger
Committee front-running	Tweak secrets known only to recipient
Signature equivocation or vote reuse	Quorum binding per UTXO and state consistency
Committee denial-of-service	Asynchronous threshold signatures tolerating f faults
Tweak extraction or forgery	Hardness of discrete logarithm in the Schnorr group

(1) **Side-chain federation.** Existing multisignature custodial federations (e.g. Liquid or RSK) already operate with $n! = !11–15$ signers and an on-chain watchtower infrastructure. Reusing those operators as committee nodes merely requires adding the ROAST signing wrapper and the replicated ownership ledger.
(2) **Stake-weighted service DAO.** A new set of operators can register Taproot public keys on-chain and bond stake via the outputs of a specific opcode [4]. A slashable log (similar to *watchtowers*) penalizes nodes that refuse to sign after a service request, aligning incentives with liveness guarantees.
(3) **Infrastructure-as-a-service.** Cloud HSM providers already expose threshold-ECDSA endpoints; exposing FROST-Schnorr shares is a minor software extension. Users can mix independent vendors to reach $3f + 1$ diversity.

In all three settings, fee revenue is straightforward: the Finalize API can deduct a small *service-fee output* in the redemption transaction, payable to a fee-collector key controlled by the committee. Because Thunderbolt's on-chain footprint is a single Taproot spend, the fee can be priced competitively against Lightning channel closures or adaptor-signature swaps.

5.2 Taproot Policy Template

Thunderbolt requires only a standard BIP-340 Schnorr key and thus fits natively into the existing Taproot commitment structure. A recommended output template is:

```
scriptPubKey = OP_1 <P_agg>
```

where $P_{\text{agg}} = P_a + P_k$ is the aggregated 2-of-2 public key. No script path is needed, preserving indistinguishability from ordinary single-party Taproot spends and avoiding extra witness bytes. Wallet-side changes are limited to recognising Thunderbolt outputs (by prior out-of-band metadata) and invoking the Finalize RPC instead of generating a local signature.

5.3 Handling Chain Re-organisations

Because ownership is tracked off-chain, a deep chain re-organisation that removes the funding transaction would invalidate all derived transfer records. We adopt the standard practice from side-chains and coin-join wallets: the committee waits for k Bitcoin confirmations (default $k! = !6$) before accepting a Prepare request. If a re-org nevertheless occurs, the ledger entry is rolled back automatically—no security violation arises because the UTXO no longer exists on-chain, and any subsequent Finalize call will fail Bitcoin consensus validation.

[4] OP_CHECKLOCKTIMEVERIFY

6 Discussion and Future Work

Thunderbolt is designed to be practical, formally analyzable, and minimally invasive to Bitcoin's consensus layer. While our protocol introduces a novel delegation model with strong security guarantees, several aspects of its current formulation open fruitful directions for future exploration.

Static Committee Configuration. Our protocol assumes a statically defined committee whose public keys are known to all participants. This simplifies threshold coordination and security analysis. In practice, many deployments may prefer a dynamic or rotating committee, especially in federated or DAO-like settings. Supporting committee reconfiguration without compromising ownership continuity or threshold guarantees is a promising extension, and could be achieved via re-keying protocols or verifiable resharing mechanisms, both of which are orthogonal to the core design of Thunderbolt.

Limited Privacy Guarantees. Thunderbolt, by design, prioritizes correctness, usability, and verifiability over privacy. Transfer commitments and on-chain settlement signatures are observable, which may leak limited metadata over time. This limitation stems from the inherent transparency of Bitcoin's base layer rather than any aspect of our protocol design. Nevertheless, integrating well-established privacy-preserving primitives—such as key blinding, zero-knowledge proofs, or adaptor signatures—could enhance confidentiality in a modular and composable fashion.

Quorum Responsiveness Assumption. Our liveness guarantees require at least $2f + 1$ responsive committee members during a transfer or finalization attempt. This reflects a standard assumption in threshold cryptographic systems and is not unique to Thunderbolt. In scenarios with extended downtime or partial outages, availability may be temporarily degraded. However, committee selection and incentive mechanisms (e.g., staking, slashing, or relay fallback) can be incorporated at the deployment level to mitigate this risk without altering the core protocol semantics.

Lack of Explicit Incentive Layer. Thunderbolt abstracts away the incentive mechanics for committee operation, focusing instead on the correctness of the protocol's core logic. This design decision follows prior work in threshold signing and multi-party computation, where incentive compatibility is treated as an orthogonal systems layer. Incorporating native fee mechanisms, committee rotation policies, or proof-of-participation could further strengthen protocol robustness without requiring changes to Thunderbolt's cryptographic foundation.

Bitcoin Layer Constraints. Because Thunderbolt operates entirely within the constraints of Bitcoin's scripting and signature model,

it inherits certain expressiveness and fee model limitations from the base chain. Notably, the absence of covenant-style constructs or introspective opcodes limits our ability to enforce recursive constraints purely on-chain. However, this design choice ensures maximal compatibility with existing infrastructure and aligns with Bitcoin's conservative consensus philosophy. Future soft forks (e.g., 'OP_CAT', 'OP_TXHASH') may enable more expressive constructions that extend Thunderbolt's capabilities even further.

7 Conclusion

We presented **Thunderbolt**, a formally verified protocol for secure, asynchronous off-chain transfer of Bitcoin UTXOs. Thunderbolt enables non-interactive ownership delegation by combining the linearity of Schnorr signatures with a quorum-based committee that tracks off-chain state under Byzantine fault tolerance. Through recursive signature tweaking, the protocol supports multi-hop transfers without requiring prior coordination, online presence, or pre-established channels, and guarantees that only the designated recipient can reconstruct the final signature.

To validate security, we modeled Thunderbolt in the Tamarin Prover and verified key safety and liveness properties, including ownership soundness, unforgeability, single-spend finality, and progress under quorum delays. Our results demonstrate that Thunderbolt remains robust even under asynchronous network conditions and partial committee compromise.

References

[1] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. 2014. Enabling blockchain innovations with pegged sidechains. *URL: http://www.opensciencereview.com/papers/123/enablingblockchain-innovations-with-pegged-sidechains* 72 (2014), 201–224.

[2] Richard Barnes, Benjamin Beurdouche, Raphael Robert, Jon Millican, Emad Omara, and Katriel Cohn-Gordon. 2023. The Messaging Layer Security (MLS) Protocol. RFC 9420. doi:10.17487/RFC9420

[3] Fabrice Benhamouda, Shai Halevi, Hugo Krawczyk, Yiping Ma, and Tal Rabin. 2023. SPRINT: High-Throughput Robust Distributed Schnorr Signatures. Cryptology ePrint Archive, Paper 2023/427. https://eprint.iacr.org/2023/427

[4] Bruno Blanchet, Ben Smyth, Vincent Cheval, and Marc Sylvestre. 2018. ProVerif 2.00: automatic cryptographic protocol verifier, user manual and tutorial. *Version from* 16 (2018), 05–16.

[5] Miguel Castro and Barbara Liskov. 1999. Practical Byzantine Fault Tolerance. In *OSDI*.

[6] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. 2017. A Comprehensive Symbolic Analysis of TLS 1.3. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 1773–1788. doi:10.1145/3133956.3134063

[7] Ivan Damgård, Thomas Pelle Jakobsen, Jesper Buus Nielsen, Jakob Illeborg Pagter, and Michael Bæksvang Østergård. 2020. Fast Threshold ECDSA with Honest Majority. Cryptology ePrint Archive, Paper 2020/501. https://eprint.iacr.org/2020/501

[8] Sourav Das, Zhuolun Xiang, Lefteris Kokoris-Kogias, and Ling Ren. 2022. Practical Asynchronous High-threshold Distributed Key Generation and Distributed Polynomial Sampling. Cryptology ePrint Archive, Paper 2022/1389. https://eprint.iacr.org/2022/1389

[9] Christian Decker and Roger Wattenhofer. 2015. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Stabilization, Safety, and Security of Distributed Systems: 17th International Symposium, SSS 2015, Edmonton, AB, Canada, August 18-21, 2015, Proceedings 17*. Springer, 3–18.

[10] Joseph Dryja and Tadge Poon. 2016. The Bitcoin Lightning Network: Scalable off-chain instant payments. https://lightning.network/lightning-network-paper.pdf.

[11] Rosario Gennaro, Steven Goldfeder, and Seny Kamara. 2020. One Round Threshold ECDSA with Identifiable Abort. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer, 520–550.

[12] Chelsea Komlo and Ian Goldberg. 2020. FROST: Flexible Round-Optimized Schnorr Threshold Signatures. In *USENIX Security Symposium*.

[13] Lightning Labs. n.d.. *Taproot Assets*. https://docs.lightning.engineering/the-lightning-network/taproot-assets Builder's Guide to the LND Galaxy.

[14] Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, Matteo Maffei, and Srivatsan Ravi. 2019. Concurrency and Privacy with Payment-Channel Networks. arXiv:1911.09148 [cs.CR] https://arxiv.org/abs/1911.09148

[15] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. 2013. The Tamarin Prover for the Symbolic Analysis of Security Protocols. In *International Conference on Computer Aided Verification (CAV)*.

[16] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. *https://bitcoin.org/bitcoin.pdf* (2008).

[17] Charmaine Ndolo and Florian Tschorsch. 2023. On the (Not So) Surprising Impact of Multi-Path Payments on Performance and Privacy in the Lightning Network. Cryptology ePrint Archive, Paper 2023/1624. https://eprint.iacr.org/2023/1624

[18] Jonas Nick, Tim Ruffing, and Yannick Seurin. 2020. MuSig2: Simple Two-Round Schnorr Multi-Signatures. Cryptology ePrint Archive, Paper 2020/1261. doi:10.1007/978-3-030-84242-0_8

[19] Andrew Poelstra. 2017. Scriptless Scripts. Presented at BPASE '17. https://diyhpl.us/wiki/transcripts/bpase/2017/scriptless-scripts/.

[20] Joseph Poon and Thaddeus Dryja. 2016. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. https://lightning.network/lightning-network-paper.pdf.

[21] Stefanie Roos, Pedro Moreno-Sanchez, Aniket Kate, and Ian Goldberg. 2017. Settling payments fast and private: Efficient decentralized routing for path-based transactions. *arXiv preprint arXiv:1709.05748* (2017).

[22] Tim Ruffing, Viktoria Ronge, Elliott Jin, Jonas Schneider-Bensch, and Dominique Schröder. 2022. ROAST: Robust Asynchronous Schnorr Threshold Signatures. Cryptology ePrint Archive, Paper 2022/550. doi:10.1145/3548606.3560583

[23] Benedikt Schmidt, Simon Meier, and Cas Cremers. 2012. Automated analysis of Diffie-Hellman protocols and advanced security properties. In *25th IEEE Computer Security Foundations Symposium (CSF)*. IEEE, 78–94.

[24] Pieter Wuille and et al. 2020. BIP340: Schnorr Signatures for secp256k1. (2020). https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki.

[25] Maofan Yin, Dahlia Malkhi, Michael K Reiter, Guy Gueta, and Ittai Abraham. 2019. HotStuff: BFT Consensus with Linearity and Responsiveness. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*.