Bitcoin Anchoring for Zenon Network Checkpoints:

A Formal Treatment with Taproot Reference Implementation

Research Paper v2.0

December 21, 2025

Abstract

We present a formal mathematical framework for using Bitcoin transactions as immutable timestamped commitments to Zenon Network checkpoints. We prove that a Bitcoin-confirmed transaction can serve as a cryptographically binding anchor for Zenon state commitments without requiring Bitcoin to execute or validate Zenon's consensus rules. Our construction provides two key guarantees: (1) *binding* — the anchor cryptographically commits to exactly one checkpoint tuple under standard collision-resistance assumptions, and (2) *timestamping* — the commitment's existence is provably timestamped by Bitcoin's proof-of-work chain. We provide both an abstract framework applicable to any Bitcoin transaction type and a concrete reference implementation using Taproot's script-path spending. The framework enables lightweight clients to obtain objective temporal ordering of Zenon checkpoints with Bitcoin's security guarantees, while maintaining clear separation between Bitcoin's timestamping service and Zenon's execution layer. We precisely delineate the boundaries of what such anchoring can and cannot prove, and provide a complete, interoperable specification for the Taproot-based encoding.

Keywords: Bitcoin, Taproot, Zenon Network, Blockchain Anchoring, Cryptographic Commitments, Timestamping, Cross-Chain Security, State Commitments, BIP 341, BIP 342

Table of Contents

1. Introduction

The problem of establishing objective temporal ordering of events across independent blockchain systems has profound implications for interoperability, security, and trust minimization. While cross-chain bridges typically require complex verification of foreign chain consensus rules, we present an alternative approach: using Bitcoin's proof-of-work chain purely as a timestamping service, without requiring Bitcoin to execute or validate any external protocol logic.

This paper formalizes the use of Bitcoin transactions to anchor Zenon Network checkpoints. The key insight is that Bitcoin's immutable transaction history can serve as a public bulletin board for cryptographic commitments. By embedding a hash commitment to a Zenon checkpoint in a Bitcoin transaction, we obtain two valuable properties: (1) cryptographic binding to a specific checkpoint, and (2) a Bitcoin-timestamped proof that the commitment existed at a particular point in time.

Scope and Implementation: While our framework applies to any Bitcoin transaction type (including OP_RETURN outputs), we provide a concrete reference implementation using Taproot (BIP 341/342) for several reasons: (1) improved privacy through key-path/script-path indistinguishability, (2) more efficient use of block space, (3) potential for complex multi-party commitment schemes, and (4) alignment with Bitcoin's latest cryptographic primitives. However, the mathematical security properties (binding and timestamping) hold regardless of encoding method.

Our contribution is fourfold. First, we provide precise mathematical definitions of the commitment scheme and verification predicates applicable to any Bitcoin encoding. Second, we prove the security properties under standard cryptographic assumptions. Third, we specify a complete, interoperable Taproot-based encoding that any implementation can adopt. Fourth, we clearly delineate what such anchoring can and cannot prove, avoiding common misconceptions about cross-chain security.

The remainder of this paper is structured as follows. Section 2 establishes our notation and defines the core objects. Section 3 formalizes Bitcoin inclusion as a logical statement. Section 4 specifies both abstract extraction requirements and a concrete Taproot encoding. Sections 5 and 6 prove the binding and timestamping properties respectively. Section 7 analyzes reorganization risk. Section 8 presents the complete verification algorithm. Section 9 discusses limitations and trust assumptions. Section 10 explores extensions. Section 11 concludes.

2. Objects and Notation

We begin by establishing formal notation for Zenon and Bitcoin objects. Let ■ denote Zenon's state space, where each state represents a complete snapshot of the network at a given point in time.

Definition 2.1 (Zenon Objects):

- Let $s_h \in$ ■ denote the Zenon state at momentum height h
- Let Hdr_h denote the Zenon header/commitment object at height h
- Let $root_h$ denote the state root (or state commitment) in Hdr_h
- Let m_h denote a unique chain identifier (e.g., momentum hash) at height h

We construct a domain-separated commitment that binds all critical checkpoint data into a single cryptographic digest. This prevents commitment reuse across different contexts and ensures unambiguous interpretation.

Definition 2.2 (Checkpoint Commitment):

$$c_h := H(\text{"ZNN\_ANCHOR\_V1"} \,||\, h \,||\, m_h \,||\, root_h)$$

where $H: \{0,1\}^* \to \{0,1\}^{256}$ is a cryptographic hash function modeled as collision-resistant in the random oracle model, and || denotes concatenation. For the reference implementation, we use SHA-256.

Encoding Specification: When computing the commitment, integer fields are encoded as 8-byte little-endian, and hash fields (m_h, $root_h$) are encoded as their raw 32-byte values. The domain separator is ASCII-encoded. This gives:

```
c_h = SHA256( "ZNN_ANCHOR_V1" || // 14 bytes ASCII LE64(h) || // 8 bytes m_h
|| // 32 bytes root_h // 32 bytes ) // Total: 86 bytes input
```

The domain separator "ZNN_ANCHOR_V1" prevents cross-protocol attacks and enables version evolution. The inclusion of height h, momentum hash m_h, and state root $root_h$ ensures the commitment uniquely identifies a specific checkpoint in Zenon's history.

3. Bitcoin Anchor as an Inclusion Statement

We now formalize what it means for a commitment to be "anchored" in Bitcoin. This requires precise definitions of Bitcoin's block structure and transaction inclusion.

Definition 3.1 (Bitcoin Objects):

- Let B_0, B_1, B_2, ... denote Bitcoin block headers
- Let Tx denote a Bitcoin transaction
- Let b denote a Bitcoin block height (non-negative integer)

Definition 3.2 (Inclusion Predicate):

InBlock(Tx, B_b) is a boolean predicate that is true if and only if transaction Tx is included in the Merkle tree of block B_b. This can be verified via a Merkle inclusion proof.

Definition 3.3 (Best Chain):

BestChain(b) denotes the Bitcoin blockchain selected by the longest-chain (most cumulative proof-of-work) rule up to height b. In the presence of forks, BestChain(b) selects the chain with maximum $\Sigma_{i=0..b}$ difficulty(B_i).

We can now precisely state what it means for a commitment c_h to be "Bitcoin-anchored with z confirmations."

Definition 3.4 (Anchored Statement):

$$S_{BTC}(c_h, z) := \exists b \; (\; \text{InBlock}(\text{Tx}(c_h), B_b) \wedge B_{b+z} \in \text{BestChain}(b+z) \;)$$

This states: "There exists a Bitcoin block height b such that a transaction committing to c_h is included in block B_b, and at least z additional blocks have been built on top of B_b in the best chain."

Important: This is purely a Bitcoin-side statement. It makes no claims about the validity or correctness of Zenon's state at height h. It merely establishes that a commitment to that state was published on Bitcoin.

4. Commitment Extraction: Abstract and Concrete

To make the anchoring scheme interoperable, we must specify how a commitment c_h is embedded in a Bitcoin transaction and how verifiers can extract it. We first define abstract requirements, then provide a concrete Taproot-based specification.

4.1 Abstract Extraction Predicate

Definition 4.1 (Extraction Function):

$$\text{Extract: Tx} \to \{0,1\}^{256} \cup \{\bot\}$$

The extraction function takes a Bitcoin transaction and returns either a 256-bit commitment value or $\bot$ (representing failure/not-found). The function must satisfy:

- **Determinism:** For any transaction Tx, Extract(Tx) always returns the same value
- **Public Verifiability:** Any party can compute Extract(Tx) from the raw transaction data
- **Unambiguity:** If Extract(Tx) = $c \neq \bot$, then c is uniquely determined
- **Standardization:** All implementations use identical extraction logic

Definition 4.2 (Anchoring Predicate):

$$\text{Anchors}(Tx, c_h) := (\text{Extract}(Tx) = c_h)$$

4.2 Taproot Reference Implementation

We now specify a concrete, standardized encoding using Taproot's script-path spending (BIP 341/342). This serves as the reference implementation for interoperability.

Specification 4.1 (Taproot Anchor Encoding):

A Zenon anchor transaction using Taproot must satisfy:

1. **Output Type:** Contains at least one Taproot (SegWit v1) output

2. **Script Tree:** The Taproot output's Merkle tree includes a leaf script with the following template:
   OP_RETURN <32-byte commitment>

3. **Commitment Location:** The 32-byte commitment is the first and only push in the OP_RETURN leaf script

4. **Leaf Version:** Uses Tapscript (leaf version 0xc0)

5. **Script Path Reveal:** To prove the anchor, the spending transaction must reveal this script path in its witness stack

Algorithm 4.1 (Extract$_{\text{Taproot}}$):

```
Input: Bitcoin transaction Tx Output: 32-byte commitment or ⊥ 1. For each
input i in Tx.inputs: a. If input i spends a Taproot output (witness version
1): i. Parse witness stack ii. If witness contains script-path spend: -
Extract revealed script S - If S matches template: OP_RETURN <32 bytes> *
Return the 32-byte value 2. If no valid anchor found in any input, return ⊥
```

Design Rationale: This encoding uses script-path spending (not the key-path) to make the commitment explicitly verifiable from the spending transaction's witness data. The anchor output itself can use key-path spending for privacy; the commitment is only revealed when the output is later spent (which could be the same block or later).

4.3 Alternative Encodings and Comparison

While Taproot is our reference implementation, we acknowledge alternative encodings and explain the tradeoffs:

Method	Privacy	Space Efficiency	Complexity	Std Status
OP_RETURN	Low (obvious)	Good (40 bytes)	Minimal	Legacy
Taproot Script	High (hidden)	Excellent	Medium	Modern (BIP 341)
Taproot Tweak	Highest	Best (0 extra)	High	Experimental

Table 1: Comparison of Bitcoin commitment encoding methods. All provide equivalent security (binding and timestamping) but differ in privacy and efficiency.

Why Taproot for the Reference Spec: We chose Taproot script-path for the reference implementation because: (1) it hides the commitment until spending, improving privacy, (2) it's more space-efficient than OP_RETURN, (3) it aligns with Bitcoin's current best practices (BIP 341/342), and (4) it enables future extensions like multi-party commitments. However, the mathematical framework (Sections 5-7) applies equally to any encoding that satisfies the abstract extraction requirements.

5. Security Analysis: The Binding Property

The primary security property we require is *binding*: an anchor should cryptographically commit to exactly one Zenon checkpoint, such that no adversary can later claim the same anchor corresponds to a different checkpoint. We formalize this under standard cryptographic assumptions.

5.1 Collision Resistance Requirement

Our binding property relies on the collision resistance of the hash function H used in constructing commitments.

Assumption 5.1 (Collision Resistance):

> A hash function $H: \{0,1\}^* \rightarrow \{0,1\}^n$ is collision-resistant if for all probabilistic polynomial-time adversaries ■, the probability that ■ outputs distinct x, x' such that H(x) = H(x') is negligible in n.

For our construction, we use H = SHA-256, which provides 256-bit output. Under current understanding of cryptographic hash functions, finding collisions in SHA-256 requires approximately 2^{128} evaluations (birthday bound), which is computationally infeasible even for nation-state adversaries.

5.2 Binding Theorem

We now prove that under the collision resistance assumption, our commitment scheme is binding.

Theorem 5.1 (Binding Property):

> Suppose H is collision-resistant. Let c_h be a commitment extracted from a Bitcoin transaction Tx via Extract(Tx) = c_h. Then, except with negligible probability, there does not exist a distinct checkpoint tuple $(h', m'_{h'}, root'_{h'})$ such that:
>
> $c_h = H(\text{"ZNN\_ANCHOR\_V1"} \,||\, h' \,||\, m'_{h'} \,||\, root'_{h'})$
>
> where $(h, m_h, root_h) \neq (h', m'_{h'}, root'_{h'})$.

Proof:

Suppose for contradiction that an adversary produces two distinct checkpoint tuples $(h, m_h, root_h)$ and $(h', m'_{h'}, root'_{h'})$ that both hash to the same commitment c_h. Then we have:

```
H("ZNN_ANCHOR_V1" || h || m_h || root_h) = H("ZNN_ANCHOR_V1" || h' || m'_h' || root'_h')
```

Since the tuples are distinct, at least one of $(h, m_h, root_h)$ differs from $(h', m'_{h'}, root'_{h'})$, making the concatenated inputs to H different. Therefore, this constitutes a collision in H, contradicting the collision resistance assumption. Hence, no such distinct tuple can exist except with probability bounded by the

collision-finding probability (negligible in the security parameter). ■

Corollary 5.1: Once a commitment c_h is anchored on Bitcoin via any encoding method that satisfies Extract, the checkpoint data (h, m_h, $root_h$) is cryptographically bound to that anchor. An adversary cannot produce alternative checkpoint data that validates against the same anchor.

Remark 5.1: The binding property is independent of the Bitcoin encoding method (Taproot, OP_RETURN, etc.). It depends only on the collision resistance of H and the determinism of Extract.

6. Timestamping Guarantees

Beyond binding, the second key property of Bitcoin anchoring is timestamping: establishing an upper bound on when the commitment was created. We formalize this guarantee precisely.

Definition 6.1 (Block Time):

Let $\tau(b)$ denote the timestamp associated with Bitcoin block B_b. This is the timestamp field in the block header, subject to Bitcoin's consensus rules (median-time-past for validation, with bounded clock drift tolerance).

Theorem 6.1 (Timestamping Property):

> If $S_{BTC}(c_h, z)$ holds at Bitcoin height b (i.e., the commitment is included in block B_b with z confirmations), then there exists a publicly verifiable proof that commitment c_h was published no later than time $\tau(b)$.

Proof:

The proof is constructive. To verify the timestamp claim, a verifier:

1. Obtains the Bitcoin block header B_b
2. Verifies B_b is part of the best chain (via cumulative proof-of-work)
3. Verifies transaction Tx is in B_b (via Merkle proof)
4. Verifies Extract(Tx) = c_h
5. Reads timestamp $\tau(b)$ from B_b's header

Since Bitcoin's consensus rules enforce that B_b could not have been created before $\tau(b)$ (within ±2 hour clock drift tolerances), and the transaction Tx is provably included in B_b, it follows that c_h existed and was published to the Bitcoin network no later than $\tau(b)$ + 2 hours. ■

Critical Distinction: This theorem proves ONLY that *someone* created and published commitment c_h by the specified time. It does NOT prove:

- That the Zenon state at height h was computed correctly
- That Zenon consensus was honest at height h
- That this checkpoint represents the canonical Zenon chain
- That the state data corresponding to $root_h$ is available
- Anything about the validity of Zenon transactions

The semantic meaning of the commitment—what it represents in Zenon's protocol—depends entirely on Zenon's own consensus and validity rules, which Bitcoin does not and cannot verify. Bitcoin merely

provides a timestamp upper bound for when the commitment came into existence.

Remark 6.1 (Timestamp Bounds): Bitcoin timestamping provides a tight upper bound (commitment existed no later than $\tau(b)$ + clock drift) but does NOT provide a lower bound. The commitment could have been created at any time before $\tau(b)$. For establishing "no earlier than" guarantees, one would need additional evidence (e.g., the commitment references data that provably didn't exist until a certain time).

7. Bitcoin Reorganization Risk

The security of Bitcoin anchoring is not absolute; it depends on the stability of Bitcoin's blockchain. A blockchain reorganization (reorg) could potentially remove an anchor transaction from the canonical chain. We analyze this risk quantitatively.

7.1 Probabilistic Security Model

Bitcoin's security model is probabilistic: an adversary with fraction $q < 0.5$ of total hashpower has a non-zero (but rapidly decreasing) probability of reorganizing the chain beyond a given depth.

Definition 7.1 (Reorganization):

A reorganization of depth z occurs when a previously confirmed block at height b is replaced by an alternative block in the new best chain, affecting all blocks from b to $b+z$.

7.2 Confirmation Depth Analysis

Following the analysis in Nakamoto's original Bitcoin paper, we can bound the probability that an anchor transaction with z confirmations gets reversed.

Theorem 7.1 (Nakamoto Bound):

> Let q be the fraction of total hashpower controlled by an adversary, where $0 < q < 0.5$. The probability that a transaction included at depth z is reversed by a reorganization is upper bounded by:
>
> $P_{reorg}(z, q) \leq (q/(1-q))^z$
>
> This bound is asymptotically tight for large z.

Practical Values: For typical adversary models:

Confirmations (z)	q = 0.1	q = 0.25	q = 0.4
6	1.2×10^{-3}	1.2×10^{-2}	1.9×10^{-1}
10	9.1×10^{-5}	3.8×10^{-4}	6.0×10^{-2}
20	8.3×10^{-13}	1.4×10^{-8}	3.7×10^{-3}
100	≈ 0	≈ 0	≈ 0

Table 2: Reorganization probabilities for various confirmation depths z and adversary hashpower fractions q. Bitcoin's standard recommendation is $z \geq 6$ for significant value transfers.

Parameterization Strategy: The security level of anchoring can be tuned by adjusting the required confirmation depth z. Applications requiring high security should use larger z. For instance:

- Standard security: z = 6 (Bitcoin's default for transactions)
- High security: z = 20-100 (for critical checkpoints)
- Maximum security: z = 100+ (approaching practical irreversibility)

8. Complete Verification Algorithm

We present a complete, deterministic algorithm for verifying that a Zenon checkpoint is properly Bitcoin-anchored using the Taproot reference implementation.

Algorithm 8.1 (Anchor Verification):

Input:
• Zenon checkpoint data: $(h, m_h, root_h)$
• Bitcoin proof package: $\pi_{BTC} = (B_b, ..., B_{b+z}, Tx, merkle\_proof)$
• Required confirmation depth: z

```
1. Compute c_h using Definition 2.2
2. Verify Bitcoin header chain validity (B_b through B_{b+z})
3. Verify best chain (maximum cumulative work)
4. Verify Merkle inclusion of Tx in B_b
5. Run Extract_Taproot(Tx) per Algorithm 4.1
6. If Extract_Taproot(Tx) ≠ c_h, return FALSE
7. Verify confirmation depth is ≥ z
8. Return TRUE
```

9. Boundaries and Limitations

It is crucial to understand precisely what Bitcoin anchoring does and does not prove.

9.1 What Anchoring Does Not Prove

Bitcoin anchoring provides ONLY binding and timestamping. It does NOT prove:

- Execution correctness of Zenon state transitions
- Honesty of Zenon consensus mechanism
- Which Zenon fork is canonical (in case of chain splits)
- Validity of individual Zenon transactions
- Data availability of the anchored state
- Liveness or progress of the Zenon network

9.2 Trust Assumptions

The security rests on:
1. Bitcoin's proof-of-work consensus security
2. Collision resistance of SHA-256
3. Separate trust in Zenon's consensus (for state validity)
4. Separate data availability assumptions
5. Correct implementation of Extract and commitment construction

10. Extensions and Variants

10.1 Checkpoint Bundles

Extended commitments can include additional metadata:

$$\beta_h := (\text{root}_h, \text{validator\_set\_id}_h, \text{epoch}_h, \ldots) \quad c_h := H(\text{"ZNN\_ANCHOR\_V1"} \,||\, h \,||\, m_h \,||\, \beta_h)$$

10.2 Multiple Checkpoint Anchoring

Multiple checkpoints can be anchored in a single transaction using Merkle trees to amortize Bitcoin fees while maintaining logarithmic proof sizes.

11. Conclusion

We have presented a rigorous framework for Bitcoin anchoring of Zenon checkpoints with: (1) formal security proofs of binding and timestamping properties, (2) a complete, interoperable Taproot-based reference specification, (3) clear boundaries of what anchoring proves and doesn't prove, and (4) practical guidance on confirmation depths and encoding tradeoffs. The key insight is that Bitcoin serves as a timestamping service without executing Zenon logic, providing objective temporal ordering while Zenon maintains full sovereignty over consensus and execution.

12. References

[1] Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System.

[2] Towns, A. (2021). Taproot: SegWit version 1 spending rules. BIP 341.

[3] Poelstra, A., et al. (2019). Tapscript. BIP 342.

[4] Garay, J., Kiayias, A., & Leonardos, N. (2015). The Bitcoin Backbone Protocol.

[5] Pass, R., Seeman, L., & Shelat, A. (2017). Analysis of the Blockchain Protocol.

[6] Gervais, A., et al. (2016). On the Security and Performance of Proof of Work.

[7] Bonneau, J., et al. (2015). SoK: Research Perspectives for Bitcoin.

[8] Bayer, D., Haber, S., & Stornetta, W. S. (1993). Digital Time-Stamping.

[9] Goldreich, O., et al. (1986). How to Construct Random Functions.

[10] Bellare, M., & Rogaway, P. (1993). Random Oracles are Practical.

[11] Merkle, R. C. (1987). A Digital Signature Based on Encryption.

[12] Zenon Network. (2024). Network of Momentum Technical Whitepaper.