Non-Interactive Translation of Winternitz Signatures to Lamport Signatures via Secret Sharing

Mikhail Sergeevitch[1], Konrad Staniec[1], David Tse[2,3], Nikhil Vanjani[2], and Robin Linus Woll[3]

[1]Babylon Labs
[2]Byzantine Research
[3]Stanford University

July 27, 2026

Abstract

BitVM2 brought arbitrary program execution to Bitcoin, yielding the first light-client-based bridge to its second layers and reducing the trust required at setup to a single honest participant. Its successors, BitVM3 and BABE, move the disputed computation off-chain into Garbled Circuits (GCs), cutting worst-case on-chain dispute costs by roughly three orders of magnitude and so opening participation beyond well-capitalized operators; BABE in turn cuts the off-chain storage and setup costs of BitVM3's garbled circuits by a comparable factor. What still reaches the chain, however, is bulkier than it need be. BitVM2 commits its data with compact Winternitz one-time signatures (WOTS), whereas BitVM3 and BABE must reveal GC input labels on-chain and so fall back on far bulkier Lamport signatures, which play two roles at once: GC input labels and Bitcoin-verifiable commitments. WOTS cannot simply be substituted, because its hash chains are monotonic—an evaluator holding one state's preimage can hash forward and obtain several active labels on a single input wire, destroying the circuit's privacy. Recovering BitVM2's compactness therefore calls for a practical GC whose input labels are WOTS signatures—a construction that has remained out of reach. We close this gap with a non-interactive *WOTS-to-Lamport translation gadget*: the garbler commits on-chain using compact WOTS chains, and the evaluator expands them off-chain into exactly the orthogonal Lamport labels the GC consumes, learning nothing about the mutually exclusive ones. A naive translation table that enumerated messages would be exponentially large. Two symmetries bring it down to quadratic: reconstruction from a Shamir sharing depends only on *how many* shares are held and not on *which*, which collapses the exponentially many messages onto the single checksum weight WOTS already computes; and monotonicity, the very property that made WOTS unusable, orders an evaluator's access by inclusion. We model the gadget as a garbling scheme and prove it adaptively private. Applied to BABE, it restores WOTS-scale commitments to a GC-based protocol, cutting total on-chain script size by more than 3× and bringing dispute transactions within Bitcoin's standardness limit.

Authors are listed in alphabetical order of last name.

Contents

1 Introduction **3**
- 1.1 Our Contributions . . . 3

2 Winternitz One-Time Signatures (WOTS) **5**
- 2.1 Key Generation . . . 5
- 2.2 Signing and Verification . . . 5
- 2.3 One-Time Existential Unforgeability . . . 6

3 The Target: Orthogonal Lamport Labels **6**

4 The Translation Mechanism **6**
- 4.1 Dynamic Degree Bound . . . 6
- 4.2 Gathering Evaluation Points . . . 7
- 4.3 The Boolean Mapping . . . 7

5 Worked Examples **9**
- 5.1 Case 1: $d = 1$ (1-Bit Digits) . . . 9
- 5.2 Case 2: $d = 2$ (2-Bit Digits) . . . 9
- 5.3 Case 3: $d = 4$ (4-Bit Digits) . . . 9
- 5.4 General Case: Arbitrary d . . . 9

6 Performance Analysis and the Chunking Optimization **11**
- 6.1 The Quadratic Spatial Blowout . . . 11
- 6.2 The Chunking Optimization . . . 11
- 6.3 Computational Complexity . . . 11

7 Proof of Correctness: Terminal State $v_i = K$ **12**

8 Security Analysis of the Translation Gadget: Overview **12**
- 8.1 Completeness . . . 12
- 8.2 Soundness (Unforgeability) . . . 12
- 8.3 Perfect Privacy of the Sharing Layer . . . 13

9 Extension to Multi-Digit Checksums **13**
- 9.1 The Carry-Over Problem . . . 13
- 9.2 The Master Key Chain and 2D Entry Grid . . . 14
- 9.3 Verification and Security . . . 15
- 9.4 Complexity Analysis: 15×32 Split Checksum . . . 15

10 Application to BABE: An Engineering Design Example **15**
- 10.1 The On-Chain Bottleneck: Public Key Commitments . . . 15
- 10.2 System Parameters and Proof Chunking . . . 17
- 10.3 The Transaction Graph Flow and Empirical Sizes . . . 17
- 10.4 Preliminary Runtime Results . . . 20

11 Conclusion **22**

1 Introduction

Verifying arbitrary computations and succinct zero-knowledge proofs on the Bitcoin blockchain has long been hindered by the limited expressiveness of Bitcoin Script and strict transaction size limits. BitVM [Lin23] first opened this door, and BitVM2 [LAA+25] made trust-minimized bridging practical by executing the disputed computation in Bitcoin Script. Its successors move that computation off-chain into Garbled Circuits (GCs), using the blockchain solely as a dispute resolution layer and cutting worst-case on-chain dispute costs by roughly three orders of magnitude, which lowers the bonds and deposits that had confined participation to well-capitalized operators. They differ in what they garble: BitVM3 [WAA+26] garbles a SNARK verifier, whereas BABE [GKS+26] garbles only an elliptic-curve scalar multiplication and layers witness encryption on top, releasing a ciphertext only if SNARK verification succeeds; BABE thereby also cuts the off-chain storage and setup costs of BitVM3's garbled circuits by a comparable factor.

What these protocols share is that the GC input labels must be revealed on the Bitcoin blockchain, and this is precisely where BitVM2's compactness is lost. BitVM2 commits its on-chain data with Winternitz One-Time Signatures (WOTS), whose hash chains encode several bits apiece and are correspondingly small. A Garbled Circuit, however, consumes orthogonal binary labels, so BitVM3 and BABE fall back instead on Lamport One-Time Signatures [Lam79], which do double duty: they are exactly the orthogonal, binary labels that Yao's Garbled Circuits natively consume, and they are simultaneously verifiable in Bitcoin Script. Lamport signatures are, however, highly spatially inefficient. GC input sizes vary across these protocols; the smallest is a single BN254 $\mathbb{G}_1$ group element at 508 bits, whose Lamport public key alone runs to over 20 Kilobytes—roughly 5 KvB of on-chain footprint once the witness discount is applied.

A single commitment therefore still fits within Bitcoin's 10 KvB standardness limit; what breaks the budget is malicious security. Achieving it requires cut-and-choose, under which the labels of every evaluated circuit must be revealed on-chain alongside the Prover's own commitment. We target 40-bit statistical security with the parameters $(181, 7)$, the regime deployed in Section 10: the Challenger generates 181 garbled circuits, of which 174 are audited during setup and the remaining 7 are actively evaluated during a dispute. Those 7 circuits, together with the Prover's own commitment, put $1 + 7 = 8$ sets of labels on the chain. With Lamport this would cost roughly 40 KvB in public keys and a further 20 KvB in signatures—some 60 KvB in total, six times over the 10 KvB limit.

Winternitz One-Time Signatures (WOTS) offer significant data compression by grouping bits into larger digits (e.g., representing 4 bits using a single hash chain of length 15). Unfortunately, Garbled Circuits cannot natively ingest WOTS signatures. WOTS hash chains are monotonic; an Evaluator holding the preimage for state 3 can trivially hash forward to compute the preimages for states 4 through 15. If fed directly into a Garbled Circuit, this monotonicity would grant the Evaluator multiple active wire labels for a single input, instantly compromising the security of the circuit (e.g., allowing the extraction of the global offset in Free-XOR).

1.1 Our Contributions

In this paper, we resolve the spatial bottleneck of Garbled Circuit verification on Bitcoin by introducing a non-interactive **WOTS-to-Lamport Translation Gadget**. Our framework allows a Garbler to commit on-chain using highly compressed WOTS chains, while the Evaluator executes an off-chain translation layer to securely expand those commitments back into orthogonal binary Lamport labels.

Specifically, our contributions are:

- **Dynamic Shamir Translation:** We construct an oblivious evaluator using Shamir Secret Sharing over a finite field. By embedding Lamport labels as the constant terms of dynamically bounded polynomials, we orchestrate a point deficit that strictly enforces the Boolean logic of the target labels.

- **Provable Adaptive Privacy:** We provide rigorous completeness and soundness proofs, demonstrating that the translation layer neutralizes the monotonicity of WOTS. An attacker attempting to reconstruct labels for an unearned state by hashing forward faces a point deficit exactly equal to the number of steps they cheated, and the underlying secret sharing hides the label *perfectly* at that deficit. We lift this into an end-to-end guarantee: modeling the gadget as a garbling scheme with sampled circuits, we prove adaptive privacy against an adversary who chooses the message after seeing the translation table. The end-to-end statement is computational—it rests on the unforgeability of WOTS, the one-wayness of the chain hash, and a masking assumption on the point encryption—and we instantiate every ingredient concretely, reaching a 105-bit security level. We defer these details to the full version of the paper.

- **Multi-Digit Checksum Duality:** We introduce a 2D Entry Grid extension that allows the overarching WOTS checksum to be split across multiple hash chains. This eliminates the requirement for sequential $\mathcal{O}(W_{\max})$ hashing depth without adding spatial overhead to the underlying polynomials.

- **BABE System Integration:** We provide an end-to-end engineering architecture applying our gadget to the BABE protocol. The committed data shrinks 5.6× (public key 20.3 → 2.7 KB, signature 10.2 → 2.7 KB); netted against the larger WOTS verification script, this yields an $> 3\times$ reduction in total on-chain script size, safely bringing zero-knowledge proof verification within Bitcoin's 10 KvB limits.

Remark 1 (Why non-interactively)**.** *Were interaction available, the problem would be easy: the Evaluator's input is public—it is committed on-chain—so the Garbler could simply wait until the message is known and send the N selected labels directly. What rules this out is not cryptography but incentives. In an optimistic dispute the Garbler is the party that loses when the Evaluator succeeds, so a protocol that depends on the Garbler answering a request at dispute time is a protocol the Garbler will decline to run; a round trip would in any case cost an on-chain transaction, its confirmation delay, and an additional timeout window to police. All label material must therefore be committed* before *the message is known, and be openable, by the message alone, without further help from the Garbler. This is a demanding requirement, and it is what makes the problem hard: material fixed in advance must in principle account for every message the Evaluator might later commit to, of which there are exponentially many. The technical core of this paper is a translation table that meets the requirement at quadratic size.*

Organization of the Paper. The body of the paper develops the gadget and its deployment; the appendices carry the formal specification and the security proof.

In Section 2, we recall the Winternitz One-Time Signature scheme and its properties. In Sections 3 and 4, we describe the target Lamport label structure and the high-level intuition of our WOTS-to-Lamport translation mechanism. We then provide intuition-building examples for different digit sizes d in Section 5. In Section 6, we analyze the spatial and computational complexity of our scheme and present chunking strategies to optimize the size of the translation table. In Sections 7 and 8, we give the intuition behind the correctness and security properties of the translation

mechanism. In Section 9, we present a multi-digit checksum extension of the translation mechanism that allows the WOTS checksum to be split across multiple hash chains, eliminating the requirement for sequential $\mathcal{O}(W_{\max})$ hashing depth without adding spatial overhead to the underlying polynomials. Finally, in Section 10, we present the engineering architecture of our implementation of the translation mechanism within the BABE protocol.

2 Winternitz One-Time Signatures (WOTS)

Before describing the translation layer, we formally define the Winternitz One-Time Signature (WOTS) scheme. Let N be the total number of bits in the message to be signed. Given a parameter $d \geq 1$ representing the number of bits encoded per digit, the message is divided into $n_0 = N/d$ digits. Each digit $i \in \{1, \ldots, n_0\}$ encodes a state $v_i \in \{0, 1, \ldots, K\}$, where $K = 2^d - 1$ is the maximum value of the digit.

The scheme utilizes a secure one-way cryptographic hash function $H : \{0,1\}^\lambda \to \{0,1\}^\lambda$, where λ is the security parameter. Let $H^x(y)$ denote the function H applied iteratively to y exactly x times, with $H^0(y) = y$.

2.1 Key Generation

To authenticate the message, the scheme uses n_0 independent message hash chains and one global checksum hash chain.

- **Secret Key (sk):** The Garbler generates $n_0 + 1$ uniformly random λ-bit strings as the starting seeds for the n_0 message chains and the 1 checksum chain:
$$sk = \left(Y_1^{(0)}, Y_2^{(0)}, \ldots, Y_{n_0}^{(0)}, S^{(0)}\right)$$
- **Public Key (pk):** The Garbler computes the terminal values of the hash chains. Each message chain is hashed K times. To prevent a forgery where an attacker increments v_i by hashing an intercepted value forward, the global checksum enforces the total "weight" of the signature. The maximum possible weight is $W_{\max} = n_0 \cdot K$. The checksum chain is strictly hashed $W_{\max}$ times:
$$pk = \left(Y_1^{(K)}, Y_2^{(K)}, \ldots, Y_{n_0}^{(K)}, S^{(W_{\max})}\right)$$
where $Y_i^{(K)} = H^K(Y_i^{(0)})$ and $S^{(W_{\max})} = H^{W_{\max}}(S^{(0)})$.

2.2 Signing and Verification

To sign a specific message vector $(v_1, v_2, \ldots, v_{n_0})$:

1. The signer computes the global checksum representing the current weight:
$$W = \sum_{i=1}^{n_0} (K - v_i)$$
2. The signer reveals the corresponding intermediate hash values. The signature σ is:
$$\sigma = \left(Y_1^{(v_1)}, Y_2^{(v_2)}, \ldots, Y_{n_0}^{(v_{n_0})}, S^{(W)}\right)$$
where $Y_i^{(v_i)} = H^{v_i}(Y_i^{(0)})$ and $S^{(W)} = H^W(S^{(0)})$.

An evaluator holding σ verifies it by hashing the elements forward to the ends of their respective chains and comparing them against pk. Because the hash chains are one-way and monotonic, an evaluator holding $Y_i^{(v_i)}$ can compute $Y_i^{(k)}$ for all $k \geq v_i$, but gains no mathematical information about preimages for $k < v_i$.

2.3 One-Time Existential Unforgeability

We formalize the security of the scheme. Explicitly, a candidate signature $\sigma^* = \left(\widetilde{Y}_1, \ldots, \widetilde{Y}_{n_0}, \widetilde{S}\right)$ on a message with digits $(v_1^*, \ldots, v_{n_0}^*)$ and weight $W^* = \sum_i (K - v_i^*)$ *verifies* under pk if

$$H^{K-v_i^*}\left(\widetilde{Y}_i\right) = Y_i^{(K)} \text{ for all } i \in [n_0], \qquad \text{and} \qquad H^{W_{\max}-W^*}\left(\widetilde{S}\right) = S^{(W_{\max})}.$$

Definition 1 (One-time existential unforgeability)**.** *Consider the following experiment between a challenger and a PPT adversary $\mathcal{A}$:*

1. *The challenger generates (sk, pk) as in Key Generation and hands pk to $\mathcal{A}$.*
2. *$\mathcal{A}$ may make* at most one *signing query: it submits a message $M \in \{0,1\}^N$ and receives the signature σ on M computed as in Signing.*
3. *$\mathcal{A}$ outputs a pair (M^*, σ^*).*

$\mathcal{A}$ wins if $M^* \neq M$ (any M^*, if no signing query was made) and σ^* verifies on M^* under pk. Let $\mathsf{Adv}^{\mathrm{WOTS}}$ denote $\mathcal{A}$'s winning probability. The WOTS scheme is one-time existentially unforgeable *if $\mathsf{Adv}^{\mathrm{WOTS}}$ is negligible for every PPT adversary.*

3 The Target: Orthogonal Lamport Labels

For a target digit i, our goal is to non-interactively translate the monotonic WOTS state v_i into a set of binary Lamport labels corresponding to the d bits of v_i. For each of the $j \in [d]$ bits, there is a mutually exclusive pair $\{L_{i,j}^0, L_{i,j}^1\}$. If the target bit is 1, the evaluator must recover $L_{i,j}^1$ and gain zero information about $L_{i,j}^0$, and vice versa.

4 The Translation Mechanism

We embed each target Lamport label $L_{i,j}^b$ as the constant term (y-intercept) of a polynomial $P_{i,j,b}(x)$ over a finite field $\mathbb{F}$, such that $P_{i,j,b}(0) = L_{i,j}^b$. The evaluator interpolates this polynomial using evaluation points provided publicly by the Garbler. For each Lamport label $L_{i,j}^b$, there is a different polynomial $P_{i,j,b}$. In the following discussion, we focus on one particular label $L_{i,j}^b$ and one polynomial $P_{i,j,b}$. Towards simplifying notations in this section and focusing on intuition-building, we often drop the indices $i \in [n_0], j \in [d], b \in \{0,1\}$ and refer to the target label as L and the corresponding polynomial as P.

4.1 Dynamic Degree Bound

To prevent unauthorized reconstruction, the required threshold to interpolate $P(x)$ must scale dynamically with the evaluator's checksum weight W.

The Garbler chooses a public **threshold shift** $\Delta \geq 0$. This determines the degree of the polynomial and its coefficients as follows:

- The degree of $P(x)$ is set to $W_{\max} + \Delta - 1$.
- The coefficients of $P(x) = \sum_{m=0}^{W_{\max}+\Delta-1} c_m \cdot x^m$ are derived from the checksum hash chain such that the coefficient of degree m is $c_m = H'(\text{poly_id} \parallel S^{(m-\Delta)})$.

Because the evaluator holds $S^{(W)}$, they can compute c_m for $m - \Delta \geq W$, which is equivalent to $m \geq W + \Delta$. This allows the evaluator to evaluate and subtract all polynomial coefficients of degree $W + \Delta$ and higher. The polynomial is mathematically reduced to strictly degree $W + \Delta - 1$, requiring exactly $W + \Delta$ **points** to interpolate.

4.2 Gathering Evaluation Points

The Garbler publishes encrypted evaluation points on $P(x)$. To maintain consistency with WOTS boundary conditions, points are only encrypted by the first K preimages $(0 \dots K-1)$ of each chain.

1. **Base Points:** For every *other* digit $i' \neq i$, the Garbler publishes exactly K evaluation points, encrypted respectively by $Y_{i'}^{(0)}, \dots, Y_{i'}^{(K-1)}$. An evaluator holding $Y_{i'}^{(v_{i'})}$ can decrypt exactly $\max(0, K - v_{i'})$ points. Summing across all $i' \neq i$, the evaluator gathers a base sum of points:

$$P_{\text{base}} = \sum_{i' \neq i} (K - v_{i'}) = W - (K - v_i)$$

2. **Helper Points:** To conditionally satisfy the remaining threshold, the Garbler publishes **helper points** evaluated at the target digit $i' = i$. Let $h_k \geq 0$ be the number of helper points encrypted by $Y_i^{(k)}$. The evaluator decrypts all points for $k \geq v_i$, gathering $\sum_{k=v_i}^{K-1} h_k$ helper points.

4.3 The Boolean Mapping

For the ith digit value $v_i \in \{0, \dots, K-1\}$, let it's bit-decomposition be $v_i = \sum_{j \in [d]} 2^{j-1} \cdot v_{i,j}$, where $v_{i,j} \in \{0, 1\}$ is the value of the jth bit. Let $B_{v_i}^{(j,b)} \in \{0, 1\}$ be defined as $B_{v_i}^{(j,b)} = 1$ if $v_{i,j} = b$ and 0 otherwise.

Then, observe that we are required to recover Lamport labels $\{L_{i,j}^b\}_{i,j}$ if and only if $B_{v_i}^{(j,b)} = 1$. So, we require the **point deficit** (points needed minus points gathered) to strictly equal $1 - B_{v_i}^{(j,b)}$. This ensures that whenever $B_{v_i}^{(j,b)} = 1$, the evaluator can interpolate $P(0) = L_{i,j}^b$ and recover the target label, and whenever $B_{v_i}^{(j,b)} = 0$, the evaluator faces a point deficit of exactly 1, leaving $L_{i,j}^b$ perfectly hidden.

Let us denote $B_{v_i} = B_{v_i}^{(j,b)}$ for brevity and calculate the formulas for public threshold shift Δ and helper point distribution $\{h_k\}_k$ that satisfy the above requirements. Recall that to interpolate $L_{i,j}^b$, the evaluator requires $W + \Delta_b$ points. So, the point deficit is the required points minus the gathered points:

$$\begin{aligned} 1 - B_{v_i} &= (W + \Delta) - \left(P_{\text{base}} + \sum_{k=v_i}^{K-1} h_k \right) \\ &= \Delta + K - v_i - \sum_{k=v_i}^{K-1} h_k \end{aligned}$$

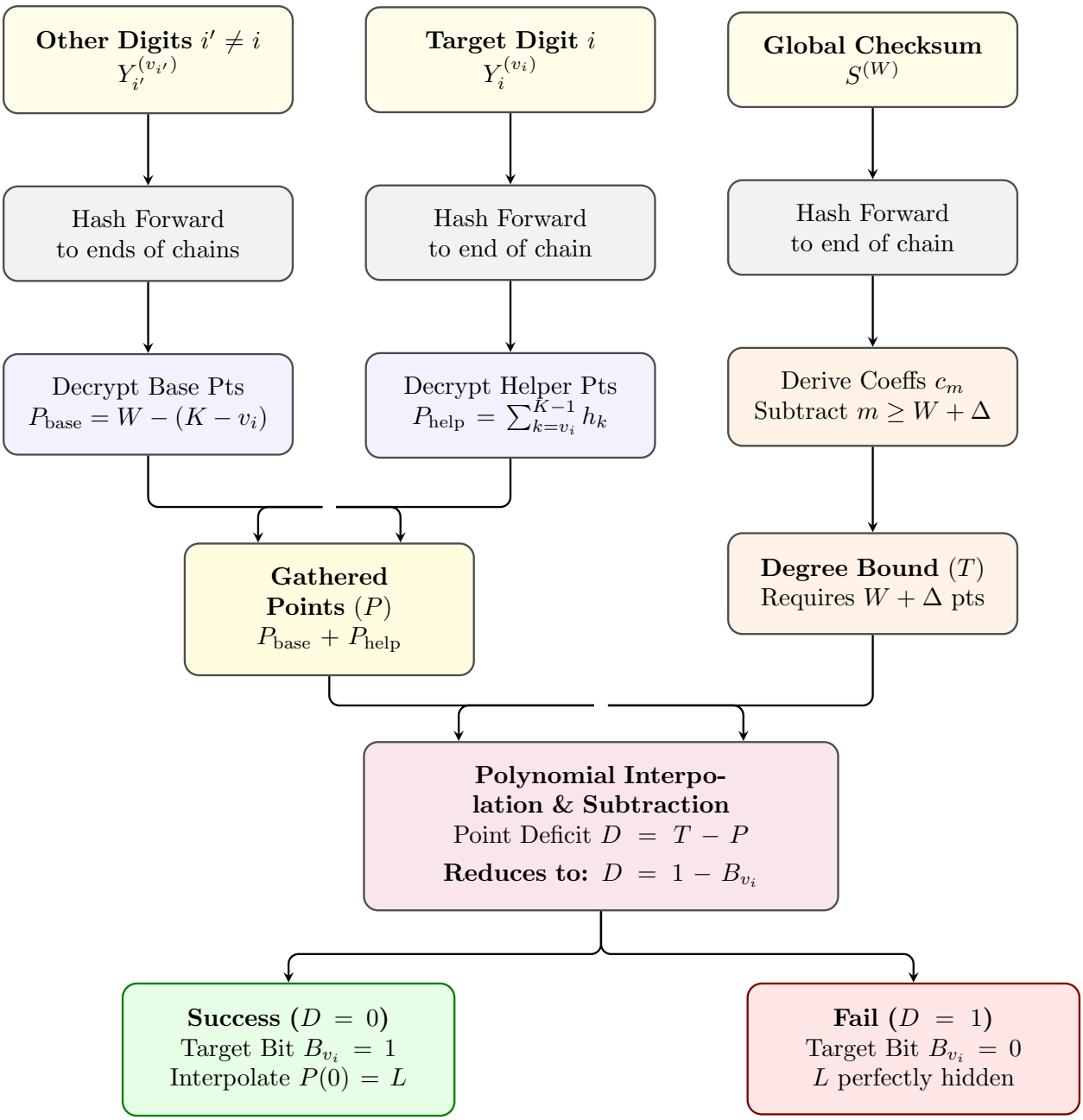

Figure 1: **Evaluator Translation Workflow.** The Evaluator parses the WOTS signature into three distinct processing tracks. The checksum establishes the dynamic Shamir threshold bound (T), while the target and base message chains yield the polynomial evaluation points (P). The interpolation logic naturally bridges the point deficit with the boolean logic of the target bit.

By writing the equation recursively for adjacent states k and $k+1$, we derive the exact construction rules:

$$\Delta := 1 - B_K$$

$$h_k := B_k - B_{k+1} + 1 \quad \text{for } k \in \{0, \ldots, K-1\}$$

This completes the informal description of the translation mechanism. In Section 5, we give examples of the construction for different digit sizes d to help build intuition.

5 Worked Examples

We now instantiate the construction for several digit sizes d, building up from the degenerate binary case to the general rule. Each example fixes a target label and reports the resulting threshold shift Δ and helper-point distribution $\{h_k\}_k$.

5.1 Case 1: $d = 1$ (1-Bit Digits)

For $d = 1$, $K = 1, v \in \{0, 1\}$.

- **Label** L^1. Here $(j, b) = (1, 1)$. Thus, $(B_1 = 1, B_0 = 0)$. Then, $\Delta = 0, h_0 = 0$.
- **Label** L^0. Here $(j, b) = (1, 0)$. Thus, $(B_1 = 0, B_0 = 1)$. Then, $\Delta = 1, h_0 = 2$.

5.2 Case 2: $d = 2$ (2-Bit Digits)

For $d = 2, K = 3$. Target vector $\mathbf{B} = (B_3, B_2, B_1, B_0)$.

- **Label** $L^1_{\mathbf{MSB}}$. Here $(j, b) = (2, 1)$. Thus, $\mathbf{B} = (1, 1, 0, 0)$. Then, $\Delta = 0, h_2 = 1, h_1 = 0, h_0 = 1$.
- **Label** $L^0_{\mathbf{MSB}}$. Here $(j, b) = (2, 0)$. Thus, $\mathbf{B} = (0, 0, 1, 1)$. Then, $\Delta = 1, h_2 = 1, h_1 = 2, h_0 = 1$.
- **Label** $L^1_{\mathbf{LSB}}$. Here $(j, b) = (1, 1)$. Thus, $\mathbf{B} = (1, 0, 1, 0)$. Then, $\Delta = 0, h_2 = 0, h_1 = 2, h_0 = 0$.
- **Label** $L^0_{\mathbf{LSB}}$. Here $(j, b) = (1, 0)$. Thus, $\mathbf{B} = (0, 1, 0, 1)$. Then, $\Delta = 1, h_2 = 2, h_1 = 0, h_0 = 2$.

5.3 Case 3: $d = 4$ (4-Bit Digits)

See Figure 2.

5.4 General Case: Arbitrary d

The translation logic generalizes to any digit parameter d. Let $\mathbf{B} = (B_K, \ldots, B_0)$ be the sequence of target bits. The parameters are defined as:

$$\boldsymbol{\Delta} = \mathbf{1} - \mathbf{B_K}$$

$$\mathbf{h_v} = \mathbf{B_v} - \mathbf{B_{v+1}} + \mathbf{1} \quad \text{for } v \in \{K-1, \ldots, 0\}$$

Because $B_v \in \{0, 1\}$, $h_v \in \{0, 1, 2\}$, which is always physically satisfiable.

Lamport Labels	**State v**															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
State (Bit 3)	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
State (Bit 2)	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
State (Bit 1)	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
State (Bit 0)	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Bit 3 (MSB)																
$L_{i,3,1}$ $(b=1)$	1	1	1	1	1	1	1	**0**	1	1	1	1	1	1	1	**0**
$L_{i,3,0}$ $(b=0)$	1	1	1	1	1	1	1	**2**	1	1	1	1	1	1	1	**0**
Bit 2																
$L_{i,2,1}$ $(b=1)$	1	1	1	**0**	1	1	1	**2**	1	1	1	**0**	1	1	1	**0**
$L_{i,2,0}$ $(b=0)$	1	1	1	**2**	1	1	1	**0**	1	1	1	**2**	1	1	1	**0**
Bit 1																
$L_{i,1,1}$ $(b=1)$	1	**0**	1	**2**	1	**0**	1	**2**	1	**0**	1	**2**	1	**0**	1	**0**
$L_{i,1,0}$ $(b=0)$	1	**2**	1	**0**	1	**2**	1	**0**	1	**2**	1	**0**	1	**2**	1	**0**
Bit 0 (LSB)																
$L_{i,0,1}$ $(b=1)$	**0**	**2**	**0**	**2**	**0**	**2**	**0**	**2**	**0**	**2**	**0**	**2**	**0**	**2**	**0**	**0**
$L_{i,0,0}$ $(b=0)$	**2**	**0**	**2**	**0**	**2**	**0**	**2**	**0**	**2**	**0**	**2**	**0**	**2**	**0**	**2**	**0**
Outside Digits	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**

Figure 2: **WOTS-to-Lamport Translation Matrix ($d = 4$).** Note that the terminal state ($v = 15$) provides no points, ensuring logic is strictly bounded by the threshold shift Δ.

6 Performance Analysis and the Chunking Optimization

We analyze the spatial and computational complexity in terms of N (message bits) and d (bits per digit). Let $n_0 = N/d$.

If a single, monolithic global checksum is used, the maximum possible weight is $W_{\max} = n_0 \cdot (2^d - 1)$.

6.1 The Quadratic Spatial Blowout

For each binary Lamport label, the Garbler must publish $W_{\max}$ evaluation points. Since there are $2N$ total labels (two for each bit), the total size of the translation table scales as:

$$\text{Table Size}_{\text{mono}} \approx 2N \cdot W_{\max} = 2N \cdot \left(\frac{N(2^d - 1)}{d} \right) = \mathcal{O}\left(\frac{N^2 2^d}{d} \right)$$

Because the spatial complexity scales quadratically with the message length N, a monolithic checksum becomes prohibitively large for payloads exceeding a few hundred bits.

6.2 The Chunking Optimization

To suppress the quadratic spatial expansion, the overarching message can be divided into C independent "chunks," each with its own localized checksum (as utilized in Section 10).

Let the message be split into C equal chunks, such that each chunk contains N/C bits and n_0/C digits. The maximum localized checksum weight for a single chunk drops proportionally to $W_{\max,c} = \frac{n_0}{C} \cdot (2^d - 1)$.

The Garbler now publishes a smaller translation table for each of the C chunks. The total system spatial complexity becomes:

$$\begin{aligned} \text{Table Size}_{\text{chunked}} &\approx C \cdot \left[2 \left(\frac{N}{C} \right) \cdot W_{\max,c} \right] \\ &= C \cdot \left[\frac{2N}{C} \cdot \frac{N(2^d - 1)}{C \cdot d} \right] \\ &= \mathbf{\frac{1}{C}} \cdot \mathcal{O}\left(\mathbf{\frac{N^2 2^d}{d}} \right) \end{aligned}$$

Spatial Compression: Chunking the proof mathematically linearizes the burden. By dividing the message into C chunks, the total spatial complexity is strictly reduced by a factor of C.

6.3 Computational Complexity

The computational bottleneck for the Garbler is the symmetric hashing required for point encryption and coefficient derivation, bounded by the table size. For the Evaluator, the dominant operations are polynomial evaluation and interpolation over $\mathbb{F}_p$.

Using standard $\mathcal{O}(T^2)$ Lagrange interpolation for a threshold $T \approx W_{\max,c}$, the Evaluator's time complexity per chunk is $\mathcal{O}((N/C)^2)$. Across C chunks, the total runtime scales as $\mathcal{O}(N^2/C)$. Similar to the spatial complexity, dividing the protocol into C independent chunks yields a strictly linear C-fold speedup in off-chain execution time, allowing protocol engineers to deterministically tune the translation layer to fit arbitrary bandwidth and runtime constraints.

7 Proof of Correctness: Terminal State $v_i = K$

We prove that at state $v_i = K$, recovery is determined solely by Δ. An evaluator at state $v_i = K$ gathers $P_{\text{base}} = W - (K - K) = W$ points. The helper sum $\sum_{k=K}^{K-1} h_k$ is empty, thus $P_{\text{target}} = 0$. The total points gathered is $P = W$. The system is solvable if the deficit $D = (W + \Delta) - W$ is zero. Since $\Delta = 1 - B_K$, then $D = 1 - B_K$. If $B_K = 1$, $D = 0$ (Success). If $B_K = 0$, $D = 1$, at which the sharing layer hides the label perfectly (Section 8.3).

8 Security Analysis of the Translation Gadget: Overview

This section gives the intuition behind the security of the WOTS-to-Lamport translation gadget, organized around three properties: Completeness, Soundness (Unforgeability), and Privacy. The arguments here are deliberately informal—they isolate *why* the point deficit enforces the Boolean logic, working directly with the idealized picture in which the evaluator simply holds a set of evaluation points.

It is worth separating up front which guarantees are unconditional and which are not, since the two are easy to conflate. The secret-sharing core is *information-theoretic*: at a point deficit of one, the unearned label is perfectly hidden, and no amount of computation recovers it (Section 8.3). What is *not* unconditional is the evaluator's inability to enlarge the set of points it holds: that rests on the one-wayness of the hash chains, the unforgeability of WOTS, and a masking assumption on the encryption of the evaluation points. The end-to-end guarantee is therefore computational, and concretely we obtain a 105-bit security level (details deferred to the full version of the paper).

8.1 Completeness

A translation gadget is complete if an honest Evaluator, possessing a valid WOTS signature σ for message vector $\mathbf{v}$, can successfully interpolate the Lamport label $L_{i,d,b}$ for every bit b such that b is the d-th bit of v_i.

Proof: As demonstrated in the Correctness Proof (Section 7), the construction rules for h_v and Δ are derived directly from the threshold equation $1 - B_{v_i} = (W + \Delta) - (P_{\text{base}} + P_{\text{target}})$. For any state v_i where the target bit $B_{v_i} = 1$, the deficit D is 0, resulting in a full-rank system of linear equations. Since $P(x)$ is a polynomial over $\mathbb{F}_p$, a full-rank system yields a unique solution for the constant term $P(0) = L$.

8.2 Soundness (Unforgeability)

The gadget is sound if an Adversary $\mathcal{A}$, holding a valid WOTS signature for state v_i, cannot recover the mutually exclusive Lamport label L' (where $b'_{v_i} = 0$) with probability significantly greater than $1/|\mathbb{F}_p|$.

Proof by Reduction: Suppose $\mathcal{A}$ successfully recovers L' for a state where $b'_{v_i} = 0$. By the construction in Section 4.3, the system of equations for L' is underdetermined with a point deficit $D = 1$. To solve the system, $\mathcal{A}$ must obtain at least one additional evaluation point. There are exactly three avenues to obtain this point:

1. **WOTS Preimage Attack:** $\mathcal{A}$ computes a preimage $Y_i^{(k)}$ for $k < v_i$ or $S^{(m)}$ for $m < W$. This reduces to breaking the one-wayness of the hash function H.

2. **Polynomial Collision:** $\mathcal{A}$ finds another valid evaluation point without knowing the preimage. Since points are encrypted by $H(\text{seed})$, this also reduces to a preimage attack or a collision in H.

3. **Brute Force:** $\mathcal{A}$ guesses the missing point or the label L' itself. The probability of success is $1/|\mathbb{F}_p|$.

If H is a secure one-way function and $|\mathbb{F}_p| \approx 2^\lambda$, the gadget is computationally sound.

8.3 Perfect Privacy of the Sharing Layer

The sharing layer provides perfect privacy if an underdetermined system (deficit $D \geq 1$) reveals zero information about the secret L. Note the scope of this claim: it is conditioned on the evaluator holding exactly the points its state entitles it to, and says nothing about how it came to hold them.

Proof: This property is inherited from Shamir's Secret Sharing scheme. Let the evaluator possess $T-1$ points for a polynomial of degree $T-1$. For every candidate label $\hat{L} \in \mathbb{F}_p$, there exists exactly one polynomial $\hat{P}(x)$ of degree $T-1$ such that $\hat{P}(0) = \hat{L}$ and $\hat{P}(x)$ passes through all $T-1$ known points. Since all such polynomials are equally likely under the uniform distribution of the polynomial coefficients $\{c_m\}$, the probability $Pr[L = \hat{L} \mid \text{Observed Points}] = Pr[L = \hat{L}]$. The evaluator's posterior knowledge is identical to their prior knowledge.

9 Extension to Multi-Digit Checksums

As the message length N and digit parameter d grow, the maximum global checksum weight $W_{\max} = n \cdot K$ scales linearly. For example, a 32-digit message with $d = 4$ ($K = 15$) yields $W_{\max} = 480$. Representing this checksum as a single monotonic hash chain requires an $\mathcal{O}(W_{\max})$ sequential execution depth. In resource-constrained environments like Zero-Knowledge proofs or Bitcoin Script, unrolling hundreds of sequential hashes creates a severe computational bottleneck.

To drastically optimize the execution depth, the checksum can be represented in a mixed-radix format across two independent hash chains (e.g., a 4-bit major digit and a 5-bit minor digit). This slashes the on-chain hashing depth from 480 to $15 + 31 = 46$ hashes. However, splitting the checksum disrupts the dynamic degree bound described in Section 4.1. An Evaluator holding two independent hash chains cannot securely derive a continuous, monotonic sequence of polynomial coefficients due to the cryptographic "carry-over" restriction.

9.1 The Carry-Over Problem

Let the checksum weight W be split into a major digit u and a minor digit v, using a radix B. The total weight is $W = u \cdot B + v$. To represent this, the Garbler instantiates two independent hash chains: S_1 with maximum length K_1, and S_2 with maximum length K_2.

To dynamically reduce the polynomial degree, the Evaluator must derive coefficients c_m for all degrees $m \geq W$. If coefficients were derived directly as $c_m = H(S_1^{(u)} \parallel S_2^{(v)})$, the Evaluator would succeed for the current degree $m = W$. However, for subsequent degrees $m > W$ (where the major digit increments to $u+1$), the Evaluator requires minor chain preimages for values $y < v$ (e.g., $y = 0$). Because WOTS hash chains are strictly one-way, the Evaluator cannot compute $S_2^{(y)}$ for $y < v$, breaking the continuous derivation of coefficients.

9.2 The Master Key Chain and 2D Entry Grid

To bypass the carry-over restriction without adding spatial overhead to the individual polynomials, the Garbler decouples coefficient derivation from the WOTS chains using an independent **Master Key Chain**.

1. **Master Key Chain:** The Garbler generates a single, independent hash chain of length $W_{\max}$:
$$\kappa_0 \to \kappa_1 \to \cdots \to \kappa_{W_{\max}}$$
where $\kappa_{m+1} = H(\kappa_m)$.

2. **Coefficient Derivation:** The coefficients for all polynomials in the system are deterministically derived strictly from this master chain:
$$c_{i,m} = H'(\text{poly_id}_i \parallel \kappa_{m-\Delta})$$

3. **The 2D Entry Grid:** To securely deliver the exact starting key κ_W to the Evaluator, the Garbler publishes a globally shared 2D grid of symmetrically encrypted keys. For every valid state $u \in \{0, \ldots, K_1\}$ and $v \in \{0, \ldots, K_2\}$, the Garbler publishes:
$$G_{u,v} = \text{Enc}\left(\text{key} = H(S_1^{(u)} \parallel S_2^{(v)}),\ \text{msg} = \kappa_{u \cdot B + v}\right)$$

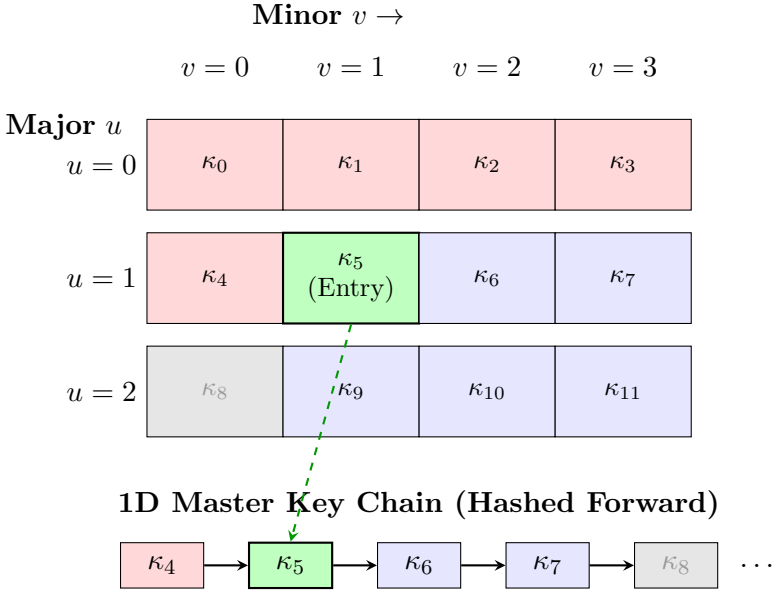

Figure 3: **2D Entry Grid for Split Checksum (Radix $B = 4$).** The Evaluator holds the split checksum state $(u = 1, v = 1)$, allowing decryption of the Entry cell yielding κ_5. Cells colored red $(\kappa_0 \ldots \kappa_4)$ are mathematically unearned; the Evaluator lacks the S_1 or S_2 preimages to decrypt them. While the Evaluator also lacks the $S_2^{(0)}$ preimage to decrypt cell κ_8 (gray), they simply hash the continuous 1D Master Key chain forward from $\kappa_5 \to \kappa_8$, cleanly bypassing the carry-over restriction.

9.3 Verification and Security

During verification, the Evaluator holds the valid checksum signature $(S_1^{(u)}, S_2^{(v)})$. They compute the symmetric decryption key $H(S_1^{(u)} \parallel S_2^{(v)})$ and decrypt the specific grid cell $G_{u,v}$ to recover κ_W.

Because κ is a continuous hash chain, the Evaluator simply hashes κ_W forward off-chain to obtain all subsequent keys $\kappa_{W+1}, \ldots, \kappa_{W_{\max}}$. This provides the exact contiguous sequence of keys needed to derive the polynomial coefficients for $m \geq W$, completely restoring the dynamic Shamir threshold reduction from Section 4.1.

Soundness of the Entry Grid: An attacker attempting to recover coefficients for an unearned weight $m < W$ must decrypt a grid cell $G_{x,y}$ where $x \cdot B + y < u \cdot B + v$. This mathematically requires either $x < u$ or $(x = u$ and $y < v)$. Because the S_1 and S_2 chains are one-way, the attacker cannot compute the necessary preimages to derive the decryption keys for these specific cells. Every cell the attacker can validly decrypt lies in the quadrant $x \geq u$ and $y \geq v$, yielding master keys κ_m strictly where $m \geq W$. Thus, coefficients for degrees $m < W$ remain inaccessible—a computational guarantee, resting on the one-wayness of the two checksum chains and on the security of the grid encryption, not an information-theoretic one.

9.4 Complexity Analysis: 15×32 Split Checksum

Consider a signature system where $W_{\max} = 480$. We optimize this by representing the checksum using a 4-bit major digit ($K_1 = 15$) and a 5-bit minor digit ($K_2 = 31$). The radix is $B = 32$.

- **Computational Savings:** The on-chain sequential hashing depth required to verify the checksum signature plummets from 480 down to exactly $15+31 =$ **46** hashes. This represents a greater than $10\times$ reduction in on-chain execution depth.
- **Spatial Overhead:** The Garbler publishes a single 2D Entry Grid containing $(15 + 1) \times (31 + 1) = 512$ ciphertexts. At 16 bytes per ciphertext, this adds a globally fixed overhead of exactly **8** KB. Because the polynomial coefficients are derived via a PRF from the Master Key Chain, the spatial overhead added to the individual polynomials is strictly **zero**, preserving maximum bandwidth efficiency.

10 Application to BABE: An Engineering Design Example

To bridge the gap between theoretical cryptography and systems engineering, this section provides an end-to-end design example intended for protocol developers. We demonstrate how the WOTS-to-Lamport translation layer natively integrates into the BABE protocol [GKS+26] to completely bypass Bitcoin's strict standardness limits without altering the underlying Garbled Circuit logic. We build the architecture from the ground up, starting with the public key footprint, moving to the exact transaction sizes, and concluding with the off-chain software architecture.

10.1 The On-Chain Bottleneck: Public Key Commitments

At the core of BABE, a Claimer (Prover) commits to a cryptographic proof on the Bitcoin blockchain. A Challenger (Verifier) then provides Garbled Circuit (GC) input labels that strictly correspond to the Claimer's committed proof. To enforce this cryptographically, the Bitcoin UTXO locking script must explicitly contain the Public Key (a series of hash commitments) for the signature scheme.

Remark 2 (Two distinct WOTS instances)**.** *The protocol uses* two *separate WOTS instances, and keeping them distinct is essential to the security argument:*

- *the* Claimer's *`Assert`-WOTS, whose signature commits the proof on-chain; the Claimer holds these seeds; and*
- *the* Challenger's *`ChallengeAssert`-WOTS, whose chain values key the translation-gadget masks (the "WOTS for the labels"); the Challenger holds these seeds.*

Neither party ever reveals the seeds of its own WOTS instance to the other. In particular, the chains consumed by the translation gadget are generated and held entirely by the Challenger, who reveals only a single signature (at the state matching the Claimer's committed proof). The Claimer acts solely as the evaluator*, receiving those revealed signature states and never the underlying seeds or earlier chain values. This separation is exactly what the gadget's soundness relies on: an evaluator in possession of the chain seeds could hash every chain forward, unmask all evaluation points, interpolate both label polynomials, and recover both mutually exclusive Lamport labels (extracting the Free-XOR offset). The evaluator must therefore never hold the signing material of the WOTS instance it is translating.*

We compare the on-chain overhead of committing to a 508-bit proof using standard 20-byte `OP_HASH160` digests:

- **Lamport Signatures (The GC Standard):** Yao's Garbled Circuits natively require binary inputs. To sign a 1-bit value using Lamport, the locking script must contain two hashes (one for `0`, one for `1`). For a 508-bit proof, the public key requires $508 \times 2 = 1,016$ hashes, and the signature reveals one preimage per bit.

$$\text{Lamport PK Size:} \quad 1,016 \times 20 \text{ bytes} = \mathbf{20,320} \text{ bytes } (\approx 20.3 \text{ KB})$$

$$\text{Lamport Sig Size:} \quad 508 \times 20 \text{ bytes} = \mathbf{10,160} \text{ bytes } (\approx 10.2 \text{ KB})$$

- **Winternitz One-Time Signatures (The Bitcoin Standard):** WOTS compresses data by grouping multiple bits into a single digit. Using a parameter $d = 4$ (4 bits per digit), a digit represents 16 possible states using only a *single* hash chain. The public key is simply the terminal hash of each chain, and the signature reveals one intermediate chain value per chain. As calculated below, our chunked WOTS design requires only 136 total chains.

$$\text{WOTS PK Size:} \quad 136 \times 20 \text{ bytes} = \mathbf{2,720} \text{ bytes } (\approx 2.7 \text{ KB})$$

$$\text{WOTS Sig Size:} \quad 136 \times 20 \text{ bytes} = \mathbf{2,720} \text{ bytes } (\approx 2.7 \text{ KB})$$

Taken in isolation, these are large gains: the public key shrinks by a factor of 7.5 and the signature by a factor of 3.7, so the committed data as a whole falls from $30,480$ to $5,440$ bytes, a $5.6\times$ reduction. The script, however, contains more than the committed data—it must also carry the opcodes implementing the verification algorithm, and here WOTS is the more expensive of the two. Verifying a Lamport signature costs one hash and one comparison per bit, whereas verifying a WOTS signature requires hashing each chain forward to its terminal value (up to $K = 15$ iterations per chain) and recomputing the checksum, so the WOTS verifier is substantially longer. This opcode overhead partially offsets the savings above.

The Engineering Solution: Garbled Circuits cannot natively ingest monotonic WOTS chains. By placing the highly compressed WOTS public key on the blockchain, and executing

our translation gadget *off-chain* to convert the WOTS signatures back into binary Lamport keys, we achieve an $> 3\times$ reduction in total on-chain script size—the net effect of a $5.6\times$ smaller committed payload against a larger verification script—without modifying the underlying Garbled Circuit logic. The empirical script sizes of the resulting transactions are reported in Table 1.

10.2 System Parameters and Proof Chunking

To integrate this safely within Bitcoin's standardness limits, the protocol parameters are chunked as follows:

1. **Cut-and-Choose** $(181, 7)$**:** To ensure malicious security, the Challenger generates 181 Garbled Circuits. 174 are audited for honesty during setup, and the remaining **7 circuits** are actively evaluated on-chain during a dispute.

2. **Proof Size** (π_1)**:** The Claimer's proof is a point on the BN254 elliptic curve, consisting of an x-coordinate (254 bits) and a y-coordinate (254 bits), totaling **508 bits**.

3. **Message Chunking:** To modularize the data and fit within standardness limits, the 508-bit proof is divided into four **127-bit chunks**: π_{xa}, π_{xb} (for the x-coordinate) and π_{ya}, π_{yb} (for the y-coordinate).

4. **WOTS Encoding:** The Challenger encodes each 127-bit chunk of the Claimer's committed proof using WOTS with $d = 4$.

   - 127 bits equates to **thirty-two 4-bit data digits**.
   - The maximum checksum weight for these 32 digits is $(32 \times 15) = 480$.
   - Representing the maximum weight of 480 requires a **9-bit checksum**. As detailed in Section 9, this overhead is split into a **4-bit major digit** and a **5-bit minor digit** to optimize hashing depth.

5. **Total Chains:** Each 127-bit chunk is perfectly encoded using exactly **34 WOTS chains** (32 message chains + 2 checksum chains). Across 4 chunks, this totals $34 \times 4 =$ **136** hash chains.

10.3 The Transaction Graph Flow and Empirical Sizes

During a dispute, the Challenger must reveal the translation evaluation points for all 7 active Garbled Circuits simultaneously. Because this payload is dense, the challenge phase is explicitly split into parallel X and Y branches.

Figure 4 maps the exact UTXO flow of the protocol during a dispute. Because the Challenger must reveal the translation labels for all 7 Garbled Circuits simultaneously, the challenge phase is explicitly split into parallel branches to keep the transaction sizes well under 10 KvB.

Table 1 details the exact empirical script and virtual byte (vB) sizes of the resulting dispute transactions.

The UTXO flow executes as follows:

1. `PegIn` & `ClaimCl1`**:** The protocol is initialized. The Claimer locks their collateral and commits to the overarching proof parameters (including the highly compressed 2.7 KB WOTS Public Key).

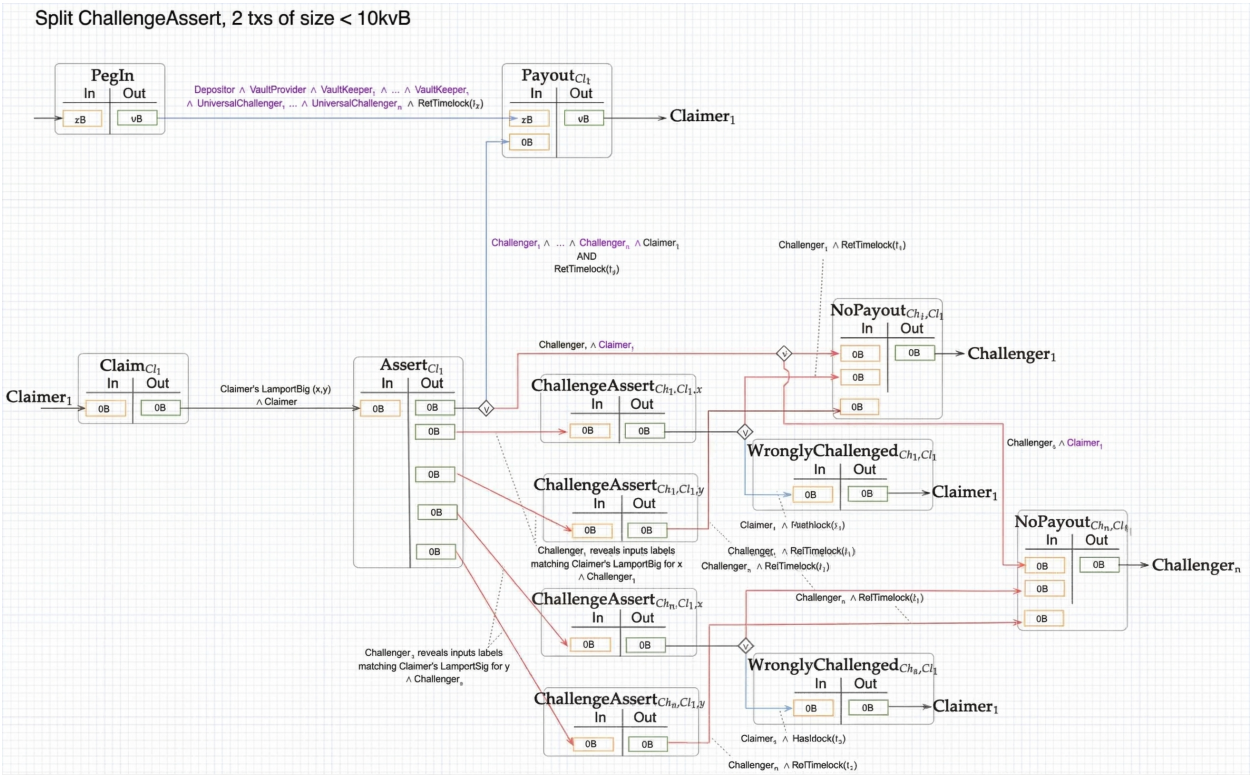

Figure 4: **Split ChallengeAssert Transaction Graph.** The Claimer initiates with the `AssertCl1` transaction. If no challenge occurs, then the Claimer simply posts `PayoutCl1`. On the other hand, the Challenger can challenge as follows: the Challenger's heavily-laden response—which carries the compressed WOTS signatures (encoding the Claimer's committed proof) that the translation gadget consumes—is split into parallel x and y transactions to adhere to standardness limits. Holding only these revealed signatures and never the WOTS seeds, the Claimer translates the labels and decrypts the GC secrets off-chain to post `WronglyChallenged` and claim the payout.

Transaction	Script (Bytes)	Vsize (vB)	Weight (WU)
`Assert`	10,970	3,748	14,990
`ChallengeAssertX`	27,194	9,869	39,474
`ChallengeAssertY`	27,194	9,869	39,474

Table 1: **Empirical On-Chain Transaction Sizes.** By splitting the Challenger's heavily-laden response into X and Y halves, the virtual sizes safely hit 9,869 vB, cleanly satisfying Bitcoin's strict 10,000 vB standardness limit. A combined transaction would have breached 19 KvB and been rejected by the network.

2. `Assert:` The Claimer posts their 508-bit proof π_1. Thanks to WOTS compression, they only need to publish the 136 WOTS hash preimages. As shown in Table 1, this evaluates to a highly efficient 3,748 vB transaction.

3. `ChallengeAssertX` & `ChallengeAssertY:` To dispute the proof, the Challenger must reveal the encrypted WOTS for the labels.

   - `ChallengeAssertX:` Contains a script replay of the Claimer's exact WOTS signatures for the x-coordinate chunks (π_{xa} and π_{xb}). Alongside this, the Challenger reveals the corresponding **WOTS for the labels** (the Lamport labels encrypted under the respective WOTS hashes) for the 7 Garbled Circuits on the exact same message. Bitcoin covenants strictly enforce that the Challenger's replay perfectly matches the Claimer's committed state.
   - `ChallengeAssertY:` Contains the exact same replay and label-reveal logic for the y-coordinate chunks (π_{ya} and π_{yb}).

Figure 5 illustrates the exact data flow from the blockchain into a Garbled Circuit instance.

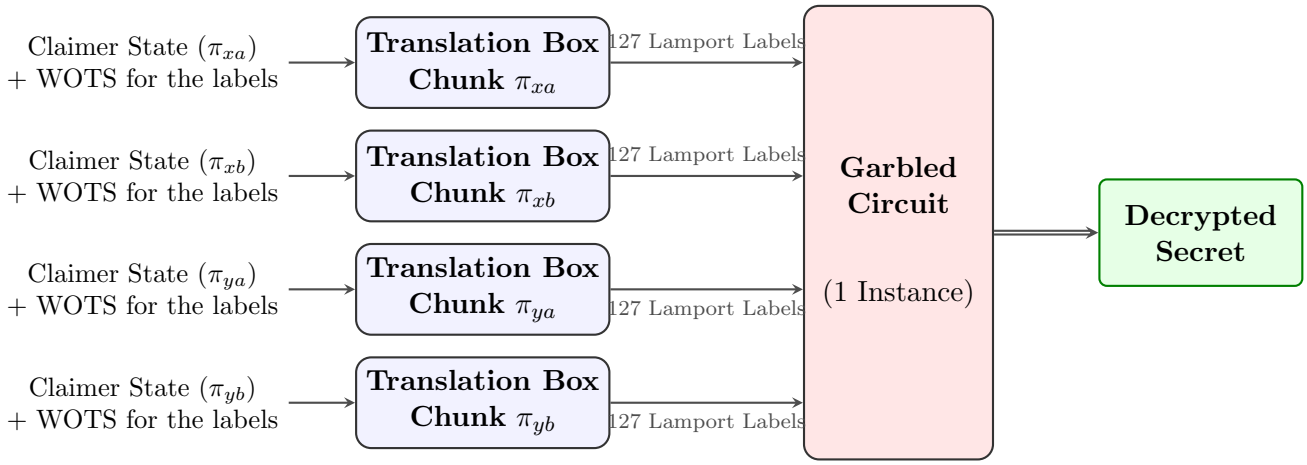

Figure 5: **Off-Chain Translation and Evaluation Architecture.** The translation layer is modularized into four independent black boxes. Each box natively ingests the Claimer's *plaintext* proof chunk (which fixes the active side of each label) alongside the Challenger's "WOTS for the labels"—the masked evaluation points together with the Challenger's revealed `ChallengeAssert`-WOTS signature—pulled from the blockchain. The boxes mathematically interpolate the points to non-interactively output the standard binary Lamport labels. *(Note: While 7 Garbled Circuits are evaluated in parallel during a dispute, this diagram traces the resolution of a single GC instance for clarity).*

The software flow operates in four deterministic stages:

1. **Data Extraction:** The Claimer parses the Bitcoin transactions and extracts the Challenger's "WOTS for the labels" for the active Garbled Circuits from the `ChallengeAssertX` and `ChallengeAssertY` transaction witnesses.

2. **The Translation Layer (The Black Box):** The Claimer feeds their known, plaintext proof chunks (which fix the active side of each label) together with the extracted WOTS for the labels—the Challenger's revealed `ChallengeAssert`-WOTS signature and the masked

evaluation points—into the four translation boxes. Note that the Claimer's own `Assert`-WOTS seeds are *not* an input here; they were consumed on-chain to commit the proof and play no role in translation. From an engineering perspective, this acts as a pure functional API. The gadget mathematically resolves the Shamir polynomials (utilizing the split checksum to dynamically hit the point threshold) and non-interactively outputs exactly 127 standard binary Yao Lamport labels per chunk.

3. **GC Evaluation:** The Claimer concatenates the Lamport labels from all 4 chunks and feeds the resulting 508 labels into the Garbled Circuit. Because the Claimer acted honestly in their `Assert` transaction, the translated labels correctly correspond to the active circuit wires, and the circuit successfully evaluates to output the decryption secret.

4. **Resolution (`WronglyChallenged`):** Armed with the decrypted secrets from the evaluated GCs, the Claimer signs and broadcasts the `WronglyChallenged` transaction. This satisfies the final script hashlocks, slashing the Challenger's collateral and resolving the dispute in the Claimer's favor.

If the Claimer lied in their `Assert` transaction, the point deficit in the translation layer mathematically prevents them from interpolating the correct Lamport labels. The Garbled Circuits evaluate to garbage, the Claimer cannot extract the secrets, and the Challenger eventually broadcasts `NoPayout` after a relative timelock expires, successfully penalizing the malicious Claimer.

10.4 Preliminary Runtime Results

The gadget buys on-chain space with off-chain work. Figure 6 sets both sides of that exchange against each other, comparing the WOTS pipeline with a Lamport baseline committing the same 508-bit proof, under the deployed $(181, 7)$ cut-and-choose parameters of Section 10.2. There, `Assert` carries four WOTS signatures covering the proof — two for the x-coordinate, whose 254 bits are split into two 127-bit chunks, and two for the y-coordinate. Counting the four signatures that jointly cover the proof as a single *logical* signature, `Assert` reveals one logical signature and each `ChallengeAssert` reveals $1 + 7 = 8$: one replayed from `Assert`, and one for each of the seven circuits evaluated during a dispute.

The on-chain figures follow from the transaction sizes reported in Table 1: 3,748 vB for `Assert` and 9,869 vB for each of `ChallengeAssertX` and `ChallengeAssertY`. Against these, a Lamport implementation needs 9,240 vB for `Assert` and 66,360 vB for a single unsplit `ChallengeAssert`. At the fee rates of 2.23 sat/vB for `Assert`, 2.22 sat/vB for `ChallengeAssert`, and 2.02 sat/vB for `WronglyChallenged`, and a price of 1 BTC = \$95,500, the two pipelines cost as shown in Table 2.

Reading Figure 6 across, on-chain cost falls 3.21×, from \$160.66 to \$50.12, comfortably exceeding the $> 3\times$ script-size reduction derived in Section 10.1. The remaining three columns move the other way: setup, the Garbler's pass over all 181 circuits, rises 3.01× (2.52 → 7.60 s); verification, the Evaluator's pass over the seven evaluated circuits, rises 5.84× (108.9 → 636.3 ms); and storage rises 2.15× (45.57 → 97.93 MiB). Verification is the largest of the three penalties, and its origin is where Section 6 predicts: much of the Winternitz verification time is spent on polynomial interpolation, where the corresponding step under Lamport is a single hash per bit. That cost is intrinsic to the construction — the same point deficit that hides an unearned label is what the Evaluator must interpolate around to obtain an earned one. In absolute terms all three penalties stay modest: setup completes in under eight seconds, verification in under a second, and the storage is under a hundred mebibytes.

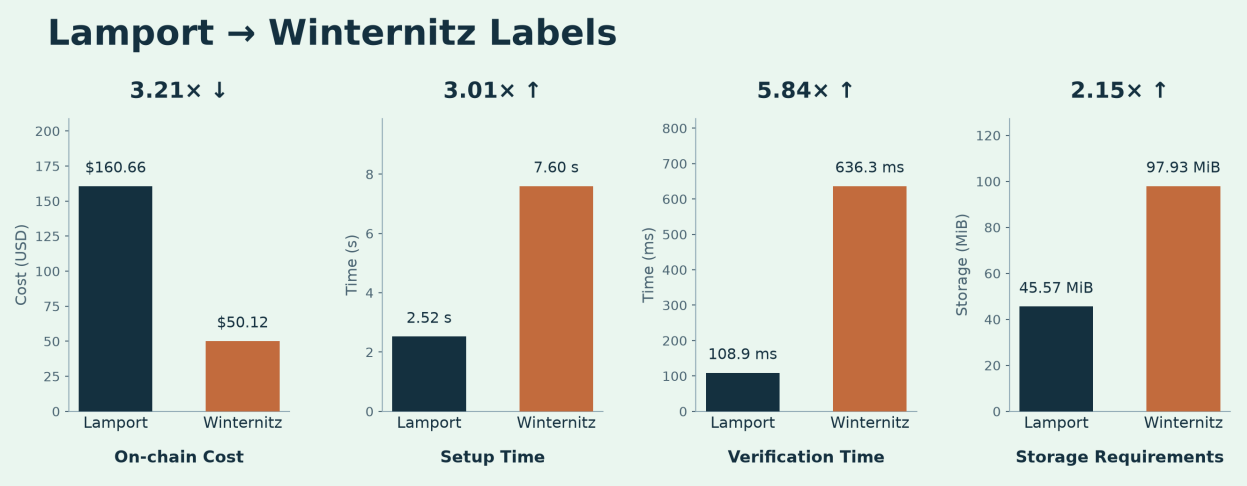

Figure 6: **Cost of the WOTS-to-Lamport translation layer.** The WOTS pipeline against a Lamport baseline committing the same 508-bit proof, under the deployed $(181, 7)$ cut-and-choose parameters. On-chain cost falls 3.21×; setup time, verification time, and storage rise by 3.01×, 5.84×, and 2.15× respectively, the verification figure being dominated by polynomial interpolation. The trade is favourable because the increases are off-chain seconds and mebibytes, while the reduction is on-chain space subject to Bitcoin's standardness limit.

	Size (vB)	**Fee (sat)**	**Cost (USD)**
Lamport			
`Assert`	9,240	20,605	19.68
`ChallengeAssert`	66,360	147,319	140.69
`WronglyChallenged`	149	301	0.29
Total			**160.66**
Winternitz			
`Assert`	3,748	8,358	7.98
`ChallengeAssertX`	9,869	21,909	20.92
`ChallengeAssertY`	9,869	21,909	20.92
`WronglyChallenged`	149	301	0.29
Total			**50.12**

Table 2: **Dispute cost under the deployed** $(181, 7)$ **parameters.** Sizes, fees, and USD cost for each transaction of a dispute, under a Lamport baseline and under the WOTS pipeline.

The more consequential difference, however, is not the total but who must fund it, and when. `ChallengeAssert` lies on the unhappy path: in the common case no dispute occurs and the transaction is never broadcast. Keeping it under 10,000 vB allows it to be posted as a zero-fee version-3 (TRUC) transaction whose fee is supplied after the fact by a child-pays-for-parent transaction, so the Challenger pays only in the event that it actually challenges: the topology restrictions of BIP 431 [Zha24] cap a TRUC parent at 10,000 vB and its fee-bearing child at 1,000 vB. At 9,869 vB the WOTS challenge transactions qualify. Lamport's, at 66,360 vB, does not: its fee must instead

be pre-funded from an output of `Assert`, which means the Claimer locks up capital in advance for a path that is usually never taken. The burden compounds, since in practice a Claimer may face up to ten Challengers and must provision for each — roughly ten times the $140.69 challenge fee, on the order of $1,400, committed up front against a dispute that in the happy path never happens. By bringing the challenge transaction under the limit, the gadget reduces that up-front commitment to zero and defers the fee to the Challenger who chooses to incur it.

11 Conclusion

In this work, we formalized a non-interactive cryptographic translation layer that securely maps monotonic Winternitz One-Time Signatures to orthogonal binary Lamport signatures. By leveraging a dynamic Shamir Secret Sharing threshold governed by a global checksum, the gadget acts as a firewall against forward-hashing attacks: an evaluator that hashes forward to an unearned state faces a point deficit at which the sharing layer hides the target label perfectly, and closing that deficit is exactly as hard as breaking the underlying hash chains. When deployed as a preprocessing layer for Garbled Circuits in resource-constrained environments like Bitcoin, our framework provides the spatial compression of WOTS with the computational compatibility of Lamport, enabling the practical deployment of expressive, zero-knowledge verification protocols.

Acknowledgements

Robin Linus Woll would like to thank Ekrem Bal of Chainway Labs and Liam Eagen of Ideal Group for some early discussions.

References

[GKS+26] Sanjam Garg, Dimitris Kolonelos, Mikhail Sergeevitch, Srivatsan Sridhar, and David Tse. BABE: Verifying proofs on bitcoin made 1000x cheaper. Cryptology ePrint Archive, Report 2026/065, 2026. `https://eprint.iacr.org/2026/065`.

[LAA+25] Robin Linus, Lukas Aumayr, Zeta Avarikioti, Matteo Maffei, Andrea Pelosi, Orfeas Thyfronitis Litos, Christos Stefo, David Tse, and Alexei Zamyatin. Bridging bitcoin to second layers via BitVM2. Cryptology ePrint Archive, Report 2025/1158, 2025. To appear in USENIX Security 2026. `https://eprint.iacr.org/2025/1158`.

[Lam79] Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, 1979.

[Lin23] Robin Linus. BitVM: Compute anything on bitcoin. Whitepaper, 2023. `https://bitvm.org/bitvm.pdf`.

[WAA+26] Robin Linus Woll, Ioannis Alexopoulos, Lukas Aumayr, Zeta Avarikioti, Matteo Maffei, and David Tse. BitVM3: Efficient bitcoin bridges via garbled circuits. Cryptology ePrint Archive, Report 2026/933, 2026. `https://eprint.iacr.org/2026/933`.

[Zha24] Gloria Zhao. BIP 431: Topology restrictions for pinning. Bitcoin Improvement Proposal 431, 2024. `https://github.com/bitcoin/bips/blob/master/bip-0431.mediawiki`.