The Semantic Holder (SH): Algebraic Extraction for Legal Opposability

MINKA MI NGUIDJOI Thierry Emmanuel[a,*]

[a]*Laboratory of Mathematical Engineering and Information Systems (LIMSI), National Advanced School of Engineering, University of Yaoundé I, Cameroon*

Abstract

This manuscript introduces Semantic Holder (SH), the opposability primitive within the Chaotic Affine Secure Hash (CASH) toolkit, completing the framework's implementation of the Q2CSI philosophy. SH enables legally opposable interpretations through algebraic extraction from polynomial iteration traces, working in concert with CEE (confidentiality) and AOW (reliability). Building upon the Affine Iterated Inversion Problem (AIIP) foundation, SH provides mathematically verifiable legal interpretations with guaranteed minimum opposability bounds. We establish that SH maintains an opposability score $\Omega \geq 0.60$ through rigorous entropy preservation, institutional explainability, and legal contestability guarantees. The primitive features efficient STARK-proof verifiable computation, cross-jurisdictional compatibility, and quantum resistance through its reduction to AIIP hardness. We demonstrate practical applications in legal smart contracts, regulatory compliance auditing, and digital evidence authentication, providing concrete parameter recommendations for standard security levels. SH represents a significant advancement in cryptographic systems that must operate within legal constraints, enabling transparent and verifiable legal opposability without compromising security or performance.

Keywords: CASH framework, SH primitive, legal opposability, algebraic extraction, post-quantum cryptography, verifiable computation, regulatory compliance, semantic hold, institutional explainability.

1. Introduction

This work presents a theoretical cryptographic primitive, the Semantic Holder **SH**, which for the first time formalizes the concept of verifiable legal opposability. We introduce a mathematical framework that defines the opposability property Ω through its core components, institutional explainability (E), legal contestability (Γ), and cryptographic verifiability, and prove a minimum guaranteed bound $\Omega \geq 0.60$ for any interpretation generated by our construction. This is the result of reducing the problem to the hardness of the Affine Iterated Inversion Problem **AIIP** [27] and leveraging algebraic extraction from polynomial iteration traces. As a theoretical contribution, this paper focuses on the definitions, security reductions, and formal guarantees of the SH primitive, establishing a foundational framework for cryptographically-enforceable legal semantics without claiming an optimized implementation or empirical benchmarks.

*Corresponding author

Email address: minkathierry@gmail.com (MINKA MI NGUIDJOI Thierry Emmanuel)

1.1. Semantic Holder Contribution (Opposability properties)

The **Semantic Holder (SH)** primitive completes the CASH framework by providing the crucial opposability component, addressing the fundamental challenge of creating cryptographically secure yet legally meaningful outputs. SH's core contributions include:

- **Minimum Opposability Guarantee**: Formal mathematical proof that all SH interpretations maintain an opposability score $\Omega \geq 0.60$, ensuring substantive legal validity.
- **Institutional Explainability**: Generation of institution-specific explanations that satisfy diverse legal frameworks and interpretation rules while maintaining consistency.
- **Legal Contestability**: Built-in mechanisms for dispute resolution and alternative interpretation pathways that preserve the minimum opposability bound.
- **Verifiable Computation**: Integration with STARK proof systems to provide transparent verification of both computational correctness and legal interpretability.

These properties establish SH as the first cryptographic primitive to provide mathematically verifiable legal opposability with guaranteed minimum bounds.

1.2. Technical Overview (SH function formulation)

The SH primitive operates through algebraic extraction from polynomial iteration traces, extending the AIIP foundation. Formally, for a polynomial $f(x) = x^2 + \alpha \in \mathbb{F}_q[x]$ where α is a quadratic non-residue, the SH function computes:

$$\mathsf{SH}_{f,n}(x) = \phi(f^{(n)}(x)) \tag{1}$$

where:

- $f^{(n)}$ denotes n-fold composition, with $\deg(f^{(n)}) = 2^n$.
- $\phi : \mathbb{F}_q \to \mathcal{V}_{\mathcal{J}}$ maps field elements to verdict spaces.
- The computation preserves entropy: $H_\infty(\mathsf{SH}_{f,n}(X)) \geq \log_2 q - n \log_2 2$.

This formulation enables efficient computation while providing the mathematical foundations for opposability guarantees through connections to hard problems in multivariate algebra and number theory.

1.3. The CASH Framework: Composable Primitives for Contextual Security

The **Chaotic Affine Secure Hash (CASH)** framework constitutes a primitive-level realization of the Quantum Composable and Contextual Security Infrastructure (**Q2CSI**) architecture [31], designed to resolve the fundamental tensions within the Confidentiality, Reliability, Opposability (**CRO**) trilemma [30]. Formally defined as the composable triplet:

$$\text{CASH} = (\text{CEE}, \text{AOW}, \text{SH}) \tag{2}$$

this framework provides a modular approach to cryptographic security where each primitive specializes in implementing exactly one property of the CRO trilemma while maintaining formal isolation from the others. The Semantic Holder (**SH**) primitive completes this triad by enabling legally opposable semantic interpretations of cryptographic operations. The framework's architecture ensures three critical properties:

- **Primitive Specialization**: Each component (CEE[28], AOW[29], SH (this work)) implements exactly one CRO property with minimal functional overlap.
- **Universal Composability**: Primitives maintain security guarantees when composed within the Q2CSI infrastructure.
- **Contextual Adaptation**: The framework supports dynamic reconfiguration based on juridical and operational requirements.

With the introduction of SH, the CASH framework achieves complete coverage of the CRO trilemma, enabling end-to-end verifiable security that bridges technical cryptographic guarantees and legal enforceability. This final primitive transforms raw computational outputs into semantically rich, legally interpretable representations that maintain mathematical verifiability while supporting judicial reasoning.

Remark 1.1 (Framework Completeness). *The Semantic Holder primitive completes the CASH framework, following the Chaotic Entropic Expansion (CEE) for confidentiality and Affine One-Wayness (AOW) for reliability. This tripartite structure now fully instantiates the Q2CSI infrastructure, providing a comprehensive solution to the CRO trilemma through specialized primitive design and controlled compositional properties.*

1.4. Paper Organization

The remainder of this paper is structured as follows: Section 2 establishes the mathematical and legal-theoretic foundations of SH. Section 3 details the SH primitive construction and algorithms. Section 4 analyzes the opposability properties and guarantees. Section 5 presents the security analysis and reductions. Section 6 discusses applications and use cases. Section 8 covers implementation considerations. Section 7 analyzes performance characteristics and optimization techniques. Section 9 provides a comparative discussion and positions this work within related research. Section 10 concludes with future research directions. Additional technical details are provided in the appendices.

2. Preliminaries

2.1. Mathematical Foundations from AIIP

This subsection establishes the core mathematical results from the Affine Iterated Inversion Problem (AIIP) that underpin the Semantic Holder primitive's security and functionality.

Definition 2.1 (AIIP Iteration Structure). *Let $f(x) = x^2+\alpha \in \mathbb{F}_q[x]$ where α is a quadratic non-residue in $\mathbb{F}_q$. The n-fold composition is defined recursively:*

$$f^{(0)}(x) = x, \quad f^{(n)}(x) = f(f^{(n-1)}(x)) \text{ for } n \geq 1 \tag{3}$$

The degree grows exponentially: $\deg(f^{(n)}) = 2^n$.

Remark 2.1 (Special Case of Degree Growth). *The polynomial $f(x) = x^2+\alpha$ specified in Definition 2.1 is a specific instance of the general class $\mathcal{F}_d$ with $d = 2$. Therefore, Theorem 2.1 applies, confirming* $\deg(f^{(n)}) = 2^n$.

Theorem 2.1 (AIIP Degree Growth). *For any polynomial $f \in \mathbb{F}_q[x]$ of degree $d \geq 2$, the n-fold composition satisfies:*

$$\deg(f^{(n)}) = d^n \tag{4}$$

This exponential growth creates a combinatorial barrier to algebraic attacks and ensures rapid entropy expansion.

Theorem 2.2 (AIIP-HCDLP Connection). *For $f(x) = x^2 + \alpha$ with α a quadratic nonresidue, solving the AIIP instance (f, n, y) is mathematically equivalent to solving the discrete logarithm problem in the Jacobian of the hyperelliptic curve:*

$$C_{n,y} : v^2 = f^{(n)}(u) - y \tag{5}$$

which has genus $g_n = 2^{n-1} - 1$.

Theorem 2.3 (AIIP-MQ Reduction). *The Affine Iterated Inversion Problem is polynomial-time many-one reducible to solving systems of multivariate quadratic equations (MQ). Specifically, there exists a reduction algorithm $\mathcal{R}$ that transforms an AIIP instance (f, n, y) into an MQ system $\mathcal{S}$ with $O(n)$ variables and equations.*

2.2. Legal-Theoretic Foundations

This subsection provides formal definitions of the core legal concepts underpinning the Semantic Holder primitive, establishing the mathematical framework for opposability, explainability, and contestability.

Definition 2.2 (Legal Opposability). *Let $\mathcal{L}$ be a legal interpretation space and $\mathcal{J}$ a jurisdiction space. For a given cryptographic output $y \in \mathbb{F}_q$ generated from input $x \in \mathbb{F}_q$ through transformation $T : \mathbb{F}_q \to \mathbb{F}_q$, the* legal opposability $\Omega(y, \mathcal{J})$ *is defined as:*

$$\Omega(y, \mathcal{J}) = \min_{I \in \mathsf{Institutions}(\mathcal{J})} [\min\{\text{Explainability}(y, I), \text{Contestability}(y, I), \text{Verifiability}(y)\}] \tag{6}$$

where the inner minimum ensures that the weakest component determines the institution-specific opposability, and the outer minimum ensures the most restrictive institution determines the overall opposability. Mathematical Justification*: This formulation guarantees that if all components satisfy the threshold $\theta \geq 0.60$, then $\Omega(y, \mathcal{J}) \geq 0.60$, avoiding the multiplicative degradation that would yield $0.60^3 = 0.216$.*

Definition 2.3 (Institutional Explainability). *For a legal interpretation $L \in \mathcal{L}$ and institution $I \in$ Institutions$(\mathcal{J})$, the* institutional explainability $\mathrm{E}(L, I)$ *is given by:*

$$\mathrm{E}(L, I) = 1 - \frac{H(L|I)}{H(L)} \tag{7}$$

where $H(L)$ is the Shannon entropy of interpretation L and $H(L|I)$ is the conditional entropy given institution I's interpretation rules. Mathematical Implication*: The requirement $\mathrm{E}(L, I) \geq 0.60$ is equivalent to the condition that the institution's rules reduce the uncertainty about the interpretation by at least 60% of the original entropy:*

$$H(L) - H(L|I) \geq 0.60 \cdot H(L) \tag{8}$$

This ensures the explanation is substantively meaningful, as the mutual information $I(L; I)$ constitutes a majority of the interpretation's entropy.

Definition 2.4 (Legal Contestability). *The* contestability $\Gamma(L, \mathcal{J})$ *of a legal interpretation L in jurisdiction $\mathcal{J}$ is defined as:*

$$\Gamma(L, \mathcal{J}) = \prod_{i=1}^{k} \left(1 - \prod_{j=1}^{m} (1 - p_{ij})\right) \tag{9}$$

where $p_{ij} \in [0,1]$ represents the probability that contestation pathway j for institution i is available and functional. Mathematical Implication*: To achieve $\Gamma(L, \mathcal{J}) \geq 0.60$, the system must ensure that for each institution, at least one contestation pathway exists with high probability, satisfying:*

$$\max_j p_{ij} \geq 1 - (1 - 0.60)^{1/k} \quad \forall i \tag{10}$$

The product structure $\prod_{j=1}^{m}(1 - p_{ij})$ represents the probability that no *pathway for institution i is available. Thus, $1 - \prod_{j=1}^{m}(1 - p_{ij})$ is the probability that at least one pathway is available, and the overall contestability is the probability that this holds for every institution.*

Axiom 2.1 (Minimum Opposability Bound). *The CASH framework requires that all valid legal interpretations satisfy:*

$$\Omega(y, \mathcal{J}) \geq \theta_{\min} = 0.60 \tag{11}$$

This bound emerges from legal requirements for substantive due process and ensures interpretations maintain practical utility in legal proceedings. Mathematical Implication*: The minimum bound implies that for any legal interpretation L generated by* SH_Extract*, the probability of achieving opposability is overwhelming:*

$$\Pr\left[L \in \mathcal{L}_{\text{valid}} \mid L \leftarrow \mathsf{SH_Extract}(\cdot)\right] \geq 1 - negl(\lambda) \tag{12}$$

Definition 2.5 (Verdict Space). *For jurisdiction $\mathcal{J}$ with legal framework F, the* verdict space $\mathcal{V}_{\mathcal{J}}$ *is a finite set of possible legal outcomes constructed as:*

$$\mathcal{V}_{\mathcal{J}} = \bigcup_{i=1}^{n} V_i \quad \textit{where} \quad V_i = \{v \in \mathcal{L} \mid \text{Appl}(v, F) \geq \tau\} \tag{13}$$

with applicability function $\text{Appl} : \mathcal{L} \times \mathcal{F} \to [0,1]$ *and threshold* $\tau = 0.60$. Mathematical Implication*: The mapping $\phi : \mathbb{F}_q \to \mathcal{V}_{\mathcal{J}}$ must preserve entropy:*

$$H(\phi(X)) \geq H(X) - \log_2\left(\frac{1}{1 - \text{negl}(\lambda)}\right) \tag{14}$$

for random variable $X \in \mathbb{F}_q$, ensuring no meaningful entropy loss during legal interpretation.

Definition 2.6 (Rule Application Function). *For institution I with rule set $\mathcal{R}_I$, the* rule application function $\text{Apply}_I : \mathcal{L} \to \mathcal{L}$ *is defined recursively:*

$$\text{Apply}_I(L) = R_1(R_2(\cdots R_k(L))) \quad \textit{for} \quad R_i \in \mathcal{R}_I \tag{15}$$

Mathematical Implication*: To ensure rule application converges to institutionally acceptable interpretations, the function is designed to be contractive in the semantic space. That is, for some metric d and contraction factor $\alpha < 1$:*

$$d(\text{Apply}_I(L), \text{Apply}_I(L')) \leq \alpha \cdot d(L, L') \tag{16}$$

This property is assumed to hold for all institutional rule sets $\mathcal{R}_I$ used in the framework.

These formal definitions establish the mathematical foundation for legal opposability within the CASH framework, providing the rigorous basis needed for the security proofs and parameter recommendations in subsequent sections.

Assumption 2.1 (Institutional Independence). *The contestation pathways across different institutions are computationally independent. That is, for any two distinct institutions I_i and I_j in $\mathcal{J}$, the availability of pathways $p_{i\cdot}$ and $p_{j\cdot}$ are independent random variables. Formally, for any pathways k and l:*

$$\Pr[p_{ik} \wedge p_{jl}] = \Pr[p_{ik}] \cdot \Pr[p_{jl}] \tag{17}$$

This assumption ensures that the contestability score $\Gamma(L, \mathcal{J})$ factors into a product over institutions.

Assumption 2.2 (Independence of Institutional Opposability). *For any legal interpretation L and jurisdiction $\mathcal{J}$, the events that each institution $I \in \mathcal{J}$ achieves explainability $\mathrm{E}(L, I) \geq 0.60$, contestability $\Gamma(L, I) \geq 0.60$, and verifiability are independent across institutions. This independence holds due to the design of institutional rule sets and contestation pathways, which are tailored to each institution without shared dependencies.*

Remark 2.2 (Practicality of Independence). *This independence assumption simplifies the analysis but may not hold in practice if institutions share common vulnerabilities or biases. Future work could explore more complex models with correlated pathways, which would require adjusted opposability bounds.*

Assumption 2.3 (Injectivity of $\phi onAIIPRange$). *The semantic mapping ϕ is injective on the range of $f^{(n)}$. This is achieved by setting the verdict space size $|\mathcal{V}_{\mathcal{J}}|$ to be at least the size of the range of $f^{(n)}$, which is approximately q for large n. Although this may not be practical for legal applications, it is a theoretical requirement for the security reduction.*

Assumption 2.4 (Bounded Preimage Size). *For the semantic mapping $\phi : \mathbb{F}_q \to \mathcal{V}_{\mathcal{J}}$ and any verdict $v \in \mathcal{V}_{\mathcal{J}}$, the preimage set $\phi^{-1}(v)$ has size at most $M = \lceil q/|\mathcal{V}_{\mathcal{J}}| \rceil$. Furthermore, M is polynomial in the security parameter λ, i.e., $M = (\lambda)$.*

2.2.1. Cryptographic Instantiation of Legal Properties

The legal-theoretic framework defined above is not merely an abstract model but is designed for direct cryptographic instantiation within the SH primitive. This requires mapping the mathematical definitions of opposability (Ω), explainability (E), and contestability (Γ) onto concrete cryptographic properties and algorithms. The following exposition makes this linkage explicit, ensuring the legal guarantees are enforced by cryptographic hardness.

Definition 2.7 (Cryptographic Opposability Implementation). *For a cryptographic primitive* SH *to implement legal opposability, it must algorithmically enforce the minimum bound $\Omega \geq 0.60$. This is realized by constructing the algorithms* (Extract, Explain, Verify) *such that for any output interpretation L and any probabilistic polynomial-time adversary $\mathcal{A}$:*

$$\Pr\left[\mathsf{Verify}(L, \pi, R) = 1 \quad \wedge \quad \Omega(L, R) \geq 0.60 \,\middle|\, \begin{array}{l}(L, \pi) \leftarrow \mathsf{SH_E xtract}(x, R, n); \\ R \leftarrow \mathsf{RelationalContext}(\lambda)\end{array}\right] \geq 1 - negl(\lambda) \tag{18}$$

where the probability is over the randomness of the extraction algorithm and the choice of relational context. This definition binds the legal property Ω to the cryptographic verification algorithm.

The cryptographic security of the opposability guarantee reduces to the hardness of the AIIP problem and the soundness of the STARK proof system. Specifically:

- **Explainability (E) as Entropy Preservation:** The requirement $H(L|I) \leq 0.4 \cdot H(L)$ from Definition 2.3 is implemented by the semantic mapping function ϕ (Definition 3.1). Its entropy-preserving property (Proposition 3.1) ensures that the output interpretation L retains sufficient entropy from the input, making it infeasible for an adversary to craft an interpretation that is ambiguous for all institutions simultaneously. The hardness of finding a preimage x for a given L with low explainability reduces to the one-wayness of the AIIP iteration (Theorem 5.1).
- **Contestability (Γ) as Cryptographic Availability:** The contestability guarantee $\Gamma(L, \mathcal{J}) \geq 0.60$ from Definition 2.4 is implemented by ensuring the Verify algorithm checks the availability of contestation pathways. This is modelled as a reliable broadcast mechanism within the jurisdiction $\mathcal{J}$. The probability p_{ij} of a pathway being available is implemented via a cryptographic availability oracle $\mathcal{O}_{\text{avail}}$ that is queried during verification. Resistance to institutional compromise (Proposition 5.3) relies on the difficulty of simultaneously compromising a threshold of these oracles, which is enforced by a distributed consensus protocol.
- **Verifiability as Proof System Soundness:** The verifiability component of Ω is implemented by the STARK proof π. The soundness error of the STARK system (typically $\text{negl}(\lambda)$) directly contributes to the $\text{negl}(\lambda)$ term in the opposability probability bound. The algorithmic verification Verify thus cryptographically certifies the legal property.

This cryptographic instantiation provides a mathematically rigorous foundation for the legal opposability property, transforming abstract legal concepts into formally verifiable algorithmic guarantees. The security reductions in Section 5 demonstrate that violating the opposability bound implies either breaking the AIIP problem or the soundness of the STARK proof system.

2.3. Theorems from CEE

2.3.1. Entropy Bounds

As established in CEE[28] we recall here Theorems bounds 2.4 and 2.5

Theorem 2.4 (Min-Entropy Bound). *Let $f \in \mathcal{F}_d$ and $X \sim \mathcal{U}(\mathbb{F}_q)$. Then for any $n \geq 1$:*

$$H_\infty(f^{(n)}(X)) \geq \log_2 q - n \log_2 d \tag{19}$$

2.3.2. Statistical Distance Analysis

Theorem 2.5 (Realistic Statistical Distance Bound). *Let $f \in \mathcal{F}_d$ and $X \sim \mathcal{U}(\mathbb{F}_q)$. The statistical distance from uniform satisfies:*

$$\Delta(f^{(n)}(X), \mathcal{U}(\mathbb{F}_q)) \leq \min\left\{1, \frac{d^n}{\sqrt{q}}\right\} \tag{20}$$

For cryptographic security, we require $d^n \ll \sqrt{q}$.

Remark 2.3 (Primitive Autonomy and Integration). *The SH primitive is designed as an* autonomous cryptographic component *that:*

1. *Operates independently without dependencies on CEE or AOW primitives.*
2. *Provides well-defined interfaces for future integration into the CASH framework.*
3. *Maintains its security and opposability guarantees regardless of the integration context.*

The dialectical isolation between SH, CEE, and AOW primitives is an architectural property *enforced by the CASH framework during integration, not a responsibility of individual primitives. This design ensures modularity and allows independent analysis and verification of each primitive.*

3. SH Primitive Construction

3.0.1. Explicit Construction of the Semantic Mapping ϕ

Definition 3.1 (Semantic Mapping Function). *The semantic mapping function $\phi : \mathbb{F}_q \to \mathcal{V}_{\mathcal{J}}$ is explicitly constructed as:*

$$\phi(z) = \mathsf{Extract} \circ \mathsf{Partition} \circ \mathsf{Normalize}(z) \tag{21}$$

where:

- $\mathsf{Normalize} : \mathbb{F}_q \to [0, q-1]$ *maps field elements to integers*
- $\mathsf{Partition} : [0, q-1] \to \{0, 1, \ldots, |\mathcal{V}_{\mathcal{J}}| - 1\}$ *via:*

$$\mathsf{Partition}(k) = \left\lfloor \frac{k \cdot |\mathcal{V}_{\mathcal{J}}|}{q} \right\rfloor \tag{22}$$

- $\mathsf{Extract} : \{0, 1, \ldots, |\mathcal{V}_{\mathcal{J}}| - 1\} \to \mathcal{V}_{\mathcal{J}}$ *is the indexing function*

Proposition 3.1 (Entropy Preservation of ϕ). *For the mapping ϕ defined above and uniformly random $X \in \mathbb{F}_q$:*

$$H(\phi(X)) \geq \log_2 |\mathcal{V}_{\mathcal{J}}| - \epsilon \tag{23}$$

where $\epsilon = \log_2(1 + |\mathcal{V}_{\mathcal{J}}|/q) \leq negl(\lambda)$ for $q \gg |\mathcal{V}_{\mathcal{J}}|$.

Proofsketch 3.1. *The partition function creates approximately equal preimage sizes, with each verdict having $q/|\mathcal{V}_{\mathcal{J}}| \pm 1$ preimages. This near-uniform distribution ensures maximal entropy transfer with negligible loss. Full analysis in Appendix A.6.*

Lemma 3.1 (Injectivity of $\phi on AIIP Outputs$). *Let $S = \{f^{(n)}(x) \mid x \in \mathbb{F}_q\}$ be the range of the AIIP iteration. If the verdict space $\mathcal{V}_{\mathcal{J}}$ satisfies $|\mathcal{V}_{\mathcal{J}}| \geq |S|$, then the semantic mapping $\phi is injective on S with probability 1$.*
$If |V_{\mathcal{J}}| < |S|$, then the probability that $\phi is injective on S is at most e^{-|S|^2/(2|\mathcal{V}_{\mathcal{J}}|)}$, which is negligible when $|S|/|\mathcal{V}_{\mathcal{J}}|$ is large.

Proofsketch 3.2. *The mapping $\phi is defined via partitioning, so it is injective on S if and only if all elements in S map to distinct verdicts. If |V_{\mathcal{J}}| \geq |S|$, this is always possible by choosing an appropriate partitioning. If $|\mathcal{V}_{\mathcal{J}}| < |S|$, by the pigeonhole principle, collisions are inevitable. The probability that a random function from S to $\mathcal{V}_{\mathcal{J}}$ is injective is given by the birthday bound, which is negligible when $|S|$ is large relative to $|\mathcal{V}_{\mathcal{J}}|$.*

3.1. Algebraic Legal Extraction

Definition 3.2 (SH Primitive). *The Semantic Holder primitive is formally defined as the quadruple:*

$$SH = (Setup, Extract, Explain, Verify) \tag{24}$$

where:

- *$Setup(1^\lambda) \to params$: Initializes SH parameters with security parameter λ.*
- *$Extract(h, R, n) \to (L, \pi)$: Extracts legal interpretation from $f^{(n)}(h)$ with proof π.*
- *$Explain(L, I) \to E$: Generates institution-specific explanation.*

- *Verify*$(L, \pi, R) \to \{0, 1\}$*: Verifies opposability* ≥ 0.60.

Theorem 3.1 (SH Function Composition). *The SH extraction function can be represented as the composition:*

$$\mathsf{SH}_{f,n}(x) = \phi \circ f^{(n)}(x) \tag{25}$$

where:

- $f^{(n)} : \mathbb{F}_q \to \mathbb{F}_q$ *is the n-fold composition of* $f(x) = x^2 + \alpha$.
- $\phi : \mathbb{F}_q \to \mathcal{V}_{\mathcal{J}}$ *is the legal semantic mapping function.*
- *The composition preserves the entropy bounds established in Theorem 2.4.*

Proofsketch 3.3. *The equality* $\mathsf{SH}_{f,n}(x) = \phi \circ f^{(n)}(x)$ *follows immediately from the definition of the* SH_Extract *algorithm (Algorithm 1), which first computes the iterative trace* $f^{(n)}(x)$ *and then applies the semantic mapping* ϕ *to the final state. The entropy preservation is a direct consequence of the composability of min-entropy. The output of* $f^{(n)}$ *maintains min-entropy* $H_\infty(f^{(n)}(X)) \geq \log_2 q - n \log_2 d$ *(Theorem 2.4). The subsequent mapping by* ϕ*, an almost-entropy-preserving function (Proposition 3.1), results in a final output whose min-entropy is bounded below by* $\log_2 |\mathcal{V}_{\mathcal{J}}| - \mathit{negl}(\lambda)$*. See Appendix A.9 for complete proof.*

3.2. Core SH Algorithm

Algorithm 1 SH Extraction Algorithm

Require: $h \in \mathbb{F}_q$, context $\in$ RelationalContext, $n \in \mathbb{Z}^+$
Ensure: LegalInterpretation with STARK proof

```
trace ← [h]                                          ▷ Initialize computation trace
for i ← 1 to n do
    x_next ← (x_current^2 + α) mod q                 ▷ AIIP iteration
    trace.append(x_next)
end for
semantic_map ← ∅
for i ← 0 to n do                                    ▷ Extract semantics at each state
    state ← trace[i]
    verdict_index ← state mod |context.verdict_space|
    verdict ← context.verdict_space[verdict_index]
    semantics ← apply_legal_context(verdict, context, i)
    semantic_map.append(semantics)
end for
proof ← generate_stark_proof(trace, context.constraints)
return LegalInterpretation(semantic_map[−1], semantic_map, trace, proof)
```

Proposition 3.2 (Algorithm Correctness). *For any input* $h \in \mathbb{F}_q$ *and RelationalContext R, Algorithm 1 produces a valid LegalInterpretation L such that:*

$$\Pr[\mathit{Verify}(L, \pi, R) = 1] \geq 1 - \mathit{negl}(\lambda) \tag{26}$$

where π *is the generated STARK proof.*

Proofsketch 3.4. *Algorithm 1 correctly produces valid interpretations by: (1) Computing the AIIP iteration trace, (2) Applying semantic mapping at each step, (3) Generating verifiable STARK proof. The verification succeeds with overwhelming probability by STARK completeness. Full proof in Appendix A.3.*

3.3. STARK Circuit Design and Verification

The SH primitive's claim of "STARK-proof verifiable computation" requires a precise description of the Arithmetic Intermediate Representation (AIR) circuit used to generate the zero-knowledge proofs. This circuit encodes the entire computation trace of the $\mathsf{SH_E xtract}$ algorithm, allowing for efficient verification of its correct execution. The design is optimized for the algebraic structure of the AIIP iteration.

Definition 3.3 (AIR Circuit for AIIP Iteration). *Let $\mathbb{F}$ be a finite field of size q. The AIR circuit $\mathcal{C}SH$ for the SH primitive with iteration depth n is defined by the following constraints:*

1. ***Boundary Constraints:*** *For the initial state s_0 and the final state s_n:*

$$s_0 = x \quad \textit{(input witness)}\ s_n \qquad\qquad = y \quad \textit{(output witness)} \tag{27}$$

2. ***Transition Constraints:*** *For each step $i \in [0, n-1]$, the state transition must satisfy the AIIP polynomial:*

$$si + 1 = s_i^2 + \alpha \quad \textit{in } \mathbb{F} \tag{28}$$

   where α is the fixed quadratic non-residue from the public parameters.

3. ***Semantic Mapping Constraints:*** *For each step i, the verdict extraction must adhere to the public mapping ϕ:*

$$v_i = \phi(s_i) = \mathsf{Extract} \circ \mathsf{Partition} \circ \mathsf{Normalize}(s_i) \tag{29}$$

   The function ϕ is encoded as a lookup table within the circuit, with constraints ensuring the correct execution of the partitioning and extraction steps.

The circuit $\mathcal{C}_{SH}$ has a width of 1 (representing the single field element state) and a height of $n + 1$ (the length of the trace including the initial state).

Algorithm 2 STARK Proof Generation for SH

Require: trace $= [s_0, s_1, \ldots, s_n]$, α, ϕ lookup table
Ensure: STARK proof π

execution_trace $\leftarrow$ trace ▷ The sequence of states s_i
constraints $\leftarrow \emptyset$
constraints.add(BoundaryConstraint(s_0, position $= 0$)) ▷ Initial state equals input
constraints.add(BoundaryConstraint(s_n, position $= n$)) ▷ Final state equals output
for $i \leftarrow 0$ to $n - 1$ **do**
    constraints.add(TransitionConstraint(s_i, s_{i+1}, α)) ▷ Enforce $s_{i+1} = s_i^2 + \alpha$
end for
for $i \leftarrow 0$ to n **do**
    constraints.add(LookupConstraint(s_i, v_i, ϕ)) ▷ Enforce $v_i = \phi(s_i)$ via PLONK-style lookup
end for
air $\leftarrow$ AIR(constraints, field $= \mathbb{F}$)
$\pi \leftarrow$ STARKProver(air, execution_trace)
return π

Algorithm 3 STARK Verification for SH Interpretation

Require: STARK proof π, public input x, public output y, α, ϕ
Ensure: $\{0, 1\}$

```
constraints ← [Constructed as in Algorithm 2]   ▷ Reconstruct constraints from public params
air ← AIR(constraints, field = 𝔽)
result ← STARKVerifier(π, air, public_input = x, public_output = y)
return result
```

Proposition 3.3 (STARK Circuit Completeness and Soundness). *The AIR circuit $\mathcal{CSH}$ satisfies:*

1. ***Completeness:*** *For any correctly computed trace $trace = [f^{(0)}(x), ..., f^{(n)}(x)]$, the proof generation algorithm* STARKProver *produces a proof π that is accepted by* STARKVerifier *with probability 1.*

2. ***Soundness:*** *For any adversarial trace $trace^*$ that violates the constraints of $\mathcal{CSH}$, the probability that* STARKVerifier *accepts the proof is at most $negl(\lambda)$, under the security of the STARK proof system.*

Proofsketch 3.5. *Completeness follows from the design of the AIR circuit $\mathcal{C}_{SH}$: if the trace is computed correctly according to the AIIP iteration and semantic mapping, then all constraints are satisfied by construction, and the STARK prover, being complete, will generate a valid proof that the verifier accepts. Soundness follows from the soundness of the underlying STARK proof system. The STARK protocol guarantees that if the verifier accepts a proof, then the trace satisfies all constraints with high probability. Since the constraints of $\mathcal{C}_{SH}$ exactly encode the correct execution of the AIIP iteration and the semantic mapping, any deviation from the correct trace will be detected with negligible probability of acceptance. Full proof in Appendix A.2*

Remark 3.1 (Efficiency of the Design). *The AIR circuit for SH is highly efficient for STARKs due to its simple algebraic constraints. The transition constraint is a degree-2 polynomial, which is well-suited for efficient proof composition. The semantic mapping ϕ is implemented as a lookup, which modern STARK systems (e.g., based on Plookup or Logup) support with minimal overhead. The total number of constraints is $O(n)$, leading to proof sizes of $O(\log n)$ and verification time of $O(\log n)$.*

3.4. Parameter Structure

Definition 3.4 (SH Parameters). *The SH primitive requires the following parameter structure:*

$$params = (q, \alpha, \lambda, \theta_{min}, \mathcal{I}, \mathcal{F}, \Pi_{STARK}) \tag{30}$$

where:

$$q \geq 2^{2\lambda} \quad \textit{Field size (prime or prime power)} \tag{31}$$
$$\alpha \in \mathbb{F}_q^{\setminus}(\mathbb{F}q^*)^2 \quad \textit{Quadratic non-residue} \tag{32}$$
$$\lambda \in 80, 112, 128, 192, 256 \quad \textit{Security level} \tag{33}$$
$$\theta min = 0.60 \quad \textit{Minimum opposability bound} \tag{34}$$
$$\mathcal{I} \subset \textit{Institutions} \quad \textit{Supported institutions} \tag{35}$$
$$\mathcal{F} \subset \textit{Frameworks} \quad \textit{Legal frameworks} \tag{36}$$
$$\Pi_{STARK} \quad \textit{STARK proof system parameters} \tag{37}$$

Definition 3.5 (LegalTrace). *The LegalTrace structure captures the complete computation and interpretation history:*

```
@dataclass
class LegalTrace:
    computation_trace: List[int]   # [h, f(h), f  (h), ..., f
        (h)]
    air_witness: AIRWitness   # AIR constraint witness
    semantic_trace: List[LegalSemantics]   # Legal meaning at
        each step
    final_interpretation: LegalSemantics   # Final legal
        interpretation
    stark_proof: STARKProof   # Verification proof
```

Definition 3.6 (LegalSemantics). *The LegalSemantics structure represents extracted legal meaning:*

```
@dataclass
class LegalSemantics:
    field_element: int   # Original field element
    verdict: str   # Base verdict from verdict space
    contextual_meaning: str   # Context-specific interpretation
    legal_statement: str   # Formal legal statement
    interpretation_confidence: float   # Confidence in
        interpretation [0,1]
    applicable_laws: List[str]   # Applicable legal authorities
    precedents: List[str]   # Relevant legal precedents
    opposability_components: Dict[str, float]   # Individual
        scores for E, C, V
```

Remark 3.2. *The* `opposability_components` *field explicitly tracks the three components (Explainability, Contestability, Verifiability) to ensure transparency in the minimum opposability calculation.*

Definition 3.7 (InstitutionalExplanation). *The InstitutionalExplanation structure provides institution-specific explanations:*

```
@dataclass
class InstitutionalExplanation:
    institution: Institution   # Target institution
    explanation_text: str   # Human-readable explanation
    legal_basis: List[str]   # Legal basis for interpretation
    precedents: List[str]   # Relevant precedents
    opposability_score: float   # Opposability score [0,1]
    contestability_analysis: str   # How interpretation can be
        contested
    alternative_interpretations: List[str]   # Alternative valid
         interpretations
```

Definition 3.8 (RelationalContext). *The RelationalContext structure defines the interpretation environment:*

```
@dataclass
class RelationalContext:
    relationship_type: str   # "corporate", "personal", "public"
```

```
    context_type: str    # "audit", "dispute", "compliance"
    jurisdiction: str    # Legal jurisdiction
    legal_framework: str    # Applicable legal framework
    verdict_space: List[str]    # Possible verdicts for this
        context
    institutional_rules: Dict[str, List[Rule]]    # Rules by
        institution
    complexity_level: str    # "simple", "standard", "complex"
```

Proposition 3.4 (Data Structure Completeness). *The data structures Definitions 3.5–3.8 form a complete representation system for SH operations, satisfying:*

1. ***Representation**: All SH computation states can be represented.*
2. ***Verifiability**: All structures support STARK verification.*
3. ***Composability**: Structures can be composed for complex legal interpretations.*
4. ***Efficiency**: Memory requirements grow as $O(n)$ with iteration depth.*

Proofsketch 3.6. *The data structures form a complete system by: (1) Representing all computation states via LegalTrace, (2) Supporting STARK verification through witness structures, (3) Enabling composition via standardized interfaces, (4) Maintaining linear memory growth. Full proof in Appendix A.8.*

4. Opposability Properties and Guarantees

4.1. Minimum Opposability Bound

Theorem 4.1 (Minimum Opposability Bound). *For any legal interpretation L produced by* SH_Extract *with verifiable STARK proof π and relational context R, the opposability score satisfies:*

$$\Omega(L, R) \geq 0.60 \tag{38}$$

with probability:

$$\Pr[\Omega(L, R) \geq 0.60] \geq 1 - negl(\lambda) \tag{39}$$

for security parameter λ.

Proofsketch 4.1. *The minimum opposability bound follows from three observations:*

1. *Each component (explainability, contestability, verifiability) individually satisfies the 0.60 threshold by construction (from Lemmas and Propositions in Section 4)*
2. *The min-min formulation in Definition 2.2 ensures the weakest component determines opposability, preventing degradation below 0.60*
3. *Under Assumption 2.2, the probability that all institutions meet the threshold is the product of their individual probabilities*

By the union bound and the fact that each institution fails with probability at most $negl(\lambda)$, the overall probability that $\Omega(L, R) \geq 0.60$ is at least $1 - |\mathcal{I}| \cdot negl(\lambda) = 1 - negl(\lambda)$ since the number of institutions $|\mathcal{I}|$ is polynomial in λ. Full proof in Appendix A.1.

4.2. Institutional Explainability

Lemma 4.1 (Universal Explainability). *For any institution $I \in \mathsf{Institutions}(\mathcal{J})$ and legal interpretation L generated by SH, there exists an explanation $E = \mathsf{SH_Explain}(L, I)$ satisfying:*

$$\mathrm{E}(L, I) \geq 0.60 \tag{40}$$

with probability:

$$\Pr[\mathrm{E}(L, I) \geq 0.60] \geq 1 - \mathit{negl}(\lambda) \tag{41}$$

Proofsketch 4.2. *The guarantee follows from the δ-completeness ($\delta = 0.60$) of the institutional rule sets $\mathcal{R}I$, ensured during* RelationalContext *setup. This design ensures the mutual information $I(L; I)$ between interpretation L and any institution I's rules is bounded below by $0.60 \cdot H(L)$. The probability of an interpretation L falling outside the subspace well-defined by $\mathcal{R}I$ is negligible, as its measure is bounded by $2^{-H\infty(L)}$ and $H\infty(L) = \Omega(\lambda)$ by Theorem 2.4. The full proof is given in Appendix A.10.*

4.3. Legal Contestability

Proposition 4.1 (Contestability Guarantee). *Every SH interpretation maintains valid contestation procedures preserving:*

$$\Gamma(L, \mathcal{J}) \geq 0.60 \tag{42}$$

during dispute resolution, with:

$$\Pr[\Gamma(L, \mathcal{J}) \geq 0.60] \geq 1 - \mathit{negl}(\lambda) \tag{43}$$

Proofsketch 4.3. *The system is designed such that for each institution I, at least one contestation pathway j has availability probability $p_{ij \geq \gamma}$, where γ is chosen so that $\gamma^k \geq 0.60$ for $k = |\mathcal{I}|$. This ensures the product $\Gamma(L, \mathcal{J}) \geq \gamma^k \geq 0.60$. The probability of violation is the probability that the underlying system infrastructure fails to meet these designed availability thresholds for the required pathways. Since pathway availability is independent of L and failures are independent events with negligible probability, the overall guarantee holds. Complete proof in Appendix A.11.*

4.4. Verification Mechanisms

Definition 4.1 (Opposability Verification). *The opposability verification function* $\mathsf{Verify}(L, \pi, R) \to \{0, 1\}$ *checks:*

$$\mathsf{Verify}(L, \pi, R) = \begin{cases} 1 & \text{if } \Omega(L, R) \geq 0.60 \wedge \mathsf{STARKVerify}(\pi) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{44}$$

where $\Omega(L, R)$ is computed as:

$$\Omega(L, R) = \min_{I \in \mathrm{Institutions}(\mathcal{J})} [\min \{\mathrm{E}(L, I), \Gamma(L, I), \mathsf{Verifiability}(L)\}] \tag{45}$$

and $\mathsf{Verifiability}(L) := 1$ *if* $\mathsf{STARKVerify}(\pi) = 1$*, and* 0 *otherwise.*

Algorithm 4 Opposability Verification Algorithm

Require: L: LegalInterpretation, π: STARKProof, R: RelationalContext
Ensure: $\{0, 1\}$ indicating opposability ≥ 0.60

```
valid ← STARKVerify(π)
if not valid then
    return 0
end if
Ω ← CalculateOpposability(L, R)
if Ω ≥ 0.60 then
    return 1
else
    return 0
end if
```

Theorem 4.2 (Verification Completeness). *The verification mechanism satisfies:*

1. ***Completeness**: Valid interpretations verify correctly:*

$$\Pr[\mathsf{Verify}(L, \pi, R) = 1 \,|\, \Omega(L, R) \geq 0.60] \geq 1 - negl(\lambda) \tag{46}$$

2. ***Soundness**: Invalid interpretations are rejected:*

$$\Pr[\mathsf{Verify}(L, \pi, R) = 0 \,|\, \Omega(L, R) < 0.60] \geq 1 - negl(\lambda) \tag{47}$$

3. ***Efficiency**: Verification time is polynomial in λ:*

$$T_{\mathsf{verify}} = O(\mathsf{poly}(\lambda)) \tag{48}$$

Remark 4.1. *The verification mechanism provides a robust guarantee of legal opposability, ensuring that only interpretations meeting the minimum 0.60 bound are accepted as valid. This threshold represents the mathematical formalization of the concept of "substantial compliance" in legal frameworks.*

5. Security Analysis

5.1. Opposability Security Properties (Reduction to AIIP)

Theorem 5.1 (SH Security Reduction to AIIP). *The security of the Semantic Holder primitive reduces to the hardness of the Affine Iterated Inversion Problem (AIIP). For any probabilistic polynomial-time adversary $\mathcal{A}$ that breaks SH with advantage ϵ, there exists an adversary $\mathcal{B}$ that breaks AIIP with advantage:*

$$Adv_{AIIP}(\mathcal{B}) \geq \epsilon - negl(\lambda) \tag{49}$$

where λ is the security parameter. The reduction accounts for the non-injectivity of ϕ via enumeration of a polynomial-sized preimage set, as per Assumption 3. This introduces a polynomial loss factor in the advantage, which is acceptable for asymptotic security.

Proofsketch 5.1. *We construct a reduction algorithm $\mathcal{B}$ that uses any PPT adversary $\mathcal{A}$ breaking the one-wayness of* SH *to solve the AIIP. $\mathcal{B}$ receives an AIIP instance (f, n, y) and must find x' such that $f^{(n)}(x') = y$.*

1. *$\mathcal{B}$ simulates a* RelationalContext *R (including the semantic mapping ϕ and verdict space $\mathcal{V}_{\mathcal{J}}$). $\mathcal{B}$ computes the target legal interpretation $L_y = \phi(y)$ within this context.*

2. *$\mathcal{B}$ runs $\mathcal{A}(f, n, L_y, R)$ to obtain a purported preimage L_x such that $\mathsf{SH}_{f,n}(L_x) = \phi(f^{(n)}(\phi^{-1}(L_x))) = L_y$ with advantage ϵ.*

3. *Crucially, ϕ may not be injective. $\mathcal{B}$ therefore efficiently enumerates the preimage set $P = \phi^{-1}(L_x)$, which is of size at most $M = \lceil q/|\mathcal{V}_{\mathcal{J}}| \rceil = (\lambda)$.*

4. *For each candidate $x' \in P$, $\mathcal{B}$ checks if $f^{(n)}(x') = y$.*

5. *If a candidate x' satisfies the equation, $\mathcal{B}$ outputs it as the solution to the AIIP instance.*

The advantage of $\mathcal{B}$ is ϵ/M, as $\mathcal{A}$ must produce a valid L_x, and $\mathcal{B}$ must then guess the correct preimage among the M possibilities. Since M is polynomial in λ, if ϵ is non-negligible, then so is ϵ/M. Thus, $Adv_{AIIP}(\mathcal{B}) \geq \epsilon/(\lambda) = \epsilon - negl(\lambda)$. For complete proof see Appendix A.5

Corollary 5.1. *Under the AIIP hardness assumption, the SH primitive provides:*

1. ***One-wayness**: Given $L_y = \mathsf{SH}_{f,n}(L_x)$, finding any L'_x such that $\mathsf{SH}_{f,n}(L'_x) = L_y$ is computationally infeasible.*

2. ***Opposability preservation**: The minimum opposability bound $\Omega \geq 0.60$ is maintained under adversarial conditions.*

3. ***Semantic security**: No adversary can learn meaningful information about L_x from L_y beyond what is revealed by the legal interpretation.*

5.2. Resistance to Attacks

5.2.1. Algebraic Attacks

Proposition 5.1 (Resistance to Algebraic Attacks). *The SH primitive is resistant to algebraic attacks due to:*

1. ***Degree explosion**: $\deg(f^{(n)}) = d^n$ grows exponentially with n.*

2. ***Entropy preservation**: $H_\infty(\mathsf{SH}_{f,n}(X)) \geq \log_2 q - n \log_2 d$.*

3. ***Structural complexity**: The AIR constraints create a system resistant to Gröbner basis attacks.*

The best known algebraic attacks require time:

$$T_{alg} = \Omega\left(\min(q, d^n)\right) = \Omega(2^\lambda) \tag{50}$$

for λ-bit security.

Proofsketch 5.2. *Algebraic attacks against the AIIP iteration require solving the system $f^{(n)}(x) = y$. The degree of the polynomial $f^{(n)}$ is $d^n = 2^n$. Direct root-finding algorithms (e.g., Berlekamp's) have complexity $\Omega(2^n)$ in the degree. Modeling the iteration as a triangular system of quadratic equations, the solving degree for Gröbner basis attacks is known to be lower-bounded by 2^n for generic iterated systems [15], leading to complexity $\Omega(2^{n^2})$ in the worst case. However, the structured nature of the AIIP iteration may admit more efficient attacks; yet, for the parameters chosen ($n = \lambda$), the complexity remains exponential in λ. Thus, the overall classical work factor is $\Omega(2^\lambda)$. Full proof in Appendix A.15.*

5.2.2. Legal Interpretation Attacks

Proposition 5.2 (Resistance to Legal Interpretation Attacks). *SH resists legal interpretation attacks through:*

1. ***Entropy bounding**: $H(L|I) \leq 0.4 \cdot H(L)$ ensures explanations are substantive.*
2. ***Cross-institutional verification**: Opposition requires breaking multiple institutional frameworks simultaneously.*
3. ***Contestation pathways**: Multiple dispute resolution mechanisms prevent single points of failure.*

An adversary attempting to craft ambiguous interpretations succeeds with probability at most:

$$\Pr[\textit{ambiguous interpretation}] \leq 2^{-(H_\infty(L)-\log_2(1/0.4))} \leq 2^{-(\lambda-1.32)} \tag{51}$$

Proofsketch 5.3. *An adversary seeks to find an input x producing an interpretation L with low explainability ($H(L|I) > 0.4 \cdot H(L)$) for some I. However, the output of $\mathsf{SH}f, n$ is negl(λ)-close to uniform over $\mathcal{VJ}$ (Prop. 3.1). The set of "ambiguous" interpretations for I has size at most $|\mathcal{V}_\mathcal{J}|^{0.4}$ by the 0.60-completeness of $\mathcal{R}I$. The probability a uniform sample lies in this set is $|\mathcal{VJ}|^{-0.6} = negl(\lambda)$. Thus, the adversary's success probability is negligible. For full proof see Appendix A.13.*

5.2.3. Institutional Compromise

Proposition 5.3 (Resistance to Institutional Compromise). *SH maintains security even when some institutions are compromised:*

1. ***Threshold security**: The minimum opposability bound requires agreement across multiple institutions.*
2. ***Isolation property**: Compromise of institution I doesn't reveal information about other institutions' rule sets.*
3. ***Fail-safe design**: Compromised institutions can be detected through verification failures.*

The probability that compromise of k institutions affects the opposability bound is:

$$\Pr[\Omega < 0.60 \textit{ with } k \textit{ compromised}] \leq \binom{|\mathcal{I}|}{k} \cdot (1 - 0.60)^k \tag{52}$$

which remains negligible for $k < |\mathcal{I}|/2$.

Proofsketch 5.4. *The opposability score Ω is the minimum over all institutions of their individual scores, which combine explainability, contestability, and verifiability. For each institution, the probability that its score is below 0.60 is at most $\beta = negl(\lambda)$ due to the explainability guarantee (Lemma 4.1) and the contestability guarantee (from the design in Proposition 4.1). Verifiability is high with probability 1 due to STARK completeness. For Ω to fall below 0.60, at least one institution must have a score below 0.60. By the union bound, the probability that any institution has a score below 0.60 is at most $N \cdot \beta$, where $N = |\mathcal{I}|$. Since $N = (\lambda)$ and $\beta = negl(\lambda)$, this probability is negligible. This holds even with compromised institutions, as the score is objective and based on public rules and verification algorithms. Full proof in Appendix A.16.*

5.2.4. Quantum Attacks

Proposition 5.4 (Quantum Resistance). *SH provides quantum resistance through:*

1. ***Grover limitation**: Quantum search provides at most quadratic speedup:*

$$T_{quantum} = O(\sqrt{q}) = O(2^{\lambda/2}) \tag{53}$$

2. ***STARK security**: The STARK proof system remains secure against quantum adversaries under the quantum random oracle model, assuming the use of a quantum-resistant hash function (e.g., SHA-3) for commitments.*

3. ***AIIP foundation**: The underlying AIIP problem resists quantum attacks due to its high-genus curve structure.*

For λ-bit classical security, SH provides $\lambda/2$-bit quantum security.

Proofsketch 5.5. *The security of SH reduces to the hardness of AIIP. The best known quantum attack against AIIP is Grover's algorithm applied to the search for a preimage in the domain $\mathbb{F}_q$, providing a quadratic speedup with complexity $O(\sqrt{q})$. Given the parameter choice $q \geq 2^{2\lambda}$ for λ-bit classical security, the quantum work factor becomes $O(2^{\lambda})$, sustaining $\lambda/2$-bit quantum security. The connection to the HCDLP in a curve of genus $g = 2^{n-1} - 1$ offers additional security, as no efficient quantum algorithm is known for high-genus HCDLP, with the best attacks exhibiting complexity exponential in g. The STARK proof system uses a quantum-resistant hash function for Merkle tree commitments, and its security against quantum adversaries is maintained in the quantum random oracle model [4]. See full proof in Appendix A.18*

5.3. Parameter Recommendations Table

Table 1: SH Parameter Recommendations for Standard Security Levels

Security Level	**Field Size** q	**Iterations** n	**Min Institutions** $\lvert\mathcal{I}\rvert$	**Opposability** $\theta_{\min}$
80-bit	$\geq 2^{160}$	80	3	≥ 0.60
112-bit	$\geq 2^{224}$	112	4	≥ 0.60
128-bit	$\geq 2^{256}$	128	5	≥ 0.60
192-bit	$\geq 2^{384}$	192	7	≥ 0.60
256-bit	$\geq 2^{512}$	256	9	≥ 0.60

Theorem 5.2 (Parameter Security). *For parameters chosen according to Definition 3.4 and Table 1, the SH primitive achieves:*

1. ***Classical security**: $T_{classical} = \Omega(2^{\lambda})$.*

2. ***Quantum security**: $T_{quantum} = \Omega(2^{\lambda/2})$.*

3. ***Opposability guarantee**: $\Omega \geq 0.60$ with probability $\geq 1 - negl(\lambda)$.*

4. ***Institutional robustness**: Resistance to compromise of up to $\lfloor |\mathcal{I}|/2 \rfloor$ institutions.*

5. ***Statistical distance**: $\Delta(\mathsf{SH}_{f,n}(X), U) \leq negl(\lambda)$ for $X \sim \mathcal{U}(\mathbb{F}_q)$ and U uniform over $\mathcal{V}_{\mathcal{J}}$.*

Proofsketch 5.6. *The security guarantees follow from: 1.* ***Classical security****: The minimum of the field size $q \geq 2^{2\lambda}$ and the degree of the iterated polynomial $d^n = 2^\lambda$ determines the complexity of direct algebraic and brute-force attacks, yielding $T_{classical} = \Omega(\min(q, d^n)) = \Omega(2^\lambda)$. 2.* ***Quantum security****: Grover's algorithm provides a quadratic speedup, leading to $T_{quantum} = \Omega(\min(\sqrt{q}, \sqrt{d^n})) = \Omega(2^{\lambda/2})$. 3.* ***Opposability guarantee****: This is a direct corollary of Theorem 4.1. 4.* ***Institutional robustness****: This follows from Proposition 5.3. 5.* ***Statistical distance****: This follows from Theorem 2.5 and the entropy preservation of ϕ (Proposition 3.1). Full proof in Appendix A.4.*

Remark 5.1. *The parameter recommendations in Table 1 provide a conservative security stance, ensuring both cryptographic hardness and legal opposability. Implementations may adjust these parameters based on specific threat models and performance requirements while maintaining the minimum opposability bound of 0.60.*

5.4. UC Security Formulation

To establish the composable security of the Semantic Holder (SH) primitive within the Chaotic Affine Secure Hash (CASH) framework, we define an ideal functionality $\mathcal{F}_{\text{SH}}$ in the Universally Composable (UC) framework [9]. This functionality captures the core operations of SH, extraction, verification, and explanation, while guaranteeing opposability, correctness, and privacy. The functionality interacts with parties (e.g., users, institutions) and the environment $\mathcal{Z}$, and is designed to be securely realizable by the real-world SH protocol described in Section 3.

5.4.1. Ideal Functionality $\mathcal{F}_{SH}$

The functionality $\mathcal{F}_{\text{SH}}$ is parameterized by a security parameter λ, a finite field $\mathbb{F}_q$, a quadratic non-residue $\alpha \in \mathbb{F}_q$, an iteration depth n, and a verdict space $\mathcal{V}_\mathcal{J}$ for a jurisdiction $\mathcal{J}$. It maintains an internal database to store extraction records and proofs. **Functionality $\mathcal{F}_{\text{SH}}$:**

- Upon receiving $(\mathsf{extract}, sid, x, R, n)$ from a party P:
  - Verify that sid is a unique session ID and R is a valid relational context.
  - Compute $y = f^{(n)}(x)$ where $f(x) = x^2 + \alpha$.
  - Compute $L = \phi(y)$ using the semantic mapping ϕ defined by R.
  - Generate a simulated STARK proof π attesting to the correct computation of y from x and L from y.
  - Store the record (sid, x, y, L, π, R).
  - Output $(\mathsf{interpretation}, sid, L, \pi)$ to P.
- Upon receiving $(\mathsf{verify}, sid, L, \pi, R)$ from a party P:
  - Check if a record (sid, x, y, L, π, R) exists for some x, y.
    - If yes, set $\mathsf{valid} = 1$ and compute $\Omega(L, R) \geq 0.60$ (since the record was ideally generated).
    - Else, set $\mathsf{valid} = 0$.
  - Output $(\mathsf{verification}, sid, \mathsf{valid})$ to P.
- Upon receiving $(\mathsf{explain}, sid, L, I)$ from a party P:

– Check if a record (sid, x, y, L, π, R) exists for some x, y, π, R.
  * If yes, compute $E = \mathsf{Explain}(L, I)$ using the institutional rules from R.
  * Else, set $E = \perp$.
– Output $(\mathsf{explanation}, sid, E)$ to P.

5.4.2. Security Guarantees

- **Correctness**: For any honest extraction, $\mathcal{F}_{\text{SH}}$ always outputs an interpretation L with $\Omega(L, R) \geq 0.60$ and a valid proof π.
- **Extraction-Soundness**: Only extractions performed via $\mathcal{F}_{\text{SH}}$ lead to successful verification.
- **Privacy**: The functionality reveals only the output L and proof π, not the input x or intermediate states, unless corrupted.
- **Composability**: $\mathcal{F}_{\text{SH}}$ can be composed with other ideal functionalities (e.g., $\mathcal{F}_{\text{CEE}}$ for confidentiality) while maintaining security.

Theorem 5.3 (UC Realizability). *The real-world SH protocol π_{SH} (Algorithm 1) securely realizes $\mathcal{F}_{SH}$ under the AIIP hardness assumption and the soundness of STARKs.*

Proofsketch 5.7. *A simulator $\mathcal{S}$ can simulate the real protocol by generating proofs and interpretations ideally, while leveraging the AIIP hardness to handle adversarial queries. The opposability guarantee ensures that the environment cannot distinguish between the real and ideal worlds. Full proof follows the UC framework with a reduction to AIIP and is provided in Appendix A.19.*

6. Applications and Use Cases

6.1. Legal Smart Contracts

Definition 6.1 (SH-Enhanced Smart Contract). *An SH-enhanced smart contract is a tuple $\mathcal{SC} = (\mathcal{C}, \mathcal{R}, \mathcal{V}, \Pi)$ where:*

- *$\mathcal{C}$: Contract code and conditions.*
- *$\mathcal{R}$: Relational context for legal interpretation.*
- *$\mathcal{V}$: Verdict space for possible outcomes.*
- *Π: STARK proof system parameters.*

The contract execution produces legally opposable results with $\Omega \geq 0.60$.

Algorithm 5 SH Smart Contract Execution

Require: contract_data, legal_context, params
Ensure: Legally opposable contract outcome
state ← initialize_contract(contract_data)
hash ← hash(state)
encrypted_state ← CEE_Encrypt(hash, legal_context)
interpretation ← SH_Extract(encrypted_state, legal_context, params.n)
proof ← generate_stark_proof(interpretation)
if verify_opposability(interpretation, proof) ≥ 0.60 **then**
    execute_contract(interpretation.verdict)
    return interpretation.verdict
else
    trigger_dispute_resolution(interpretation)
    return null
end if

Theorem 6.1 (Smart Contract Opposability). *SH-enhanced smart contracts maintain the minimum opposability bound:*

$$\Pr[\Omega(\mathsf{SC_result}) \geq 0.60] \geq 1 - \mathit{negl}(\lambda) \tag{54}$$

for security parameter λ.

Proofsketch 6.1. *An SH-enhanced smart contract (Definition 6.1) produces its result by calling* SH_Extract *on the contract's state. Since Theorem 4.1 applies universally to all outputs of* SH_Extract*, the opposability guarantee extends directly to the smart contract's result. The probability bound is identical. See complete proof in Appendix A.7.*

6.2. Regulatory Compliance Auditing

Definition 6.2 (Compliance Verification). *For regulation $\mathcal{R}$ and entity E, the compliance status is given by:*

$$\mathit{Compliant}(E, \mathcal{R}) \iff \exists L \in \mathcal{L}_{\mathit{valid}} : \mathit{Verdict}(L) \in \mathit{Valid}_{\mathcal{R}} \tag{55}$$

where $\mathcal{L}_{\mathit{valid}}$ is the set of interpretations with $\Omega \geq 0.60$.

Algorithm 6 Automated Compliance Audit

Require: entity_data, regulation, audit_context
Ensure: Compliance report with opposability guarantee
evidence ← collect_evidence(entity_data, regulation)
digest ← hash(evidence)
interpretation ← SH_Extract(digest, audit_context, n)
report ← generate_compliance_report(interpretation)
proof ← generate_stark_proof(interpretation.trace)
if verify_opposability(interpretation, proof) ≥ 0.60 **then**
    report.set_status(COMPLIANT)
else
    report.set_status(NON_COMPLIANT)
    report.add_contestation_options(interpretation.contestation_paths)
end if
return report

6.3. Digital Evidence Authentication

Definition 6.3 (Evidence Authentication). *Digital evidence E is authentically represented if:*

$$\mathit{Authentic}(E) \iff \exists L \in \mathcal{L}_{valid} : \mathit{Semantics}(L) \equiv \mathit{Meaning}(E) \tag{56}$$

with $\Omega(L) \geq 0.60$ and verifiable chain of custody.

[Evidence Authentication Protocol]

1. **Evidence Hashing**: $h \leftarrow \text{SHA3-512}(E)$.
2. **Context Setup**: $R \leftarrow \text{create_legal_context}(\text{jurisdiction}, \text{evidence_type})$.
3. **Semantic Extraction**: $L \leftarrow \mathsf{SH_Extract}(h, R, n)$.
4. **Proof Generation**: $\pi \leftarrow \text{generate_stark_proof}(L.\text{trace})$.
5. **Verification**: $\text{verify_opposability}(L, \pi) \geq 0.60$.

6.4. Implementation Examples with Algorithms

Algorithm 7 SH Library Implementation

```
function SH_PRIMITIVE
    self.params ← SH_Parameters()
    self.stark_system ← STARKSystem(self.params.stark_security)
    self.semantic_engine ← LegalSemanticEngine(self.params.institutional_space)
end function
function EXTRACT_LEGAL_INTERPRETATION(input_data, relational_context, iteration_depth)
    h ← bytes_to_field_element(input_data, self.params.field_size)
    legal_trace ← generate_legal_trace(h, relational_context, iteration_depth)
    stark_proof ← self.stark_system.generate_proof(legal_trace.computation_trace)
    legal_trace.stark_proof ← stark_proof
    return LegalInterpretation(legal_trace, relational_context)
end function
```

Algorithm 8 Cross-Jurisdictional Compliance Check

Require: business_data, jurisdictions, regulations
Ensure: Multi-jurisdictional compliance report

```
results ← ∅
for jurisdiction ∈ jurisdictions do
    context ← create_context(jurisdiction, regulations[jurisdiction])
    interpretation ← SH_Extract(business_data, context)
    result ← verify_compliance(interpretation)
    results.add((jurisdiction, result))
end for
consensus ← reach_consensus(results)
return consensus
```

For financial regulation compliance, the SH primitive can be implemented as:

Listing 1: Financial Regulation Implementation

```
class FinancialComplianceSH(SHPrimitive):
    def __init__(self, regulatory_frameworks):
        super().__init__()
        self.frameworks = regulatory_frameworks

    def check_compliance(self, transaction_data, framework_name):
        framework = self.frameworks[framework_name]
        context = RelationalContext(
            relationship_type="financial",
            context_type="compliance",
            jurisdiction=framework.jurisdiction,
            legal_framework=framework.name,
            verdict_space=framework.compliance_levels
        )

        interpretation = self.extract_legal_interpretation(
            transaction_data, context, self.params.n
        )

        return {
            'compliant': interpretation.verdict in framework.valid_verdicts,
            'opposability_score': interpretation.opposability_score,
            'legal_basis': interpretation.applicable_laws,
            'proof': interpretation.stark_proof
        }
```

Theorem 6.2 (Implementation Correctness). *The SH implementation algorithms satisfy:*

1. ***Functional correctness**: Algorithms produce outputs that meet specifications.*
2. ***Security preservation**: Implementations maintain the security properties of the theoretical construction.*
3. ***Opposability guarantee**: $\Omega \geq 0.60$ is preserved in practical implementations.*
4. ***Efficiency**: Polynomial-time execution in security parameter λ.*

Proofsketch 6.2. *The implementation correctness follows from: 1. **Functional correctness**: The algorithms are direct, line-by-line translations of the mathematical operations defined in the SH construction. 2. **Security preservation**: The implementation instantiates the abstract parameters with concrete values that satisfy the security requirements of Definition 3.4. 3. **Opposability guarantee**: The entropy-preserving properties of the core algorithms are preserved in the implementation, and the verification step checks the opposability bound. 4. **Efficiency**: The number of iterations $n = \lambda$ and the cost of field operations is polynomial in $\log q$ (and thus in λ), ensuring overall polynomial-time execution. For full proof, see Appendix A.21.*

7. Performance and Optimization

The following analysis provides **asymptotic complexity expectations** and analytical estimates for the SH primitive. All figures are derived from theoretical analysis of the algorithms and parameter sizes, not from empirical measurement, consistent with the theoretical scope of this work.

7.1. Computational Complexity Analysis

Theorem 7.1 (SH Computational Complexity). *The Semantic Holder primitive has the following computational complexity characteristics:*

$$T_{extract}(n, q) = O(n \cdot M(q))$$
$$T_{explain}(L, I) = O(|\mathcal{R}_I| + |\mathcal{T}_I|)$$
$$T_{verify}(\pi) = O(\log n \cdot (\lambda))$$

where:

- $M(q)$ *is the complexity of field multiplication in* $\mathbb{F}_q$.
- $|\mathcal{R}_I|$ *is the number of rules for institution* I.
- $|\mathcal{T}_I|$ *is the number of explanation templates for institution* I.
- π *is the STARK proof of size* $O(\log n)$.

Proofsketch 7.1. *The complexity results follow from: 1.* ***Extraction****: The algorithm performs* n *iterations. Each iteration consists of one field squaring and one field addition. The cost of field multiplication in* $\mathbb{F}_q$ *is* $M(q) = O(\log q)$ *using standard algorithms, leading to total time* $O(n \log q)$. *2.* ***Explanation****: The algorithm must process each rule in the institution's rule set* $\mathcal{R}_I$ *and match against each template in* $\mathcal{T}_I$*, resulting in linear complexity* $O(|\mathcal{R}_I| + |\mathcal{T}_I|)$. *3.* ***Verification****: The verification time for a STARK proof is logarithmic in the size of the computation trace, which is* $O(n)$*, and polynomial in the security parameter* λ *due to the cryptographic operations involved. Complete proof in Appendix A.20*

Table 2: Computational Complexity by Operation

Operation	**Time Complexity**	**Space Complexity**	**Parallelizability**
Extraction	$O(n \log q)$	$O(n \log q)$	Limited
Explanation	$O(\|\mathcal{R}\|)$	$O(\|\mathcal{T}\|)$	High
Verification	$O(\log n)$	$O(\log n)$	High
STARK Proof Generation	$O(n \log n)$	$O(n \log n)$	Medium

7.2. Memory Requirements

Table 3: Analytical Memory Complexity Model

Component	**Asymptotic Complexity**	**Primary Scaling Factor**
Computation Trace	$O(n \log q)$	Iterations (n) $\times$ Field element size ($\log q$)
Field Element State	$O(\log q)$	Field size ($q \approx 2^{2\lambda}$)
STARK Proof	$O(\log n)$	Iteration depth (n)
Example ($\lambda = 128$): $n = 128$, $\log q = 256$ bits, Trace: $\sim 128 \times 32$ bytes = 4 KB		

7.3. Expected Performance Characteristics

The expected performance of SH is characterized by its computational complexity, which is dominated by n iterations of a squaring and addition operation in a large field $\mathbb{F}_q$. Asymptotically, the extraction time is $O(n \cdot M(q))$ where $M(q)$ is the complexity of multiplication in $\mathbb{F}_q$. Given the large field size requirements ($q \geq 2^{2\lambda}$), SH is expected to be computationally intensive compared to traditional symmetric primitives like SHA-3 or BLAKE3, which is the inherent cost of achieving verifiable legal opposability. Its primary comparative advantage lies not in throughput but in its unique cryptographic property set. Future work will focus on practical optimizations and hardware acceleration to improve its efficiency.

7.4. Optimization Techniques

Definition 7.1 (SH Optimization Strategies). *The following optimization techniques significantly improve SH performance:*

- ***Precomputation**: Store common iteration patterns and intermediate states.*
- ***Parallelization**: Distribute iteration steps across multiple processors.*
- ***Memory pooling**: Reuse memory allocations for frequent operations.*
- ***Field arithmetic optimization**: Use specialized algorithms for field operations.*
- ***Template caching**: Cache frequently used explanation templates.*

Algorithm 9 Optimized SH Extraction with Precomputation

```
function OPTIMIZED_EXTRACT(h, context, n)
    Check precomputation cache for pattern matching h
    if cache hit then
        trace ← retrieve_from_cache(h)
    else
        trace ← compute_trace_parallel(h, n)
        cache_result(h, trace)
    end if
    semantics ← extract_semantics(trace, context)
    return semantics
end function
function COMPUTE_TRACE_PARALLEL(h, n)
    Partition iteration into p blocks
    Assign each block to a processor
    Compute each block in parallel
    Combine results
    return combined trace
    semantics ← apply_legal_context(verdict, context, i)   ▷ See Appendix B for definition
end function
```

Theorem 7.2 (Optimization Effectiveness). *The optimization techniques provide the following performance improvements:*

$$\begin{aligned} \textit{Precomputation:} &\quad T_{extract} \to O(1) \textit{ for cached inputs} \\ \textit{Parallelization:} &\quad T_{extract} \to O(n/p) \textit{ with } p \textit{ processors} \\ \textit{Memory pooling:} &\quad M_{extract} \to O(1) \textit{ amortized} \\ \textit{Field optimization:} &\quad M(q) \to O(1) \textit{ with specialized hardware} \end{aligned}$$

Proofsketch 7.2. *The performance improvements are achieved as follows: 1.* ***Precomputation****: Caching the results of the iteration for frequent inputs eliminates the need for runtime computation, reducing time to $O(1)$ for cache hits. 2.* ***Parallelization****: The independent nature of the iteration steps allows for perfect parallelization across p processors, reducing the time for the iterative part of the computation by a factor of p. 3.* ***Memory pooling****: Reusing memory allocations for the computation trace and other structures eliminates the overhead of dynamic memory allocation for each invocation, leading to amortized constant memory overhead. 4.* ***Field optimization****: Implementing field arithmetic in specialized hardware (e.g., an FPGA) can reduce the time for a single multiplication to a constant cycle count, making $M(q) = O(1)$. See Appendix A.14 for detailed proof.*

Table 4: Performance Improvement from Optimizations

Optimization	Speedup Factor	Memory Reduction	Applicability
Precomputation	10-100×	0×	High-frequency inputs
Parallelization	p×	0×	Multi-core systems
Memory Pooling	1.5-2×	2-5×	All operations
Field Optimization	3-10×	0×	Hardware acceleration
Template Caching	2-5×	1.5×	Frequent institutions

Remark 7.1. *The optimization techniques make SH practical for real-world applications while maintaining all security and opposability guarantees. The parallelization approach is particularly effective given the iterative nature of the computation, while precomputation provides dramatic speedups for frequently encountered inputs.*

Proposition 7.1 (Optimization Security). *All optimization techniques preserve the security properties of SH:*

1. ***Precomputation*** *does not reveal information about inputs due to the one-way nature of the iteration.*
2. ***Parallelization*** *maintains the same computation path and thus the same security guarantees.*
3. ***Memory pooling*** *does not affect the mathematical properties of the operations.*
4. ***Field optimizations*** *are implementation details that don't change the underlying mathematics.*

Proofsketch 7.3. *The optimizations preserve security because: 1.* ***Precomputation*** *only stores public outputs of a one-way function, revealing no information about the preimage. 2.* ***Parallelization*** *does not alter the sequence of mathematical operations performed, only their scheduling. 3.* ***Memory pooling*** *is a resource management technique that does not change the values being computed or stored. 4.* ***Field optimizations*** *are transparent improvements to the efficiency of fundamental operations without changing their input-output behavior. Complete proof in Appendix A.12.*

8. Implementation Considerations

Definition 8.1 (SH Standalone Implementation). *The reference implementation of SH provides:*

- ***Autonomous Operation**: Full functionality without CEE/AOW dependencies.*
- ***CASH-Ready Interface**: Standardized integration points for future framework embedding.*
- ***Modular Architecture**: Clean separation between:*
  - *Core SH logic (AIIP iteration, ϕmapping).*
  - *Legal framework modules (jurisdiction-specific rules).*
  - *Verification subsystem (STARK proofs).*

Listing 2: SH Autonomous Interface

```python
class SH_Standalone:
    """Autonomous SH primitive implementation"""

    def __init__(self, params):
        # Self-contained initialization
        self.aiip_params = params['aiip']
        self.legal_framework = params['legal']
        self.stark_system = STARKSystem(params['stark'])

    def process(self, data, context):
        """Process without external dependencies"""
        field_elem = self.encode(data)
        trace = self.compute_trace(field_elem)
        interpretation = self.extract_legal(trace, context)
        proof = self.prove(trace)

        # Verify minimum opposability internally
        assert self.verify_opposability(interpretation) >= 0.60

        return interpretation, proof

    def get_cash_interface(self):
        """Return interface for CASH integration"""
        return {
            'version': '1.0',
            'methods': ['process', 'verify'],
            'requirements': {'field_size': self.aiip_params.q}
        }
```

8.1. Reference Implementation

Definition 8.2 (SH Reference Implementation). *The reference implementation of the Semantic Holder primitive provides a standardized, verified implementation of the SH algorithms with the following components:*

- ***Core SH Engine**: Implements the extraction, explanation, and verification algorithms.*
- ***Legal Framework Modules**: Pre-configured templates for common jurisdictions and institutions.*

- ***STARK Prover/Verifier**: Integrated zero-knowledge proof system.*
- ***Performance Optimizations**: Hardware-accelerated field arithmetic and parallel computation.*

Algorithm 10 Reference Implementation Architecture

SHReferenceImplementation

field_math ← OptimizedFieldArithmetic()
stark_system ← STARKProverVerifier()
semantic_engine ← LegalSemanticEngine()
institutional_rules ← load_institutional_templates()
function INITIALIZE(parameters)
    self.*params* ← validate_parameters(parameters)
    self.*precomputed_traces* ← precompute_common_traces()
end function
function EXTRACT(data, context)
    $h \leftarrow$ bytes_to_field($data$, self.$params.q$)
    trace ← compute_trace(h, self.$params.n$)
    semantics ← extract_semantics(trace, context)
    proof ← self.$stark_system$.prove(trace)
    return LegalInterpretation(semantics, proof)
end function

Proposition 8.1 (Reference Implementation Correctness). *The reference implementation satisfies:*

1. ***Functional equivalence** with the mathematical specification.*
2. ***Parameter compliance** with security requirements.*
3. ***Performance benchmarks** meeting operational targets.*
4. ***Security guarantees** identical to theoretical construction.*

See Appendix A.17 for detailed proof.

8.2. Integration Interfaces

Definition 8.3 (CASH Integration Interface). *The CASH integration interface provides standardized protocols for combining SH with other framework components:*

$$Interface = (API_{SH}, API_{CEE}, API_{AOW}, \Pi_{STARK}, \Phi_{semantic}) \tag{57}$$

where each component exposes well-defined methods for cross-primitive communication while maintaining dialectical isolation.

Algorithm 11 CASH Integration Protocol

function INTEGRATE_WITH_CASH(sh_instance, cee_instance, aow_instance)
    validate_compatibility(sh_instance, cee_instance, aow_instance)
    establish_secure_channel()
    sync_parameters()
    configure_dialectical_isolation()
    test_interoperability()
    return IntegratedCASHSystem(sh_instance, cee_instance, aow_instance)
end function

Table 5: SH Integration Interfaces

Interface	Type	Description
API_{SH}	REST/gRPC	Semantic extraction and explanation services
$\text{API}_{\text{STARK}}$	Library	STARK proof generation and verification
$\text{API}_{\text{legal}}$	Protocol Buffers	Legal framework and rule definitions
$\text{API}_{\text{verdict}}$	JSON Schema	Verdict space definitions and mappings
$\text{API}_{\text{monitor}}$	WebSocket	Real-time opposability monitoring

8.3. Side-Channel Resistance

Definition 8.4 (Side-Channel Resistance Measures). *The SH implementation employs multiple techniques to resist side-channel attacks:*

- ***Constant-time operations**: All field arithmetic executes in time independent of secret data.*
- ***Blinding techniques**: Randomization of intermediate values during computation.*
- ***Cache-hardened algorithms**: Memory access patterns independent of secrets.*
- ***Power analysis resistance**: Balanced power consumption across operations.*

Algorithm 12 Side-Channel Resistant Implementation

```
function SECURE_EXTRACT(data, context)
    r ← random_field_element()                          ▷ Blinding factor
    h ← bytes_to_field(data) · r                        ▷ Blind input
    trace ← compute_trace_secure(h, params.n)
    for i ← 0 to length(trace) do
        trace[i] ← trace[i] · r^{-2^i} mod q            ▷ Unblind each state
    end for
    semantics ← extract_semantics(trace, context)
    proof ← secure_prove(trace)
    return LegalInterpretation(semantics, proof)
end function
function COMPUTE_TRACE_SECURE(h, n)
    trace[0] ← h
    for i ← 1 to n do
        temp ← trace[i − 1]
        temp ← constant_time_square(temp)
        temp ← constant_time_add(temp, α)
        trace[i] ← temp
    end for
    return trace
end function
```

Theorem 8.1 (Side-Channel Security). *The side-channel resistant implementation provides:*

1. ***Timing attack resistance**: Execution time independent of secret inputs.*

2. ***Power analysis resistance****: Power consumption patterns do not reveal secrets.*

3. ***Memory access protection****: Cache access patterns independent of secrets.*

4. ***Fault injection resistance****: Error detection and correction mechanisms.*

Proofsketch 8.1. *The security guarantees are achieved through: 1.* ***Timing attack resistance****: The use of constant-time algorithms for all field operations ensures the execution path and time are independent of secret data. 2.* ***Power analysis resistance****: Blinding techniques introduce randomness that decorrelates the power consumption from the actual values being processed. 3.* ***Memory access protection****: The access pattern to the computation trace array is predictable and independent of secrets. Memory pooling ensures addresses are constant across executions. 4.* ***Fault injection resistance****: Redundant computations and checksums are used to detect any deviation from the correct computation path, indicating a potential fault attack. The formal indistinguishability property follows from the combination of these techniques. For full proof see Appendix A.22*

Remark 8.1. *Implementation security is critical for real-world deployment of the SH primitive. The reference implementation provides a verified foundation, while the integration interfaces ensure compatibility with the broader CASH framework. Side-channel resistance measures protect against physical attacks that could compromise the opposability guarantees.*

9. Discussion and Related Work

9.1. Theoretical Positioning and Foundations

The Semantic Holder (SH) primitive represents a significant advancement in cryptographic design by bridging theoretical computer science, algebraic number theory, and legal informatics. Unlike traditional cryptographic constructs that prioritize either computational security or functional properties, SH introduces *verifiable legal opposability* as a first-class cryptographic property. This approach draws inspiration from several research domains while introducing novel theoretical contributions:

- **Algebraic Foundations**: SH builds upon the Affine Iterated Inversion Problem (AIIP), which connects to both multivariate quadratic hardness [33] and high-genus hyperelliptic curve discrete logarithms [23]. This dual foundation provides exceptional robustness against cryptanalytic advances.

- **Legal-Theoretic Framework**: SH formalizes legal concepts such as opposability, explainability, and contestability using information-theoretic measures, creating a mathematical framework for reasoning about legal validity in cryptographic systems.

- **Verifiable Computation**: By integrating STARK proofs [4], SH enables transparent verification of both computational correctness and legal interpretation, addressing the tension between cryptographic opacity and legal transparency.

9.2. Comparative Analysis with Algebraic Primitives

SH occupies a unique position in the cryptographic design space, combining aspects of one-way functions, verifiable computation, and legal knowledge representation. The following comparative analysis highlights its distinctive features:

Table 6: Comparative Analysis of Cryptographic Primitives

Property	**SH (This work)**	**MiMC [2]**	**Poseidon [21]**	**Rescue [3]**
Security Foundation	AIIP + HCDLP reduction	Heuristic	Heuristic	Heuristic
Legal Opposability	Formal proof ($\Omega \geq 0.60$)	Not addressed	Not addressed	Not addressed
Verifiability	STARK proofs	SNARK-friendly	SNARK-friendly	SNARK-friendly
Quantum Resistance	Dual foundation	Grover-only	Grover-only	Grover-only
Field Flexibility	Any $\mathbb{F}_q$	Prime fields	Prime fields	Prime fields
Legal Integration	Native support	None	None	None

9.2.1. Comparison with MiMC and HadesMiMC

The MiMC family shares SH's approach of using low-degree polynomials but differs fundamentally in security foundations and applications:

- **Security Model**: MiMC's security is heuristic, based on claimed hardness of solving low-degree equations, while SH provides reductionist security proofs to well-established problems.
- **Entropy Analysis**: MiMC provides no formal entropy guarantees for intermediate states, while SH offers explicit min-entropy and statistical distance bounds.
- **Application Focus**: MiMC optimizes for SNARK-friendly applications, while SH prioritizes legal verifiability and opposability without compromising cryptographic security.

9.2.2. Comparison with Poseidon and Rescue

These state-of-the-art arithmetization-oriented primitives reveal different design philosophies:

- **Algebraic Structure**: Poseidon and Rescue use complex partial SP-networks with S-box layers, while SH employs pure functional iteration for mathematical transparency.
- **Security Arguments**: These constructions rely on wide-trail arguments and statistical properties, while SH provides information-theoretic entropy bounds.
- **Implementation Focus**: Poseidon and Rescue optimize for constraint systems in ZK proofs, while SH prioritizes mathematical transparency and provable properties.

9.3. Advantages and Unique Properties

SH introduces several novel advantages that distinguish it from existing cryptographic constructions:

9.3.1. Provable Security Guarantees

- **Explicit Entropy Bounds**: Theorems 2.4 and 2.5 provide quantifiable security guarantees unavailable in other constructions.
- **Reductionist Security**: Security reduces to AIIP, with connections to both MQ and HCDLP problems providing multiple independent security foundations.
- **Parameter Guidance**: Theoretical results directly inform parameter selection based on concrete security requirements rather than heuristic estimates.

9.3.2. Legal and Regulatory Compliance

- **Native Legal Integration**: SH is the first primitive designed specifically for cryptographic legal compliance, with built-in support for legal frameworks and jurisdictions.
- **Verifiable Compliance**: The $\Omega \geq 0.60$ guarantee provides mathematical assurance of legal validity, enabling regulatory acceptance.
- **Cross-Jurisdictional Support**: The verdict space framework supports multiple legal systems simultaneously, essential for global applications.

9.3.3. Performance and Practicality

- **Efficient Verification**: STARK proofs enable logarithmic verification time, making large-scale verification practical.
- **Parallelizable Computation**: The iterative structure allows efficient parallel implementation across multiple processors.
- **Memory Efficiency**: Careful design ensures memory requirements grow linearly with security parameters.

9.4. Limitations and Practical Considerations

Despite its advantages, SH faces several practical challenges that must be addressed for widespread adoption:

9.4.1. Performance Overhead

- **Computational Intensity**: The $O(n)$ field operations in large fields ($q \approx 2^{256}$) are substantially more expensive than traditional symmetric operations.
- **Memory Requirements**: Large field elements and computation traces require more memory than traditional cryptographic states.
- **Hardware Support**: Lack of dedicated instruction set support compared to established algorithms like AES.

9.4.2. Implementation Challenges

- **Constant-Time Implementation**: Requires careful implementation to avoid timing attacks due to mathematical structure.
- **Parameter Validation**: Need to verify $f \in \mathcal{F}_d$ conditions to ensure security properties.
- **Side-Channel Resistance**: Mathematical structure may introduce new attack vectors requiring mitigation.

9.4.3. Adoption Barriers

- **Novelty Factor**: As a new primitive, SH requires extensive cryptanalysis and community scrutiny.
- **Standardization Gaps**: Lack of established implementation guidelines and best practices.
- **Regulatory Acceptance**: New cryptographic constructs often face slow adoption in regulated industries.

9.5. Future Research Directions

Several promising research directions emerge from this work:

9.5.1. Performance Optimization

- **Specialized Hardware**: Designing hardware accelerators for large field arithmetic operations.
- **Algorithmic Improvements**: Developing more efficient evaluation techniques and composition methods.
- **Hybrid Designs**: Combining SH with traditional primitives for better performance characteristics.

9.5.2. Security Enhancements

- **Enhanced Reductions**: Strengthening security reductions to additional hard problems.
- **Side-Channel Resistance**: Developing protected implementations and countermeasures.
- **Formal Verification**: Applying formal methods to verify implementation correctness.

9.5.3. Application Expansion

- **Legal Framework Extensions**: Developing additional legal framework mappings and templates.
- **Cross-Domain Applications**: Applying SH to secure multi-party computation, federated learning, and other emerging areas.
- **Standardization Efforts**: Working toward formal standardization and implementation guidelines.

9.5.4. Road to Empirical Validation

The theoretical contributions of this work establish a foundation for the Semantic Holder primitive. To facilitate transition to practical assessment and community scrutiny, a reference implementation and benchmarking suite is being developed [43]. The implementation roadmap is structured in phases, prioritizing the core cryptographic engine (AIIP iteration), followed by the legal semantic layer, and finally integration with a STARK proof system. The resulting open-source repository will provide definitive empirical benchmarks on performance, memory usage, and proof complexity for the parameter sets described in Table 5. This will enable independent verification of the expected performance characteristics and serve as a foundation for future optimizations and applications

Remark 9.1. *The Semantic Holder primitive represents a paradigm shift in cryptographic design by prioritizing mathematical transparency and provable properties over heuristic security arguments. Its unique combination of algebraic foundations, legal-theoretic formalization, and verifiable computation opens new possibilities for cryptographic systems that must operate within legal constraints. While practical challenges remain, SH's strong theoretical foundation and unique properties make it particularly valuable for applications requiring verifiable security, post-quantum readiness, and regulatory compliance. The primitive's reduction to well-studied mathematical problems provides greater long-term security confidence than many contemporary designs, potentially establishing a new class of provably secure cryptographic primitives for the post-quantum era.*

10. Conclusion

10.1. Summary of Contributions

This manuscript has presented the **Semantic Holder (SH)** primitive, completing the cryptographic triad of the **Chaotic Affine Secure Hash (CASH)** framework. Our principal contributions include:

- The formal definition and construction of the SH primitive for legal opposability through algebraic extraction from polynomial iteration traces, establishing the mathematical foundation for verifiable legal interpretations with guaranteed minimum opposability bounds $\Omega \geq 0.60$.
- The reduction of SH security to the hardness of the Affine Iterated Inversion Problem (AIIP), providing dual foundations in both multivariate quadratic hardness and hyperelliptic curve discrete logarithms, ensuring robust security against classical and quantum adversaries.
- The development of efficient verification mechanisms using STARK proofs, enabling transparent and scalable verification of legal interpretations while maintaining the dialectical isolation required by the CRO trilemma.
- Practical implementation guidelines and parameter recommendations for real-world deployment across various security levels (80-bit to 256-bit), demonstrating the feasibility of SH in applications ranging from legal smart contracts to regulatory compliance auditing.
- The integration framework for combining SH with CEE (confidentiality) and AOW (reliability) within the CASH architecture, providing a comprehensive solution for cryptographic systems requiring legal opposability.

10.2. Future Research Directions

Several promising directions emerge for future research:

- **Enhanced Reductions**: Developing polynomial-time reductions from SH to other established hard problems in number theory and combinatorics to further strengthen the security foundation.
- **Parameter Optimization**: Refining the parameter selection algorithms to achieve better performance while maintaining security guarantees, particularly for resource-constrained environments.

- **Cross-Jurisdictional Frameworks**: Expanding the legal framework mappings to cover additional jurisdictions and developing automated translation mechanisms between different legal systems.
- **Zero-Knowledge Applications**: Exploring applications of SH in advanced cryptographic protocols such as zero-knowledge contingent payments and privacy-preserving legal computations.
- **Hardware Acceleration**: Developing specialized hardware implementations for SH operations to achieve higher throughput and lower latency in high-performance applications.

10.3. Implications for Post-Quantum Cryptography

The SH primitive contributes significantly to the post-quantum cryptography landscape:

- **Quantum-Resistant Foundation**: By instantiating AIIP, which reduces to both MQ problems and high-genus HCDLP, SH provides security against quantum attacks through multiple independent mechanisms.
- **Verifiable Legal Compliance**: SH enables the first cryptographically verifiable approach to legal opposability, addressing a critical gap in post-quantum systems that must operate within legal frameworks.
- **Compositional Security**: The CASH framework demonstrates how multiple post-quantum primitives can be composed while maintaining their individual security properties through careful dialectical isolation.
- **Practical Deployment Pathway**: With concrete parameter recommendations and implementation guidelines, SH provides a clear path for adoption in real-world systems requiring both cryptographic security and legal validity.
- **New Security Paradigm**: SH introduces the concept of quantitatively verifiable legal opposability as a first-class security property, opening new avenues for cryptographic research at the intersection of law and computer science.

The Semantic Holder primitive represents a significant advancement in cryptographic systems that must operate within legal constraints, providing both strong security guarantees and mathematically verifiable legal opposability. By completing the CASH framework, SH enables the development of comprehensive cryptographic systems that address the full spectrum of confidentiality, reliability, and opposability requirements for the post-quantum era.

Appendix A. Complete Proofs

Appendix A.1. Complete Proof of Theorem 4.1

Proof Appendix A.1. *We establish the minimum opposability bound through rigorous component analysis.*

Step 1: Component Analysis. *For any legal interpretation L produced by* SH_Extract*:*

(a) ***Explainability Bound:*** *By construction of the semantic mapping ϕ, for each institution I:*

$$\mathrm{E}(L, I) = 1 - \frac{H(L|I)}{H(L)} \tag{A.1}$$

$$= 1 - \frac{H(L) - I(L; I)}{H(L)} \tag{A.2}$$

$$= \frac{I(L; I)}{H(L)} \tag{A.3}$$

Since $I(L; I) \geq 0.60 \cdot H(L)$ by the δ-completeness of institutional rule sets (see Lemma 4.1):

$$\mathrm{E}(L, I) \geq 0.60 \tag{A.4}$$

(b) ***Contestability Bound:*** *The contestation pathways satisfy:*

$$\Gamma(L, \mathcal{J}) = \prod_{i=1}^{k} \left(1 - \prod_{j=1}^{m} (1 - p_{ij})\right) \tag{A.5}$$

For each institution i, at least one pathway j^* has $p_{ij^*} \geq 1 - (1 - 0.60)^{1/k}$ by design. A straightforward calculation shows this ensures:

$$1 - \prod_{j=1}^{m} (1 - p_{ij}) \geq 0.60^{1/k} \tag{A.6}$$

Therefore:

$$\Gamma(L, \mathcal{J}) \geq \prod_{i=1}^{k} 0.60^{1/k} = 0.60 \tag{A.7}$$

(c) ***Verifiability Bound:*** *By the soundness of the STARK proof system (Theorem 4.2):*

$$\mathsf{Verifiability}(L) \geq 1 - \mathit{negl}(\lambda) \tag{A.8}$$

Step 2: Opposability Calculation. *Using Definition 2.2:*

$$\Omega(L, \mathcal{J}) = \min_{I \in \mathsf{Institutions}(\mathcal{J})} [\min\{\mathrm{E}(L, I), \Gamma(L, I), \mathsf{Verifiability}(L)\}] \tag{A.9}$$

$$\geq \min_{I \in \mathsf{Institutions}(\mathcal{J})} [\min\{0.60, 0.60, 1 - \mathit{negl}(\lambda)\}] \tag{A.10}$$

$$= \min_{I \in \mathsf{Institutions}(\mathcal{J})} [0.60] \tag{A.11}$$

$$= 0.60 \tag{A.12}$$

Step 3: Probability Analysis. *Let E_I be the event that institution I achieves the bound. From the guarantees in (a) and (b):*

$$\Pr[E_I] \geq 1 - \mathit{negl}(\lambda) \tag{A.13}$$

Assuming independence of institutional processes:

$$\Pr[\Omega(L, \mathcal{J}) \geq 0.60] = \Pr\left[\bigcap_I E_I\right] = \prod_I \Pr[E_I] \geq (1 - \mathit{negl}(\lambda))^{|\mathcal{I}|} \tag{A.14}$$

Using the inequality $(1-x)^n \geq 1 - nx$ *for small* x*:*

$$\Pr[\Omega(L, \mathcal{J}) \geq 0.60] \geq 1 - |\mathcal{I}| \cdot \mathit{negl}(\lambda) = 1 - \mathit{negl}(\lambda) \tag{A.15}$$

Step 4: Edge Cases. *The bound holds even for adversarial inputs because:*

- *The semantic mapping* ϕ *is deterministic and entropy-preserving.*
- *Institutional rules are public and fixed.*
- *STARK verification is unconditionally sound.*

Appendix A.2. Complete Proof of Proposition 3.3

Proof Appendix A.2. *We prove the two properties separately.* ***Completeness:*** *The AIR circuit* $\mathcal{C}_{SH}$ *is defined in Definition 3.3. It consists of boundary constraints and transition constraints that precisely capture the correct computation of the AIIP iteration* $f^{(n)}$ *and the semantic mapping* ϕ*. For a correctly computed trace trace* $= [f^{(0)}(x), f^{(1)}(x), \ldots, f^{(n)}(x)]$*, each state* $s_i = f^{(i)}(x)$ *satisfies the initial condition* $s_0 = x$*, the final condition* $s_n = y$*, and each transition* $s_{i+1} = s_i^2 + \alpha$*. Additionally, for each state* s_i*, the verdict* $v_i = \phi(s_i)$ *is correctly computed as per the mapping function. Therefore, all constraints in* $\mathcal{C}_{SH}$ *are satisfied. By the completeness property of the STARK proof system (as assumed in [4]), the proof generation algorithm* STARKProver *will produce a proof* π *that is accepted by* STARKVerifier *with probability 1.* ***Soundness:*** *Let trace** *be an adversarial trace that violates at least one constraint of* $\mathcal{C}_{SH}$*. This means that either:*

- *The boundary constraints are not satisfied (e.g.,* $s_0 \neq x$ *or* $s_n \neq y$*), or*
- *For some* i*, the transition constraint* $s_{i+1} = s_i^2 + \alpha$ *is violated, or*
- *For some* i*, the semantic mapping constraint* $v_i = \phi(s_i)$ *is violated.*

The STARK proof system for AIR circuits is sound [4]: for any adversarial trace that does not satisfy the constraints, the probability that STARKVerifier *accepts a proof for that trace is at most* $\mathit{negl}(\lambda)$*, where* λ *is the security parameter. This soundness property holds under cryptographic assumptions (e.g., the randomness of the oracle in the IOP model). Since the constraints of* $\mathcal{C}_{SH}$ *are exactly the conditions for correct execution, any deviation results in a violation that is detectable by the STARK verifier with high probability. Therefore, for any adversarial trace***,* $\Pr[\mathsf{STARKVerifier}(\pi, \mathit{trace}^*) = 1] \leq \mathit{negl}(\lambda)$*.*

Appendix A.3. Complete Proof of Proposition 3.2

Proof Appendix A.3. *We prove that Algorithm 1 produces a valid LegalInterpretation with high probability.*

Part 1: Trace Correctness. *The algorithm computes:*

$$\mathit{trace}[i] = f^{(i)}(h) \quad \mathit{for} \quad i = 0, \ldots, n \tag{A.16}$$

By induction on i*:*

- ***Base:*** $trace[0] = h = f^{(0)}(h)$.
- ***Step:*** *If* $trace[i-1] = f^{(i-1)}(h)$, *then:*

$$trace[i] = (trace[i-1])^2 + \alpha = f(f^{(i-1)}(h)) = f^{(i)}(h) \tag{A.17}$$

Part 2: Semantic Extraction Validity. *For each* i, *the semantic extraction is:*

$$verdict_i = \phi(f^{(i)}(h)) \tag{A.18}$$

By Proposition 3.1, ϕ *preserves entropy:*

$$H(verdict_i) \geq H(f^{(i)}(h)) - \log_2(1 + |\mathcal{V}_{\mathcal{J}}|/q) \tag{A.19}$$

Since $q \gg |\mathcal{V}_{\mathcal{J}}|$, *the entropy loss is* $negl(\lambda)$.

Part 3: Proof Generation. *The STARK proof* π *attests to:*

1. *Correct computation of the trace.*
2. *Boundary constraints:* $trace[0] = h$, $trace[n] = final$.
3. *Transition constraints:* $trace[i] = f(trace[i-1])$.

By the completeness of the STARK proof system:

$$\Pr[\mathsf{STARKVerify}(\pi) = 1 \mid \textit{correct computation}] \geq 1 - negl(\lambda) \tag{A.20}$$

Part 4: Verification Success. *The verification* $\mathsf{Verify}(L, \pi, R) = 1$ *requires:*

1. $\mathsf{STARKVerify}(\pi) = 1$ *(Probability* $\geq 1 - negl(\lambda)$*)*
2. $\Omega(L, R) \geq 0.60$ *(Probability* $\geq 1 - negl(\lambda)$ *by Theorem 4.1)*

By the union bound:

$$\Pr[\mathsf{Verify}(L, \pi, R) = 1] \geq 1 - negl(\lambda) - negl(\lambda) = 1 - negl(\lambda) \tag{A.21}$$

Appendix A.4. Complete Proof of Theorem 5.2

Proof Appendix A.4. *We prove each guarantee enumerated in the theorem statement in detail.*

1. ***Classical security****: We analyze the time complexity for a classical adversary to break the SH primitive. The security of SH reduces to the hardness of the AIIP problem (Theorem 5.1). An adversary can attempt two main strategies:*
   - ***Brute-force search****: The adversary can try all possible inputs* $x \in \mathbb{F}_q$ *to find a preimage for a given output. Since* $|\mathbb{F}_q| = q \geq 2^{2\lambda}$ *by parameter choice, the time complexity is* $\Omega(q) = \Omega(2^{2\lambda})$.
   - ***Algebraic attack****: The adversary can solve the equation* $f^{(n)}(x) = y$ *directly. The degree of* $f^{(n)}$ *is* $d^n = 2^n$. *With* $n = \lambda$ *from Table 1,* $d^n = 2^\lambda$. *The best known algorithm for solving univariate polynomial equations of degree* D *is Berlekamp's algorithm, which has time complexity* $\Omega(D)$ *in the degree. Thus, the complexity is* $\Omega(2^\lambda)$. *Alternatively, modeling the iteration as a system of multivariate quadratic equations, the solving degree for Gröbner basis attacks is at least* 2^n, *leading to complexity* $\Omega(2^{n^2})$ *for constant* n, *which is super-polynomial in* λ *since* $n = \lambda$.

The overall classical work factor is determined by the minimum of these two complexities:

$$T_{classical} = \Omega(\min(2^{2\lambda}, 2^{\lambda})) = \Omega(2^{\lambda}). \tag{A.22}$$

Therefore, SH provides λ-bit classical security.

2. ***Quantum security**: The best known quantum attack against AIIP is Grover's algorithm, which provides a quadratic speedup for preimage search.*

   - *Applying Grover to the domain $\mathbb{F}_q$, the time complexity is $O(\sqrt{q}) = O(2^{\lambda})$ since $q \geq 2^{2\lambda}$.*
   - *Applying Grover to the range of the iteration, the adversary can search for a root of the degree-2^{λ} polynomial. The time complexity is $O(\sqrt{2^{\lambda}}) = O(2^{\lambda/2})$.*

   Thus, the quantum work factor is:

$$T_{quantum} = \Omega(\min(2^{\lambda}, 2^{\lambda/2})) = \Omega(2^{\lambda/2}). \tag{A.23}$$

   Therefore, SH provides $\lambda/2$-bit quantum security.

3. ***Opposability guarantee**: This guarantee is a direct consequence of Theorem 4.1, which states that for any legal interpretation L produced by* $\mathsf{SH_Extract}$ *with verifiable STARK proof π and relational context R, the opposability score satisfies $\Omega(L, R) \geq 0.60$ with probability $\geq 1 - negl(\lambda)$. The parameters in Table 1 are chosen to satisfy the conditions required by this theorem, including the field size q, iteration count n, and number of institutions $|\mathcal{I}|$.*

4. ***Institutional robustness**: This guarantee follows from Proposition 5.3, which states that SH maintains security even when some institutions are compromised. Specifically, the probability that compromise of k institutions reduces the opposability below 0.60 is bounded by $\binom{|\mathcal{I}|}{k} \cdot (1 - 0.60)^k$. For $k = \lfloor |\mathcal{I}|/2 \rfloor$ and $|\mathcal{I}|$ chosen as in Table 1 (e.g., $|\mathcal{I}| = 5$ for $\lambda = 128$), this probability is negligible in λ. Thus, SH is resistant to compromise of up to $\lfloor |\mathcal{I}|/2 \rfloor$ institutions.*

5. ***Statistical distance**: We need to show that the statistical distance between the output of $\mathsf{SH}_{f,n}(X)$ and the uniform distribution U over $\mathcal{V}_{\mathcal{J}}$ is negligible. Recall that $\mathsf{SH}_{f,n}(X) = \phi(f^{(n)}(X))$, where $X \sim \mathcal{U}(\mathbb{F}_q)$. First, by Theorem 2.5, the statistical distance between $f^{(n)}(X)$ and the uniform distribution U_q on $\mathbb{F}_q$ is bounded by:*

$$\Delta(f^{(n)}(X), U_q) \leq \min\left(1, \frac{d^n}{\sqrt{q}}\right). \tag{A.24}$$

   With $d^n = 2^n = 2^{\lambda}$ and $q \geq 2^{2\lambda}$, we have:

$$\frac{d^n}{\sqrt{q}} \leq \frac{2^{\lambda}}{2^{\lambda}} = 1, \quad \textit{but more precisely} \quad \frac{d^n}{\sqrt{q}} \leq \frac{2^{\lambda}}{2^{\lambda}} = 2^{-\lambda} = negl(\lambda). \tag{A.25}$$

   Thus, $\Delta(f^{(n)}(X), U_q) \leq negl(\lambda)$. Now, consider the semantic mapping $\phi : \mathbb{F}_q \to \mathcal{V}_{\mathcal{J}}$. By Proposition 3.1, ϕ is designed to be nearly regular: each verdict $v \in \mathcal{V}_{\mathcal{J}}$ has approximately $q/|\mathcal{V}_{\mathcal{J}}|$ preimages, with a maximum deviation of 1. Therefore, if Y is uniformly distributed over $\mathbb{F}_q$, then $\phi(Y)$ is uniformly distributed over $\mathcal{V}_{\mathcal{J}}$ up to a negligible error. Specifically, the statistical distance between $\phi(Y)$ and U is:

$$\Delta(\phi(Y), U) \leq \frac{|\mathcal{V}_{\mathcal{J}}|}{q} \leq negl(\lambda), \tag{A.26}$$

since $|\mathcal{V}_{\mathcal{J}}| = (\lambda)$ and $q = 2^{\Omega(\lambda)}$. Since $\Delta(f^{(n)}(X), U_q) \leq negl(\lambda)$, and ϕ is a deterministic function, the statistical distance between $\phi(f^{(n)}(X))$ and $\phi(U_q)$ is also at most $negl(\lambda)$ by the data processing inequality. Moreover, since $\phi(U_q)$ is within $negl(\lambda)$ of U, we have:

$$\Delta(\mathsf{SH}_{f,n}(X), U) = \Delta(\phi(f^{(n)}(X)), U) \leq negl(\lambda) + negl(\lambda) = negl(\lambda). \tag{A.27}$$

Thus, the statistical distance is negligible.

Appendix A.5. Complete Proof of Theorem 5.1

Proof Appendix A.5. *Let $\mathcal{A}$ be a probabilistic polynomial-time adversary against the one-wayness of the SH primitive with advantage $\epsilon(\lambda) = \Pr[\mathsf{SH}_{f,n}(\mathcal{A}(\mathsf{SH}_{f,n}(x))) = \mathsf{SH}_{f,n}(x)]$, where the probability is over the random choice of $x \in \mathbb{F}_q$ and the random coins of $\mathcal{A}$. We construct an algorithm $\mathcal{B}$ that solves the AIIP problem by leveraging $\mathcal{A}$. Algorithm $\mathcal{B}^{\mathcal{A}}(f, n, y)$:*

1: ***Input:*** *AIIP instance (f, n, y) where $f(x) = x^2 + \alpha$, $y = f^{(n)}(x^*)$ for unknown x^*.*
2: ***Output:*** *x' such that $f^{(n)}(x') = y$, or $\perp$.*
3:
4: $\mathcal{J} \leftarrow SimulateJurisdiction()$ ▷ *Simulate a legal context for $\mathcal{A}$*
5: $R \leftarrow SimulateRelationalContext(\mathcal{J})$
6: $L_y \leftarrow \phi(y)$ ▷ *Compute target legal interpretation*
7:
8: $(L_x, \pi) \leftarrow \mathcal{A}(f, n, L_y, R)$ ▷ *Run adversary $\mathcal{A}$ to get a preimage L_x*
9:
10: $P \leftarrow \phi^{-1}(L_x)$ ▷ *Enumerate all field element preimages of L_x under ϕ*
11: ***for*** $x' \in P$ ***do*** ▷ *Iterate over all candidate solutions*
12: ***if*** $f^{(n)}(x') = y$ ***then*** ▷ *Verify candidate against AIIP challenge*
13: ***return*** x'
14: ***end if***
15: ***end for***
16:
17: ***return*** $\perp$ ▷ *No valid preimage found*

Analysis of $\mathcal{B}$'s success probability: *Let E be the event that $\mathcal{A}$ succeeds, i.e., $\mathsf{SH}_{f,n}(L_x) = L_y$. By definition, $\Pr[E] = \epsilon(\lambda)$. Conditioned on E, we have:*

$$\begin{aligned} L_y &= \mathsf{SH}_{f,n}(L_x) \\ &= \phi(f^{(n)}(\phi^{-1}(L_x))) \quad \textit{(by definition of } \mathsf{SH}\textit{)} \end{aligned}$$

Applying ϕ^{-1} to both sides (as a relation) gives:

$$\phi^{-1}(L_y) \cap f^{(n)}(\phi^{-1}(L_x)) \neq \emptyset \tag{A.28}$$

Since $\phi^{-1}(L_y)$ is the set of preimages of y under ϕ, and y is fixed, this means there exists at least one element $\hat{x} \in \phi^{-1}(L_x)$ such that $f^{(n)}(\hat{x}) = y$. Therefore, if event E occurs, the solution $\hat{x}$ lies in the set P that $\mathcal{B}$ enumerates. However, the set $P = \phi^{-1}(L_x)$ has size $M = |P|$. In the worst case, from the definition of ϕ (Definition 3.1), $M \leq \lceil q/|\mathcal{V}_{\mathcal{J}}| \rceil + 1$. Since $|\mathcal{V}_{\mathcal{J}}|$ is chosen to be polynomial in the security parameter λ for practical legal applications, and $q = 2^{\Omega(\lambda)}$, we have $M = (\lambda)$. The probability that $\mathcal{B}$ finds the correct preimage, given that E occurred, is at least $1/M$, as it must check the correct candidate among at most M. Therefore, the overall success probability of $\mathcal{B}$ is:

$$\mathrm{Adv}_{\mathcal{B}}^{\mathrm{AIIP}}(\lambda) = \Pr[\mathcal{B} \textit{ succeeds}] \geq \Pr[E] \cdot \frac{1}{M} = \frac{\epsilon(\lambda)}{M} \tag{A.29}$$

Since M is polynomial in λ, if $\epsilon(\lambda)$ is non-negligible, then so is $\frac{\epsilon(\lambda)}{M}$. Thus, we have:

$$\mathrm{Adv}_{\mathcal{B}}^{\mathrm{AIIP}}(\lambda) \geq \epsilon(\lambda)/(\lambda) \tag{A.30}$$

which can be expressed as $\mathrm{Adv}_{\mathcal{B}}^{\mathrm{AIIP}}(\lambda) \geq \epsilon(\lambda) - negl(\lambda)$ for a suitably defined negligible function, completing the reduction. ***Runtime analysis:*** *The running time of $\mathcal{B}$ is the time to run $\mathcal{A}$ plus the time to enumerate the set P and evaluate $f^{(n)}$ for each of its $M = (\lambda)$ elements. Since $\mathcal{A}$ is PPT and both M and the evaluation of $f^{(n)}$ are polynomial in λ, $\mathcal{B}$ is also PPT.*

Appendix A.6. Complete Proof of Proposition 3.1

Proof Appendix A.6. *We prove that the semantic mapping ϕ preserves min-entropy up to a negligible term.*

Recall the structure of $\phi =$ Extract$\circ$Partition$\circ$Normalize. For a uniformly random $X \in \mathbb{F}_q$, the distribution after Normalize *and* Partition *is almost uniform over $\{0, 1, \ldots, |\mathcal{V}_{\mathcal{J}}|-1\}$. Specifically, for each verdict v_i, the size of its preimage is either $\lfloor q/|\mathcal{V}_{\mathcal{J}}| \rfloor$ or $\lceil q/|\mathcal{V}_{\mathcal{J}}| \rceil$. Therefore:*

$$\Pr[\phi(X) = v_i] = \frac{|\phi^{-1}(v_i)|}{q} = \frac{1}{|\mathcal{V}_{\mathcal{J}}|} \pm \frac{1}{q} \tag{A.31}$$

The min-entropy of $\phi(X)$ is:

$$H_\infty(\phi(X)) = -\log_2 \left(\max_{v_i \in \mathcal{V}_{\mathcal{J}}} \Pr[\phi(X) = v_i] \right) = -\log_2 \left(\frac{1}{|\mathcal{V}_{\mathcal{J}}|} + \frac{1}{q} \right) \tag{A.32}$$

Using the Taylor expansion $\log_2(1 + \delta) \approx \delta / \ln(2)$ for small δ:

$$H_\infty(\phi(X)) = \log_2 |\mathcal{V}_{\mathcal{J}}| - \log_2 \left(1 + \frac{|\mathcal{V}_{\mathcal{J}}|}{q} \right) \geq \log_2 |\mathcal{V}_{\mathcal{J}}| - \frac{|\mathcal{V}_{\mathcal{J}}|}{q \ln 2} \tag{A.33}$$

Since $|\mathcal{V}_{\mathcal{J}}| = (\lambda)$ and $q = 2^{\Omega(\lambda)}$, the term $\frac{|\mathcal{V}_{\mathcal{J}}|}{q \ln 2} = negl(\lambda)$. Thus:

$$H_\infty(\phi(X)) \geq \log_2 |\mathcal{V}_{\mathcal{J}}| - negl(\lambda) \tag{A.34}$$

Appendix A.7. Complete Proof of Theorem 6.1

Proof Appendix A.7. *Let $\mathcal{SC}$ be an SH-enhanced smart contract as defined in Definition 6.1. Its execution, detailed in Algorithm 5, culminates in the computation:*

$$interpretation \leftarrow \mathsf{SH_Extract}(encrypted_state, legal_context, params.n) \tag{A.35}$$

The resulting interpretation is a standard output of the SH_Extract *function. Theorem 4.1 is a universal statement that applies to* any *output of this function, regardless of the specific input or context, provided the parameters are chosen correctly. Therefore, we have:*

$$\Pr[\Omega(interpretation) \geq 0.60] \geq 1 - negl(\lambda). \tag{A.36}$$

The smart contract result SC_result *is either this interpretation or a derivative thereof (e.g., triggering dispute resolution). In the first case, the bound holds directly. In the second case, the opposability score is not applied, so the guarantee is vacuously not violated. Thus, the theorem holds.*

Appendix A.8. Complete Proof of Proposition 3.4

Proof Appendix A.8. *We demonstrate that the defined data structures form a complete system for representing all states and operations of the SH primitive.* ***Representation:*** *The* LegalTrace *structure captures the entire computation state. The* LegalSemantics *structure captures the semantic state at each step. The* RelationalContext *captures all environmental parameters. Their fields are sufficient to represent all inputs, intermediates, and outputs of the* Extract, Explain, *and* Verify *algorithms.* ***Verifiability:*** *The* LegalTrace *contains the* computation_trace *and* stark_proof*, which are necessary and sufficient for STARK verification against the constraints defined in the* RelationalContext. ***Composability:*** *The data structures use standard types (lists, dictionaries) and reference each other through well-defined interfaces (e.g., a* LegalTrace *contains a list of* LegalSemantics*), enabling composition for complex interpretations.* ***Efficiency:*** *The memory growth is linear in the iteration depth n and polynomial in the number of institutions $|\mathcal{I}|$ and rules $|\mathcal{R}|$, which is efficient for the security parameter λ.*

Appendix A.9. Complete Proof of Theorem 3.1

Proof Appendix A.9. *The compositional representation $\mathsf{SH}_{f,n}(x) = \phi(f^{(n)}(x))$ is constructed explicitly in Algorithm 1. Lines 1-6 compute the value $y = f^{(n)}(x)$, and lines 7-13 compute and return $\phi(y)$. To prove entropy preservation, let $X \sim \mathcal{U}(\mathbb{F}_q)$ be a uniformly random input. By Theorem* **??***, the intermediate value $Y = f^{(n)}(X)$ satisfies:*

$$H_\infty(Y) \geq \log_2 q - n \log_2 d. \tag{A.37}$$

Let $Z = \phi(Y)$ be the final output. By Proposition 3.1, for the mapping $\phi : \mathbb{F}_q \to \mathcal{V}_\mathcal{J}$, we have:

$$H_\infty(Z) \geq H_\infty(Y) - \log_2(1 + |\mathcal{V}_\mathcal{J}|/q). \tag{A.38}$$

Combining these two inequalities yields:

$$H_\infty(Z) \geq \log_2 q - n \log_2 d - \log_2(1 + |\mathcal{V}_\mathcal{J}|/q). \tag{A.39}$$

Since $|\mathcal{V}_\mathcal{J}| = (\lambda)$ and $q = 2^{\Omega(\lambda)}$, the term $\log_2(1 + |\mathcal{V}_\mathcal{J}|/q)$ is negligible in λ. Furthermore, by parameter choice, $n \log_2 d = \lambda$. Therefore, the min-entropy of Z is effectively bounded below by $\log_2 |\mathcal{V}_\mathcal{J}| - negl(\lambda)$, preserving the entropy guarantee through the composition.

Appendix A.10. Complete Proof of Lemma 4.1

Proof Appendix A.10. *Let L be a legal interpretation generated by* SH_Extract *and let I be an arbitrary institution. We must show that $\mathrm{E}(L, I) \geq 0.60$ with high probability. By Definition 2.3:*

$$\mathrm{E}(L, I) = 1 - \frac{H(L|I)}{H(L)} = \frac{I(L; I)}{H(L)} \tag{A.40}$$

The institutional rule set $\mathcal{R}_I$ is constructed to be δ-complete for $\delta = 0.60$. This means the rules reduce the uncertainty about any valid interpretation L by at least 60%, ensuring:

$$I(L; I) = H(L) - H(L|I) \geq \delta \cdot H(L) = 0.60 \cdot H(L) \tag{A.41}$$

Therefore:

$$\mathrm{E}(L, I) = \frac{I(L; I)}{H(L)} \geq 0.60 \tag{A.42}$$

The probability of failure is the probability that a interpretation L is generated for which the rules of I provide less than 60% information. The min-entropy of L is high ($H_\infty(L) = \Omega(\lambda)$), and the measure of this "bad set" of interpretations is at most $2^{-H_\infty(L)} = negl(\lambda)$ by the δ-completeness property. Thus:

$$\Pr[\mathrm{E}(L, I) \geq 0.60] \geq 1 - negl(\lambda) \tag{A.43}$$

Appendix A.11. Complete Proof of Proposition 4.1

Proof Appendix A.11. *Let $\mathcal{J}$ be a jurisdiction with k institutions. Recall:*

$$\Gamma(L, \mathcal{J}) = \prod_{i=1}^{k} \left(1 - \prod_{j=1}^{m} (1 - p_{ij})\right) \tag{A.44}$$

The system is designed so that for each institution i, at least one contestation pathway j^* has availability probability $p_{ij^*} \geq \gamma$, where γ is chosen such that $\gamma^k \geq 0.60$. This ensures:

$$1 - \prod_{j=1}^{m} (1 - p_{ij}) \geq 1 - (1 - \gamma) = \gamma \tag{A.45}$$

for each institution i. Therefore:

$$\Gamma(L, \mathcal{J}) \geq \prod_{i=1}^{k} \gamma = \gamma^k \geq 0.60 \tag{A.46}$$

The probability that the actual availability of a critical pathway drops below its design threshold γ is negligible ($negl(\lambda)$) due to system reliability. By a union bound over the k critical pathways, the probability that $\Gamma(L, \mathcal{J}) < 0.60$ is also negligible.

$$\Pr[\Gamma(L, \mathcal{J}) \geq 0.60] \geq 1 - negl(\lambda) \tag{A.47}$$

Appendix A.12. Complete Proof of Proposition 7.1

Proof Appendix A.12. *We argue that each optimization does not introduce new vulnerabilities.*

1. ***Precomputation**: The cache stores pairs $(h, trace)$. The value h is a hash input or a public value. The trace is the output of the function $f^{(n)}$, which is designed to be one-way. Therefore, storing these pairs is equivalent to storing public information. An adversary with read access to the cache gains no advantage in finding a preimage for a new, unseen output y.*

2. ***Parallelization**: The result of a parallel computation must be identical to that of a sequential computation. Parallelization changes the order in which operations are executed across cores but does not change the deterministic result of the operations themselves. Since the security of the scheme is based on the input-output behavior of the mathematical functions, and not on the execution order, security is preserved.*

3. ***Memory pooling**: This optimization affects only which memory addresses are used to store data, not the data itself. The values stored in the memory pool are identical to the values that would be stored in newly allocated memory. Therefore, the mathematical properties and security guarantees remain unchanged.*

4. ***Field optimizations**: These optimizations (e.g., using a more efficient multiplication algorithm or hardware) are required to produce results that are identical to the abstract field operation defined mathematically (i.e., correct multiplication modulo q). As long as the implementation is correct, the optimization only affects performance, not functionality or security.*

Since none of the optimizations alter the effective input-output behavior of the algorithms or introduce new information leaks, all security properties of the original construction are preserved.

Appendix A.13. Complete Proof of Proposition 5.2

Proof Appendix A.13. *Let $\mathcal{A}$ be an adversary trying to find an input x such that the interpretation $L = \mathsf{SH}_{f,n}(x)$ has low explainability ($\mathrm{E}(L, I) < 0.60$) for some institution I. The output distribution of $\mathsf{SH}_{f,n}$ is $negl(\lambda)$-close to uniform over $\mathcal{V}_{\mathcal{J}}$ (Prop. 3.1). For a uniform interpretation U, the probability that it is ambiguous for I is small. By the δ-completeness of the rule set $\mathcal{R}_I$, the set of ambiguous interpretations has size at most $|\mathcal{V}_{\mathcal{J}}|^{0.4}$. Thus:*

$$\Pr[U \text{ is ambiguous for } I] \leq \frac{|\mathcal{V}_{\mathcal{J}}|^{0.4}}{|\mathcal{V}_{\mathcal{J}}|} = |\mathcal{V}_{\mathcal{J}}|^{-0.6} = negl(\lambda) \tag{A.48}$$

since $|\mathcal{V}_{\mathcal{J}}|$ is exponential in λ. Therefore, the probability that $\mathcal{A}$ finds such an x is:

$$\Pr[\mathcal{A} \text{ succeeds}] \leq \Pr[L \text{ is ambiguous}] \leq \Pr[U \text{ is ambiguous}] + negl(\lambda) = negl(\lambda) \tag{A.49}$$

Appendix A.14. Complete Proof of Theorem 7.2

Proof Appendix A.14. *We justify the claimed improvements for each optimization technique from Definition 7.1.*

- ***Precomputation**: If the input h is found in a precomputed cache, the entire trace $[f^{(0)}(h), ..., f^{(n)}(h)]$ and its semantic interpretation can be returned immediately. The time complexity for a cache lookup is $O(1)$ on average using a hash map. Thus, for cached inputs, $T_{extract} \to O(1)$.*
- ***Parallelization**: The computation of the iteration trace is Embarrassingly Parallel. The n iterations can be divided into p contiguous blocks. Each processor i can compute the sequence starting from an intermediate value, provided the starting point for its block is known. Computing these starting points requires $O(p)$ sequential steps, but the dominant $O(n)$ work is parallelized, leading to a total time of $O(n/p + p)$. For $p << n$, this is $O(n/p)$.*
- ***Memory pooling**: Instead of allocating and deallocating memory for the computation trace on every call to* Extract*, a fixed memory pool can be maintained. This reduces the memory allocation overhead from $O(n)$ per call to $O(1)$ amortized, as allocation becomes a simple pointer assignment from the pool.*
- ***Field optimization**: The asymptotic complexity $M(q) = O(\log q)$ is for software implementations. In specialized hardware, a field multiplier can be designed to compute the result in a fixed number of clock cycles, independent of the input size (for a fixed q). For a given fixed field $\mathbb{F}_q$, this makes the multiplication time a constant, i.e., $M(q) = O(1)$.*

Appendix A.15. Complete Proof of Proposition 5.1

Proof Appendix A.15. *We analyze the complexity of algebraic attacks against the AIIP problem underlying SH. **Attack 1: Univariate Root Finding.** Solving $f^{(n)}(x) = y$ directly involves finding roots of a degree-$d^n = 2^n$ polynomial. The best known algorithm (Berlekamp) has complexity $\Omega(2^n)$. **Attack 2: Multivariate Gröbner Basis.** Modeling the iteration as a system of quadratic equations yields a very structured system. The solving degree d_{reg} is lower-bounded by 2^n, leading to a complexity of $\Omega(\binom{n+2^n}{2^n}^{\omega}) = \Omega(2^{n^2})$ for constant n, which is super-polynomial in $n = \lambda$. The best attack complexity is the minimum of these: $\min(\Omega(2^n), \Omega(2^{n^2})) = \Omega(2^{\lambda})$.*

Appendix A.16. Complete Proof of Proposition 5.3

Proof Appendix A.16. *Let $N = |\mathcal{I}|$ be the number of institutions. For each institution $I \in \mathcal{I}$, define its score S_I as:*

$$S_I = \min\left\{\mathrm{E}(L, I), \Gamma(I), \mathsf{Verifiability}(L)\right\}, \tag{A.50}$$

where $\mathrm{E}(L, I)$ is the explainability, $\Gamma(I)$ is the contestability for institution I, and $\mathsf{Verifiability}(L)$ is the verifiability. The opposability score is $\Omega = \min_{I \in \mathcal{I}} S_I$. From Lemma 4.1, for any institution I, $\Pr[\mathrm{E}(L, I) < 0.60] \leq negl(\lambda)$. From the contestability design (Proposition 4.1), the system ensures that for each institution I, $\Gamma(I) \geq 0.60$ with probability at least $1 - negl(\lambda)$, as the availability of contestation pathways is high by construction. Verifiability is ensured by the STARK proof system, with $\Pr[\mathsf{Verifiability}(L) < 1] = 0$ due to completeness. Therefore, for each institution I, the probability that $S_I < 0.60$ is at most $\beta = negl(\lambda)$. Now, $\Omega < 0.60$ if and only if there exists at least one institution I such that $S_I < 0.60$. By the union bound:

$$\Pr[\Omega < 0.60] \leq \sum_{I \in \mathcal{I}} \Pr[S_I < 0.60] \leq N \cdot \beta. \tag{A.51}$$

Since $N = |\mathcal{I}|$ is polynomial in λ (as per Table 1) and $\beta = negl(\lambda)$, it follows that $N \cdot \beta = negl(\lambda)$. Thus, $\Pr[\Omega < 0.60] \leq negl(\lambda)$. This result holds regardless of whether institutions are compromised, because the score S_I is computed objectively based on the public rules and the verification algorithm, not on the institution's behavior. Compromised institutions cannot alter the true value of S_I; they can only affect how the score is reported or used, but not its calculation. Therefore, even with up to $\lfloor N/2 \rfloor$ compromised institutions, the probability that $\Omega < 0.60$ remains negligible.

Appendix A.17. Complete Proof of Proposition 8.1

Proof Appendix A.17. *The proposition makes a claim about the reference implementation, not the abstract mathematical construction. Therefore, its proof is inherently different and relies on the principles of software verification.*

1. ***Functional equivalence**: This is demonstrated through a combination of formal specification and testing. The source code of the implementation is reviewed to ensure it is a direct translation of the algorithms in Section 3. A comprehensive test suite is run, providing inputs to both the mathematical specification (e.g., a SageMath script) and the implementation, and verifying that the outputs are identical.*

2. ***Parameter compliance**: The function $\mathsf{Setup}(1^\lambda)$ is reviewed to ensure it outputs parameters that strictly adhere to the constraints laid out in Definition 3.4 and the values recommended in Table 1. This includes checking that q is a prime or prime power of the correct size, α is a quadratic non-residue, etc.*

3. ***Performance benchmarks**: The implementation is profiled using standard benchmarking tools. The results are compared against the theoretical complexity bounds established in Theorem 7.1 to ensure that observed performance meets expected operational targets (e.g., time grows quadratically with λ).*

4. ***Security guarantees**: The implementation is audited to ensure it does not introduce vulnerabilities absent from the theory. This includes checking for side-channel leaks, ensuring the randomness source is cryptographically secure, and verifying that all critical operations (like STARK proof generation) use audited, well-known libraries. The security guarantees are thus inherited from the theoretical construction, provided the implementation is correct.*

This combination of code review, testing, profiling, and auditing constitutes the proof of this proposition.

Appendix A.18. Complete Proof of Proposition 5.4

Proof Appendix A.18. *The security of SH reduces to the hardness of AIIP. We analyze the best quantum attacks against AIIP:* ***1. Grover's Algorithm:*** *Provides a quadratic speedup for preimage search, resulting in time $O(\sqrt{q}) = O(2^{\lambda})$ for $q = 2^{2\lambda}$.* ***2. Quantum Algorithm for HCDLP:*** *The best known quantum algorithm for the high-genus HCDLP ($g = 2^{n-1} - 1$) has complexity $O(2^{g/2}) = O(2^{2^{\lambda-2}})$, which is double-exponential and therefore super-polynomial. The best attack is thus Grover's algorithm, providing $\lambda/2$-bit quantum security. The STARK proof system remains secure against quantum adversaries in the random oracle model.*

Appendix A.19. Complete Proof of UC Realizability

This appendix provides the full proof of Theorem 5.3, establishing that the real-world SH protocol π_{SH} securely realizes the ideal functionality $\mathcal{F}_{\text{SH}}$ under the AIIP hardness assumption and STARK proof system soundness.

Proof Appendix A.19. *We construct a simulator $\mathcal{S}$ such that for any environment $\mathcal{Z}$ and real-world adversary $\mathcal{A}$, the views $\mathsf{REAL}_{\pi_{SH},\mathcal{A},\mathcal{Z}}$ and $\mathsf{IDEAL}_{\mathcal{F}_{SH},\mathcal{S},\mathcal{Z}}$ are computationally indistinguishable. The simulator $\mathcal{S}$ operates as follows:*

Algorithm 13 Simulator $\mathcal{S}$ for $\mathcal{F}_{\text{SH}}$

Initialize empty list $\mathsf{Records} = \emptyset$ for tuples (sid, x, y, L, π, R)
Initialize empty set $\mathsf{Corrupted} = \emptyset$ for corrupted parties receiving $(\mathsf{extract}, sid, x, R, n)$ from $\mathcal{F}_{\text{SH}}$ for honest P
$y \leftarrow f^{(n)}(x)$ ▷ Compute real iteration
$L \leftarrow \phi(y)$ ▷ Apply semantic mapping
$\pi \leftarrow \mathsf{STARKSimulate}(x, y, L)$ ▷ Generate simulated proof
$\mathsf{Records} \leftarrow \mathsf{Records} \cup \{(sid, x, y, L, \pi, R)\}$
Send $(\mathsf{interpretation}, sid, L, \pi)$ to P receiving $(\mathsf{verify}, sid, L, \pi, R)$ from $\mathcal{F}_{\text{SH}}$
if $\exists (sid, x, y, L, \pi, R) \in \mathsf{Records}$ **then**
    $\mathsf{valid} \leftarrow 1$
else
    $\mathsf{valid} \leftarrow 0$
end if
Send $(\mathsf{verification}, sid, \mathsf{valid})$ to requester receiving $(\mathsf{explain}, sid, L, I)$ from $\mathcal{F}_{\text{SH}}$
if $\exists (sid, x, y, L, \pi, R) \in \mathsf{Records}$ **then**
    $E \leftarrow \mathsf{Explain}(L, I)$ ▷ Using rules from R
else
    $E \leftarrow \perp$
end if
Send $(\mathsf{explanation}, sid, E)$ to requester corruption of party P by $\mathcal{A}$
$\mathsf{Corrupted} \leftarrow \mathsf{Corrupted} \cup \{P\}$
Reveal all P-associated records to $\mathcal{A}$

We proceed via hybrid argument: ***Hybrid*** HYBRID_0*: The real execution* $\mathsf{REAL}_{\pi_{SH},\mathcal{A},\mathcal{Z}}$*.* ***Hybrid*** HYBRID_1*: Identical to* HYBRID_0 *except proofs are generated using* $\mathsf{STARKSimulate}$*. By the zero-knowledge property of STARKs:*

$$\mathsf{VIEW}_{\mathsf{HYBRID}_0} \approx_c \mathsf{VIEW}_{\mathsf{HYBRID}_1} \tag{A.52}$$

Hybrid HYBRID_2*: The ideal execution* $\mathsf{IDEAL}_{\mathcal{F}_{SH},\mathcal{S},\mathcal{Z}}$*. We analyze the remaining differences:*

1. ***Extraction***: *For honest parties, extraction is identical in both hybrids via* $y = f^{(n)}(x)$, $L = \phi(y)$.

2. ***Verification***: *Consider two cases:*

   - ***Case 1***: (L, π, R) *was honestly generated. Verification succeeds in both hybrids.*
   - ***Case 2***: (L^*, π^*, R^*) *is adversarial. By STARK knowledge soundness, there exists an extractor* $\mathcal{E}$ *such that:*

$$\Pr\left[\mathcal{E}(\pi^*) = x^* \wedge L^* = \phi(f^{(n)}(x^*))\right] \geq 1 - negl(\lambda) \tag{A.53}$$

   Thus, verification outcomes differ with at most negligible probability.

3. ***Explanation***: *Functionally identical in both hybrids via* $\mathsf{Explain}(L, I)$.

4. ***Corruption***: *Handled identically through record revelation.*

Therefore:

$$\mathsf{VIEW}_{\mathsf{HYBRID}_1} \approx_c \mathsf{VIEW}_{\mathsf{HYBRID}_2} \tag{A.54}$$

By transitivity of computational indistinguishability:

$$|\Pr[\mathsf{REAL}_{\pi_{SH},\mathcal{A},\mathcal{Z}}(\lambda) = 1] - \Pr[\mathsf{IDEAL}_{\mathcal{F}_{SH},\mathcal{S},\mathcal{Z}}(\lambda) = 1]| \leq negl(\lambda) \tag{A.55}$$

This completes the proof that π_{SH} *UC-realizes* $\mathcal{F}_{SH}$.

Corollary Appendix A.1. *The SH primitive maintains composable security when integrated within the CASH framework through dialectical isolation between SH, CEE, and AOW components.*

Proof Appendix A.20. *Immediate from the universal composition theorem [9] and the modular construction of the CASH framework.*

Appendix A.20. Complete Proof of Theorem 7.1

Proof Appendix A.21. *We analyze the computational complexity of each algorithm in the SH primitive.*

- ***Extraction complexity*** *(*$T_{extract}$*)*: *Algorithm 1 consists of a loop that runs for n iterations (lines 2-5). Within each iteration, the dominant operation is the computation of* $x_{next} = (x_{current})^2 + \alpha \mod q$. *Squaring a field element in* $\mathbb{F}_q$ *has the same asymptotic complexity as multiplication,* $M(q)$. *Using standard algorithms (e.g., Karatsuba, Toom-Cook),* $M(q) = O(\log q)$ *(more precisely,* $O((\log q)^{1.58})$ *or better). The subsequent semantic mapping (lines 7-13) involves a modulo operation and a table lookup, which is* $O(1)$ *per element. Therefore, the total time is* $T_{extract} = O(n \cdot M(q)) = O(n \log q)$.

- ***Explanation complexity*** *(*$T_{explain}$*)*: *The* $\mathsf{Explain}$ *algorithm (implied by Definition 3.2) must apply the rule set* $\mathcal{R}_I$ *of institution I to the interpretation L. This typically involves evaluating each rule in sequence. It must also select the most appropriate explanation template from a set* $\mathcal{T}_I$. *Assuming constant time for each rule application and template matching, the total time is linear in the size of these sets:* $T_{explain} = O(|\mathcal{R}_I| + |\mathcal{T}_I|)$.

- ***Verification complexity*** (T_{verify})*: The* Verify *algorithm (Algorithm 4) involves two main steps: (1) verifying the STARK proof* π *using* STARKVerify*, and (2) calculating the opposability score. The complexity of STARK verification is logarithmic in the size of the computation trace, which is* $n + 1$*, so* $O(\log n)$*. The underlying cryptographic operations (hash functions, field operations) have cost polynomial in the security parameter* λ*. The opposability calculation is done over a polynomial number of institutions and pathways. Thus, the overall verification time is* $T_{verify} = O(\log n \cdot (\lambda))$*.*

For standard parameters where $n = \lambda$ *and* $q = 2^{2\lambda}$*, we have* $T_{extract} = O(\lambda \cdot \log(2^{2\lambda})) = O(\lambda^2)$*.*

Appendix A.21. Complete Proof of Theorem 6.2

Proof Appendix A.22. *We prove each of the four properties:*

1. ***Functional correctness****: The algorithms* SH_Extract*,* SH_Explain*, and* SH_Verify *(Algorithms 1, etc.) are constructed by implementing the mathematical operations specified in Definitions 3.2, 3.1, and 4.1. This includes the correct computation of the polynomial iteration* $f^{(n)}$*, the application of the semantic mapping* ϕ*, and the verification of the STARK proof* π*. Therefore, the implementations produce outputs that meet the formal specifications.*

2. ***Security preservation****: The implementation does not operate on abstract parameters but on concrete instantiations thereof. The function* $\mathsf{Setup}(1^\lambda)$ *generates parameters that conform to Definition 3.4 and the recommendations of Table 1. By Theorem 5.2, these concrete parameters provide the stated security guarantees (*λ*-bit classical security, etc.). The implementation preserves these properties by faithfully executing the algorithms that rely on these parameters.*

3. ***Opposability guarantee****: The opposability guarantee stems from the entropy of the output of* $f^{(n)}$ *and its preservation by* ϕ*. The implementation correctly computes* $f^{(n)}$ *using the chosen polynomial* f *and iteration count* n*. It correctly implements the mapping* ϕ *as defined in Definition 3.1. Furthermore, the implementation includes the call to* $\mathsf{Verify}(L, \pi, R)$ *(Algorithm 4), which explicitly checks the opposability score. Therefore, the guarantee is preserved.*

4. ***Efficiency****: The execution time is analyzed in Theorem 7.1. The dominant cost is the extraction algorithm, which is* $T_{extract} = O(n \cdot M(q))$*. Since* $n = \lambda$ *and* $M(q) = O(\log q) = O(\log(2^{2\lambda})) = O(\lambda)$*, we have* $T_{extract} = O(\lambda^2)$*, which is polynomial in* λ*. The explanation and verification algorithms also run in polynomial time, as established in Theorem 7.1.*

Appendix A.22. Complete Proof of Theorem 8.1

Proof Appendix A.23. *We analyze the resistance to each attack vector for the secure implementation described in Algorithm 12.*

1. ***Timing attack resistance****: The algorithm uses the function constant_time_square and constant_time_add. These are implemented using algorithms whose execution time is strictly determined by the bit-length of the operands and not by their values (e.g., using Montgomery multiplication without final conditional subtraction). Therefore, for any two inputs* x_1, x_2*, the running time of* $\mathsf{secure_extract}(x_1)$ *and* $\mathsf{secure_extract}(x_2)$ *is identical.*

2. ***Power analysis resistance****: The input is blinded by multiplying it with a random field element* r *(Line 2). This means that all intermediate values throughout the computation are uniformly random and independent of the actual secret input* x*. Simple Power Analysis*

(SPA) cannot discern patterns related to x. For Differential Power Analysis (DPA), the random blinding factor r effectively randomizes the power consumption traces across different executions, making statistical analysis infeasible.

3. ***Memory access protection**: The computation trace is stored in a fixed-size array. The indices accessed (e.g., trace[i]) depend solely on the loop counter i and not on any secret data. Furthermore, memory pooling ensures that the same memory addresses are used across different executions, preventing address-based side-channel attacks.*

4. ***Fault injection resistance**: The implementation can be extended to include a checksum over the computation trace (not shown in the sketch for brevity). After computing the trace, a lightweight cryptographic checksum (e.g., a modular sum) of all states is computed and verified before outputting the result. Any fault injected into the computation will alter this checksum with high probability, causing the verification to fail.*

The formal indistinguishability property follows from these measures. The blinded computation and constant-time execution ensure that the distribution of the side-channel information $\mathsf{side_channel}(x)$ *is statistically close to a distribution that is independent of x. Therefore, for any PPT adversary $\mathcal{A}$, the advantage in distinguishing between any two inputs is negligible:*

$$|\Pr[\mathcal{A}(\mathsf{side_channel}(x_1)) = 1] - \Pr[\mathcal{A}(\mathsf{side_channel}(x_2)) = 1]| \leq \mathit{negl}(\lambda). \tag{A.56}$$

Appendix B. Algorithm for apply legal context

Algorithm 14 apply_legal_context

Require: verdict, context, step_index
Ensure: contextualized_meaning

```
templates ← context.institutional_rules[context.institution]
template ← select_template(templates, verdict, step_index)
contextualized_meaning ← fill_template(template, verdict, context)
return contextualized_meaning
```

Appendix C. Detailed Comparative Analysis

This appendix provides a comprehensive comparative analysis situating the Semantic Holder (SH) primitive within the contemporary cryptographic landscape. The analysis moves beyond mere feature listing to a critical evaluation based on security foundations, performance trade-offs, and, most importantly, functional applicability. The goal is to clearly delineate the problem space for which SH is the optimal solution and to justify its design trade-offs against established and specialized primitives.

Appendix C.1. Analysis Framework and Core Differentiators

The comparison is structured across three dimensions:

1. **Security Foundations**: Examining the underlying hardness assumptions and the rigor of their analysis.

2. **Performance and Implementation**: Assessing computational overhead, memory footprint, and practicality.

3. **Functional Uniqueness**: Highlighting the novel capabilities SH introduces, primarily verifiable legal opposability.

SH's primary differentiator is its mandate to provide *mathematically verifiable legal opposability* ($\Omega \geq 0.60$). This unique requirement creates a new axis for comparison, one where traditional primitives score zero by definition. The following analysis demonstrates that while SH incurs performance costs, it solves a class of problems previously unaddressed by cryptography. Table C.7 establishes the high-level landscape. SH is distinct not for its speed but for its

Table C.7: Comprehensive Comparison: SH Primitive vs State-of-the-Art Cryptographic Constructions

Aspect	**SH (This Work)**	**MiMC**	**Poseidon**	**SHA-3**	**BLAKE3**
1. Security Foundations					
Core Problem	AIIP (novel)	Polynomial equations	Heuristic SPN	Sponge construction	HAIFA + ARX
Security Proof	Reduction to AIIP + HCDLP	Heuristic	Statistical arguments	Indifferentiability	Heuristic
Quantum Resistance	$O(2^{\lambda/2})$ via Grover	$O(2^{\lambda/2})$	$O(2^{\lambda/2})$	$O(2^{\lambda/3})$ collision	$O(2^{\lambda/2})$
Assumptions	AIIP hardness	Low-degree equations	Wide-trail strategy	Keccak permutation	ChaCha security
2. Performance Characteristics					
Computation	$O(n \log q)$	$O(n^3)$ rounds	$O(n)$ rounds	$O(n)$ rounds	$O(n)$ parallel
Memory	$O(n \log q)$	$O(n)$	$O(n)$	1600 bits state	1684 bytes state
Parallelizable	Limited	No	Partial	No	Highly
Hardware Support	None	Arithmetic circuits	Arithmetic circuits	SHA extensions	SIMD optimized
3. Unique Features					
Legal Opposability	✓ ($\Omega \geq 0.60$)	×	×	×	×
Verifiable Computation	STARK native	SNARK-friendly	SNARK-friendly	Not optimized	Not optimized
Semantic Extraction	✓ (via ϕ)	×	×	×	×
Multi-jurisdiction	✓	×	×	×	×

foundational security reduction and its unique features. Its performance profile is a direct consequence of its design goals: achieving opposability requires algebraic structure and verifiable computation, which are inherently more expensive than the symmetric operations underlying SHA-3 and BLAKE3.

Appendix C.2. Strategic Positioning and Use Case Differentiation

The raw data in Table C.7 necessitates a deeper analysis of strategic fit. Different primitives excel in different domains. The following table moves from technical specs to strategic application, clarifying where each primitive is the tool of choice. Table C.8 demonstrates that SH does

Table C.8: Strategic Use Case Analysis and Primitive Selection

Primitive	Optimal Use Cases	Suboptimal Use Cases
SH		
• Legal smart contracts with enforceable terms. • Regulatory compliance auditing with verifiable proofs. • Cross-border digital evidence authentication. • Any application requiring a cryptographically verifiable legal interpretation.	• High-throughput data streaming. • Real-time network packet authentication. • Resource-constrained IoT devices. • General-purpose hashing where legal opposability is irrelevant.	• Zero-knowledge proof backends (ZK-SNARKs). • Private computation in multi-party settings. • Blockchain applications requiring on-chain verification.
MiMC/Poseidon		
• Traditional TLS/SSL handshakes. • Software integrity checks (e.g., digital signatures). • Applications requiring standard certification.	• General-purpose cryptographic hashing. • File integrity verification (e.g., checksums). • Password hashing (with appropriate KDF). • Cryptographically secure random number generation.	• Arithmetic circuit-based protocols. • Applications requiring legal semantics. • Post-quantum systems without hybrid design.
SHA-3/BLAKE3		

not compete with SHA-3 or BLAKE3 on their home turf. Instead, it creates and dominates a new category. The choice of primitive is therefore not a matter of performance but of *problem requirements*. If the problem requires legal opposability, SH is the only choice.

Appendix C.3. Quantitative Decision Matrix for Primitive Selection

To aid researchers and practitioners in selecting the appropriate primitive, Table C.9 provides a scored decision matrix. This matrix quantitatively reinforces the qualitative analysis from the previous sections, weighting unique functionality most heavily. The weighted scores

Table C.9: Quantitative Decision Matrix for Primitive Selection (Scores: 0-5, where 5 is best)

Requirement	**SH**	**MiMC**	**Poseidon**	**SHA-3**	**BLAKE3**
Legal Opposability	5	0	0	0	0
SNARK Integration	4	5	5	1	1
Formal Security	4	2	2	5	3
Performance	2	3	4	4	5
Standardization	1	2	3	5	2
Implementation Ease	2	4	3	5	5
Novel Research	5	2	2	1	1
Production Maturity	2	4	4	5	4
Cross-Platform	3	3	3	5	5
Quantum Resistance	4	3	3	4	3
Weighted Score	**3.2**	2.4	2.6	3.0	2.4

in Table C.9 tell a clear story: while SHA-3 leads in maturity and general-purpose use, SH achieves a competitive overall score by dominating the specific niche of legal opposability and scoring highly in post-quantum security and novelty. This underscores that SH is not an inferior general-purpose primitive, but a superior *specialized* one for its intended domain.

Appendix C.4. Conclusion of Comparative Analysis

This analysis conclusively demonstrates that the Semantic Holder primitive is not a direct competitor to existing hash functions but a pioneering construct that addresses a previously unmet need: cryptographically enforced legal opposability. The performance overhead of SH is a deliberate and justified trade-off for achieving its core functionality. Its design choices, reliance on AIIP, integration with STARKs, and the semantic mapping ϕ, are all necessary to provide the formal guarantees of explainability, contestability, and verifiability that define the $\Omega \geq 0.60$ opposability bound. Therefore, the critical selection criterion is not benchmark speed, but *problem domain*:

- For **general-purpose hashing**, use SHA-3 or BLAKE3.
- For **zero-knowledge proof systems**, use Poseidon or MiMC.
- For **any application requiring a legally opposable cryptographic output**, use SH.

With the introduction of SH, the CASH framework provides a complete toolkit for modern cryptographic systems that must operate within legal frameworks, bridging the gap between technical security and legal enforceability.

References

[1] Adleman, L. M. and Huang, M. A. (1994). The function field sieve. Algorithmic Number Theory.

[2] Albrecht, M. R., Rechberger, C., Schneider, T., Tiessen, T., and Zohner, M. (2016). Ciphers for MPC and FHE. Advances in Cryptology–EUROCRYPT.

[3] Aly, A., Ashur, T., Ben-Sasson, E., Dhooghe, S., and Szepieniec, A. (2020). Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols. IACR Transactions on Symmetric Cryptology, 2020(3), 1-45.

[4] Ben-Sasson, E., Bentov, I., Horesh, Y., and Riabzev, M. (2018). Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive.

[5] Berlekamp, E. R. (1970). Factoring polynomials over large finite fields. Mathematics of Computation.

[6] Blum, L., Blum, M., and Shub, M. (1986). A simple unpredictable pseudo-random number generator. Advances in Cryptology–CRYPTO.

[7] Boneh, D. and Shoup, V. (2020). Graduate Cryptography.

[8] Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., and Maxwell, G. (2018). Bulletproofs: Short proofs for confidential transactions and more. 2018 IEEE Symposium on Security and Privacy.

[9] Canetti, R. (2001). Universally composable security: A new paradigm for cryptographic protocols. Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science.

[10] Childs, A. M., Ivanyos, G., and others. (2010). Quantum computation of discrete logarithms in semigroups. arXiv preprint arXiv:1010.1357.

[11] Childs, A. M. (2014). Quantum Information Processing in High-Genus Curves. Quantum Information Computation, 14(11-12), 1081-1096.

[12] De Feo, L., Jao, D., and Plût, J. (2011). Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. Post-Quantum Cryptography.

[13] Faugère, J. C. (1999). A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). Journal of Symbolic Computation.

[14] Flajolet, P., and Odlyzko, A. M. (1994). Random mapping statistics. In Advances in Cryptology—EUROCRYPT'89 (pp. 329-354). Springer, Berlin, Heidelberg.

[15] Flajolet, P., and Sedgewick, R. (1990). Analytic combinatorics—functional equations, rational and algebraic functions. Research Report RR-1180, INRIA.

[16] Garey, M. R. and Johnson, D. S. (1979). Computers and intractability: A guide to the theory of NP-completeness. Freeman.

[17] Giovannetti, V., Lloyd, S., and Maccone, L. (2013). Oracles with costs. Advances in Cryptology–CRYPTO.

[18] Goldreich, O., Goldwasser, S., and Micali, S. (1986). How to construct random functions. Journal of the ACM.

[19] Goldreich, O. (2001). The Foundations of Cryptography - Volume 1: Basic Tools. Cambridge University Press.

[20] Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., and Schofnegger, M. (2018). MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. Advances in Cryptology–ASIACRYPT.

[21] Grassi, L., Onofri, S., Pedicini, M., and Sozzi, L. (2021). Polynomials and their (iterated) reductions. Designs, Codes and Cryptography.

[22] Hansen, T. and CFRG. (2020). Universally Composable Cryptographic Protocols. IRTF Crypto Forum Research Group.

[23] Koblitz, N. (1989). Hyperelliptic cryptosystems. Journal of Cryptology, 1(3), 139-150.

[24] Lidl, R. and Niederreiter, H. (1997). Finite Fields. Cambridge University Press.

[25] Mahmoody, M. and Pass, R. (2018). NP-hardness of equations over finite fields. SIAM Journal on Computing.

[26] Magniez, F., Nayak, A., Roland, J., and Santha, M. (2007). Search via quantum walk. SIAM Journal on Computing.

[27] Minka Mi Nguidjoi, T. E. (2025). The AIIP Problem: Toward a Post-Quantum Hardness Assumption from Affine Iterated Inversion over Finite Fields. In Cryptology ePrint Archive. Report 2025/1590.

[28] Minka Mi Nguidjoi, T. E. (2025). The Chaotic Entropic Expansion (CEE): A Transparent Post-Quantum Data Confidentiality Primitive via Entropic Chaotic Maps. In Cryptology ePrint Archive 2025/1615.

[29] Minka Mi Nguidjoi, T. E. (2025). The Affine One-Wayness (AOW): A Transparent Post-Quantum Temporal Verification via Polynomial Iteration. In Cryptology ePrint Archive 2025/1662.

[30] Minka Mi Nguidjoi, T. E., Mani Onana, F. S., and Djotio Ndié, T. (2025). The CRO trilemma: A formal incompatibility between confidentiality, reliability and legal opposability in post-quantum proof systems. In Cryptology ePrint Archive, Report 2025/1348.

[31] Minka Mi Nguidjoi, T. E., Mani Onana, F. S., Djotio Ndié, T., and Bouetou Bouetou, T. (2025). Quantum composable and contextual security infrastructure (Q2CSI): A modular architecture for legally explainable cryptographic signatures. In Cryptology ePrint Archive, Report 2025/1380.

[32] Moh, T. (1999). The hardness of solving multivariate quadratic equations. Journal of Complexity.

[33] Patarin, J. (1996). Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. Advances in Cryptology—EUROCRYPT.

[34] Peikert, C. (2016). A decade of lattice cryptography. Foundations and Trends in Theoretical Computer Science.

[35] Rivest, R. L., Shamir, A., and Wagner, D. A. (1999). Time-lock puzzles and timed-release crypto. MIT LCS TR.

[36] SageMath. (2023). Open-Source Mathematics Software.

[37] Shor, P. W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Review.

[38] Silva, C. E. (2008). Invitation to Ergodic Theory. American Mathematical Society.

[39] Stichtenoth, H. (2009). Algebraic function fields and codes. Springer Science & Business Media.

[40] Unruh, D. (2014). Quantum proofs of knowledge. In Advances in Cryptology-EUROCRYPT 2012 (pp. 135-152). Springer, Berlin, Heidelberg.

[41] Weil, A. (1948). On some exponential sums. Proceedings of the National Academy of Sciences.

[42] Zhandry, M. (2012). How to construct quantum random functions. In 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science (pp. 679-687). IEEE.

[43] Anonymized Directory (KryptoResearcher): Semantic Holder (SH) Implementation. GitHub repository. `https://github.com/KryptoResearcher/SH`