Fast Authenticated and Interoperable Multimedia Healthcare Data over Hybrid-Storage Blockchains

Jucai Yang, Liang Li, Yiwei Gu, and Haiqin Wu*
East China Normal University, Shanghai, China
Email: {jcyang, liangli, ywgu}@stu.ecnu.edu.cn, hqwu@sei.ecnu.edu.cn

***Abstract*—The integration of blockchain technology into healthcare presents a paradigm shift for secure data management, enabling decentralized and tamper-proof storage and sharing of sensitive Electronic Health Records (EHRs). However, existing blockchain-based healthcare systems, while providing robust access control, commonly overlook the high latency in user-side re-computation of hashes for integrity verification of large multimedia data, impairing their practicality, especially in time-sensitive clinical scenarios. In this paper, we propose FAITH, an innovative scheme for Fast Authenticated and Interoperable[1] mulTimedia Healthcare data storage and sharing over hybrid-storage blockchains. Rather than user-side hash re-computations, FAITH lets an off-chain storage provider generate verifiable proofs using recursive Zero-Knowledge Proofs (ZKPs), while the user only needs to perform lightweight verification. For flexible access authorization, we leverage Proxy Re-Encryption (PRE) and enable the provider to conduct ciphertext re-encryption, in which the re-encryption correctness can be verified via ZKPs against the malicious provider. All metadata and proofs are recorded on-chain for public verification. We provide a comprehensive analysis of FAITH's security regarding data privacy and integrity. We implemented a prototype of FAITH, and extensive experiments demonstrated its practicality for time-critical healthcare applications, dramatically reducing user-side verification latency by up to 98%, bringing it from 4 s down to around 70 ms for a 5 GB encrypted file.**

***Index Terms*—healthcare data sharing, data integrity verification, privacy, blockchain, zero-knowledge proofs**

I. Introduction

The application of blockchain in healthcare is reshaping conventional isolated systems, with hybrid-storage architectures [1]–[3] emerging as a prevailing strategy to scale medical services. In this model, large raw electronic health records (EHRs) are stored in an off-chain service provider (SP), while only the metadata (e.g., hashes) are recorded on-chain. However, this model introduces significant security concerns as the off-chain SP is often untrusted, posing risks to data privacy and integrity [4]. In data sharing scenarios, clinicians or researchers as data users request patients' (i.e., data owners') EHRs for further treatment or academic studies, and should be able to check the integrity of data received from the SP.

For secure data storage and sharing in healthcare scenarios, existing blockchain-based solutions commonly employ cryptographic primitives like Proxy Re-Encryption (PRE) [1], [3], [5], [6] or Attribute-Based Encryption (ABE) [2], [7]–[10] for flexible access delegation or fine-grained access control over encrypted EHRs, respectively. However, they work in an honest-but-curious model and overlook the SP's misbehavior (e.g., incorrect re-encryption). Some studies [1]–[3] adopted a hash re-computation-based approach for data integrity checking, which unfortunately requires users to hash entire large multimedia files (e.g., MRIs) to verify their integrity. The incurred significant latency is unsuitable for time-sensitive scenarios such as emergency care where delays can have severe consequences.

To address the above issues, we introduce FAITH, a novel system for Fast Authenticated and Interoperable mulTimedia Healthcare data storage and sharing over hybrid-storage blockchains. FAITH overcomes the performance bottleneck of integrity verification for large EHRs. Our core innovation is to transform user-side intensive hash re-computation of large data to a lightweight proof verification for data integrity checking, in which the proof generation is offloaded to the SP who proves the integrity of its stored data based on recursive Zero-Knowledge Proofs (ZKPs). We further incorporate PRE for flexible access authorization and data sharing, and anyone can verify the off-chain ciphertext re-encryption with ZKPs recorded on-chain. Overall, FAITH provides a holistic solution for efficiently verifiable and interoperable secure sharing of large multimedia EHRs among patients and healthcare institutions. This caters to time-critical healthcare scenarios. Our main contributions are summarized below:

1) We are the first to identify and resolve the key performance bottleneck of user-side integrity verification for large multimedia files in blockchain healthcare systems.
2) We propose FAITH, an interoperable healthcare architecture that enables fast integrity verification by shifting the burden from user-side hash re-computation to lightweight verification of on-chain ZKPs.
3) We demonstrate FAITH's security guarantees and performance, achieving around a 98% reduction in verification time for a 5 GB file compared to hash recomputation.

II. Preliminaries

A. Proxy Re-Encryption

Proxy Re-Encryption (PRE) is an advanced public-key encryption that enables a proxy to transform a ciphertext from

*Corresponding author: Haiqin Wu. This work was supported in part by the National Key R&D Program of China under Grant 2022YFB2701400, and in part by the National Natural Science Foundation of China under Grants 62202167, 62172162, and 62132005.

[1]We focus on the secure data sharing across healthcare institutions.

one key to another without learning the underlying plaintext, making it ideal for secure data sharing [11]. A PRE scheme consists of the following algorithms:

- $\mathsf{PRE.KeyGen}(1^\lambda) \rightarrow (pk, sk)$: Generates a key pair.
- $\mathsf{PRE.Enc}(pk_A, m) \rightarrow c_A$: Encrypts a message m under public key pk_A to get ciphertext c_A.
- $\mathsf{PRE.ReKeyGen}(sk_A, pk_B) \rightarrow rk_{A\rightarrow B}$: Generates a re-encryption key $rk_{A\rightarrow B}$ using sk_A and pk_B.
- $\mathsf{PRE.ReEnc}(rk_{A\rightarrow B}, c_A) \rightarrow c_B$: Uses $rk_{A\rightarrow B}$ to transform ciphertext c_A into a new ciphertext c_B.
- $\mathsf{PRE.Dec}(sk_B, c_B) \rightarrow m$: Decrypts c_B with secret key sk_B to recover message m.

B. Recursive Zero-Knowledge Proofs

A Zero-Knowledge Proof (ZKP) [12] allows a prover to prove a statement's validity without revealing secret information. A Zero-Knowledge Succinct Non-interactive Argument of Knowledge (zk-SNARK) is a highly efficient ZKP that generates a succinct proof. A zk-SNARK scheme includes:

- $\mathsf{ZKP.Setup}(\mathcal{C}) \rightarrow (prk, vrk)$: Generates a proving key prk and a verification key vrk for a given circuit $\mathcal{C}$.
- $\mathsf{ZKP.Prove}(prk, x, w) \rightarrow \pi$: Using a private witness w, generates a proof π for a public input x.
- $\mathsf{ZKP.Verify}(vrk, x, \pi) \rightarrow 0/1$: Verifies the proof π.

Recursive ZKPs [13] extend this by allowing one ZKP to verify other ZKPs. Since a zk-SNARK verifier itself is a circuit, its execution can be proven within another zk-SNARK. This enables aggregating multiple proofs $\{\pi_1, \ldots, \pi_n\}$ into a single, compact proof that attests to their collective validity.

III. Construction

A. System Model

As illustrated in Fig. 1, our proposed system, FAITH, consists of five main entities:

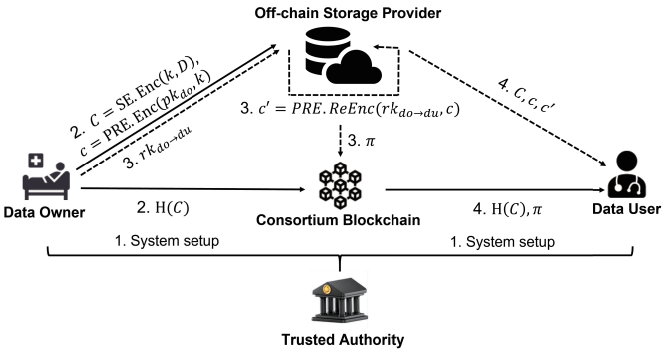


Fig. 1: System model of FAITH.

- **Trusted Authority (TA):** The TA is a trusted third party (government agency or healthcare authority) responsible for the system setup and public parameter generation.
- **Data Owner (DO):** A patient who encrypts their multimedia healthcare data and grants access rights to others.
- **Data User (DU):** A clinician or researcher who requests authorized access to the DO's data.
- **Off-chain Storage Provider (SP):** The SP stores large encrypted data, performs proxy re-encryption upon request, and is tasked with generating ZKPs for integrity.
- **Consortium Blockchain:** A tamper-proof ledger for storing metadata (e.g., data hashes) and verifiable proofs.

The overall workflow of FAITH is as follows. First, the TA conducts system setup, including public parameter generation and cryptographic primitive definition (step 1). Then the DO encrypts a large multimedia EHR D with a symmetric key k, and derives a ciphertext C. The symmetric key k itself is encrypted using a PRE scheme under the DO's public key pk_o, which generates an encrypted key c. Both C and c are sent to the SP while the metadata $H(C)$ is recorded on-chain (step 2). When a DU requests access to the EHR, the DO generates a re-encryption key $rk_{o\rightarrow u}$ and sends it to the SP, who then transforms c into a new ciphertext c' with $rk_{o\rightarrow u}$. Meanwhile, it generates a zero-knowledge proof π certifying the honest execution of re-encryption and data storage integrity, π is anchored to the blockchain (step 3). Then SP returns the corresponding result to the DU. With the on-chain recorded $H(C)$ and proof π, the DU finally performs an integrity check and then recovers the original EHR D (step 4).

B. Scheme Implementation

The FAITH system is constructed through four main phases: setup, data upload, access granting, and data retrieval.

1. System Setup. The TA performs a one-time setup, defining system-wide parameters and cryptographic primitives: a security parameter λ, bilinear map parameters $\{\mathbb{G}_1, \mathbb{G}_2, p, e, g_1, g_2\}$, a hash function $\mathsf{H} : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$, a symmetric encryption scheme $\mathsf{SE} = \{\mathsf{SE.KeyGen}(1^\lambda), \mathsf{SE.Enc}(k, D), \mathsf{SE.Dec}(k, C)\}$, a proxy re-encryption scheme PRE (see Section II-A), and a zero-knowledge proof system ZKP (see Section II-B). The TA generates proving and verification keys for three ZKP circuits: $\{prk_{\text{int}}, vrk_{\text{int}}\} \leftarrow \mathsf{ZKP.Setup}(\mathcal{C}_{\text{int}}, 1^\lambda)$ for data integrity, $\{prk_{\text{pre}}, vrk_{\text{pre}}\} \leftarrow \mathsf{ZKP.Setup}(\mathcal{C}_{\text{pre}}, 1^\lambda)$ for re-encryption correctness, and $\{prk_{\text{agg}}, vrk_{\text{agg}}\} \leftarrow \mathsf{ZKP.Setup}(\mathcal{C}_{\text{agg}}, 1^\lambda)$ for proof aggregation. The details of $\mathsf{ZKP.Setup}$ construction can be found in [13]. Each user (DO or DU) generates their own PRE key pair by running $\mathsf{PRE.KeyGen}(1^\lambda)$. Specifically, s/he chooses two random numbers $sk_1, sk_2 \in \mathbb{Z}_p^*$ as secret key $sk = (sk_1, sk_2)$ and the public key is generated as $pk = (g_2^{sk_1}, g_1^{sk_2})$. To improve clarity, we denote the DO's key pair as (pk_o, sk_o) where $sk_o = (o_1, o_2)$ and $pk_o = (g_2^{o_1}, g_1^{o_2})$, and the DU's key pair as (pk_u, sk_u) where $sk_u = (u_1, u_2)$ and $pk_u = (g_2^{u_1}, g_1^{u_2})$.

2. Data Encryption and Upload. To securely store a multimedia file D, the DO performs the following:

1) Encrypts the file D with a fresh symmetric key $k \leftarrow \mathsf{SE.KeyGen}(1^\lambda)$ to get ciphertext $C \leftarrow \mathsf{SE.Enc}(k, D)$.
2) Encrypts the key k using their own public key pk_o, resulting in a ciphertext $c \leftarrow \mathsf{PRE.Enc}(pk_o, k)$, where

$$c = (c_1, c_2), c_1 = g_1^r,\ c_2 = k \cdot (g_2^{o_1})^r = k \cdot g_2^{o_1 r}. \quad (1)$$

3) Records the hash of the data ciphertext, $h \leftarrow \mathsf{H}(C)$, on the blockchain and sends the tuple $\langle id, C, c\rangle$ to SP.

Upon receipt, the SP generates an initial integrity proof $\pi_{\text{int}} \leftarrow \mathsf{ZKP.Prove}(prk_{\text{int}}, x_{\text{int}}, w_{\text{int}})$ which attests that it stores the correct data corresponding to the on-chain hash h, where $x_{\text{int}} \leftarrow h$ is a public statement, and $w_{\text{int}} \leftarrow C$ is the witness. Then the SP stores $\langle id, C, c, \pi_{\text{int}}\rangle$ in its storage.

3. Access Granting and Proof Aggregation. When a DO grants access to a DU, the following steps occur:

1) The DO generates a re-encryption key $rk_{o\to u} \leftarrow \mathsf{PRE.ReKeyGen}(sk_o, pk_u)$ where $rk_{o\to u} = (g_1^{u_2})^{o_1} = g_1^{o_1 u_2}$, and sends it to the SP.
2) The SP uses $rk_{o\to u}$ to transform the key ciphertext c into a re-encrypted version c' for the DU, where

$$c'_1 = e(c_1, rk_{o\to u}) = e(g_1^r, g_1^{o_1 u_2}) = g_2^{r o_1 u_2},$$
$$c'_2 = c_2 = k \cdot g_2^{o_1 r}. \tag{2}$$

3) The SP generates a proof π_{pre} certifying the correctness of the re-encryption operation:

$$\pi_{\text{pre}} \leftarrow \mathsf{ZKP.Prove}(prk_{\text{pre}}, x_{\text{pre}}, w_{\text{pre}}), \tag{3}$$

where $x_{\text{pre}} \leftarrow c' \| c$ is public and $w_{\text{pre}} \leftarrow rk_{o\to u}$ is private.
4) Critically, the SP uses a recursive ZKP to aggregate the integrity proof π_{int} and the re-encryption proof π_{pre} into a single, compact proof π_{agg}:

$$\pi_{\text{agg}} \leftarrow \mathsf{ZKP.Prove}(prk_{\text{agg}}, x_{\text{agg}}, w_{\text{agg}}), \tag{4}$$

where where $x_{\text{agg}} \leftarrow h \| c' \| c$ and $w_{\text{agg}} \leftarrow C \| rk_{o\to u}$. This aggregated proof is then published on the blockchain.

4. Data Request and Decryption. To access the data, the DU sends a query $Q = \langle id\rangle$ to SP who then returns the tuple $\langle id, C, c, c', \pi_{\text{agg}}\rangle$. The DU then performs the following steps:

1) Fetches the data hash h and the aggregated proof π_{agg} from the blockchain, and performs a single, lightweight verification $\mathsf{ZKP.Verify}(vrk_{\text{agg}}, x_{\text{agg}}, \pi_{\text{agg}})$. This check simultaneously validates both the integrity of the file C and the correctness of the key re-encryption.
2) If verification succeeds, the DU uses their secret key sk_u to decrypt c' to recover the symmetric key k:

$$\begin{aligned} c'_2 \cdot (c'_1)^{-1/u_2} &= (k \cdot g_2^{o_1 r}) \cdot (g_2^{r o_1 u_2})^{-1/u_2} \\ &= (k \cdot g_2^{o_1 r}) \cdot (g_2^{-r o_1}) = k, \end{aligned} \tag{5}$$

and then decrypts C to obtain $D \leftarrow \mathsf{SE.Dec}(k, C)$.

This process can be extended to handle multiple files, where proofs for all files can be combined into a single aggregated proof, further reducing the verification overhead for the DU.

TABLE I: Performance of SE encryption and decryption.

Message size (GB)	1	2	3	4	5
SE.Enc (s)	2.800	5.604	8.381	11.167	13.967
SE.Dec (s)	0.820	1.656	2.482	3.249	4.0527

TABLE II: Time cost of PRE algorithms.

PRE algorithms	**KeyGen**	**ReKeyGen**	**Enc**	**ReEnc**	**Dec**
Time cost (ms)	3.135	0.292	1.506	2.615	3.791

IV. Evaluation

A. Security Analysis

FAITH provides robust security guarantees through its cryptographic design, including data privacy and integrity.

Data Privacy. Privacy is ensured by a two-layer encryption mechanism. First, large multimedia healthcare data is encrypted via a secure symmetric scheme (e.g., AES), rendering it unintelligible to the off-chain SP. Second, the symmetric key itself is encrypted using a PRE scheme. The fundamental security of PRE ensures that only an authorized DU, holding the correct private key, can decrypt the re-encrypted key. The SP, even while performing re-encryption, learns nothing about the key's content. This dual approach provides end-to-end confidentiality, effectively protecting sensitive data from both the SP and any unauthorized parties.

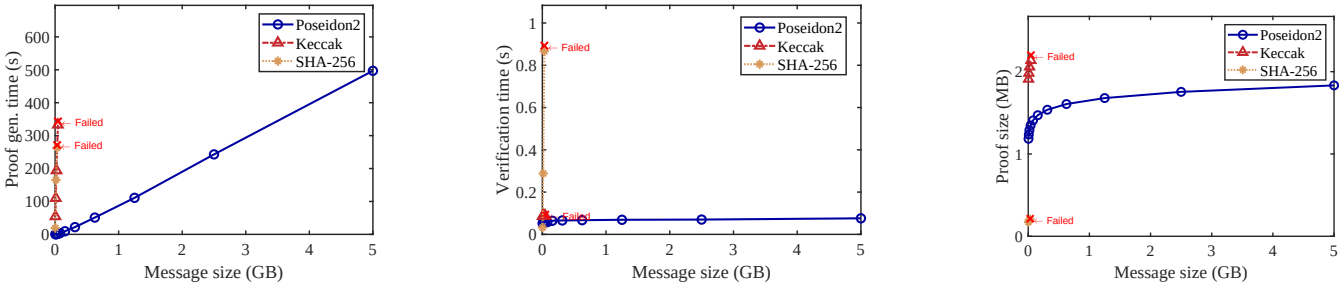
Integrity. FAITH guarantees both data integrity and re-encryption correctness using on-chain hashes and ZKPs. The hash of the large encrypted file is anchored on the immutable blockchain. Instead of requiring the DU to download the entire file for hash re-computation, our system offloads this task to the SP. The SP generates a ZKP attesting that its stored data matches the on-chain hash. Another ZKP is generated to prove the correctness of the PRE operation. Crucially, these proofs are recursively aggregated into a single and compact proof that is stored on-chain. This allows any DU to perform a fast, public, and lightweight verification in milliseconds, regardless of the data size, immediately detecting any malicious tampering or incorrect re-encryption by the SP.

B. Performance Analysis

We implemented and evaluated a prototype of FAITH using AFGH-06 PRE [11], AES [14], and the Plonky2 [13] ZKP system. Our evaluation compared the performance of the ZKP-friendly Poseidon2 hash against traditional hashes like SHA-256 for files up to 5 GB.

The performance of the core cryptographic primitives validates our hybrid design. Symmetric encryption of a 5 GB file takes approximately 14 s, a practical one-time cost, while all PRE operations on the small symmetric key complete in just a few milliseconds (Tables I & II).

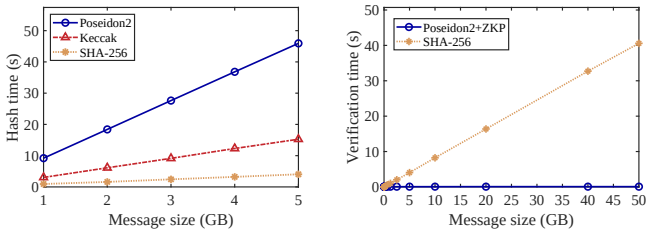
For the ZKP-based integrity check, our results demonstrate that using a ZKP-friendly hash function like Poseidon2 is critical. As shown in Fig. 2, our ZKP-based verification time remains near-constant at around 70 ms, regardless of the file size. As observed, Poseidon2 presents the fastest integrity verification performance, merely incurring tens of milliseconds, which is more lightweight than hash recomputation (at least a few seconds). In contrast, the conventional approach of re-computing a SHA-256 hash scales linearly, taking around 4 s



(a) Proof generation time (b) Verification time (c) Proof size

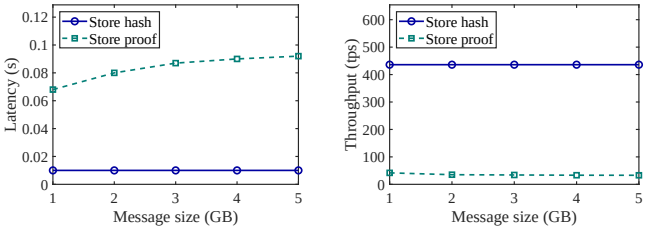
Fig. 2: Comparison of different hash functions under ZKP.

Notes: × ← Failed indicates the proving process terminated at that message size due to errors such as out-of-memory.



(a) Naive hash re-computation (b) Comparison with benchmark

Fig. 3: Integrity verification time under different approaches.



(a) Latency (b) Throughput

Fig. 4: On-chain performance for storing hash and proof.

for a 5 GB file (see Fig. 3). Overall, FAITH presents an approximately 98% verification time reduction by transforming hash re-computation to a near-instantaneous operation. Finally, Fig. 4 depicts the on-chain latency and throughput for our hash and proof storage, confirming FAITH's overall efficiency (the latency was under 0.1 s) and practicality.

V. Conclusion

This paper introduced FAITH, a novel framework for fast, interoperable healthcare data sharing that resolves the critical user-side integrity verification bottleneck. It replaces the costly hash re-computation of large files with a lightweight verification of server-generated ZKPs. FAITH also integrates Proxy Re-Encryption for flexible access, using ZKPs to ensure re-encryption correctness against a malicious server. Our analysis confirms robust data privacy and integrity, while experiments demonstrate that FAITH reduces verification time to a constant few milliseconds, irrespective of file size, validating its feasibility for real-world medical applications.

References

[1] Y. Lv, X. Li, Y. Wang, K. Chen, Z. Hou, and R. Feng, "Cross-chain sharing of personal health records: Heterogeneous and interoperable blockchains," in *2024 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2024, pp. 3588–3591.

[2] X. Wu, L. Zhang, and H. Huang, "Patient-Centered Data Sharing and Revision Framework Based on Redactable Blockchain," in *2023 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2023, pp. 4452–4458.

[3] M. N. Hasan, S. Datta, and S. Namasudra, "Inter-hospital secure healthcare data exchange process by using proxy re-encryption and blockchain technology," *Computers in Biology and Medicine*, vol. 194, p. 110462, 2025.

[4] C. Dive, "Data breach at healthcare services firm episource affects 5.4m," 2025. [Online]. Available: https://www.cybersecuritydive.com/news/episource-healthcare-data-breach-impacts-5-4-million/751960/

[5] Raghav, N. Andola, S. Venkatesan, and S. Verma, "Privacy-preserving cloud data sharing for healthcare systems with hybrid blockchain," *Peer-to-Peer Networking and Applications*, vol. 16, no. 5, pp. 2525–2547, 2023.

[6] X. Feng, Q. Shen, C. Li, X. Wang, N. Xie, L. Xie, Y. Fang, and Z. Wu, "A privacy preserving computer-aided medical diagnosis framework with outsourced model," in *2023 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2023, pp. 1080–1087.

[7] G. Han, Y. Ma, Z. Zhang, and Y. Wang, "A hybrid blockchain-based solution for secure sharing of electronic medical record data," *PeerJ Computer Science*, vol. 11, p. e2653, 2025.

[8] S. Xu, J. Ning, X. Li, J. Yuan, X. Huang, and R. H. Deng, "A privacy-preserving and redactable healthcare blockchain system," *IEEE Transactions on Services Computing*, vol. 17, no. 2, pp. 364–377, 2024.

[9] Y. Tao, L. Zhou, Y. Zhu, Y. Li, and C. Ge, "Care-emrs: Efficient and privacy-preserving data sharing framework for electronic medical records," in *2024 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2024, pp. 3758–3761.

[10] L. Fang, C. Yin, J. Zhu, C. Ge, M. Tanveer, A. Jolfaei, and Z. Cao, "Privacy protection for medical data sharing in smart healthcare," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 16, no. 3s, Dec. 2020.

[11] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, no. 1, pp. 1–30, 2006.

[12] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *STOC*, 2019, pp. 203–225.

[13] PolygonZero, "Plonky2: Fast recursive arguments with plonk and fri," https://github.com/mir-protocol/plonky2/blob/main/plonky2/plonky2.pdf, 2022.

[14] J. Daemen and V. Rijmen, "AES proposal: Rijndael," 1999.