



DESIGN AND PERFORMANCE EVALUATION OF A HYBRID HOMOMORPHIC ENCRYPTION AND HASHING-BASED CLOUD SECURITY ARCHITECTURE

¹Pooja Gupta, ²Dr. Udai Shankar (Assistant Professor)

¹Research Scholar, ²Research Supervisor

¹⁻²Department of Computer Science, Sunrise University, Alwar, Rajasthan

Abstract

Cloud computing provides elastic storage and computational resources but transfers sensitive information into infrastructure that is not fully controlled by the data owner. Conventional encryption protects outsourced data at rest and in transit, yet ordinary computation generally requires plaintext exposure. Homomorphic encryption (HE) addresses this limitation by enabling selected operations directly over ciphertext; however, HE introduces substantial computational cost, ciphertext expansion, parameter-management complexity, and does not by itself provide a complete mechanism for application-level integrity verification. This study designs and analytically evaluates a hybrid cloud security architecture that combines homomorphic encryption with cryptographic hashing. The architecture separates responsibilities: HE protects confidentiality during cloud-side computation, while SHA-256/HMAC-based verification protects the integrity of stored ciphertext, immutable metadata, and version information. The design contains client preprocessing, key management, HE encryption, digest generation, encrypted cloud storage, homomorphic evaluation, verification, and authorized decryption stages. Performance is assessed through encryption/decryption cost, homomorphic evaluation latency, hash-verification cost, storage expansion, communication overhead, scalability, and security coverage. Recent experimental literature is used as a performance reference rather than fabricating a new benchmark dataset. The analysis indicates that hashing adds comparatively small computational and storage cost, whereas HE dominates runtime and ciphertext growth.

Keywords: Homomorphic encryption; cryptographic hashing; cloud security; SHA-256; HMAC; privacy-preserving computation; integrity verification; performance evaluation.

1. Introduction

Cloud computing has become a foundational platform for enterprise information systems, healthcare analytics, financial services, Internet of Things applications, research computing, and artificial intelligence. Organizations gain scalability and lower infrastructure-management burden by outsourcing data and computation, but outsourcing changes the security boundary. Sensitive information may be stored, replicated, processed, and transmitted through systems that are administratively separated from the data owner. The resulting security problem is not limited to preventing unauthorized reading. Cloud users must also detect unauthorized modification, ensure that processing does not unnecessarily expose plaintext, manage cryptographic keys, and maintain acceptable performance.

Traditional symmetric and public-key encryption provide strong confidentiality for data at rest and in transit, but conventional encrypted information must normally be decrypted before the server can perform meaningful arithmetic or analytical operations. This creates an exposure point during processing. Homomorphic encryption provides an alternative model: permitted mathematical operations are performed on ciphertext, and decryption of the evaluated ciphertext yields a result corresponding to computation on the underlying plaintext. Recent cloud-security literature therefore treats HE as an important privacy-preserving technology, while consistently identifying computational cost and storage overhead as major barriers to widespread deployment.

Integrity creates a second requirement. Even if the cloud cannot read encrypted data, corrupted, substituted, replayed, or maliciously modified ciphertext may still disrupt the service or produce invalid results. Cryptographic hashing offers a lightweight method for representing a data object with a fixed-length digest. A secure digest changes when the protected representation changes, allowing the client to detect tampering when the reference digest itself is trusted. Hashing does not provide confidentiality and HE does not automatically solve every integrity problem. The architecture developed in this study therefore combines the two mechanisms as complementary security layers.



2. Research Problem

The central problem is how to enable useful cloud-side processing while preserving confidentiality and providing explicit verification that protected cloud objects have not been altered outside authorized operations. Encryption-only approaches can protect content but may require decryption for processing. Hashing-only approaches can reveal tampering but do not hide content. Standalone HE enables encrypted computation but adds high overhead and still requires careful integrity, freshness, and metadata protection. A hybrid design must therefore provide stronger security coverage without introducing unnecessary cryptographic work.

- Confidentiality must continue during storage, transmission, and selected cloud computation.
- Integrity verification must identify unauthorized modification of ciphertext and security-relevant metadata.
- The architecture must avoid treating a simple hash as proof that the cloud performed a requested computation correctly.
- Performance evaluation must account for encryption time, evaluation latency, ciphertext expansion, verification time, communication cost, and scalability.
- Key management and authenticated metadata must remain outside an attacker-controlled trust domain.

3. Objectives of the Study

1. To design a layered cloud security architecture integrating homomorphic encryption with cryptographic hashing.
2. To analyze the security role of HE in protecting confidentiality during outsourced computation.
3. To evaluate the contribution of hashing/HMAC to ciphertext and metadata integrity verification.
4. To identify the major computational and storage overheads introduced by the hybrid architecture.
5. To compare conventional encryption, standalone HE, and hybrid HE-plus-hashing approaches on security and performance dimensions.
6. To identify optimization strategies and deployment conditions for practical cloud environments.

4. Background and Related Security Concepts

4.1 Homomorphic Encryption

Homomorphic encryption permits computation over encrypted values. In simplified notation, if $c = \text{Enc}(\text{pk}, m)$ and Eval applies an allowed function f to ciphertext c , then decryption should recover the corresponding plaintext result: $\text{Dec}(\text{sk}, \text{Eval}(\text{pk}, f, c)) = f(m)$. Partially homomorphic systems support a restricted operation, somewhat or leveled HE supports bounded circuits, and fully homomorphic encryption aims to support arbitrary computations. Paillier is widely associated with additive homomorphism, while BFV, BGV, and CKKS are commonly used in modern lattice-based HE implementations. CKKS is particularly relevant to approximate numerical computation, whereas BFV/BGV are suitable for exact modular arithmetic.

The practical benefit is that a cloud provider can process protected values without receiving the secret decryption key. The cost is substantial mathematical work on large ciphertext structures. Recent benchmark studies demonstrate that performance varies by scheme and library; therefore, the architecture should not bind itself unnecessarily to a single implementation. Instead, the HE component should be selected according to data type, permitted operations, precision, depth, and latency requirements.

4.2 Cryptographic Hashing and HMAC

A cryptographic hash function maps variable-length input to a fixed-length digest. SHA-256 generates a 256-bit output and is widely used in integrity mechanisms. Collision resistance makes it computationally infeasible to deliberately construct two distinct inputs with the same digest under normal security assumptions. For cloud integrity, the client can compute a digest over ciphertext and immutable metadata before upload and compare it with a recomputed digest after retrieval. However, an unkeyed digest stored next to the data can be replaced by an attacker who controls both objects. HMAC adds a secret key to the integrity calculation and is therefore preferable when the same authorized domain can securely maintain the MAC key. Digital signatures are another option when public verification is required.

4.3 Need for Hybridization

Security Requirement	HE Contribution	Hash/HMAC Contribution
Data confidentiality	Keeps protected values encrypted, including during supported computation.	Does not conceal data.
Integrity	Limited unless combined with authentication/proof mechanisms.	Detects unauthorized changes to protected representation.
Encrypted computation	Core capability.	No computational privacy.
Metadata binding	Can encrypt selected metadata but may be expensive.	Efficiently binds IDs, versions and policy fields.
Tamper detection	Not a substitute for an integrity protocol.	Primary role when digest/tag is trusted.

5. Proposed Hybrid Cloud Security Architecture

The proposed architecture separates the data path into security-specific stages. The client retains control of secret keys and prepares the object before outsourcing. A homomorphic encryption layer transforms protected numerical or structured fields into ciphertext. In parallel, an integrity layer creates a trusted digest or HMAC over the ciphertext and immutable metadata. The cloud stores ciphertext and performs only authorized homomorphic functions. When an object or result is returned, the client first validates applicable integrity information and metadata, then decrypts the authorized ciphertext. This ordering prevents unverified data from being silently accepted.

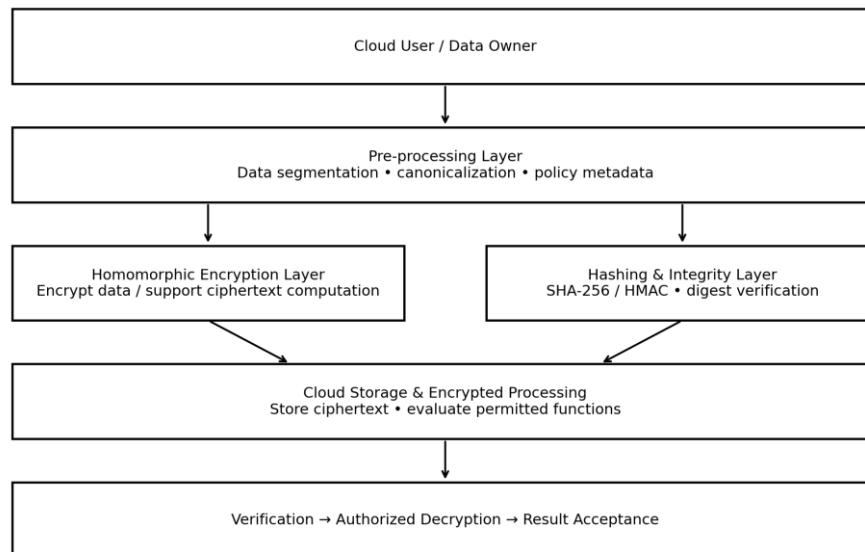


Figure 1. Proposed hybrid homomorphic-encryption and hashing-based cloud security architecture.

5.1 Architectural Components

- Client/Data Owner: prepares data, controls keys, defines allowed computations, and maintains trusted verification information.
- Pre-processing Module: canonicalizes data, segments fields, attaches object identifiers, versions, and policy metadata.
- HE Module: generates keys, encrypts sensitive computational fields, and decrypts authorized outputs.
- Hash/HMAC Module: creates and verifies integrity tags for ciphertext and immutable metadata.
- Cloud Storage Layer: stores encrypted objects without requiring plaintext access.
- Encrypted Processing Layer: evaluates permitted functions over HE ciphertext.



- Audit/Logging Layer: records object versions, verification events, and security failures without exposing protected content.

5.2 Processing Algorithm

1. Generate HE public/secret keys and, where HMAC is used, a separate integrity key.
2. Canonicalize the data and assign a unique object identifier and version.
3. Encrypt computation-sensitive data with the selected HE scheme.
4. Construct immutable metadata and compute an integrity value over ciphertext plus metadata.
5. Upload ciphertext and required metadata to cloud storage while protecting the trusted verification value.
6. Execute the authorized homomorphic function on ciphertext without exposing the secret key.
7. Retrieve the requested object or result and validate identifiers, versions, and integrity information.
8. Decrypt only after applicable verification succeeds; reject or quarantine data when verification fails.

5.3 Integrity Scope

The design distinguishes storage integrity from computational correctness. A hash of an uploaded ciphertext can show whether that ciphertext has changed since the trusted digest was generated. It cannot prove that a newly generated homomorphic result is mathematically the correct output of the requested function. A digest of the result can protect that result against later alteration, but correctness against a malicious evaluator requires an additional mechanism such as a homomorphic signature, zero-knowledge proof, SNARK, trusted execution evidence, or another verifiable-computation technique. This boundary prevents the architecture from overstating the security provided by hashing.

6. Threat Model and Security Analysis

The baseline threat model assumes a cloud that may be honest-but-curious, faulty, or partially compromised. The provider may inspect ciphertext and metadata, attempt to infer information, lose or replace objects, replay older valid objects, or return altered stored data. External attackers may intercept network traffic or obtain unauthorized cloud credentials. The design assumes that cryptographic algorithms and parameters are securely selected and that client-side secret keys are not compromised.

Threat	Architecture Response	Residual Concern
Cloud reads outsourced data	HE keeps protected fields encrypted.	Metadata/access-pattern leakage may remain.
Ciphertext modification	Hash/HMAC verification detects changed representation.	Reference tag must itself be trusted.
Object substitution	Bind object ID and context into authenticated metadata.	Incorrect metadata design can permit substitution.
Rollback/replay	Authenticate version/counter/timestamp policy.	Freshness source must be trustworthy.
Key theft	Key isolation and rotation reduce exposure.	Cryptography cannot protect data after secret-key compromise.
False computation result	HE preserves confidentiality, not automatically correctness.	Add verifiable-computation proof for malicious evaluator model.

7. Performance Evaluation Framework

Performance evaluation is organized around measurable components rather than a single overall speed number. The architecture should be tested with fixed security parameters, documented hardware, a defined HE library, representative data sizes, and repeatable workloads. The most important metrics are listed below.

- Encryption latency: time required to convert client plaintext into HE ciphertext.
- Homomorphic evaluation latency: time required for cloud-side addition, multiplication, aggregation, or application-specific circuits.

- Hash/HMAC latency: time required to generate and verify integrity values.
- Decryption latency: time required to recover the authorized result.
- Ciphertext expansion ratio: encrypted object size divided by original protected data size.
- Communication overhead: additional bytes transmitted for ciphertext, tags, keys/metadata, and proofs.
- Throughput: number of protected operations or objects processed per unit time.
- Scalability: change in latency/throughput as file count, data size, user count, or circuit depth increases.
- Resource utilization: CPU, memory, and, where relevant, accelerator usage.

7.1 Evidence from Recent Performance Studies

Recent published experiments demonstrate why HE dominates the performance discussion. A 2025 edge-cloud simulation comparing standard HE, lazy bootstrapping, and a hybrid HE configuration reported that its hybrid configuration reduced encryption time by 76% and ciphertext size by 97.7% relative to the study's standard HE configuration. These percentages are specific to that experimental design and should not be generalized to every workload, but they show that architecture and optimization strategy can materially affect feasibility.

A 2025 benchmark of Microsoft SEAL, HELib, OpenFHE, and Lattigo across BGV, BFV, and CKKS likewise found that the fastest library depended on the scheme and operation. The study reported strong SEAL performance in CKKS and low memory use, while OpenFHE performed competitively in BFV operations. This supports a workload-specific selection strategy rather than declaring one HE library universally superior.

A 2026 Scientific Reports cloud-storage framework combining hybrid encryption, HMAC-SHA256, Paillier-based encrypted search, and integrity verification reported 245.7 ms encryption time for 1 MB files and 7.3% storage overhead in its own implementation. The same study identified Paillier operations as a major contributor to cryptographic processing cost. These results are useful reference points for hybrid design, although the architecture in the present study is not identical and therefore should not claim those measurements as its own.

7.2 Analytical Cost Model

Let T_{total} denote client-to-result cryptographic processing time. A simplified model can be written as $T_{\text{total}} = T_{\text{prep}} + T_{\text{HE-enc}} + T_{\text{hash}} + T_{\text{upload}} + T_{\text{eval}} + T_{\text{download}} + T_{\text{verify}} + T_{\text{HE-dec}}$. For large or computationally deep workloads, T_{eval} and $T_{\text{HE-enc}}$ are expected to dominate, whereas T_{hash} and T_{verify} are normally much smaller. The model also clarifies where optimization should occur: reducing multiplicative depth, batching values, parallelizing independent ciphertext operations, and selecting appropriate parameters can produce larger gains than attempting to optimize an already inexpensive hash operation.

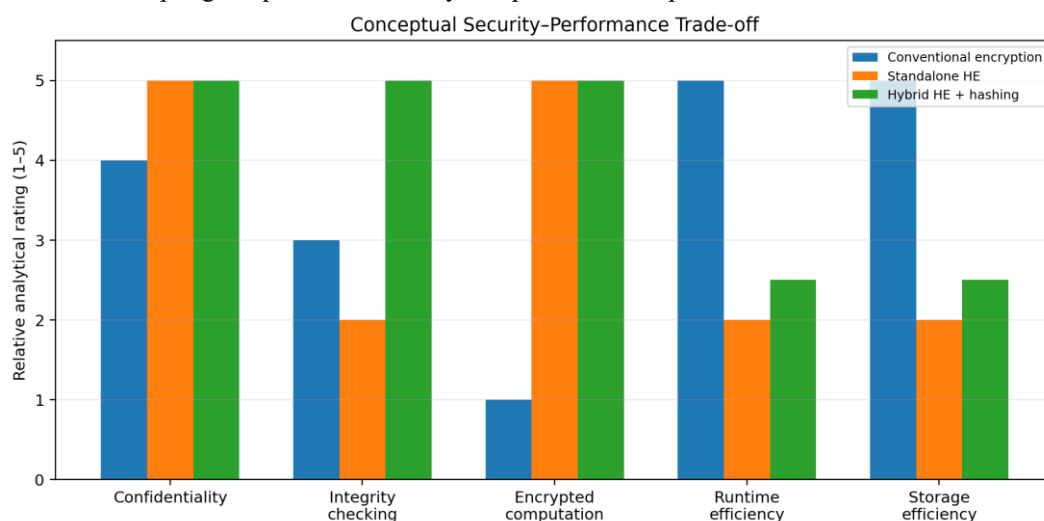


Figure 2. Conceptual security-performance trade-off



8. Comparative Analysis

Criterion	Conventional Encryption	Standalone HE	Hybrid HE + Hashing
Confidentiality at rest	Strong	Strong	Strong
Confidentiality during supported cloud computation	Usually requires plaintext exposure	Strong	Strong
Explicit ciphertext integrity checking	With authenticated mode/additional mechanism	Not inherently complete	Integrated hash/HMAC layer
Computational overhead	Low	High–very high	High–very high; hashing adds little
Storage overhead	Low	High	High plus small tag/metadata
Implementation complexity	Low–moderate	High	High
Best fit	Storage/transport protection	Privacy-preserving computation	Privacy-preserving computation plus explicit integrity assurance

The comparison shows that the hybrid architecture is not intended to replace efficient conventional encryption in every cloud application. If the cloud only stores files and never needs to compute on protected values, authenticated symmetric encryption can be simpler and faster. The hybrid architecture becomes valuable when encrypted computation is a functional requirement and the owner also needs explicit tamper detection. Its performance penalty is justified only when those requirements are genuine.

9. Optimization Strategies

- Use partially or leveled HE when the application does not require arbitrary fully homomorphic computation.
- Batch multiple values into ciphertext slots when supported by the chosen scheme.
- Minimize multiplicative depth because multiplication and bootstrapping are typically much more expensive than simple additions.
- Separate bulk non-computational file content from fields that actually require HE; protect bulk content with efficient authenticated encryption where appropriate.
- Compute hashes incrementally or with Merkle structures for large block-based objects when partial verification is needed.
- Parallelize independent encryption/evaluation tasks across cloud CPU cores or accelerators.
- Cache safe public parameters while protecting secret material.
- Benchmark BFV/BGV/CKKS or Paillier against the actual workload before deployment rather than selecting by popularity.

10. Application Scenarios

10.1 Healthcare Analytics

Sensitive numerical health records can remain encrypted while an approved cloud service performs aggregate statistics or selected predictive computations. The hash/HMAC layer verifies that encrypted records and version metadata have not been silently modified. Regulatory compliance, patient consent, and access governance remain separate requirements and should not be assumed to follow automatically from cryptography.

10.2 Financial Cloud Processing

Banks and financial platforms can use HE for selected risk aggregation, encrypted scoring, or confidential statistics. Integrity tags can bind transaction batches, object identifiers, and versions. Because financial systems are latency-sensitive, the architecture should isolate only the fields and computations that require HE instead of encrypting every



operation homomorphically.

10.3 IoT and Edge–Cloud Systems

Edge devices can preprocess and protect sensitive readings before cloud aggregation. Resource constraints make lightweight preprocessing, batching, and partial HE particularly important. Recent edge-cloud studies suggest that hybrid designs can substantially reduce overhead compared with unoptimized HE under specific configurations, making this an important deployment area for further experimentation.

11. Limitations and Research Challenges

- HE remains computationally expensive compared with plaintext and conventional encrypted workflows.
- Ciphertext expansion increases storage, memory, and network cost.
- Parameter selection is technically demanding and errors can weaken security or correctness.
- A hash verifies a defined representation but does not prove that a remote computation was executed correctly.
- Unprotected metadata can leak object relationships, timing, sizes, or access patterns.
- Multi-user key sharing, revocation, and delegation complicate enterprise deployment.
- Performance results from one HE scheme, library, hardware platform, or dataset cannot be directly generalized to all cloud systems.
- Future cryptographic migration and post-quantum security planning require algorithm agility.

12. Discussion

The architecture demonstrates a clear division of cryptographic responsibility. HE is used where confidentiality must survive computation; hashing or HMAC is used where efficient integrity checking is required. This separation avoids the common design error of treating encryption and integrity as interchangeable. It also improves performance reasoning because the dominant HE cost can be optimized independently from the lightweight verification layer.

Performance evidence from recent research suggests that practical HE is moving toward optimized, workload-specific configurations rather than one universal FHE pipeline. Hybridization, batching, reduced bootstrapping, and careful library selection can materially improve efficiency. Nevertheless, the exact magnitude of improvement depends on experimental conditions. A scientifically defensible evaluation of the proposed architecture should therefore report security parameters, hardware, software versions, dataset characteristics, operation counts, and repeated-run statistics.

From a security perspective, the architecture should prefer authenticated integrity rather than a bare digest stored beside the cloud object. HMAC is appropriate when verification remains within a shared secret domain; digital signatures are preferable when independent public verification is needed. If malicious computation is in scope, an additional proof-of-correctness layer is required. These choices allow the architecture to scale from an honest-but-curious cloud model to a stronger adversarial model without misrepresenting the guarantees of HE or hashing.

13. Conclusion and Future Scope

This analytical study designed and evaluated a hybrid homomorphic encryption and hashing-based cloud security architecture. The proposed design preserves confidentiality during selected cloud-side computation through HE and adds an explicit integrity layer through SHA-256/HMAC-based verification of ciphertext and immutable metadata. The performance analysis indicates that the principal cost originates from HE encryption, ciphertext expansion, and homomorphic evaluation, while hashing contributes comparatively small overhead. Consequently, practical deployment depends less on optimizing the hash function and more on selecting an HE scheme appropriate to the workload, minimizing circuit depth, batching data, and avoiding homomorphic processing for fields that do not require it.

Future work should implement the architecture with at least two representative HE configurations, such as BFV for exact integer operations and CKKS for approximate numerical analytics, and benchmark them against a conventional authenticated-encryption baseline. Experimental evaluation should measure encryption, hashing, upload, evaluation, verification, decryption, throughput, memory consumption, ciphertext expansion, and communication cost across



increasing data sizes. Additional research should integrate Merkle-tree verification for block-level integrity, threshold or hardware-backed key management, zero-knowledge or succinct proofs for computation correctness, and adaptive workload routing between conventional and homomorphic encryption.

References

1. Ayitey Junior, M., Appiahene, P., Appiah, O., et al. (2025). Cloud data privacy protection with homomorphic algorithm: A systematic literature review. *Journal of Cloud Computing*, 14, 84. <https://doi.org/10.1186/s13677-025-00774-5>
2. Ahmed, H. M. A., Zubair, S., & Tawfik, M. (2026). A privacy-preserving cloud storage framework with hybrid encryption, homomorphic keyword search, and blockchain-based integrity verification. *Scientific Reports*, 16, 16697. <https://doi.org/10.1038/s41598-026-48705-x>
3. Comparative Simulation of Homomorphic Encryption Techniques in Edge-Cloud Computing: Reducing Computational and Storage Overhead. (2025). *Procedia Computer Science*, 269, 438–448. <https://doi.org/10.1016/j.procs.2025.08.296>
4. Performance Analysis of Leading Homomorphic Encryption Libraries: A Benchmark Study of SEAL, HELib, OpenFHE, and Lattigo. (2025). *Proceedings of the 2025 4th International Conference on Cyber Security, Artificial Intelligence and the Digital Economy*. <https://doi.org/10.1145/3729706.3729711>
5. Selvi, P., & Sakthivel, S. (2025). A hybrid ECC-AES encryption framework for secure and efficient cloud-based data protection. *Scientific Reports*, 15, 30867. <https://doi.org/10.1038/s41598-025-01315-5>
6. Fully homomorphic encryption-based optimal key encryption for privacy preservation in the cloud sector. (2025). *Journal of Information Security and Applications*, 91, 104048. <https://doi.org/10.1016/j.jisa.2025.104048>
7. Parthasarathy, K., Panga, N. K. R., Bobba, J., Bolla, R. L., Ayyadurai, R., & Hemnath, R. (2025). Fortifying cloud security with advanced data encryption technique. *Journal of Science & Technology*, 10(3), 20–27. <https://doi.org/10.46243/jst.2025.v10.i03.pp20-27>
8. Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, 169–178.
9. Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. *Advances in Cryptology—EUROCRYPT '99*, 223–238.
10. Brakerski, Z., Gentry, C., & Vaikuntanathan, V. (2014). (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory*, 6(3), Article 13.
11. National Institute of Standards and Technology. (2015). Secure Hash Standard (SHS), FIPS PUB 180-4.
12. Boneh, D., Goh, E.-J., & Nissim, K. (2005). Evaluating 2-DNF formulas on ciphertexts. *Theory of Cryptography Conference*, 325–341.

