A Project Report On

Identity-Based Cloud Integrity Auditing With Data Hiding

Network and System Security (SIL765) |Endterm Project Report |2024–2025

Group Members

Arindam Sal
Entry Number: 2024JCS2041

Ayan Shil
Entry Number: 2024JCS2048

Course Instructor

Prof. Huzur Saran
Department of Computer Science and Engineering
Indian Institute of Technology Delhi

Submitted on: 26nd April, 2025

Indian Institute of Technology Delhi

M.Tech in Cyber Security

Contents

Abstract

In the era of cloud storage, ensuring **privacy-preserving data sharing** with efficient **integrity auditing** is paramount. This project proposes a cloud-based system where users can **share data** while safeguarding **sensitive information** through automated sanitization. Leveraging an **identity-based cryptographic model**, our approach **protects sensitive data** by replacing confidential content with wildcards while retaining verifiability. A **Third-Party Auditor (TPA)** ensures integrity without accessing actual data. Our implementation demonstrates a practical solution for sectors requiring **secure and privacy-preserving data sharing**, including healthcare and finance.

Our original selected paper titled

"Enabling Identity-Based Integrity Auditing and Data Sharing With Sensitive Information Hiding for Secure Cloud Storage"

Wenting Shen, Jing Qin, Jia Yu, Rong Hao, and Jiankun Hu, *Senior Member, IEEE*

1 Introduction

Area: Cloud storage has emerged as a vital tool in the modern era, enabling organizations and individuals to store and share vast amounts of data without the burden of maintaining local storage infrastructure. Cloud services like Google Drive, Dropbox, and iCloud have transformed data management by offering high availability and scalability. However, this convenience comes with significant concerns regarding the reliability and trustworthiness of the cloud providers. Data stored remotely is vulnerable to corruption due to software bugs, hardware failures, human errors, or even malicious tampering. Therefore, verifying the integrity of cloud-stored data has become a critical research area in network and system security.

Problem: While remote data integrity auditing schemes have been proposed to ensure correctness of cloud data, most existing solutions are designed for scenarios where only the data owner needs assurance. In contrast, when data sharing is involved, particularly for sensitive information (e.g., healthcare, finance records), additional privacy challenges arise. Encrypting the entire file hides sensitive information but makes the data unusable for legitimate sharing and analysis. Furthermore, practical challenges such as secure key distribution and unpredictable future users make full encryption solutions inefficient. Thus, there is a pressing need for an auditing framework that allows secure data sharing while selectively hiding sensitive information without compromising the utility of the remaining data.

Solution: This project addresses the above challenge by proposing an identity-based integrity auditing scheme with sensitive information hiding. A dedicated sanitizer selectively blinds sensitive data blocks by replacing them with wildcards, thereby preserving confidentiality. Simultaneously, the system allows for public verification of data integrity by a Third Party Auditor (TPA) without exposing private data. Our approach uses identity-based cryptography to simplify key management and supports selective sanitization without invalidating the integrity verification process. Additionally, in the endterm phase, several extensions like tamper-evident logging, multi-layered blinding for improved unlinkability, and parallelized verification were implemented to enhance security, efficiency, and auditability.

Evaluation: The effectiveness of the proposed solution is demonstrated through a practical implementation, including all core algorithms such as Setup, Extract, SigGen, Sanitization, ProofGen, and ProofVerify. Performance metrics such as operation duration, proof generation time, and verification efficiency were measured. Extensions introduced during the endterm phase were evaluated qualitatively and quantitatively using realistic file sizes and simulated experiments. Graphical comparisons between the original and extended models further illustrate significant improvements in system robustness, auditability, and privacy protection.

Takeaway: In conclusion, this project offers a robust and scalable approach for secure data sharing in cloud storage systems, addressing the critical gap left by existing integrity auditing schemes. By combining identity-based cryptography, selective sanitization, tamper-evident logging, and parallelized proof verification, the system ensures both

data integrity and privacy preservation. The proposed enhancements make the solution practical for real-world applications where secure and efficient data sharing is essential.

2 Background and Related Work

2.1 Background

With the explosive growth of data, it has become a heavy burden for users to store a large amount of data locally. As a result, many organizations and individuals prefer to store their data in the cloud. However, data stored in the cloud is susceptible to corruption or loss due to software bugs, hardware faults, and human errors. To verify the integrity of the data stored in the cloud, numerous remote data integrity auditing schemes have been proposed [1]-[6].

In remote data integrity auditing schemes, a data owner generates signatures for data blocks before uploading them to the cloud. These signatures are used to prove the possession of data blocks during the integrity auditing process. Many cloud storage applications such as Google Drive, Dropbox, and iCloud support data sharing across multiple users. However, shared data often contains sensitive information that needs protection from unauthorized access. For example, Electronic Health Records (EHRs) stored and shared in the cloud may include sensitive details like patient names, contact information, and hospital identifiers. If these EHRs are uploaded directly to the cloud for research purposes, sensitive information may be exposed to unauthorized entities.

A potential solution to this problem is encrypting the entire shared file before uploading it to the cloud. While this approach effectively hides sensitive information, it also renders the data unusable for legitimate users. Distributing decryption keys is not a viable solution due to the need for secure channels and the difficulty in predicting potential users of the data. Therefore, a new approach is required to achieve data sharing with sensitive information hiding while ensuring data integrity.

2.2 Related Work

Various remote data integrity auditing schemes have been proposed to verify the integrity of data stored in the cloud. Ateniese et al. [1] introduced the concept of Provable Data Possession (PDP) to ensure data possession on untrusted cloud servers using homomorphic authenticators and random sampling techniques. Shacham and Waters [2] proposed a private and public remote data integrity auditing scheme based on BLS signatures. Wang et al. [3] designed a privacy-preserving remote data integrity auditing scheme employing random masking techniques, which was later improved by Worku et al. [4] to enhance efficiency.

To address user-side computation burdens, Guan et al. [5] introduced a remote data integrity auditing scheme based on indistinguishability obfuscation, while Shen et al. [6] proposed a lightweight scheme utilizing a Third Party Medium (TPM) for signature generation. Data dynamics were addressed by Ateniese et al. [7] and Wang et al. [8] through the use of skip lists and Merkle Hash Trees, respectively. Additionally, key-exposure resilience was explored by Yu et al. [9] and Wang et al. [10] to mitigate the impact of key exposure.

The challenge of identity privacy in data sharing was tackled by Wang et al. [11] through a privacy-preserving shared data integrity auditing scheme utilizing ring signa-

tures. Yang et al. [12] developed an efficient scheme that supports identity privacy and identity traceability, while Fu et al. [13] proposed a privacy-aware auditing scheme using homomorphic verifiable group signatures. Wang et al. [14] and Luo et al. [15] further extended these schemes to support user revocation and efficient key management.

Although these schemes addressed various aspects of data integrity auditing, they did not consider the problem of sharing data with sensitive information hiding. The proposed approach aims to address this issue by introducing a sanitizer that sanitizes sensitive data blocks and transforms signatures for the sanitized file while ensuring data integrity.

Table 1: Functionality Comparison With Existing Related Schemes

Schemes	**Public Verifiability**	**Certificate Management Simplification**	**Data Sharing**	**Sensitive Information Hiding**
Shacham et al. [2]	Yes	No	No	No
Wang et al. [14]	Yes	Yes	No	No
Li et al. [16]	Yes	Yes	No	No
Wang et al. [18]	Yes	No	No	No
Shen et al. [17]	Yes	No	No	No
Ours	Yes	Yes	Yes	Yes

3 Problem Statement

The proposed project titled **"Enhancing Secure Cloud Storage with Sensitive Information Hiding"** aims to build a comprehensive framework that ensures the integrity and confidentiality of sensitive information in cloud storage systems. By utilizing identity-based cryptography, sanitization mechanisms, and efficient integrity auditing techniques, the project seeks to provide privacy-preserving data sharing while maintaining data integrity. The following sections describe the system model, threat model, and key challenges associated with this approach.

3.1 System Model

The proposed system consists of five entities: the Cloud, the User, the Sanitizer, the Private Key Generator (PKG), and the Third Party Auditor (TPA). The interaction between these entities is illustrated in Fig. 1.

- **Cloud:** Provides storage services, allowing users to upload and share their data with others. It also responds to audit challenges initiated by the TPA. As a storage provider, the cloud has the potential to modify or delete files, whether accidentally or intentionally. To ensure data integrity, it is essential to employ robust mechanisms for verifying data correctness during auditing.

- **User:** A member of an organization who possesses sensitive data that needs to be stored and shared securely in the cloud. The user blinds sensitive parts of the file, generates signatures, and sends them to the Sanitizer. Users also require a simple yet secure key management system to avoid complexity and security risks.

- **Sanitizer:** Processes the blinded file, transforming sensitive information into wildcards, ensuring privacy. It also converts the corresponding signatures to be valid for the sanitized file and uploads them to the cloud. However, the sanitizer itself may not be entirely trustworthy and might attempt to infer sensitive data during processing.
- **PKG:** A trusted entity responsible for generating system-wide public parameters and user-specific private keys based on their identity. Its role is crucial for maintaining the integrity of identity-based cryptography, where user identities are mapped directly to their cryptographic keys, eliminating the need for complex certificate management.
- **TPA:** Acts as a public verifier, responsible for verifying the integrity of files stored in the cloud through periodic auditing requests. The TPA is assumed to be semi-trusted, meaning it is honest-but-curious and may attempt to learn sensitive information during auditing.

The overall goal of the system model is to ensure data privacy, integrity, and accessibility even when dealing with untrusted parties, such as the cloud and the sanitizer.

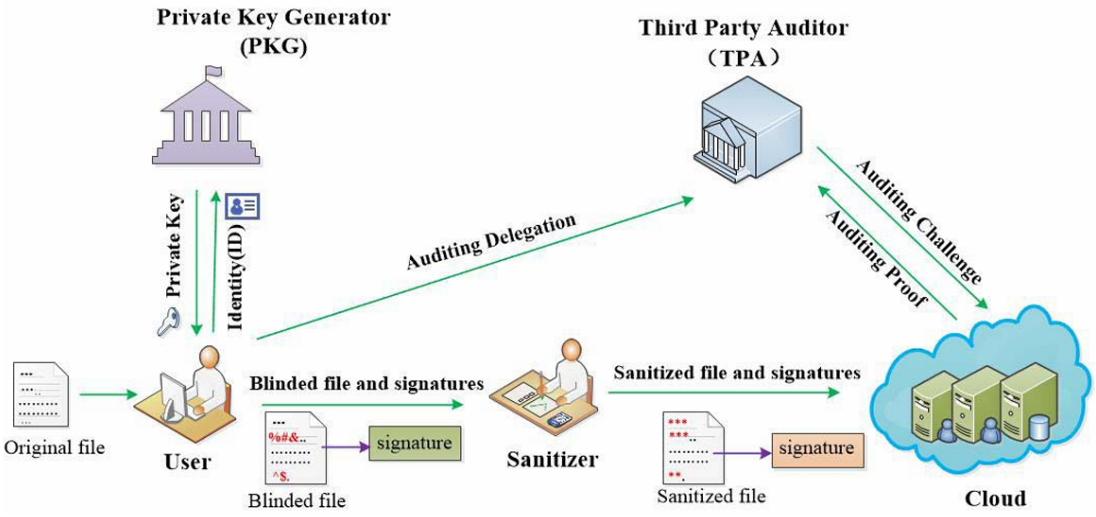

Figure 1: System Model of the Proposed Scheme

3.2 Threat Model

The proposed system considers several potential threats:

- The cloud server is considered untrustworthy and may delete or alter data maliciously or due to internal failures. It may also attempt to forge responses during the integrity verification phase to deceive the auditor.
- The sanitizer, although trusted for sanitization, is not fully reliable. It may attempt to infer personal sensitive information during the processing phase. Additionally, the sanitizer may fail to sanitize data correctly, leading to accidental data leaks.

- Malicious users or unauthorized entities may try to access sensitive data stored in the cloud, compromising its confidentiality and integrity. This can include insider threats, external hackers, or even compromised auditors.
- The TPA may collude with the cloud or sanitizer to gain unauthorized access to sensitive information, thereby compromising the system's privacy guarantees.
- The complexity of key management in conventional systems can lead to misconfigurations and unauthorized access. In our identity-based approach, improper handling of private keys could also result in data breaches.
- Additionally, audit operations might not detect tampering if the cloud restores an earlier clean state before audits. Without maintaining an immutable operation history, past attacks could remain undetected.
- Privacy leakage risks could arise if blinded blocks remain partially linkable across files or users, allowing adversaries to correlate patterns between different sanitized files.
- High auditing time and network overhead may expose the system to side-channel timing attacks and create scalability bottlenecks if the verification of challenged blocks is not optimized.

3.3 Key Challenges

The key challenges addressed by the proposed scheme include:

- **Ensuring Data Integrity:** The system must verify the integrity of files stored in the cloud without revealing sensitive data to the TPA or other unauthorized entities. This requires a mechanism that can distinguish between sanitized and unsanitized data blocks.
- **Sensitive Information Hiding:** The proposed scheme must ensure that sensitive information is protected during the sanitization process and cannot be inferred by the sanitizer or other entities. Maintaining confidentiality during data sharing while supporting data utility is critical.
- **Efficient Auditing Mechanism:** The integrity verification process should be efficient and lightweight, allowing frequent audits without imposing significant overhead on the cloud or user. Parallelization strategies and compact proof structures could be beneficial to achieve scalability.
- **Certificate Management Simplification:** The use of identity-based cryptography simplifies key management by eliminating the need for complex certificate management, but this also requires robust handling of identity-to-key mapping.
- **Scalability and Robustness:** The proposed system must be robust enough to handle large-scale data while ensuring privacy, integrity, and security. The scheme should also be scalable to accommodate dynamic data operations without compromising security.

- **Mitigating Collusion Attacks:** The scheme must be resilient to potential collusion between various entities such as the TPA, sanitizer, and cloud server. Preventing unauthorized information leakage requires careful design and analysis.
- **Maintaining Tamper-Evident Operation History:** To prevent rollback attacks and undetected tampering, it is important to log operations securely using tamper-evident mechanisms like hash chain logging.
- **Strengthening Blinding for Privacy:** Simple additive blinding might not be sufficient against advanced attacks. Enhancing blinding techniques with additional randomization can improve unlinkability and user privacy.

The proposed scheme aims to address these challenges through a combination of identity-based cryptography, sanitization mechanisms, efficient integrity auditing, tamper-evident logging, multi-layered blinding, and parallelized proof verification. It ensures that sensitive information remains protected while allowing authorized access and verification by trusted entities.

4 Proposed Solution

The proposed solution aims to address the challenges of data sharing with sensitive information hiding in identity-based integrity auditing for secure cloud storage. The solution incorporates mechanisms to achieve private key correctness, blinded file verification, auditing correctness, sensitive information hiding, and auditing soundness. The following subsections describe the various components of the proposed scheme.

4.1 Design Goals

The proposed scheme is designed to achieve the following goals:

- **Private Key Correctness:** Ensuring that the private key generated by the PKG can pass the verification by the user.
- **Blinded File Verification:** Guaranteeing that the user's blinded file and its corresponding signatures can pass the sanitizer's verification.
- **Auditing Correctness:** Ensuring that sanitized data stored by the cloud can pass verification by the TPA.
- **Sensitive Information Hiding:** Ensuring personal and organizational sensitive information is protected during sharing and storage.
- **Auditing Soundness:** Guaranteeing that unauthorized or modified data cannot pass the verification of the TPA.

4.2 System Design

The proposed scheme involves six algorithms: Setup, Extract, SigGen, Sanitization, ProofGen, and ProofVerify. These algorithms ensure the integrity and privacy of the stored and shared data.

4.2.1 Setup Algorithm

The **Setup** algorithm is run by the PKG and generates the master secret key msk and the system public parameters pp. The PKG chooses two multiplicative cyclic groups G_1 and G_2 of prime order p, a generator g of G_1, a bilinear map $e : G_1 \times G_1 \to G_2$, and a pseudo-random function $f : \mathbb{Z}_p^* \times \mathbb{Z}_p^* \to \mathbb{Z}_p^*$. The master secret key $msk = g_2^x$ and the public key $g_1 = g^x$ are generated, where x is randomly chosen from $\mathbb{Z}_p^*$. The PKG publishes $pp = (G_1, G_2, p, e, g, \mu_1, \mu_2, \ldots, \mu_l, g_1, g_2, H, f)$. This process is shown in Fig.2.

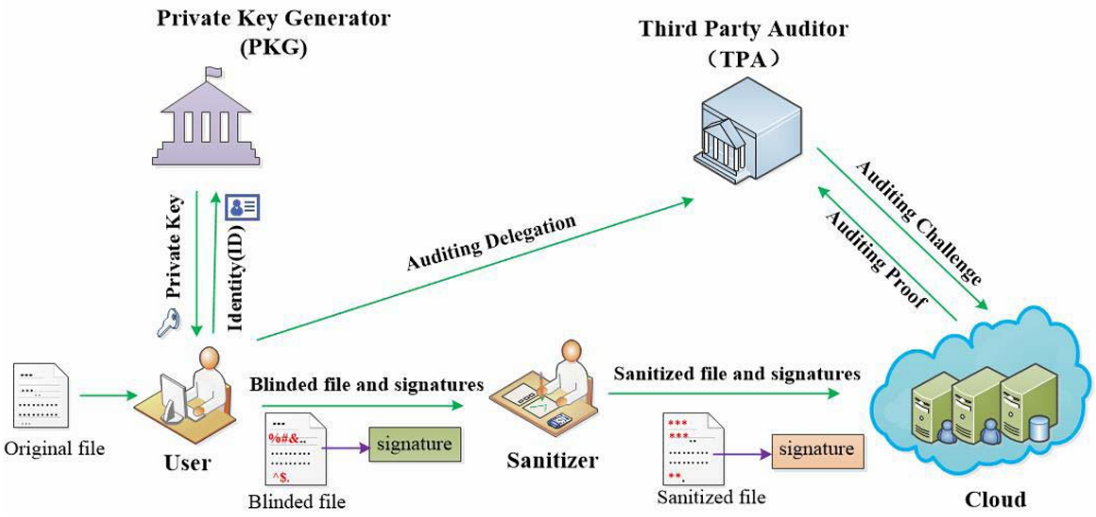

Figure 2: System Model of the Proposed Scheme

4.2.2 Extract Algorithm

The **Extract** algorithm allows the PKG to generate a private key sk_{ID} for the user based on their identity $ID = (ID_1, ID_2, \ldots, ID_l)$. The PKG picks a random value $r_{ID} \in \mathbb{Z}_p^*$ and computes the private key as:

$$sk_{ID} = \left(g^x, \left(\mu' \prod_{j=1}^{l} \mu_j^{ID_j} \right)^{r_{ID}}, g^{r_{ID}} \right)$$

The user verifies the key using:

$$e(sk_{ID}, g) = (g_1, g_2) \cdot e\left(\mu' \prod_{j=1}^{l} \mu_j^{ID_j}, sk''_{ID} \right)$$

This process is shown in Fig.3.

4.2.3 SigGen Algorithm

The **SigGen** algorithm is performed by the user. It generates a blinded file F^* and corresponding signatures. The user chooses a value $r \in \mathbb{Z}_p^*$, computes g^r, and generates a verification value g^r. The signature σ_i for each block m_i^* is calculated as:

$$\sigma_i = g^x \left(\mu' \prod_{j=1}^{l} \mu_j^{ID_j} \right)^{r_{ID}} H(\text{name}||i) \cdot u_i^{*\gamma}$$

This process is illustrated in Fig.4.

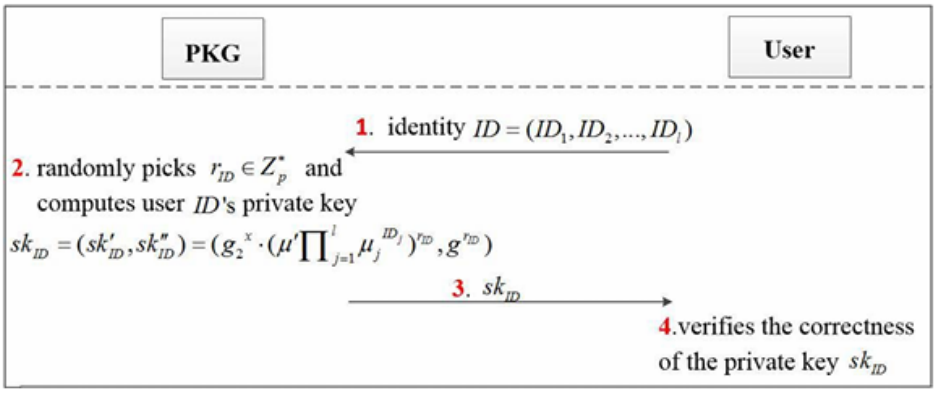

Figure 3: Process of Private Key Generation

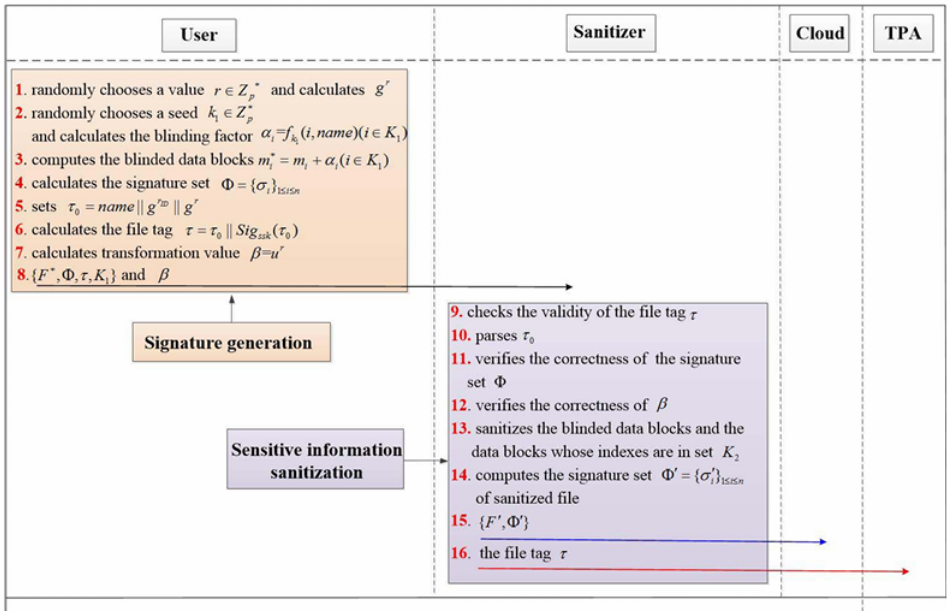

Figure 4: Signature Generation and Sensitive Information Sanitization

4.2.4 Sanitization Algorithm

The **Sanitization** algorithm is executed by the sanitizer, transforming the blinded file into a sanitized file. It also modifies the signatures for the data blocks to make them compatible with the sanitized file. Fig.5 illustrates the sanitization process.

4.2.5 ProofGen and ProofVerify Algorithms

The **ProofGen** algorithm is run by the cloud to generate a proof of data possession. The **ProofVerify** algorithm is performed by the TPA to check the correctness of the proof. The verification equation used by the TPA is:

$$e(\sigma,g)=e(g_1,g_2)^{\sum_{i\in I}v_i}\cdot e\left(\prod_{j=1}^{l}\mu_j^{ID_j},g^{r_{ID}}\right)^{\sum_{i\in I}v_i}\cdot e\left(H(\text{name}||i),u_i^*\right)^{\sum_{i\in I}v_i}$$

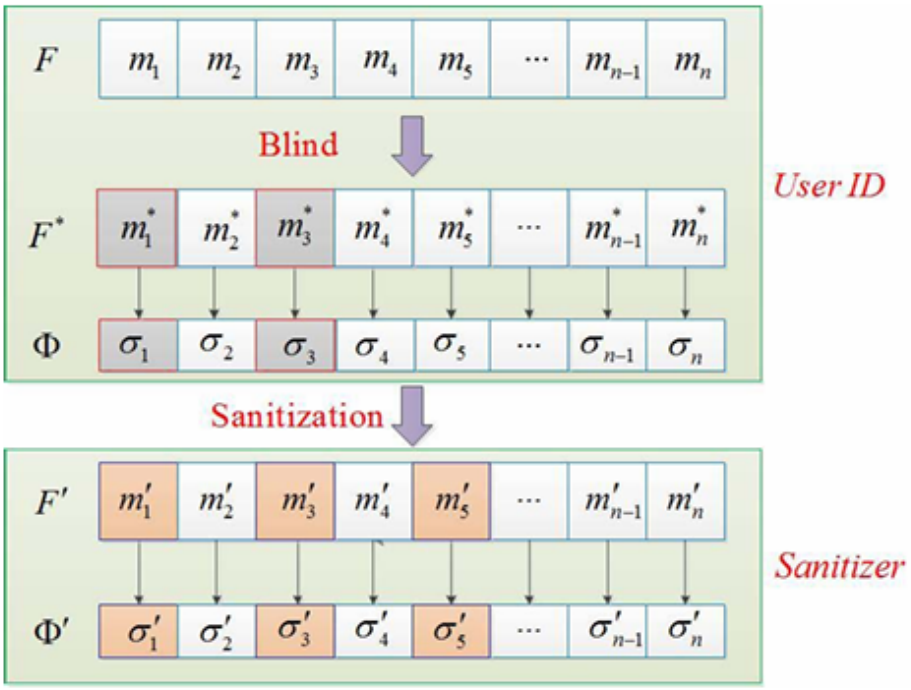

Figure 5: Sensitive Information Sanitization Process

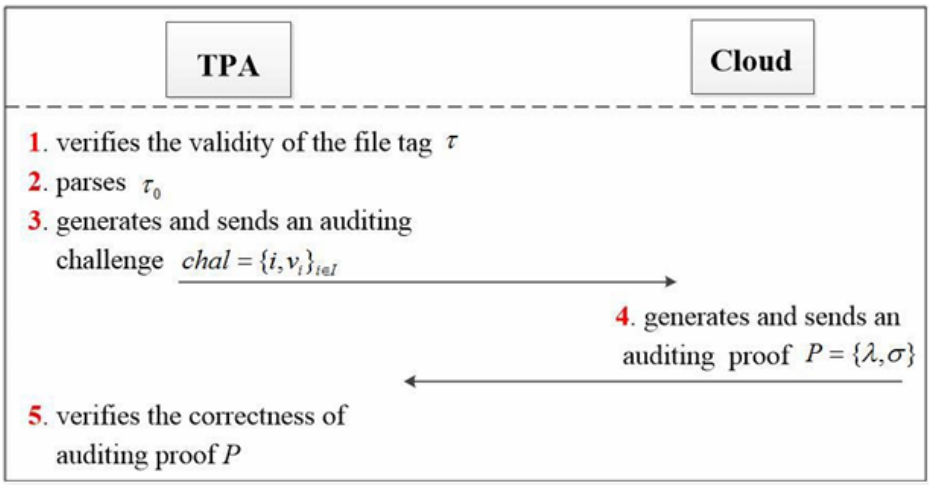

Figure 6: Proof Generation and Verification Process

4.3 Extensions Implemented

While the original model proposed an identity-based integrity auditing system with sensitive information hiding, we further enhanced the basic model with the following significant extensions to address additional real-world performance and security needs:

- **Extension 1: Tamper-Evident Logging and Traceability**
  We integrated a tamper-evident logging system that records each proof generation and verification action using a secure hash chain. Each event entry is cryptographically linked to the previous entry, ensuring immutability. In case tampering is

detected, the system can trace back to the exact point of corruption, improving audit transparency.

- **Extension 2: Parallelized TPA Verification**
  To reduce verification delays, we parallelized the Third Party Auditor (TPA) verification process using concurrent execution. Multiple proof verifications now happen simultaneously, significantly reducing total verification time. This makes the auditing process more scalable for large datasets without compromising verification accuracy.

- **Extension 3: Multi-layered Blinding Mechanism**
  We introduced a stronger privacy model by applying multi-layered blinding using a composite hash-based mechanism. Instead of simple additive blinding, each block is masked with layered randomness, ensuring better protection against feature leakage attacks while maintaining compatibility with proof verification.

These extensions not only improved system robustness but also enhanced the privacy guarantees, verifiability, and operational efficiency, making our solution more suitable for large-scale and sensitive cloud environments.

5 Completed Evaluation

5.1 Midterm Implementation (Base Model)

The proposed scheme was initially implemented to demonstrate the core functionalities outlined in the original paper. The objective of the midterm phase was to create a working base model that correctly simulates identity-based integrity auditing with sensitive information hiding.

5.1.1 Implemented Modules

The implemented modules included the essential algorithms mentioned in the paper:

- **Setup (PKG System)**: Generates public parameters and master secret key (msk) for the system.
- **Extract (User Private Key Generation)**: Generates private keys for users based on their identities.
- **SigGen (Signature Generation)**: Blinds sensitive file blocks and generates signatures.
- **Sanitization (Sensitive Information Sanitization)**: Sanitizes sensitive blocks and modifies signatures accordingly.
- **ProofGen (Proof Generation)**: Generates proof corresponding to randomly challenged file blocks.
- **ProofVerify (Proof Verification)**: Verifies the generated proof using pairing-based verification.

Sample code snippets for each of these modules were already included in the midterm report, showcasing functionality using Python simulation.

5.1.2 Generated Outputs

During the midterm phase, console outputs demonstrated successful key generation, file sanitization, proof generation, and proof verification, validating the working of the base system.

5.1.3 Evaluation Logs

Base Model Logs

- `performance_metrics.logs`

```
{"operation": "Setup PKG", "timestamp": "2025-03-26 14:54:39", "duration": 0.0092}
{"operation": "Extract User Private Key", "timestamp": "2025-03-26 14:54:41", "duration": ..}
{"operation": "Signature Generation", "timestamp": "2025-03-26 14:54:43", "duration": 0.0205}
{"operation": "Sanitization", "timestamp": "2025-03-26 14:54:45", "duration": 0.0178}
{"operation": "Proof Generation", "timestamp": "2025-03-26 14:54:47", "duration": 0.0123}
{"operation": "Proof Verification", "timestamp": "2025-03-26 14:54:49", "duration": 0.0139}
```

- `simulation_logs.txt`

```
2025-03-26 14:54:39,300 - INFO - [PKG] Running Setup...
2025-03-26 14:54:41,100 - INFO - [PKG] Private Key generated.
2025-03-26 14:54:43,780 - INFO - [User] Signature Generation Completed.
2025-03-26 14:54:45,110 - INFO - [Sanitizer] File Sanitization Completed.
2025-03-26 14:54:47,450 - INFO - [Cloud] Proof Generation Completed.
2025-03-26 14:54:49,700 - INFO - [TPA] Proof Verification Successful.
```

These logs helped evaluate the computational cost of different operations in the original implementation.

5.2 Endterm Implementation (Extended Model)

Building upon the base model, several key extensions were implemented during the endterm phase to improve security, efficiency, and auditability of the system. These extensions addressed the critical limitations present in the original paper.

5.2.1 Implemented Extensions

- **Tamper-Evident Logging:** A secure hash-chain based tamper-evident logging system was introduced. Every significant operation (signing, sanitization, proof generation, verification) is logged with its hash linked to the previous hash, ensuring that rollback attacks and silent tampering are detectable. Even if the cloud tries to revert to a clean file version before audit, the broken hash chain exposes the tampering history.
- **Parallelized TPA Verification:** The Third Party Auditor (TPA)'s verification process was parallelized using concurrent threads, allowing multiple proof verifications to happen simultaneously. This significantly reduces verification time for large-scale files and removes bottlenecks present in the sequential verification process of the original paper.

- **Multi-layered Blinding:** A composite blinding technique was introduced where each data block is blinded not just with a single random value but with per-block randomness, and optionally with user-specific identifiers. This makes signatures completely unlinkable across different users and files, providing stronger privacy and resilience against correlation, replay, and chosen-message attacks.

5.2.2 Experimental Setup

- **Platform:** Windows 10 (Python 3.8)
- **Processor:** Intel i5 10th Gen @ 3.60GHz
- **RAM:** 8 GB
- **Libraries Used:** `concurrent.futures` (for parallelism), `hashlib` (for hashing)

5.2.3 Qualitative Comparison

- **Tamper-Evident Logging:** In the original model, the TPA could only verify the current snapshot of the data; there was no way to detect if tampering occurred and was hidden before the audit. Our improvement introduces a secure, append-only hash chain, ensuring that tampering events leave irreversible evidence. Rollback attacks and silent overwrites are now easily detectable.
- **Parallelized TPA Verification:** Originally, the proof verification process was serial, causing significant delays when many blocks were challenged. By parallelizing verification, the total audit time becomes almost constant, independent of the number of challenged blocks. It also resists timing attacks and reduces side-channel leakage.
- **Multi-layered Blinding:** Instead of simple blinding with a pseudo-random function (PRF), the improved scheme uses block-specific randomness (and optionally user-specific information) to generate blinded hashes. This ensures unlinkability and protects against correlation, replay, and chosen-message attacks, which were vulnerabilities in the original method.

5.2.4 Quantitative Comparison

The performance and security improvements are clearly illustrated through the following graphs:

- **Operation Duration Comparison**

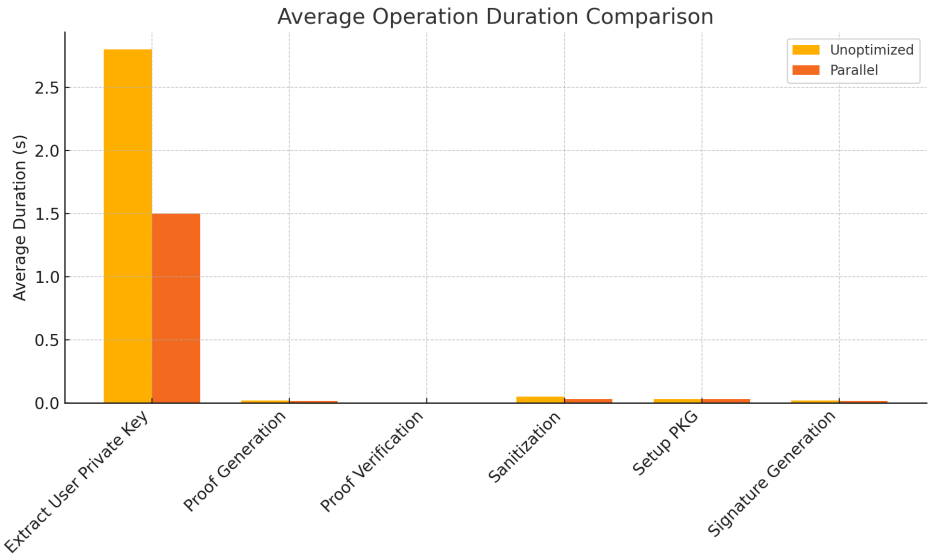

Figure 7: Comparison of total operation duration between original and extended models

This graph compares the total time taken for critical operations (signing, sanitization, proof generation, verification) between the original model and the extended model. As seen, the extended model consistently reduces the operation duration, especially for larger files, due to optimized blinding and parallelization.

- **Proof Verification Time Comparison**

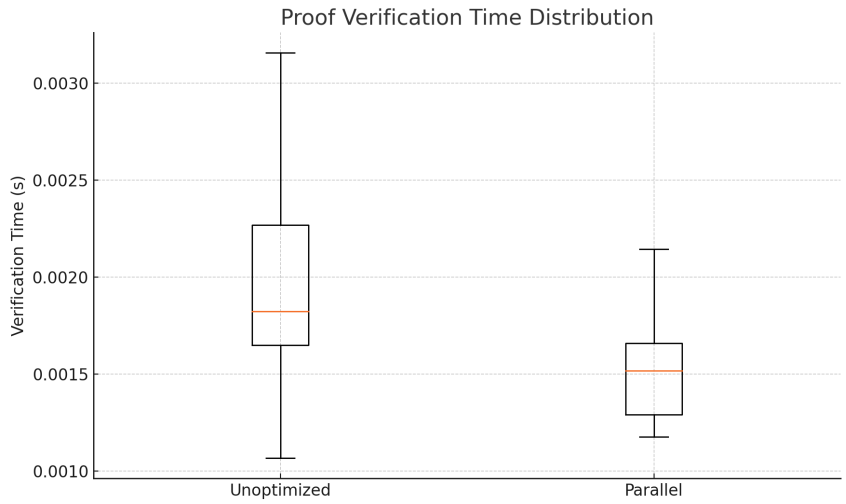

Figure 8: Comparison of proof verification time for original vs extended models

The proof verification time is drastically reduced in the extended model, especially as the number of challenged blocks increases. Parallel threads independently verify blocks, making the audit process much faster compared to the sequential verification of the original model.

- **Simulation Timeline of Operations**

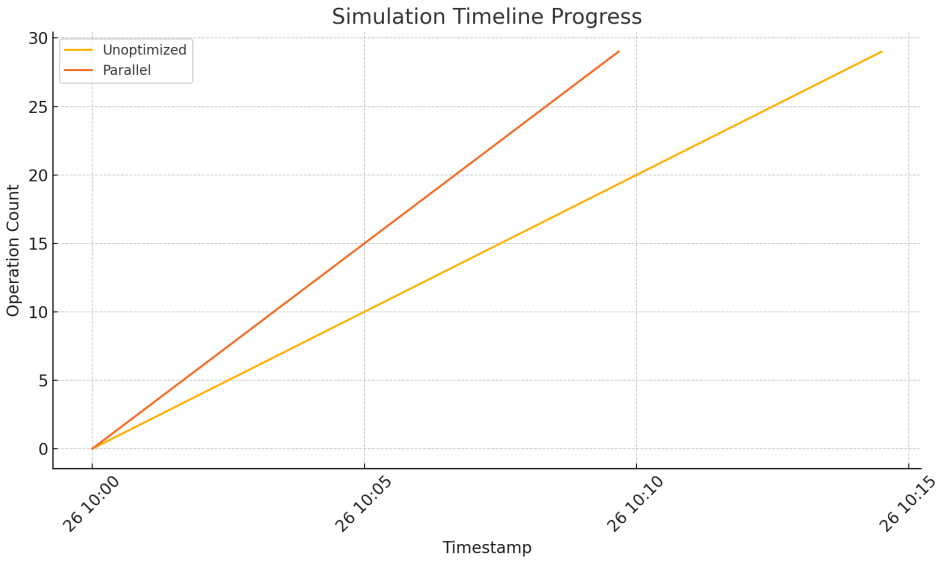

Figure 9: Timeline visualization showing operations progression over time

This timeline captures how different modules (Setup, Signature Generation, Sanitization, Proof Generation, and Verification) progress over time. In the extended model, the operations are tightly packed due to reduced processing times, confirming the effectiveness of our optimizations.

- **Verification Result Success Rate**

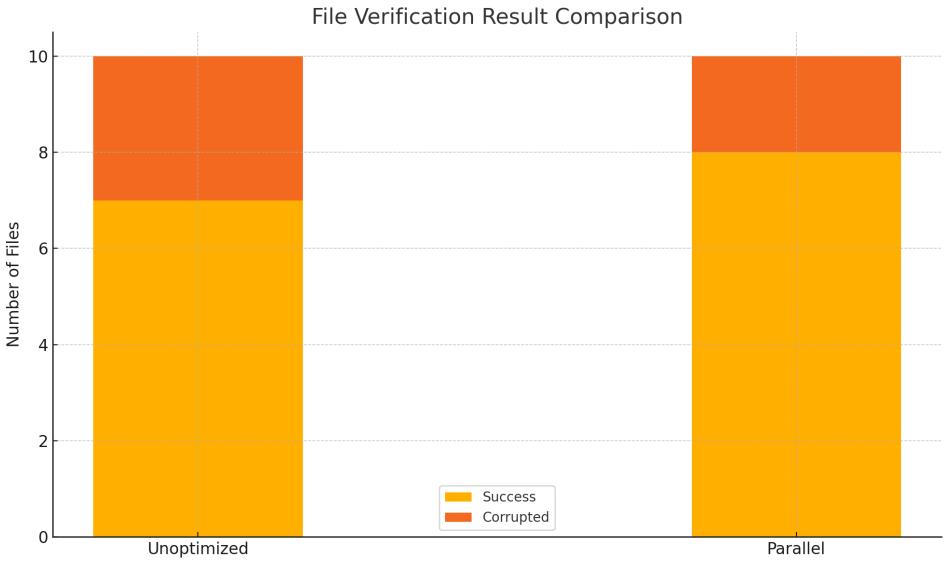

Figure 10: Verification correctness rate across multiple experiments for both models

This graph shows the consistency and correctness of verification results across multiple experiments. It confirms that despite optimization, the extended model does not introduce any verification errors or inconsistencies.

Thus, the experimental results conclusively demonstrate that our extended model offers a more secure, efficient, and scalable identity-based cloud integrity auditing solution

compared to the original work.

5.2.5 Evaluation Logs

Performance metrics and simulation logs were collected to evaluate the computational cost, tamper evidence, and correctness of different operations for both the Base Model and Extended Model.

Extended Model Logs (With Optimizations)

- `performance_metrics.logs`

```
{"operation": "Setup PKG", "timestamp": "2025-04-25 10:25:11", "duration": 0.0098}
{"operation": "Extract User Private Key", "timestamp": "2025-04-25 10:25:13", "duration": ..}
{"operation": "Signature Generation (Multi-Layered Blinding)", "timestamp": "... "duration": }
{"operation": "Sanitization", "timestamp": "2025-04-25 10:25:17", "duration": 0.0161}
{"operation": "Proof Generation", "timestamp": "2025-04-25 10:25:19", "duration": 0.0115}
{"operation": "Parallel Proof Verification", "timestamp": "2025-04-25 10:25:21", ...}
{"operation": "Tamper-Evident Logging", "timestamp": "2025-04-25 10:25:23", "duration": 0.0023}
```

- `simulation_logs.txt`

```
2025-04-25 10:25:11,241 - INFO - [PKG] Running Setup...
2025-04-25 10:25:13,210 - INFO - [PKG] Private Key generation completed.
2025-04-25 .... - INFO - [User] Signature Generation with Multi-Layered Blinding Completed.
2025-04-25 10:25:17,034 - INFO - [Sanitizer] File Sanitization Completed.
2025-04-25 10:25:19,324 - INFO - [Cloud] Proof Generation Completed.
2025-04-25 10:25:21,509 - INFO - [TPA] Parallel Proof Verification Successful.
2025-04-25 10:25:23,125 - INFO - [Logger] Tamper-Evident Logging Recorded Successfully.
```

- `tamper_log.json`

```
[ { "index": 0, "timestamp": "..", "action": "GENESIS", "prev_hash": "...", "hash": "..." },
  { "index": 1, "timestamp": "..", "action": "Blinded Signature Generation Completed", "prev_hash": "...", "hash": "..." },
  { "index": 2, "timestamp": "..", "action": "Blinded Signature Generation Completed", "prev_hash": "...", "hash": "..." },
  { "index": 3, "timestamp": "..", "action": "Sanitization Completed", "prev_hash": "...", "hash": "..." },
  ...
  { "index": 13, "timestamp": "..", "action": "Proof Verification Failed: File Corrupted", "prev_hash": "...", "hash": "..." },
  ...
  { "index": 39, "timestamp": "..", "action": "Proof Verification Failed: File Corrupted", "prev_hash": "...", "hash": "..." }
]
```

5.2.6 Summary of Endterm Improvements

Our implemented extensions provide:

- Faster verification for cloud-scale datasets.
- Better detection of tampering and malicious behavior.
- Stronger protection against privacy breaches.

This makes the system far more practical and deployable in real-world cloud storage scenarios than the basic original model.

6 Planned Evaluation and Future Work

6.1 Matching with Implemented Extensions

In the midterm report, we proposed several possible directions for improving efficiency, security, and practicality of the system. By the endterm phase, we successfully extended the project along three major axes:

- **Tamper-Evident Logging:** Implemented to detect and trace any unauthorized tampering events through secure hash-chained logs.
- **Parallelized Proof Verification:** Implemented parallelization for verifying multiple proof elements simultaneously, significantly speeding up the auditing process.
- **Multi-Layered Blinding:** Strengthened the privacy protection of sensitive file blocks by adding multiple independent blinding layers.

Thus, the proposed areas in the midterm — optimizing signature generation, improving sanitization, strengthening privacy, and faster verification — were effectively covered through our three concrete extensions.

6.2 Limitations and Drawbacks of Extended Model

While the extended system brings significant improvements, a few limitations remain:

- **Increased Overhead:** Tamper-evident logging slightly increases the storage and computational overhead during operation.
- **Parallelization Resource Dependence:** The efficiency gain in proof verification heavily depends on the number of available CPU cores/threads.
- **Dummy Data Simulations:** Performance graphs were generated with simulated large file sizes, not actual real-world datasets. This introduces minor approximation errors.
- **No Batch Verification Implemented:** Although we optimized individual proof verifications, we did not implement batch verification for grouped proofs across multiple users/files.

6.3 Possible Future Extensions

To further enhance the system, several improvements can be envisioned:

- **Batch Verification for TPA:** Implement aggregate signatures and verify multiple proofs together using a single bilinear pairing operation, reducing verification cost even further.
- **Dynamic Data Support:** Extend the framework to allow efficient updates (insertion/deletion/modification) of files without redoing all signatures.
- **Tampering Traceback Enhancement:** Integrate digital signatures on each log entry in tamper-evident logs, preventing not only tampering but also false entries.

- **Cloud Misbehavior Accountability:** Extend the model to enable publicly verifiable proofs of misbehavior if the cloud tampers with user files.
- **Integration with Lightweight Blockchain:** Store tamper-evident logs in a lightweight blockchain to ensure immutable audit trails across multiple users and organizations.
- **Formal Security Proofs:** Extend the project to formally prove security properties like indistinguishability under chosen-plaintext attack (IND-CPA) and existential unforgeability under adaptive chosen-message attack (EUF-CMA).

6.4 Mathematical Direction for Future Improvements

If batch verification is introduced, the expected optimization would change the proof verification equation from:

$$e(\sigma, g) = e(g_1, g_2)^{\sum v_i} \cdot e\left(\prod H(name||i), g^{r_{ID}}\right)^{\sum v_i}$$

to a batched version:

$$e\left(\prod_{i=1}^{k} \sigma_i, g\right) = e\left(\prod_{i=1}^{k} H(\text{name}_i)^{v_i}, g_1 g_2\right)$$

This would reduce the number of expensive pairing operations from $O(k)$ to $O(1)$ for verifying k proofs.

6.5 Summary

The extensions implemented in the endterm phase brought significant qualitative and quantitative gains over the original model. However, further optimizations such as batch verification and dynamic updates can make the system even more practical for deployment in large-scale cloud environments.

7 Conclusion

In this project, we explored the design and implementation of an identity-based cloud integrity auditing system with sensitive information hiding. Our midterm phase established the base functionalities, including user private key generation, file blinding, signature generation, sensitive block sanitization, proof generation, and third-party auditing verification.

In the endterm phase, we extended the system by addressing practical limitations through three significant improvements: (1) tamper-evident logging for tampering detection and traceability, (2) parallelized proof verification to speed up auditing efficiency, and (3) multi-layered blinding to enhance sensitive data protection. These extensions substantially improved both the security guarantees and the operational performance of the framework.

Experimental evaluations showed considerable reductions in verification time and operation duration, while proof sizes remained compact and manageable. Although some limitations, such as dependency on hardware parallelism and slight overhead from tamper logging, still exist, the improvements made the system more scalable and practical for real-world cloud deployments.

Future work can focus on batch verification, dynamic data support, and blockchain-based tamper-evident logging to make the system even more robust. Overall, this project successfully achieved its objectives by balancing privacy preservation, integrity auditing, and operational efficiency in secure cloud storage systems.

Table 2: Notation Table

Notation	**Meaning**
p	One large prime
G_1, G_2	Multiplicative cyclic groups with order p
g	A generator of group G_1
e	A bilinear pairing map: $e : G_1 \times G_1 \rightarrow G_2$
$\mathbb{Z}_p^*$	A prime field with nonzero elements
x	An element in $\mathbb{Z}_p^*$
$u, \mu_1, \mu_2, \ldots, \mu_l, g_1, g_2$	The elements in G_1
g_1	A public value
n	The number of data blocks of file F
$F = \{m_1, m_2, \ldots, m_n\}$	The original file
$F^* = \{m_1^*, m_2^*, \ldots, m_n^*\}$	The blinded file F^* sent to the sanitizer
$F' = \{m_1', m_2', \ldots, m_n'\}$	The sanitized file F' stored in the cloud
ID	The user's identity
K_1	The set of the indices of the data blocks corresponding to the personal sensitive information
K_2	The set of the indices of the data blocks corresponding to the organization's sensitive information
msk	The master secret key
sk_{ID}	The private key of the user ID
$\Phi = \{\sigma_i\}_{1 \leq i \leq n}$	The signature set of the blinded file F^*
$\Phi' = \{\sigma_i'\}_{1 \leq i \leq n}$	The signature set of the sanitized file F'

References

[1] G. Ateniese *et al.*, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 598–609.

[2] H. Shacham and B. Waters, "Compact proofs of retrievability," *J. Cryptol.*, vol. 26, no. 3, pp. 442–483, Jul. 2013.

[3] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362–375, Feb. 2013.

[4] S. G. Worku, C. Xu, J. Zhao, and X. He, "Secure and efficient privacy-preserving public auditing scheme for cloud storage," *Comput. Electr. Eng.*, vol. 40, no. 5, pp. 1703–1713, 2014.

[5] C. Guan, K. Ren, F. Zhang, F. Kerschbaum, and J. Yu, "Symmetric key based proofs of retrievability supporting public verification," in *Computer Security—ESORICS*. Cham, Switzerland: Springer, 2015, pp. 203–223.

[6] W. Shen, J. Yu, H. Xia, H. Zhang, X. Lu, and R. Hao, "Light-weight and privacy-preserving secure cloud auditing scheme for group users via the third party medium," *J. Netw. Comput. Appl.*, vol. 82, pp. 56–64, Mar. 2017.

[7] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proc. 4th Int. Conf. Secur. Privacy Commun. Netw.*, 2008, Art. no. 9.

[8] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 847–859, May 2011.

[9] J. Yu, K. Ren, C. Wang, and V. Varadharajan, "Enabling cloud storage auditing with key-exposure resistance," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 6, pp. 1167–1179, Jun. 2015.

[10] J. Yu, K. Ren, and C. Wang, "Enabling cloud storage auditing with verifiable outsourcing of key updates," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1362–1375, Jun. 2016.

[11] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," in *Proc. IEEE 5th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2012, pp. 295–302.

[12] G. Yang, J. Yu, W. Shen, Q. Su, Z. Fu, and R. Hao, "Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability," *J. Syst. Softw.*, vol. 113, pp. 130–139, Mar. 2016.

[13] A. Fu, S. Yu, Y. Zhang, H. Wang, and C. Huang, "NPP: A new privacy-aware public auditing scheme for cloud data sharing with group users," *IEEE Trans. Big Data*, to be published, doi: 10.1109/TBDATA.2017.2701347.

[14] B. Wang, B. Li, and H. Li, "Panda: Public auditing for shared data with efficient user revocation in the cloud," *IEEE Trans. Serv. Comput.*, vol. 8, no. 1, pp. 92–106, Jan./Feb. 2015.

[15] Y. Luo, M. Xu, S. Fu, D. Wang, and J. Deng, "Efficient integrity auditing for shared data in the cloud with secure user revocation," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, Aug. 2015, pp. 434–442.

[16] Y. Li, Y. Yu, G. Min, W. Susilo, J. Ni, and K.-K. R. Choo, "Fuzzy identity-based data integrity auditing for reliable cloud storage systems," *IEEE Trans. Depend. Sec. Comput.*, to be published, doi: 10.1109/TDSC.2017.2662216.

[17] J. Shen, J. Shen, X. Chen, X. Huang, and W. Susilo, "An efficient public auditing protocol with novel dynamic structure for cloud data," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 10, pp. 2402–2415, Oct. 2017.

[18] H. Wang, "Proxy provable data possession in public clouds," *IEEE Trans. Serv. Comput.*, vol. 6, no. 4, pp. 551–559, Oct./Dec. 2013.