Provable Co-Owned Data Deletion with Zero-Residuals and Verifiability in Multi-Cloud Environment

Darwish, Marwan Adnan; Markatou, Evangelia Anna; Smaragdakis, Georgios

DOI
10.1145/3722041.3723104

Publication date
2025

Document Version
Final published version

Published in
Proceedings of the 18th European Workshop on Systems Security, EuroSec 2025, Part of EuroSys 2025

Provable Co-Owned Data Deletion with Zero-Residuals and Verifiability in Multi-Cloud Environment

Marwan Adnan Darwish
Delft University of Technology
The Netherlands

Evangelia Anna Markatou
Delft University of Technology
The Netherlands

Georgios Smaragdakis
Delft University of Technology
The Netherlands

ABSTRACT

The rapid expansion of multi-cloud environments and the growing prevalence of collaborative data ownership present significant challenges in ensuring the verifiable deletion of co-owned data. Current approaches predominantly address individual ownership and often rely on simplistic one-bit result protocols where a deletion command merely outputs success or failure, turning the deletion into a black box without proper verification. This paper tackles the problem of secure processing and verifiable deletion of shared outsourced data in multi-cloud environments. We design a framework that enables a data owner to outsource encrypted data to multiple co-owners, who perform computations directly within their respective cloud providers—ensuring that sensitive data never leaves the cloud. Our system leverages readily available cloud *Hardware Security Modules* (HSMs) to manage cryptographic keys from generation to controlled destruction—ensuring data remains inaccessible beyond its intended use. Secure Enclaves enforce on-cloud data computation, eliminating local copies and preventing unauthorized exposure. Encrypted data is structured within a fixed storage model, ensuring controlled allocation and strict storage constraints. When data expires or must be deleted to meet regulatory requirements, our framework triggers zero-residual permuted overwriting to remove the data traces irreversibly. Verifiability is achieved at two levels: *Bounded Merkle Hash Tree* (BMHT) ensures bounded storage and verifiable deletion within each cloud provider. In contrast, *Global Merkle Forest* (GMF) aggregates BMHT roots across providers, enabling consistent global verification. The data owner maintains a log of these BMHT roots, allowing independent verification of secure deletion across the multi-cloud environment.

CCS CONCEPTS

• **Security and privacy → Security services; Privacy-preserving protocols**.

KEYWORDS

Provable Data Deletion; Multi-Cloud; Zero-Residuals Overwriting

ACM Reference Format:
Marwan Adnan Darwish, Evangelia Anna Markatou, and Georgios Smaragdakis. 2025. Provable Co-Owned Data Deletion with Zero-Residuals and Verifiability in Multi-Cloud Environment. In *The 18th European Workshop on Systems Security (EuroSec'25), March 30-April 3, 2025, Rotterdam, Netherlands.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3722041.3723104

1 INTRODUCTION

Modern digital services depend on data-intensive systems, making secure and provable data deletion crucial for privacy protection [27, 37]. As sensitive data moves to the cloud, concerns over integrity, availability, and deletion grow [14, 24]. Failure to ensure proper erasure can lead to privacy breaches, security risks, and regulatory violations [22, 36]. Researchers have proposed solutions compliant with *General Data Protection Regulation* (GDPR) [10, 17], broadly classified into the following approaches.

a) Deletion by Unlinking [11, 25, 28]: This method severs the data's link to the file system, returning a one-bit success/failure response on deletion. However, the data remains physically intact, making it vulnerable to forensic recovery and lacking reliable, secure deletion.

b) Deletion by Cryptography [4, 7, 8, 12, 18]: Data is encrypted, securing it through its key. The data becomes unrecoverable once the key is destroyed, though the ciphertext remains in the cloud. This method simplifies deletion by eliminating the key instead of overwriting multiple copies, but secure key deletion remains a critical challenge.

c) Deletion by Overwriting [20, 26, 29, 31, 34]: In this approach, old data is replaced with new or random data of the same size to ensure it is irretrievable. Overwriting offers physical deletion, making it more robust than other methods. However, improper implementation may leave residual traces (i.e., incomplete removal of data or unintended storage expansion) of the original data. Advanced microscopic tools can exploit these traces, exposing the physical remnants and compromising the data's integrity [12].

These limitations become even more pronounced in multi-cloud environments, where co-owned data—shared among multiple stakeholders across diverse providers—faces significant challenges [30, 33]. Previous studies lack synchronized management to address inconsistent deletion practices, cross-provider protocol gaps, and the need for verifiable proofs at each step [15]. For example, European hospitals often collaborate with multiple research institutes and universities on specific projects, such as health data analysis, to perform experiments to identify anomalies, outliers, and patterns for privacy enhancement. The data, co-owned by multiple stakeholders, is stored across different providers, such as AWS [35], Azure [5], or GCP [2], depending on operational needs or resources provided. Under recently established *European Health Data Space* (EHDS) [9, 13], which mandates strict data-sharing and retention policies, managing this co-owned data securely and transparently is essential. When data expires or requires deletion, the hospital (data owner) requests deletion from all relevant cloud providers. Providers verify

EuroSec'25, March 30-April 3, 2025, Rotterdam, Netherlands

ACM ISBN 979-8-4007-1563-1/2025/03
https://doi.org/10.1145/3722041.3723104

consent, execute secure deletion, and return cryptographic proofs. Co-owners and regulators use these proofs to ensure compliance with EHDS regulations. If research institutes (co-owners) agree to modify or delete data before expiration, the framework supports collaborative governance, with cloud providers executing updates and generating verifiable proofs. Secure data-sharing, decryption, and access occur within the protocol, ensuring compliance with EHDS requirements for sensitive data. We propose a comprehensive framework for secure and provable co-owned data deletion in multi-cloud environments, ensuring that data never leaves the cloud and computations remain within each respective provider's infrastructure. The system supports a Data Owner (DO) and multiple Co-Owners (COs), each associated with a distinct Cloud Provider (CP) with separate infrastructures. No local copies are ever created, and providers maintain zero knowledge of actual data or computations. To achieve this, we integrate *Hardware Security Modules (HSMs)* [21] at each CP to securely handle cryptographic keys for encrypting and decrypting outsourced data, managing them from generation to controlled destruction upon expiry, and ensuring data remains inaccessible after its designated lifecycle. Additionally, Secure Enclaves [1] enable on-cloud computations without external data storage; however, their high cost restricts their use to only the most critical operations. The framework ensures zero-residual overwriting by applying a 3-cycle process—PRF masking to obscure patterns, bit-order reversal to disrupt structure, and PRP-based repositioning to eliminate recoverable traces—aligning with NIST SP 800-88[16]. This ensures that overwritten data is computationally irrecoverable while maintaining storage integrity. While the scheme enforces bounded storage, we acknowledge that filesystem or firmware-level remapping could allocate new physical blocks instead of reusing existing ones. However, since all data remains encrypted at rest, any reallocation would be useless without the proper decryption keys. Furthermore, to verify the 3-cycle process, each overwriting step is correctly executed, and a new BMHT root is recalculated. Each CP constructs a *Bounded Merkle Hash Tree (BMHT)*, a structured variant of *Merkle Hash Tree* (MHT) [19], to restrict storage expansion, ensuring that only authorized updates and deletions occur within a fixed allocation to the CP's storage. Copies made outside the scope of the protocol are useless without the private keys within the HSMs. Verifiable deletion is guaranteed across two distinct levels: local verification using BMHT within each CP and global verification via the GMF, aggregating BMHT roots from all providers, ensuring consistency. The system produces cryptographic evidence—including storage, membership/non-membership, overwriting, and global proofs.
In summary, our paper makes the following key contributions:

- We present a framework for provable co-owned data deletion, ensuring co-owned data never leaves the cloud via HSMs for key management and Secure Enclaves for on-cloud processing.
- We propose BMHT for strict storage enforcement and deletion verifiability at each provider, leveraging zero-residual permuted overwriting for complete data erasure.
- We provide cryptographic proofs for storage, membership, non-membership, and overwriting, enabling verifiable data removal.
- We introduce GMF for verifiable global proof, integrating generated local overwriting proofs across providers for consistency.

2 DESIGN GOALS

2.1 Asynchronous and Secure Communication

The solution operates in asynchronous environments where cloud providers and stakeholders may experience communication delays or inconsistencies. All communications, including proof exchanges, key management, and data transmission between providers, are protected using Transport Layer Security (TLS) or equivalent cryptographic protocols to ensure security.

2.2 Threat Model

Byzantine Cloud Providers (CPs): CPs are potential Byzantine adversaries capable of deviating from protocols, retaining data, or falsifying proofs. They store encrypted data and initiate Secure Enclaves for processing. We assume no collusion between the co-owners CPs, with communication limited to the primary owner.
Trusted Data Owners (DOs): Data owners are considered trusted entities responsible for initiating data sharing, storage, and deletion processes. They coordinate data management activities with co-owners and cloud providers.
Trusted-But-Forgetful Co-Owners (COs): COs, such as research institutions, operate independently in cloud environments, adhering to the security model by processing data within Secure Enclaves. They are trusted but may forget to delete local copies if they exist.
Trusted Hardware Manufacturers: Provide HSMs for secure key management and Secure Enclaves for isolated execution. The framework assumes HSMs are FIPS $140-2$ Level 3 certified for key isolation [3], and Secure Enclaves process data in-memory, preventing persistence or external access. Security relies on the certified integrity of the hardware.

2.3 Security Model

Correctness: Correctness holds when a valid proof, generated by the prover (CPs), enables the verifier (DOs or COs) to validate storage, membership, non-membership, and overwriting operations. Let $D = \{d_1, d_2, \ldots, d_n\} \in \mathbb{B}^*$ be the dataset stored in the system, where $d_i = (k_i, D_i)$ is a unique key-value pair stored as a leaf in the BMHT. For a leaf at index i, the hash is: $H_{\text{leaf}} = \text{Hash}(k_i \parallel D_i)$. Let $\mathcal{H}_{\text{root}}$ denote the BMHT root hash. Correctness guarantees the following for all proofs:
1. Storage Proof: If the Merkle tree constructed on dataset D has root $\mathcal{H}_{\text{root}}$, then the verifier can check:

$$V_{\pi_{\text{storage}}}(D, \mathcal{H}_{\text{root}}, \pi_{\text{storage}}) = 1.$$

where $V_{\pi_{\text{storage}}}$ is the verification function, and π_{storage} is the proof that $\mathcal{H}_{\text{root}}$ correctly represents D.
2. Membership Proof: Confirms that a key-value pair $d_i = (k_i, D_i)$ is present in D:

$$V_{\text{membership}}(d_i, \mathcal{H}_{\text{root}}, \pi_{\text{membership}}) = 1$$

3. Non-Membership Proof: Confirms that a key $k \notin D$, using its predecessor k_{pred} and successor k_{succ}:

$$V_{\text{non-membership}}(k, \mathcal{H}_{\text{root}}, \pi_{\text{non-membership}}) = 1 \quad \text{, if } k_{\text{pred}} < k < k_{\text{succ}}$$

4. Overwriting Proof: Validates that an old key-value pair $(k_{\text{old}}, D_{\text{old}})$ has been securely replaced by a new pair $(k_{\text{new}}, D_{\text{new}})$:

$$V_{\text{overwrite}}((k_{\text{old}}, D_{\text{old}}), (k_{\text{new}}, D_{\text{new}}), \mathcal{H}_{\text{root}}, \pi_{\text{overwrite}}) = 1$$

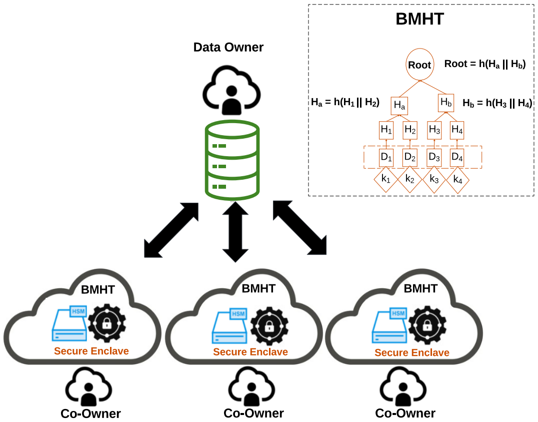

Figure 1: The Framework Employs BMHT for Fixed Storage, HSMs for Secure Key Management, and Secure Enclaves to Ensure On-the-Fly Data Access Without Local Copies.

Security: Security ensures that no malicious cloud provider can tamper with proofs, generate false proofs, or introduce inconsistencies between BMHT roots of the GMF root. Let π_{storage}, $\pi_{\text{membership}}$, $\pi_{\text{non-membership}}$, $\pi_{\text{overwrite}}$ be valid proofs, and let π' denote adversarially generated false proofs. For any efficient adversarial algorithm P', the probability of successfully generating a false proof or tampering with $\mathcal{H}_{\text{GMF}}$ is negligible:

$$\Pr\left[V_{\text{proof}}(\pi') = 1 \mid \pi' \neq \pi \text{ or } \mathcal{H}^{(i)}_{\text{root}} \notin \mathcal{H}_{\text{GMF}}\right] < \epsilon(k)$$

where $\epsilon(k)$ is a negligible function of the security parameter $k \in N$, V_{proof} represents the verification function for any proof type, and $\mathcal{H}_{\text{GMF}} = \text{Hash}(\mathcal{H}^{(1)}_{\text{root}} \parallel \mathcal{H}^{(2)}_{\text{root}} \parallel \cdots \parallel \mathcal{H}^{(n)}_{\text{root}})$ serves as a globally verifiable proof.

3 PRELIMINARIES

3.1 Bounded Merkle Hash Trees (BMHT)

The BMHT extends the traditional MHT [19] by incorporating a fixed and sorted tree structure, ensuring efficient storage management without dynamic growth concerns. The BMHT consists of a constant number of leaf nodes N, each representing a predefined storage block of size S, ensuring predictable and enforceable height. The leaf nodes are sorted in ascending order by their keys to facilitate efficient proofs. With N leaf nodes, the total storage capacity is: $C = N \cdot S$. This fixed structure ensures consistent storage limits and avoids the growth issues associated with MHT in constrained environments. Each leaf node hash is computed as: $H_{\text{leaf}_i} = \text{Hash}(k_i \parallel D_i)$, where k_i is the binary key and D_i is the data block (i.e., will be encrypted). Including the key in the hash ensures positional integrity. Internal nodes derive their hashes recursively from child nodes, culminating in a root hash H_{root}, which serves as a cryptographic commitment to the entire data's integrity. The key differences between BMHT and MHT are: (*i*) BMHT maintains a fixed number of leaf nodes, enforcing strict storage limits for predefined constraints. (*ii*) Its sorted leaf structure enables efficient non-membership proofs via predecessor-successor verification. (*iii*) BMHT's fixed order ensures smaller and more predictable proof sizes for non-membership and overwriting compared to MHT.

3.2 Global Merkle Forest (GMF)

The GMF serves as global proof by aggregating the local root hashes of BMHT instances from multiple cloud providers, ensuring tamper-resistant evidence of data operations across a multi-cloud environment. For $M \in N$ providers, where the j-th provider manages a BMHT root hash $H^{(j)}_{\text{root}}$, the global root hash H_{GMR} is computed as: $H_{\text{GMR}} = h(H^{(1)}_{\text{BMHT}} \parallel H^{(2)}_{\text{BMHT}} \parallel \cdots \parallel H^{(M)}_{\text{BMHT}})$, where $h(\cdot)$ denotes a cryptographic hash function and $\parallel$ represents concatenation. This structure integrates fixed storage enforcement at each provider through BMHT while offering a unified cryptographic commitment to the global data state. To verify the integrity of a data block D_i stored at provider j, the proof includes the path from the leaf node L_i (representing D_i) to $H^{(j)}_{\text{BMHT}}$, along with the aggregated hashes required to reconstruct H_{GMR}. The proof is valid if: $H_{\text{GMR}} = h(\pi_{\text{global}})$, where π_{global} combines the local proof from $H^{(j)}_{\text{BMHT}}$ at each provider. This approach guarantees transparent and verifiable operations (e.g., overwriting) across providers while preserving the independence of each BMHT.

4 SYSTEM ARCHITECTURE

Figure 1 outlines the overall system architecture in key steps:
Initialization: The DO initializes the system by dividing the dataset into fixed storage blocks (e.g., 256 MB) and encrypting each block using a provided key. The encrypted blocks are mapped to a local Bounded Merkle Hash Tree (BMHT) leaf node, which enforces fixed storage. The padding ensures symmetric tree structure and consistent block sizes. The DO computes the BMHT root and shares it with Co-Owners, allowing them to verify storage integrity. The DO retains key mappings and metadata for proof verification.
Preprocessing: The DO prepares the data by associating each block with a unique key and sorting the BMHT leaf nodes in ascending order by their keys. This facilitates efficient proof generation for both membership and non-membership verification.
Data Sharing: The DO securely shares data with COs using their public keys (PKI) for encryption via HSMs, ensuring data remains within the cloud until expiration. The HSMs decrypts data internally, without exposing it to the CP, and immediately re-encrypts it using a session-specific key before processing. The Secure Enclave is used only when data needs to be processed, preventing unauthorized access. The DO periodically publishes a signed digest of sorted key identifiers and BMHT root, allowing COs to verify CP-provided membership/ non-membership proofs.
Outsourcing: The CP computes the BMHT root hash over the encrypted blocks via HSMs to enable verifiable integrity checks by the DO and COs, who can request and verify proofs when needed. The BMHT structure enforces fixed storage, preventing uncontrolled expansion of stored data or an increase in the number of leaf nodes.
Data Access and Update: Secure Enclaves facilitate confidential on-cloud data processing without exposing sensitive information or creating local copies. Instead of decrypting data outside the enclave, a session-specific symmetric key is generated by HSMs to encrypt data within the enclave before processing. This key ensures the enclave can process encrypted data without revealing the PKI keys. The key is securely destructed upon request, preventing unauthorized access and ensuring confidentiality.

Proofs Request: The CP provides BMHT roots to be verified later when requested by the DO or COs, ensuring freshness and avoiding stale proofs. The CP does not proactively provide stored proofs but instead generates them dynamically upon request: ***a)*** *Storage Proof:* Confirms that the data remains within the BMHT structure, ensuring fixed storage limits are intact; ***b)*** *Membership Proof:* Verifies including specific data blocks using their BMHT path; and ***c)*** *Non-Membership Proof:* Confirms specific data exclusion by verifying its predecessor and successor in the sorted BMHT. DOs or Co-Owners (COs) verify these proofs to ensure storage limits and data integrity.
Data Overwriting: Expired or updated data is securely erased using zero-residual overwriting, following NIST SP 800 − 88 [16]. A 3-cycle process—PRF masking, bit reversal, and entropic shuffling—ensures complete removal. The BMHT root is updated, with overwriting verified via membership (new data) and non-membership (old data) proofs. Overwriting can be triggered by expiration time or uploading new data from DO.
Global Proof Synchronization: The GMF aggregates BMHT roots across CPs, ensuring consistent global storage verification. This guarantees that all parties remain consistent, even in asynchronous communication environments.

5 SCHEME DESCRIPTION

Figure 2 presents an overview of the provable structure for co-owned data deletion in multi-cloud systems.

5.1 Initialization

The initialization phase establishes fixed storage, encrypts data blocks, and constructs the Bounded Merkle Hash Tree (BMHT) for efficient proof generation and secure storage.
Fixed Storage: Let the dataset be $D = \{d_1, d_2, \ldots, d_m\}$ in dedicated storage medium, where each D_i is a data block. The dataset is divided into fixed-sized storage blocks, each of size S. The total storage capacity is $C = N \cdot S$, where $N = \lceil C/S \rceil$ is the number of leaf nodes. Each block D_i is padded as much as necessary to ensure symmetry as follows: $D'_i = D_i \parallel P_i, \quad P_i = S - |D_i|, \quad |D'_i| = S$.
Encryption: Each padded block D'_i is encrypted using a symmetric key K generated by the HSMs: $E_i = \text{Enc}_K(D'_i)$, where $\text{Enc}_K(\cdot)$ denotes the symmetric encryption function.
BMHT Construction: The BMHT consists of N fixed and sorted leaf nodes, each representing an encrypted block E_i with an assigned key k_i. The hash of each leaf node is: $H_{\text{leaf}_i} = \text{Hash}(k_i \parallel E_i)$. Sorting the leaf nodes by their keys k_i facilitates efficient proofs verifications (i.e., $\{H_{\text{leaf}_1}, H_{\text{leaf}_2}, \ldots, H_{\text{leaf}_N}\}$ where $k_1 < k_2 < \ldots < k_N$). Internal nodes are computed recursively as: $H_{\text{node}} = \text{Hash}(H_{\text{left}} \parallel H_{\text{right}})$, with the root hash serving as a commitment to data integrity: $H_{\text{root}} = \text{Hash}(H_{\text{left-subtree}} \parallel H_{\text{right-subtree}})$.
Tree Symmetry: To maintain a balanced tree, the height h is determined by: $h = \lceil \log_2(C/S) \rceil$. Symmetry ensures consistent proof sizes and predictable storage management.

5.2 Verification via BMHT and Proofs

After data is stored, in addition to the DO, the CP also constructs the BMHT to support storage, membership, and non-membership proofs, while the DO provides COs with a signed, sorted list of key identifiers to verify that the CP's proofs reference the correct keys.
Storage Proof: The storage proof ensures that the dataset is correctly stored in the BMHT and remains within the fixed number of leaves, preventing expansion. The BMHT root is cryptographically linked to the dataset using an HSMs-signed commitment: $\tau = \text{Sign}_{\text{HSMs}}(H_{\text{root}} \parallel N \parallel S), \quad H'_{\text{root}} = \text{Hash}(H_{\text{root}} \parallel \tau)$.
The CP provides: $\pi_{\text{storage}} = \{H_{\text{leaf}_1}, H_{\text{leaf}_2}, \ldots, H_{\text{leaf}_N}, H'_{\text{root}}\}$, where each leaf is computed as $H_{\text{leaf}_i} = \text{Hash}(k_i \parallel E_i)$. The verifier (DO/CO) checks: $H'_{\text{root}} \overset{?}{=} \text{Hash}(H_{\text{root}} \parallel \tau)$, and reconstructs H_{root} from π_{storage} to verify its consistency. The proof is valid if the recomputed H_{root} matches the expected value, confirming that the storage structure has not been altered.
Membership Proof: The membership proof verifies the inclusion of a specific block D_i in the BMHT. The provider provides the Merkle path $\pi_{\text{membership}}$ from the corresponding leaf H_{leaf_i} to the H_{root}, while the DO provides the expected root hash H_{root} for verification.: $\pi_{\text{membership}} = \{H_{\text{sibling}_1}, H_{\text{sibling}_2}, \ldots, H_{\text{sibling}_h}\}$, where $h = \lceil \log_2(N) \rceil$ is the height of the tree. The verifier performs the following steps: 1. Computes the root hash iteratively:

$$H^{(1)}_{\text{path}} = \text{Hash}(H_{\text{leaf}_i} \parallel H_{\text{sibling}_1}),$$

$$H^{(j+1)}_{\text{path}} = \begin{cases} \text{Hash}(H^{(j)}_{\text{path}} \parallel H_{\text{sibling}_{j+1}}), & \text{if left-child,} \\ \text{Hash}(H_{\text{sibling}_{j+1}} \parallel H^{(j)}_{\text{path}}), & \text{if right-child} \end{cases} \tag{1}$$

for $j = 1, \ldots, h-1$. 2. Verifies: $H^{(h)}_{\text{path}} = H_{\text{root}}$. The proof is valid if only the computed root matches the DO's provided H_{root}.
Non-Membership Proof: The non-membership proof ensures that a queried key $k \notin \{k_1, k_2, \ldots, k_N\}$, leveraging the sorted property of BMHT leaves and the concepts of frontier and boundary sets [23]. Sentinel nodes $-\infty$ and $+\infty$ are introduced to handle edge cases where k is smaller than k_1 or larger than k_N. These are fixed, predefined virtual keys serving as placeholders at the BMHT boundaries [6].
Definitions and Proof Generation by CP:
Sentinel Nodes: Predefined values are assigned to sentinel hashes:

$$H_{-\infty} = \text{Hash}(\text{`-infinity"}), \quad H_{+\infty} = \text{Hash}(\text{+infinity"}),$$

ensuring they are publicly known and verifiable constants. Sentinel nodes $-\infty$ and $+\infty$ are included as the first and last leaves in the BMHT: Leaves $= [H_{-\infty}, H_1, H_2, \ldots, H_N, H_{+\infty}]$.
Frontier Set (*F*): The smallest subset of keys required to validate $k \notin \{k_1, \ldots, k_N\}$: $F = \{k_{\text{pred}}, k_{\text{succ}}\}, \quad$ where $k_{\text{pred}} < k < k_{\text{succ}}$. For edge cases: $k_{\text{pred}} = -\infty$ if $k < k_1, \quad k_{\text{succ}} = +\infty$ if $k > k_N$.
Boundary Set (*B*): The Merkle hashes along the paths from $H_{\text{leaf}_{\text{pred}}}$ and $H_{\text{leaf}_{\text{succ}}}$ to the root: $B = \{\pi_{\text{pred}}, \pi_{\text{succ}}\}$, where π_{pred} and π_{succ} are the Merkle paths.
Proof Generation: The CP identifies k_{pred} and k_{succ} such that $k_{\text{pred}} < k < k_{\text{succ}}$, then computes π_{pred} and π_{succ}.
Verification by the Verifier:
Check Key Order: Verify: $k_{\text{pred}} < k < k_{\text{succ}}$. For edge cases, confirm: $k_{\text{pred}} = -\infty$ or $k_{\text{succ}} = +\infty$, ensuring sentinel nodes are correctly referenced.
Recomputing Root from Boundary Set: The root is recomputed using π_{pred} and π_{succ} as defined in Equation (1), where the initial condition is $H^{(1)}_{\text{path-pred}} = H_{\text{root}}$. Likewise, the successor path satisfies $H^{(h)}_{\text{path-succ}} = H_{\text{root}}$.

```
Initialization$(\lambda, N, S, D, \{PK_i, SK_{\mathbf{HSM}}\})$:
        $\lambda \in \mathbb{N}$ (security parameter), $N, S \in \mathbb{N}$ (blocks, block size)
        $C = N \cdot S$ (total storage capacity), $D_i \in \mathbb{B}^*$ (data blocks),
        $k_i \in \mathbb{B}^*$ is the key index assigned to $D_i$,
        $PK_i, SK_i$ (public/private key pairs for co-owners),
        $SK_{\text{HSM}}$ (HSM private key for signing operations).
    For each $i \in [N]$ :
        Pad: $D'_i = D_i \parallel \text{pad}(S - |D_i|)$,    ensuring $|D'_i| = S$.
        Encrypt: $E_i \leftarrow \text{Enc}(D'_i, PK_i)$.
        Compute leaf hash: $H(L_i) = h(k_i \parallel E_i)$.
    BMHT Construction: $H_{\text{root}} \leftarrow \text{BMHT}(\{H(L_i)\}_{i=1}^{N})$.
    Storage Commitment: $\tau \leftarrow \text{Sign}_{\text{HSM}}(H_{\text{root}} \parallel N \parallel S)$.
    Output: $H_{\text{root}}, \tau, \{E_i\}_{i=1}^{N}, SK_{\text{HSM}}$.
Preprocessing$(N, S, \{E_i\})$:
        Map blocks: $L_i \leftarrow E_i, \quad \forall i \in [N]$.
        Sort by keys: Order $\{L_i\}_{i=1}^{N}$ in ascending order by $k_i$.
        Output: $H_{\text{root}}, \{H(L_i)\}_{i=1}^{N}$.

Proof Generation$(H_{\text{root}}, \{H(L_i)\}, SK_{\mathbf{HSM}})$:
    Membership proof: $\pi_{\text{mem}} = \{H_{\text{path}}\}$.
    Non-membership proof: $\pi_{\text{non-mem}} = \{H_{\text{pred}}, H_{\text{succ}}\}$.
    Overwrite proof: $\pi_{\text{overwrite}} = \pi_{\text{mem}} \cup \pi_{\text{non-mem}}$.
    Storage proof: $H'_{\text{root}} = \text{Hash}(H_{\text{root}} \parallel \tau)$.
    Zero-residual overwrite: $D_i \leftarrow \pi_k(\text{ReverseBits}(D_i \oplus \text{PRF}(k, H(D_i))))$, $\forall i$.
    Output: $\pi = (\pi_{\text{store}}, \pi_{\text{mem}}, \pi_{\text{non-mem}}, \pi_{\text{overwrite}})$.

Proof Verification$(\pi, \{H(L_i)\}, H_{\text{root}}, \{PK_i\})$:
        Membership check: $H_{\text{path}}^{\text{comp}} \stackrel{?}{=} H_{\text{root}}$.
        Non-membership check: $k_{\text{pred}} < k < k_{\text{succ}}$, $H_{\text{pred}}, H_{\text{succ}} \stackrel{?}{=} H_{\text{root}}$.
        Overwrite check: $\pi_{\text{overwrite}} \stackrel{?}{=} \pi_{\text{mem}} \cup \pi_{\text{non-mem}}$.
        Storage proof check: $H'_{\text{root}} \stackrel{?}{=} \text{Hash}(H_{\text{root}} \parallel \tau)$.
        Decision: return VALID if all checks pass.
```

Figure 2: `Provable_Structure_Function` Overview for Secure Deletion in Multi-Cloud with Zero-Residuals and Verifiability.

Validate Non-Membership: The proof is valid if: $H_{\text{path-pred}}^{(h)} = H_{\text{root}}$, $H_{\text{path-succ}}^{(h)} = H_{\text{root}}$, and: $k \notin \{k_{\text{pred}}, k_{\text{succ}}\}$. If $k = k_{\text{pred}}$, $k = k_{\text{succ}}$, or the paths fail to recompute H_{root}, the proof is invalid. Verification is efficient, with storage proofs requiring $O(N)$ hash operations for root computation, while incremental updates rehash only affected parts, reducing complexity to $O(\log_2 N)$. BMHT leverages an HSM-signed commitment, enabling $O(1)$ verification by checking a single hash signature. Membership proofs operate in $O(\log_2 N)$, while non-membership requires verifying two paths, yielding $O(\log_2 N)$. Security relies on collision-resistant hash functions, ensuring data integrity, with adversarial forgery probability bounded by $\epsilon(k)$, where k is the security parameter.

5.3 Data Sharing, Secure Access, Zero-Residuals Overwriting, and Global Consistency

This phase ensures secure data sharing, controlled access, zero-residual overwriting, and verifiable deletion across CPs using HSMs, PKI, Secure Enclaves, and GMF.

Data Sharing and Encryption: The HSMs generates asymmetric key pairs for co-owners (COs) at the respective provider. For each CO$_j$, let: $\{PK_j, SK_j\} \leftarrow \text{KeyGen}_{\text{HSM}}(1^k)$, where PK_j and SK_j are the public and private keys, respectively, and k is the security parameter. The main HSM in the organizer CP encrypts each block D_i for a CO using $E_i = \text{Enc}_{PK_j}(D_i)$ provided by their HSM. The encrypted blocks $\{E_1, E_2, \ldots, E_N\}$ are uploaded to CPs, ensuring only authorized COs with SK_j can decrypt: $D_i = \text{Dec}_{SK_j}(E_i)$. Their HSMs is responsible for decrypting the data.

Secure Access and Session Keys: Data access occurs within a Secure Enclave initialized by the CP to prevent persistent local copies. The HSMs generates a session-specific symmetric key (i.e., Time-bounded key): $K_{\text{session}} \leftarrow \text{KeyGen}_{\text{HSM}}(1^k)$. The session key K_{session} encrypts the data for communication between the enclave and COs: $C_i = \text{Enc}_{K_{\text{session}}}(D_i)$. Once the session ends, the HSM securely destructs K_{session}, ensuring the key's irrecoverability: Destroy(K_{session}).

Zero-Residual Permuted Overwriting: Old data D_{old} is securely overwritten with D_{new} using a three-phase process based on PRFs and PRPs [16, 32], ensuring zero-residual deletion:
1. PRF Masking: Mask D_{old} via PRF: $D_1 = D_{\text{old}} \oplus f_k(H(D_{\text{old}}))$.
2. Bit Reversal: Reverse bits and re-mask: $D_2 = \text{ReverseBits}(D_1) \oplus f_k(H(D_1))$.
3. Entropic Shuffling: Apply PRP for randomness: $D_{\text{new}} = \pi_k(D_2)$.
The PRP assigns D_{new} a new position: $P_{\text{new}} = \pi_k(P_{\text{old}})$, updating the BMHT root: $H_{\text{leaf}_{\text{new}}} = \text{Hash}(k_{\text{new}} \parallel D_{\text{new}})$ at each cycle.
Irreversibility Guarantee: Reconstructing D_{old} requires inverting PRF, PRP, and bit reversals:

$$D_{\text{old}} = f_k^{-1}\Big(\text{ReverseBits}^{-1}\big(\pi_k^{-1}(D_{\text{new}}) \oplus f_k(H(D_1))\big)\Big),$$

Overwriting Proof with Storage Commitment: The proof ensures that D_{old} at position P_{old} is securely replaced by D_{new} at P_{new}, where $P_{\text{old}}, P_{\text{new}} \in \mathbb{N}$, while maintaining strict storage enforcement: $\pi_{\text{overwrite}} = \pi_{\text{membership}} \cup \pi_{\text{non-membership}}$
After overwriting, the updated root must satisfy the fixed storage commitment, ensuring integrity: $H'_{\text{root}} = \text{Hash}(H_{\text{root}} \parallel \tau)$. To verify the proof, the verifier checks inclusion by ensuring $H_{\text{path-new}}^{(h)} = H_{\text{root}}$, confirming that D_{new} is correctly inserted. Exclusion is verified by ensuring that $k_{\text{pred}} < k_{\text{old}} < k_{\text{succ}}$ holds, along with the recomputed paths satisfying $H_{\text{pred}}^{(h)} = H_{\text{succ}}^{(h)} = H_{\text{root}}$, guaranteeing that D_{old} is removed. The storage enforcement is validated using $\text{Verify}_{\text{HSM}}(\tau, H_{\text{root}}, N, S)$. The proof is accepted if all checks hold:

$$\pi_{\text{overwrite}} = \begin{cases} 1, & \text{if all checks are satisfied,} \\ 0, & \text{otherwise.} \end{cases}$$

This ensures that permuted overwriting respects the limited storage constraints (N, S) and prevents residual data from persisting.

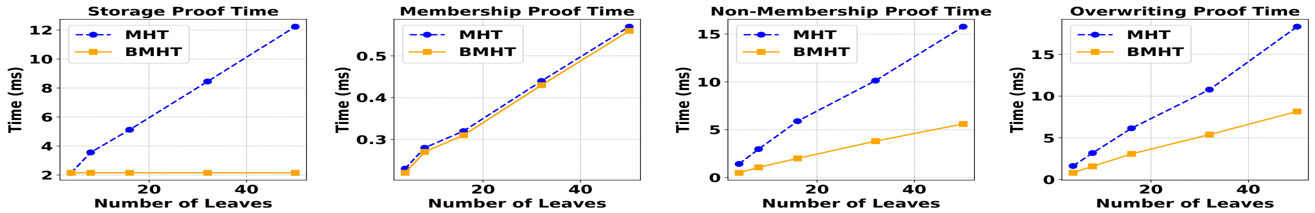

Figure 3: Comparison of BMHT and Traditional MHT Verification Times for Different Proofs.

Global Proof Synchronization: The GMF aggregates BMHT roots H_{root} from all CPs, ensuring global consistency. The aggregated root is periodically (i.e., Based on configuration) recomputed and stored in the organizer CP's HSMs: $H_{\text{GMF}} = \text{Hash}(H_{\text{root}_1} \parallel H_{\text{root}_2} \parallel \dots \parallel H_{\text{root}_m})$, where M is the total number of CPs. This guarantees consistent verification across asynchronous environments. Encryption operations scale linearly with the number of blocks, as asymmetric encryption for data sharing requires $O(N)$ computations. Overwriting via PRF/PRP occurs per block, maintaining $O(N)$ complexity. Verifying overwriting proofs, including membership and non-membership, follows $O(\log_2 N)$ complexity. The GMF aggregates BMHT roots from M providers into a single-level Merkle structure. Construction requires hashing all M roots, yielding $O(M)$, while verification is constant-time at $O(1)$, ensuring minimal overhead. The security of overwriting and GMF proofs relies on collision-resistant hashes, PRFs, PRPs, and bit reversals, ensuring irreversibility with adversarial success probability $\epsilon(k)$.

6 FORMAL ANALYSIS

Theorem 6.1 (Correctness). *A valid proof π generated by the CP enables the verifier (DO or CO) to verify storage, membership, non-membership, and overwriting operations (Def. 2.3). Formally, correctness holds if:* $\Pr[V_\pi(D, H_{root}, \pi) = 1] \geq 1 - \epsilon(k)$, *where V_π is the verification function (i.e., mentioned §2.3), and the probability is taken over the randomness of the cryptographic hash function and pseudorandom permutations (PRPs) [32].*

Theorem 6.2 (Security). *A malicious CP attempting to forge a proof or introduce inconsistencies across the GMF will be detected with high probability (Def. 2.3). For any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, we have:*

$$\Pr\left[\begin{array}{c|} V_\pi(\pi') = 1 \\ \left(\pi' \neq \pi\right) \vee \left(\exists\, i \in \{1, \dots, n\} : \mathcal{H}^{(i)}_{BMHT} \notin \mathcal{H}_{GMF}\right) \end{array} \right] \leq \epsilon(k),$$

where $\epsilon(k)$ is negligible in the security parameter k.

7 RESULTS AND DISCUSSION

Verification Time (Figure 3): This experiment was conducted on a local MacBook with a 2.3 GHz Intel Core *i*5 and 8 GB RAM. It evaluates the verification time for storage, membership, non-membership, and overwriting proofs in both MHT and BMHT locally. Table 1 highlights BMHT's efficiency, particularly in non-membership and overwriting proofs. For storage proof, BMHT achieves constant-time storage frequent verification ($O(1)$), maintaining 2.15 ms across all leaf sizes due to cryptographic enforcement by the HSMs, which eliminates the need for full-tree recomputation. While MHT requires full-tree recomputation at all times, increasing from 2.15 ms at $N = 4$ to 12.23 ms at $N = 50$. Membership proofs are nearly identical, as both follow a Merkle path to the root ($O(\log N)$), yielding 0.56 ms for BMHT and 0.57 ms for MHT at $N = 50$. For non-membership proofs, BMHT efficiently finds predecessor-successor nodes in $O(\log N)$, reducing verification time to 5.60 ms at $N = 50$ compared to MHT's 15.77 ms, which requires scanning all leaves. Similarly, BMHT's overwriting proofs leverage its structured design, achieving 8.16 ms at $N = 50$ versus MHT's 18.32 ms.

Table 1: Comparison of BMHT and MHT Verification Time Complexities.

Proof Type	BMHT Complexity	MHT Complexity
Storage Proof	$O(1)$	$O(N)$
Membership Proof	$O(\log N)$	$O(\log N)$
Non-Membership Proof	$O(\log N)$	$O(N)$
Overwriting Proof	$O(\log N)$	$O(N)$

8 RELATED WORK

Data link removal, encryption, and overwriting are the three main strategies that now constitute secure data deletion in the cloud environment. Although these techniques cover several data deletion issues, their shortcomings are especially noticeable regarding co-owned data erasure in multi-cloud scenarios. The link removal strategy severs a file's logical connection to the filesystem [11, 28]. Kavous et al. [25] improved this with delayed deletion to prevent accidental loss. However, the data remains in storage and is recoverable via forensic methods, making this approach unsuitable for securely erasing co-owned data in multi-cloud settings where physical deletion is essential. The data encryption strategy renders data inaccessible by encrypting it and deleting the key. Cachin et al. [4] enforced deletion via policy-based key removal. Darwish et al. [7, 8] introduced key-decay schemes, either corrupting ephemeral keys or enforcing audience-based expiration. Jaeheung et al. [18] stored keys with metadata, deleting them to make ciphertext inaccessible. Junfeng et al. [12] proposed TPM-based verification, ensuring deletion but limiting scalability due to TPM constraints. While effective for logical deletion, encryption-based methods rely on secure key removal, leaving ciphertext vulnerable in multi-cloud environments. Managing keys across providers adds complexity, requiring verifiable deletion to prevent unauthorized access. The overwriting strategy ensures data irretrievability by replacing it with random or predefined patterns. Yuchuan et al. [20] optimized overwriting efficiency with a rule transposition algorithm (RTA). Daniele and Gene [26] introduced PoSE-s, securely overwriting storage media to prevent data recovery. Tian et al. [31] proposed SEAD-OO, integrating orderly overwriting with attribute-based encryption for fine-grained access control and blockchain-based deletion verification. Singh et al. [29] highlighted SSD wear-leveling issues that hinder zero residual overwriting. Existing approaches lack verifiable deletion for co-owned data in multi-cloud environments, where global and verifiable proof of deletion is essential.

9 CONCLUSION & FUTURE WORK

We propose a provable co-owned data deletion framework for multi-cloud environments, ensuring data never leaves the cloud. HSMs manage key lifecycles, while Secure Enclaves enable on-cloud processing without local remnants. Verifiability is enforced through BMHT for storage constraints and GMF for global proof verification. Zero-residual overwriting guarantees complete erasure with cryptographic proofs. Future work includes deploying the solution on AWS using CloudHSM, Nitro Enclaves, and Python-based BMHT, evaluating scalability, computational overhead, proof efficiency, and cost analysis and formal validation.

ACKNOWLEDGMENTS

This work was supported by the European Commission under the Horizon Europe Programme as part of the projects RECITALS (Grant Agreement #101168490) and SafeHorizon (Grant Agreement #101168562). The content of this article does not reflect the official opinion of the European Union. Responsibility for the information and views expressed therein lies entirely with the authors.

REFERENCES

[1] Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. 2013. Innovative technology for CPU based attestation and sealing. In *Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy*, Vol. 13. ACM New York, NY, USA.

[2] Ekaba Bisong. 2019. An overview of google cloud platform services. *Building Machine Learning and Deep Learning Models on Google Cloud Platform* (2019), 7–10.

[3] Stanislav Boboň. 2021. Analysis of NIST FIPS 140-2 Security Certificates. *Masaryk University: Brno, Czech Republic* (2021).

[4] Christian Cachin, Kristiyan Haralambiev, Hsu-Chun Hsiao, and Alessandro Sorniotti. 2013. Policy-based secure deletion. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 259–270.

[5] Michael Collier and Robin Shahan. 2015. *Microsoft Azure Essentials-fundamentals of Azure*. Microsoft Press.

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2022. *Introduction to algorithms*. MIT press.

[7] Marwan Adnan Darwish and Georgios Smaragdakis. 2024. Disjunctive Multi-Level Digital Forgetting Scheme. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*. 112–121.

[8] Marwan Adnan Darwish and Apostolis Zarras. 2023. Digital Forgetting Using Key Decay. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*. 34–41.

[9] European Commission. Accessed: 2025-01-26. European Health Data Space. https://health.ec.europa.eu/ehealth-digital-health-and-care/european-health-data-space_en

[10] European Union. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council. *Regulation (eu)* 679 (2016), 2016.

[11] Simson L. Garfinkel and Abhi Shelat. 2003. Remembrance of Data Passed: A Study of Disk Sanitization Practices. *IEEE Security & Privacy* 1, 1 (2003), 17–27.

[12] Feng Hao, Dylan Clarke, and Avelino Francisco Zorzo. 2015. Deleting Secret Data with Public Verifiability. *IEEE Transactions on Dependable and Secure Computing* 13, 6 (2015), 617–629.

[13] Rada Hussein, Lucas Scherdel, Frederic Nicolet, and Fernando Martin-Sanchez. 2023. Towards the European Health Data Space (EHDS) Ecosystem: A Survey Research on Future Health Data Scenarios. *International Journal of Medical Informatics* 170 (2023), 104949.

[14] Maritza Johnson, Serge Egelman, and Steven M Bellovin. 2012. Facebook and Privacy: It's Complicated. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*. 1–15.

[15] Ruogu Kang, Laura Dabbish, Nathaniel Fruchter, and Sara Kiesler. 2015. Data Just Goes Everywhere: User Mental Models of the Internet and Implications for Privacy and Security. In *Eleventh symposium on usable privacy and security (SOUPS 2015)*. 39–52.

[16] Richard Kissel, Matthew Scholl, Steven Skolochenko, and Xing Li. 2006. Guidelines for Media Sanitization. *NIST Special Publication 800-88* (2006).

[17] Mariusz Krzysztofek. 2018. *GDPR: General Data Protection Regulation (EU) 2016/679: Post-reform Personal Data Protection in the European Union*. Kluwer Law International BV.

[18] Jaeheung Lee, Sangho Yi, Junyoung Heo, Hyungbae Park, Sung Y Shin, and Yookun Cho. 2010. An Efficient Secure Deletion Scheme for Flash File Systems. *J. Inf. Sci. Eng.* 26, 1 (2010), 27–38.

[19] Chang Liu, Rajiv Ranjan, Chi Yang, Xuyun Zhang, Lizhe Wang, and Jinjun Chen. 2014. MuR-DPA: Top-Down Levelled Multi-Replica Merkle Hash Tree Based Secure Public Auditing for Dynamic Big Data Storage on Cloud. *IEEE Trans. Comput.* 64, 9 (2014), 2609–2622.

[20] Yuchuan Luo, Ming Xu, Shaojing Fu, and Dongsheng Wang. 2016. Enabling Assured Deletion in the Cloud Storage by Overwriting. In *Proceedings of the 4th ACM international workshop on security in cloud computing*. 17–23.

[21] Stathis Mavrovouniotis and Mick Ganley. 2013. Hardware Security Modules. In *Secure Smart Embedded Devices, Platforms and Applications*. Springer, 383–405.

[22] Yinbin Miao, Ximeng Liu, Kim-Kwang Raymond Choo, Robert H Deng, Jiguo Li, Hongwei Li, and Jianfeng Ma. 2019. Privacy-preserving attribute-based keyword search in shared multi-owner setting. *IEEE Transactions on Dependable and Secure Computing* 18, 3 (2019), 1080–1094.

[23] Silvio Micali, Michael Rabin, and Joe Kilian. 2003. Zero-knowledge Sets. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.* IEEE, 80–91.

[24] Ambar Murillo, Andreas Kramm, Sebastian Schnorf, and Alexander De Luca. 2018. "If I press delete, it's gone"-User Understanding of Online Data Deletion and Expiration. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*. 329–339.

[25] Kavous Salehzadeh Niksirat, Diana Korka, Quentin Jacquemin, Céline Vanini, Mathias Humbert, Mauro Cherubini, Sylvain Métille, and Kévin Huguenin. 2024. Security and Privacy with Second-Hand Storage Devices: A User-Centric Perspective from Switzerland. *Proceedings on Privacy Enhancing Technologies (PoPETs)* 2024, 2 (2024), 22.

[26] Daniele Perito and Gene Tsudik. 2010. Secure Code Update for Embedded Devices via Proofs of Secure Erasure. In *Computer Security–ESORICS 2010: 15th European Symposium on Research in Computer Security, Athens, Greece, September 20-22, 2010. Proceedings 15*. Springer, 643–662.

[27] Kopo Marvin Ramokapane and Awais Rashid. 2023. ExD: Explainable Deletion. In *Proceedings of the 2023 New Security Paradigms Workshop*. 34–47.

[28] Joel Reardon, David Basin, and Srdjan Capkun. 2013. Sok: Secure data deletion. In *2013 IEEE Symposium on Security and Privacy*. IEEE, 301–315.

[29] Bhupendra Singh, Ravi Saharan, Gaurav Somani, and Gaurav Gupta. 2016. Secure File Deletion for Solid State Drives. In *Advances in Digital Forensics XII: 12th IFIP WG 11.9 International Conference, New Delhi, January 4-6, 2016, Revised Selected Papers 12*. Springer, 345–362.

[30] Emil Stefanov and Elaine Shi. 2013. Multi-Cloud Oblivious Storage. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 247–258.

[31] Junfeng Tian and Tianfeng Zhang. 2022. Secure and Effective Assured Deletion Scheme with Orderly Overwriting for Cloud Data. *The Journal of Supercomputing* 78, 7 (2022), 9326–9354.

[32] Marten Van Dijk, Ari Juels, Alina Oprea, Ronald L. Rivest, Emil Stefanov, and Nikos Triandopoulos. 2012. Hourglass Schemes: Howto Prove that Cloud Files Are Encrypted. In *Proceedings of the 2012 ACM conference on Computer and Communications Security*. 265–280.

[33] Vinod Veeramachaneni. 2025. Integrating Zero Trust Principles into IAM for Enhanced Cloud Security. *Recent Trends in Cloud Computing and Web Engineering* 7, 1 (2025), 78–92.

[34] Michael Wei, Laura Grupp, Frederick E Spada, and Steven Swanson. 2011. Reliably Erasing Data From Flash-Based Solid State Drives. In *9th USENIX Conference on File and Storage Technologies (FAST 11)*.

[35] Andreas Wittig and Michael Wittig. 2023. *Amazon Web Services in Action: An in-depth guide to AWS*. Simon and Schuster.

[36] Jinbo Xiong, Ximeng Liu, Zhiqiang Yao, Jianfeng Ma, Qi Li, Kui Geng, and Patrick S. Chen. 2014. A Secure Data Self-Destructing Scheme In Cloud Computing. *IEEE Transactions on Cloud Computing* 2, 4 (2014), 448–458.

[37] Dong Zheng, Liang Xue, Chao Yu, Yannan Li, and Yong Yu. 2020. Toward Assured Data Deletion in Cloud Storage. *IEEE Network* 34, 3 (2020), 101–107.