AKESO: Bringing Post-Compromise Security to Cloud Storage (abstract #30)

Lily Gloudemans *William & Mary* Pankaj Niroula *William & Mary* Aashutosh Poudel *William & Mary* Stephen Herwig *William & Mary*

Problem. As organizations increasingly store their data in the cloud, a primary concern is preventing— and recovering from—data leaks. While cloud service providers (CSPs) offer various options for encrypting an organization's data server-side, these approaches unfortunately rely on the CSP having access to the encryption key, thereby allowing for unconsented data disclosure. Alternatively, an organization could first apply client-side encryption of the data, but this incurs a heavy burden of key management, thus increasing the likelihood of key compromise. **In this work, we present AKESO—an efficient, encrypted cloud storage system that does not share the encryption keys with the cloud, and which automatically recovers from a key compromise.**

Challenges and assumptions. We assume the storage clients are outside of the cloud. The CSP is semi-trusted: it faithfully runs the cloud systems, but may be compelled to disclose data to law enforcement agencies. An adversary may *completely*—but only *temporarily*—compromise a storage client; notably the adversary is unable to achieve *persistence* on either the client's device or its network.

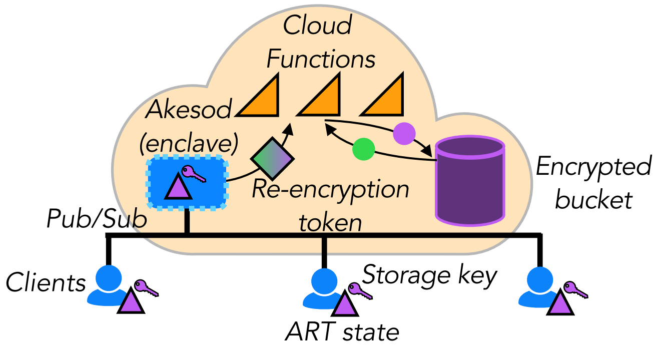

Figure 1: Architectural overview of AKESO

Our approach. Figure 1 depicts a high-level overview of the AKESO architecture. To efficiently establish and rotate a group storage encryption key, an AKESO storage group uses the Asynchronous Ratcheting Tree (ART) [2]—a Diffie-Hellman tree-based data structure that has found broad application in secure end-to-end group messaging. ART notably guarantees post-compromise security: for an adversary to maintain indefinite access to the fresh group key, it is not enough to temporarily compromise a group member; rather, the adversary must be a persistent man-in-the-middle to all key updates.

To efficiently re-encrypt all objects, AKESO uses an updatable encryption scheme. Abstractly, a group member sends a small update token to untrusted cloud functions, and the functions use this token to rotate the encrypted storage from the old key to the new key without learning the plaintext. Since many updatable encryption schemes require that a trusted group member download a portion of the ciphertext to generate the token, AKESO runs a special member called AKESOD in a cloud enclave, thus minimizing data egress.

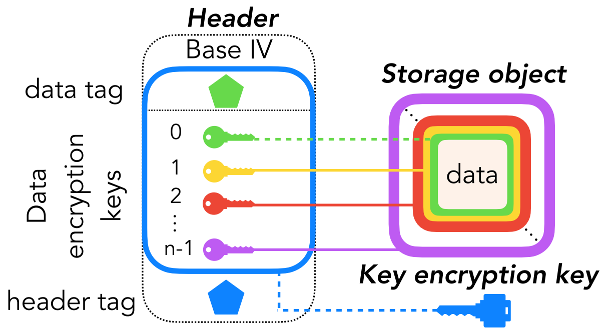

Figure 2: Updatable encryption using nested AES. The scheme encrypts an object with multiple DEKs, and encrypts this list with the ART group key. The group key and initial DEK are private; the subsequent DEKs are the update tokens.

Implementation and evaluation. We implement AKESO using Google Cloud Platform (GCP). For the clients, we modify Google's open source FUSE (Filesystem in Userspace) adapter—`gcsfuse`—which allows users to mount a storage bucket as a local filesystem. Specifically, we integrate the ART protocol into `gcsfuse`, and amend `gcsfuse` to use Google's Pub/Sub for key update notifications. Figure 2 shows how AKESO implements a practical version [1] of updatable encryption based on nested-AES. Figure 3 shows that AKESO's latency for reading a bucket is competitive with Google's current encryption options.

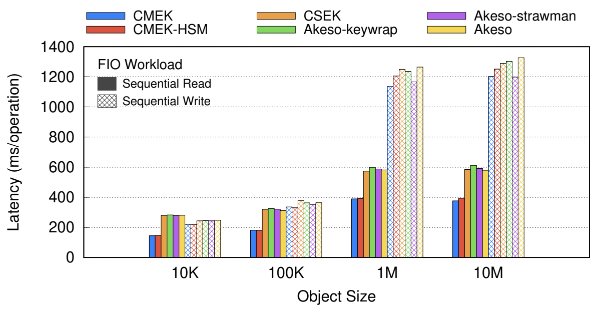

Figure 3: AKESO's sequential I/O performance

References

[1] Dan Boneh, Saba Eskandarian, Sam Kim, and Maurice Shih. Improving speed and security in updatable encryption schemes. In *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, 2020.

[2] Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. On ends-to-ends encryption: Asynchronous group messaging with strong security guarantees. In *ACM Conference on Computer and Communications Security (CCS)*, 2018.