Decentralised access control project

DARE 2024 - Final Project

Made by Dinis Sousa and Gustavo Costa

Motivation

This project is an implementation of the algorithm presented in Martin Kleppmann's "Recovering from key compromise in decentralised access control systems".

Set-up

Set up a Python virtual environment in a new, empty directory:

```
python3 -m venv venv
source venv/bin/activate # or venv/Scripts/activate
pip install -r requirements.txt
```

Structure

The repository consists of the following key files:

`acl_operations.py` : Defines the operations for group management:

- `create_op` : Creates a new group.
- `add_op` : Adds a new member to the group.
- `remove_op` : Removes an existing member from the group. All operations are signed and include dependencies to ensure causal consistency.

`acl_test_operations.py` : Contains unit tests to validate the access control implementation. It:

- Tests scenarios like adding/removing members and checking operation precedence.
- Includes methods to simulate and validate the system's behavior under various conditions, such as conflicting operations.

`acl_validation.py` : Implements algorithms to validate and manage operations:

- `compute_seniority` : Assigns seniority based on the operation graph.
- `authority_graph` : Builds a graph to track the influence of operations.
- `compute_validity` : Determines valid operations based on graph analysis.

`acl_helpers.py` : Provides utility functions:

- Cryptographic hashing (`hex_hash`) and message signing/verification (`sign_msg` , `verify_msg`).
- Helpers for transitive closure in operation graphs.

What is the problem

In decentralized systems like secure group messaging or collaboration tools, users can concurrently modify access permissions. This creates challenges, especially when devices or credentials are compromised. For example:

```
An adversary using stolen keys can manipulate membership.
Concurrent removal of users may lead to inconsistencies in group membership.
```

The paper introduces an algorithm that:

```
Ensures legitimate users can reliably revoke compromised devices.
Resolves conflicts in decentralized systems without relying on a central authority.
```

Solution

The proposed solution addresses the challenge of recovering from key compromise and resolving conflicts in decentralized access control systems. Key elements of the solution include:

Conflict Resolution with Seniority Ranking: Devices and their operations are assigned a seniority based on the order of their (first) inclusion in the group. When two devices try to remove each other (or perform conflicting operations), the conflict is resolved in favor of the more senior device. This deterministic approach ensures consistency across all devices.

Hash-based Directed Acyclic Graph (DAG): Operations (e.g., add/remove members) are encoded in a cryptographically secure hash-DAG. The DAG captures the causal relationships between operations, enabling detection of concurrent and conflicting changes.

Revocation Handling: Compromised devices can be reliably removed, and any unauthorized operations performed by them (e.g., adding fake members) are automatically invalidated during conflict resolution.

Decentralized Convergence: The algorithm ensures that all devices converge to the same group membership, regardless of the order in which they receive operations, leveraging Conflict-free Replicated Data Type (CRDT) principles.

Security Guarantees: Even if adversaries exploit stolen keys to perform malicious actions, the system ensures that their actions are reversible and do not impact legitimate users, given that there is a more senior legitimate user than any of the stolen users keys.