

SafeVault : Stockage distribué sécurisé avec secret sharing proactif

Encadrant : Joachim Bruneau-Queyreix

Mots clés : Cryptographie à seuil, secret sharing proactif, stockage distribué, adversaire mobile, tolérance aux pannes

Contexte

Un fichier sensible est souvent chiffré avec une clé secrète. Si cette clé est stockée sur une seule machine, une panne ou une compromission de cette machine peut rendre le fichier inaccessible ou exposer son contenu. Le *secret sharing* permet d'éviter ce point de défaillance unique en découpant un secret en n parts. Dans un schéma à seuil (t, n) , il faut réunir au moins t parts de secret pour retrouver le secret. Moins de t parts ne doivent rien pouvoir révéler.

Cette idée améliore à la fois la confidentialité et la disponibilité. Par exemple, une clé peut être répartie sur cinq serveurs, tout en imposant la coopération de trois serveurs pour déchiffrer un fichier. Toutefois, si un attaquant compromet successivement plusieurs serveurs, il peut finir par accumuler suffisamment de parts. Le *secret sharing proactif* permet de mitiger ce problème. Cela consiste à ce que les parts soient renouvelées régulièrement, sans changer la clé. Les parts volées lors d'une ancienne période ne sont alors plus utiles (tant qu'il y en a moins que t part volées dans la période précédente).

Objectif du projet

Vous développerez un prototype de **stockage distribué sécurisé** basé sur le partage de secret proactif. Dans ce système chaque client chiffre un fichier avant son envoi. Les blocs chiffrés sont distribués sur plusieurs nœuds, tandis que la clé de chiffrement est partagée entre ces nœuds avec un schéma de Shamir. Pour récupérer un fichier, le client contacte au moins t nœuds, reconstruit la clé localement, puis déchiffre le fichier. La solution doit aussi renouveler périodiquement les parts de clé sans jamais reconstruire cette clé sur un serveur.

Architecture du projet

Le prototype comprendra quatre composants principaux :

1. **Client** : il s'authentifie, chiffre et déchiffre les fichiers, et dépose et récupère les données.
2. **Nœuds de stockage** : ils conservent les blocs chiffrés et une part de la clé
3. **Service de coordination** : il gère l'authentification des clients, le partage de fichier, les nœuds disponibles, les périodes de renouvellement et les métadonnées. Il ne connaît jamais les clés en clair.
4. **Banc d'essai** : il simule des pannes et des attaques, puis démontre et mesure le comportement du système.

Scénario de sécurité étudié

La version de base suppose qu'un attaquant peut prendre le contrôle de quelques nœuds, lire leur part et passer ensuite à d'autres nœuds. Il ne peut cependant compromettre qu'au plus c nœuds pendant une même période, avec $c < t$. Il faudra alors être en mesure de montrer et vérifier que le système conserve les propriétés suivantes : moins de t parts ne permettent pas de retrouver la clé, un ensemble d'au moins t parts valides permet de récupérer le fichier, le système continue de fonctionner malgré l'arrêt de certains nœuds, le renouvellement rend les anciennes parts inutilisables pour l'attaquant.

Tâches techniques et livrables

Le projet relevera à minima les tâches suivantes :

- Comprendre le principe de partage de secret Shamir et réaliser une courte revue de l'existant sur le *secret sharing proactif*.
- Implémenter une première version simple : création de parts, reconstruction de la clé et stockage de fichiers chiffrés.
- Implémenter le renouvellement périodique des parts, sans transmettre ni stocker la clé complète sur un serveur.
- Ajouter des communications authentifiées, la gestion des clients, le partage de fichiers, ainsi que des nœuds indisponibles et des tests automatisés.
- Simuler des pannes et des compromissions successives ; comparer le système avec et sans renouvellement des parts.
- Mesurer le temps de dépôt et de récupération, le coût réseau, le temps de renouvellement et le nombre de pannes tolérées.

Les livrables attendus sont :

- Un prototype fonctionnel, documenté et déployable, par exemple avec Docker Compose.
- Des tests couvrant le dépôt, la récupération, les pannes, le renouvellement et les scénarios d'attaque.
- Un rapport expliquant les choix techniques, le modèle de menace, les résultats et les limites du prototype.
- Une démonstration de la solution avec dépôt d'un fichier, arrêt d'un nœud, renouvellement des parts, puis récupération du fichier avec le seuil requis.

Déroulement

Le projet suit une démarche agile / scrum. Les premières séances seront consacrées à la prise en main du sujet, à la définition du scénario de sécurité et à une première démonstration. Les séances suivantes permettront d'intégrer les composants, de tester les pannes et d'analyser les résultats.

Profils et compétences

Le projet s'adresse à des élèves-ingénieurs intéressés par la sécurité, les systèmes distribués et le développement logiciel. Il ne suppose pas de connaissance préalable approfondie en cryptographie : les notions nécessaires seront étudiées au début du projet. Technologies envisageables : Linux, Docker, Git, Go, Python, Rust ou Java, API REST ou gRPC et bibliothèques cryptographiques.