Proactive Dynamic Secret Sharing: Implementation, Benchmarking, Applications to Distributed Password Authenticated Symmetric Key Encryption

Monday 12th December, 2022 - 11:45

Georgi Tyufekchiev
University of Luxembourg
Email: georgi.tyufekchiev.001@student.uni.lu

Ce rapport a été produit sous la supervision de:
Qingju Wang, Marius Lombard-Platet
University of Luxembourg
Email: qingju.wang@uni.lu
Email: marius.lombard-platet@uni.lu

***Abstract*—Les violations de données constituent une menace sérieuse tant pour les entreprises que pour leurs clients. Des mots de passe compromis peuvent entraîner des dommages irréparables dans la vie de quiconque. Pour cette raison, des protocoles d'authentification de mot de passe forts sont nécessaires pour protéger les informations personnelles de l'utilisateur. L'objectif de ce rapport est d'examiner comment les schémas de partage de secrets dynamiques peuvent aider au développement de tels protocoles. Nous fournirons une mise en œuvre de preuve de concept d'un tel schéma, qui a été développé pour récupérer les secrets de la blockchain.**

1. Introduction

Au cours des dernières décennies, la technologie a considérablement évolué, depuis l'invention du World Wide Web jusqu'à l'émergence des services en nuage. Cependant, plus les systèmes deviennent complexes, plus ils sont sujets à des failles de sécurité. Les cybercriminels sont de plus en plus créatifs dans leurs méthodes d'attaque et d'exfiltration des données. L'information étant l'un des actifs les plus précieux, il est essentiel que nous protégions nos données personnelles.
Une solution à ce problème est l'utilisation de protocoles d'authentification par mot de passe distribués. L'objectif de ce projet de semestre de licence est d'examiner le partage dynamique proactif des secrets, qui peut aider au développement de tels protocoles.

2. Qu'est-ce que le DPSS et son application ?

Dynamic proactive secret sharing st une méthode permettant de mettre à jour les clés distribuées après certaines périodes de temps, de sorte que l'attaquant n'ait pas le temps de compromettre la ou les clés. Il s'agit d'une technique utile lorsque l'attaquant peut attaquer dynamiquement des parties (serveurs). De tels schémas fournissent également une solution au "problème des généraux byzantins" [LSP82]. Dans ce problème, les généraux byzantins communiquent uniquement avec des messagers, qui peuvent être corrompus et fournir de fausses informations aux autres parties dans le but de les confondre et de les saboter.

Nous nous intéressons principalement à l'article sur la manière de stocker et de récupérer les secrets de la blockchain [Goy+22], ui est basée sur DPSS..

3. Domaine Scientifique

3.1. Partage du secret

Imaginons que notre actrice Alice ait un mot de passe pour son compte bancaire et qu'elle veuille le stocker quelque part pour pouvoir y accéder à tout moment. Ce faisant, elle court un risque, car ce lieu de stockage constitue un point de défaillance unique. Pour résoudre ce problème, Alice peut "partager" son mot de passe avec quelques-uns de ses amis. Lorsqu'un certain nombre d'entre eux se réunissent, Alice peut récupérer son mot de passe. Dans ce cas, même si quelqu'un est son adversaire, il ne possédera qu'une partie du mot de passe et ne pourra pas le compromettre.

3.2. Promesses polynomiales

Imaginons que nous ayons effectué un grand nombre de calculs et que nous voulions prouver que nous les avons effectués. Tout recalculer et envoyer à un vérificateur est coûteux et inefficace. Dans ce cas, nous pouvons simplement "commettre" [KZG10] les opérations en un seul point de courbe elliptique. Même lorsque les données sont compressées, les données originales restent "séparées" et nous pouvons effectuer des opérations mathématiques sur elles.

3.3. Courbes elliptiques

Les courbes elliptiques sont principalement utilisées en cryptographie comme chiffrement à clé publique. La raison en est que la cryptographie par courbes elliptiques utilise peu le processeur et la mémoire, ce qui permet un cryptage et un décryptage rapides.

3.4. DPSS

Nous allons maintenant examiner brièvement les DPSS définis dans [Goy+22]. Comme indiqué dans l'article, ils "permettent aux utilisateurs d'encoder un secret accompagné d'une condition de libération". Il y a 3 phases principales, chacune composée de quelques protocoles. Nous donnons une brève description des phases, des informations plus détaillées et des preuves de sécurité peuvent être trouvées dans l'article.

3.4.1. Phase de Préparation. Pendant cette phase, les parties obtiendront un partage du ou des secrets. Nous avons les protocoles suivants

- **Distribution**
- **Accusation-Réponse**
- **Vérification**
- **Output**
- **Fresh**

3.4.2. Phase de Transfert. Dans cette phase, l'ancien comité des parties transfère le secret à un nouveau groupe de parties. Les mêmes protocoles sont exécutés, sauf le dernier qui est

- **Refresh**

3.4.3. Phase de Reconstruction. Il s'agit de la phase la plus simple, composée d'un seul protocole. Le client récupère son secret.

3.5. Analyse des Performances

Nous avons évalué uniquement le cas où les parties agissent honnêtement. Pour le benchmarking, un outil fourni par Python est utilisé, appelé "cProfiler". D'après les données recueillies, le système a une complexité de $O(n)$, ce qui est suffisant pour une première version du programme. Bien sûr, pour un protocole cryptographique, il doit être beaucoup plus rapide.

4. Mise en œuvre de la DPSS

4.1. Exigences

Dans cette section, nous décrivons les exigences fonctionnelles et non fonctionnelles que le système aura. Pour les exigences non fonctionnelles, nous utiliserons la norme internationale **ISO/IEC 25051:2014**.

4.1.1. Exigences Fonctionnelles.

FR#1. Dans la phase d'installation et la phase de transfert, les parties sont en mesure de distribuer des parts et des témoins entre elles. Chaque partie est en mesure de publier ses promesses.
FR#2. Si la validation des actions échoue, la partie est en mesure d'accuser la partie qui les a envoyées. La partie accusée est en mesure de publier les partages et les preuves pour une nouvelle validation.
FR#3. Dans le **Protocole de vérification**, la partie peut publier certaines partages et preuves, utilisés pour la vérification.
FR#4. Dans le **Protocole Fresh**, les parties sont en mesure d'envoyer leurs partages depuis le **Protocole Output** vers le client.

FR#5. Le client est capable de valider les partages. Il est également capable de générer un polynôme aléatoire et de l'utiliser pour masquer son secret. Enfin, il est capable de publier le masque. Chaque partie est capable de calculer sa part du secret.
FR#6. Dans le protocole **Refresh**, l'ancien comité est capable de choisir un *king party* et d'envoyer ses partages et ses preuves à ce parti. Le *king* est capable de reconstruire les actions. Les autres parties sont en mesure d'accuser le king d'être compromis.
FR#7. Le client est capable de demander les partages du secret aux parties. Les parties peuvent envoyer leurs partages. Le client est alors capable de vérifier la validité et de reconstruire le secret.

4.1.2. Exigences non Fonctionnelles.

NFR#1. *Complète fonctionnelle*, ce qui signifie que tous les protocoles doivent être mis en œuvre pour que le système fonctionne comme prévu.
NFR#2. *Correction fonctionnelle*, ce qui signifie que toutes les fonctions doivent effectuer des calculs précis et envoyer les résultats corrects à la partie concernée.
NFR#3. *Intégrité*, ce qui signifie que les parties malhonnêtes ne peuvent pas violer l'intégrité des secrets.
NFR#4. *Responsabilité*, ce qui signifie que toutes les parties malhonnêtes peuvent être identifiées si elles décident de frauder.
NFR#5. *Modularité*, ce qui signifie que le système est composé d'éléments distincts, ce qui facilite les modifications et la maintenance du code.

NFR#6. *Réutilisabilité*, ce qui signifie qu'une fonction peut être utilisée pour construire différents composants du système.
NFR#7. *Modifiabilité*, c'est-à-dire que les composants peuvent être modifiés sans introduire de défauts ou dégrader le système.
Exigences non fonctionnelles supplémentaires, qui sont spécifiques au système.

NFR#8. *Multiprocessing*, qui permet de paralléliser certaines parties du système.
NFR#9. Il faut utiliser une bibliothèque cryptographique professionnelle, reconnue par le public.
NFR#10. Les courbes BLS ou BN peuvent être utilisées dans le système.
NFR#11. Seuls les hachages SHA-3 peuvent être utilisés dans le système.

4.2. Processus de développement

Dans cette section, nous décrivons le processus de développement logiciel du système. Nous suivrons le processus incrémental, qui nous permet d'être flexibles et de repérer rapidement les erreurs. Le processus de développement comportera deux phases principales.

4.2.1. Parties Honnêtes.
Durant cette phase, nous supposons que les parties agiront honnêtement durant chaque protocole. Tous les protocoles doivent être implémentés, au moins partiellement, de manière à ce que le client puisse partager son secret et le reconstruire sur demande.

4.2.2. Parties Malveillantes.
Au cours de cette phase, nous supposons que les parties peuvent être malveillantes. Il est obligatoire que tous les protocoles soient mis en œuvre. Les promesses et les preuves doivent être calculées, le cas échéant, et les partages doivent être validés si nécessaire.

5. Conclusion

En conclusion, les recherches menées dans ce rapport ont fourni des informations précieuses sur le thème du partage proactif des secrets. La mise en œuvre et le benchmarking d'un tel protocole indiquent qu'il s'agit effectivement d'une méthode efficace pour partager en toute sécurité des informations sensibles entre plusieurs parties. Des recherches plus approfondies sur ce sujet peuvent aider les organisations qui ont besoin de partager des informations sensibles de manière efficace. Elle est également utile pour le développement des futures technologies blockchain, qui pourraient être utilisées par tout le monde un jour.

6. Déclaration de plagiat

Je déclare avoir connaissance des faits suivants :

- En tant qu'étudiant de l'Université du Luxembourg, je dois respecter les règles de l'honnêteté intellectuelle, notamment ne pas recourir au plagiat, à la fraude ou à toute autre méthode illégale ou contraire à l'intégrité scientifique.
- Mon rapport sera vérifié pour le plagiat et si le contrôle du plagiat est positif, une procédure interne sera engagée par mon tuteur. Il m'est conseillé de demander une vérification préalable par mon tuteur pour éviter tout problème.
- Comme déclaré dans la procédure d'évaluation de l'Université du Luxembourg, le plagiat est commis lorsque la source d'information utilisée dans un travail, un rapport de recherche, un article ou tout autre travail publié/circulé n'est pas correctement reconnue. En d'autres termes, le plagiat consiste à faire passer pour siens les mots, les idées ou le travail d'une autre personne, sans en mentionner l'auteur. L'omission d'une telle reconnaissance revient à revendiquer la paternité du travail d'une autre personne. Le plagiat est commis quelle que soit la langue de l'œuvre originale utilisée. Le plagiat peut être délibéré ou accidentel. Les exemples de plagiat comprennent, sans s'y limiter, les cas suivants :
    1) Ne pas mettre de guillemets à une citation du travail d'une autre personne.
    2) Faire semblant de paraphraser alors qu'il s'agit en fait d'une citation
    3) Citer de manière incorrecte ou incomplète
    4) Ne pas citer la source d'un travail cité ou paraphrasé
    5) Copier/reproduire des sections du travail d'une autre personne sans en reconnaître la source
    6) Paraphraser le travail d'une autre personne sans en reconnaître la source
    7) Faire écrire/rédiger une œuvre par une autre personne pour soi-même et la soumettre/publier (avec la permission, avec ou sans compensation) sous son propre nom (”ghostwriting”)
    8) Utiliser le travail non publié d'une autre personne sans attribution et sans autorisation (”vol”)
    9) Présenter comme sien un travail qui contient une forte proportion de textes cités/copiés ou paraphrasés (images, graphiques, etc.), même s'ils sont correctement référencés.

  L'auto-plagiat ou l'autoplagiat, c'est-à-dire la reproduction de (parties d'un) texte précédemment écrit par l'auteur sans citer ce texte, c'est-à-dire faire passer un texte précédemment écrit pour un nouveau, peut être considéré comme une fraude s'il est jugé suffisamment grave.

References

[Goy+22] Vipul Goyal et al. “Storing and Retrieving Secrets on a Blockchain”. In: *Public-Key Cryptography - PKC 2022 - 25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8-11, 2022, Proceedings, Part I*. Ed. by Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe. Vol. 13177. Lecture Notes in Computer Science.

Springer, 2022, pp. 252–282. DOI: 10.1007/978-3-030-97121-2_10. URL: https://doi.org/10.1007/978-3-030-97121-2%5C_10.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. "Constant-Size Commitments to Polynomials and Their Applications". In: *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*. Ed. by Masayuki Abe. Vol. 6477. Lecture Notes in Computer Science. Springer, 2010, pp. 177–194. DOI: 10.1007/978-3-642-17373-8_11. URL: https://doi.org/10.1007/978-3-642-17373-8%5C_11.

[LSP82] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. "The Byzantine Generals Problem". In: *ACM Trans. Program. Lang. Syst.* 4.3 (1982), pp. 382–401. DOI: 10.1145/357172.357176. URL: http://doi.acm.org/10.1145/357172.357176.