Utilisation d'un PRE pour le partage de fichiers chiffrés en mode SaaS

Anass Sbai, C. Drocourt, Gilles Dequen

Compas'2022 : Parallélisme/ Architecture/ Système
MIS/UPJV - Amiens France, 5-8 juillet 2022

Utilisation d'un PRE pour le partage de fichiers chiffrés en mode SaaS

Anass Sbai, Cyril Drocourt, Gilles Dequen

Université de Picardie Jules Vernes,
Laboratoire MIS (UR4290), 33 rue St-Leu, 80000 Amiens, France
{anass.sbai,cyril.drocourt,gilles.dequen}@u-picardie.fr

Résumé
La confidentialité des données est devenue aujourd'hui essentielle mais chiffrer les données directement par les utilisateurs reste difficile. De plus, la problématique du partage de fichiers est souvent inexistante ou mal gérée. En effet, les outils qui le permettent utilisent majoritairement un chiffrement asymétrique peu performant ni très simple d'utilisation. Dans cet article nous proposons de résoudre le problème du partage de fichiers en utilisant les PRE (Proxy Re-Encryption) qui permettent le re-chiffrement des données par un tiers via une clé de re-chiffrement. Nous proposons également d'améliorer les problèmes de performance en modifiant un algorithme PRE existant utilisant des courbes elliptiques. Nous proposons une implémentation de notre solution PRE en tant que service (PREaaS) utilisable à la demande et en complément d'un service de stockage existant.

Mots-clés : Cloud data privacy, Proxy re-encryption, Encrypted file sharing, SECaaS.

1. Introduction

L'utilisation du chiffrement aujourd'hui se limite majoritairement au flux de données par l'utilisation du protocole TLS. Il existe peu de solutions permettant de chiffrer directement les données, et surtout de les partager. Les principaux acteurs du Cloud (Google, Microsoft, ...) proposent des solutions mais qui sont bien souvent insuffisantes, inadaptées ou peu documentées. De manière théorique, la solution la plus simple pour chiffrer des données est d'utiliser le chiffrement symétrique comme AES qui est rapide et facile à mettre en œuvre côté client. Cependant, cette solution ne permet pas le partage d'informations avec un tiers puisqu'il faut partager la clé de chiffrement par un canal sécurisé qu'il faut mettre en place. Dans les protocoles existants, la solution habituellement utilisée est le mécanisme d'encapsulation de clé (KEM). Ce principe consiste à utiliser un chiffrement asymétrique (comme RSA) et à chiffrer la clef de session (clef symétrique) à l'aide de la clef publique du destinataire. Cette méthode est cependant contraignante dans le cas de stockage distant puisqu'elle nécessite de récupérer la clef de chiffrement stockée chiffrée sur le serveur distant, de la déchiffrer avec la clef privée, de la chiffrer à nouveau avec la clef publique du destinataire, puis de l'envoyer. Ce mécanisme est donc coûteux en termes d'échange de données, et ne permet pas de fonctionnement asynchrone. La solution que nous proposons dans cette article se base sur l'utilisation d'un Proxy de Re-chiffrement, qui permet d'éviter au client les multiples opérations en délégant l'opération de re-chiffrement à un tiers, sans exposer la donnée.

2. Travaux existants

Plusieurs travaux visant à protéger la vie privée ont vu le jour comme par exemple CryptDB [1] et ESPRESSO [2]. Ces solutions fournissent des services de chiffrement pour maintenir la confidentialité. Néanmoins, chaque solution a ses limites. Par exemple, CryptDB ne peut pas être utilisé pour les bases de données SQL ou les systèmes de fichiers et la complexité de la gestion des clés augmente avec le nombre d'utilisateurs. ESPRESSO utilise uniquement un chiffrement symétrique et nécessite de faire confiance à un tiers ou à des CSP pour assurer l'opération de chiffrement et la gestion des clés. En ce qui concerne le partage de données, les deux solutions doivent passer par un processus de déchiffrement/chiffrement.
Ce problème de partage de données a été résolu par diverses solutions dans la littérature, à commencer par le chiffrement par diffusion conçu par [3]. Dans ce cas, chaque utilisateur peut accéder aux données indépendamment des autres. Cela nécessite de savoir au moment du chiffrement qui aura le privilège d'accéder aux données. Une autre approche similaire, introduite par [4], est le chiffrement basé sur les attributs (ABE). Inspiré par les travaux de D. Boneh sur les schémas de chiffrement basé sur l'identité (IBE), leur idée était de créer un nouveau type de système IBE [5] pour combiner chiffrement et contrôle d'accès. Dans ce cas, les privilèges d'accès ne sont pas adressés à un ensemble d'utilisateurs mais uniquement à des utilisateurs disposant d'un nombre spécifique d'attributs. Malheureusement, cela ne permet pas un partage sélectif.
Nous choisissons ici d'utiliser le proxy de re-chiffrement (PRE) qui permet de transférer les droits de déchiffrement à des entités spécifiques. Le proxy de re-chiffrement (PRE) est un crypto-système qui permet de transformer des messages chiffrés d'une entité A en messages pouvant être déchiffrés par une entité B. On retrouve alors les trois acteurs principaux :

- Le délégant (Alice) : noté "A" est l'entité propriétaire des données qui délègue les droits de déchiffrement à une autre entité, à savoir le délégué, en créant une clé de re-chiffrement envoyée au proxy de re-chiffrement.
- Le délégué (Bob) : noté "B" est l'entité à laquelle nous avons accordé les droits de déchiffrement des messages chiffrés qui ne lui sont pas destinés à l'origine. Ces messages chiffrés doivent être re-chiffrés par la clé de re-chiffrement qui fait l'objet d'une autorisation d'accès.
- Le proxy : cette entité gère le processus de re-chiffrement des messages chiffrés initialement destinés au déléguant, en messages pouvant être déchiffrés par le délégué. La clé de re-chiffrement utilisée lors du processus de transformation ne doit permettre aucune divulgation d'informations au proxy.

Le premier PRE est bidirectionnel et a été conçu par BBS (Blaze, Bleum et Strauss) [6] sur la base du système de chiffrement asymétrique ElGamal. L'article [7] formalise la conception des proxies de re-chiffrement en classant ces systèmes en deux types : unidirectionnels et bidirectionnels et [8] se concentre davantage sur la création d'un PRE unidirectionnel. Les auteurs de [9] ont proposé le premier schéma PRE sécurisé contre les attaques IND-CCA où ils prouvent la sécurité de leur schéma basé sur la composabilité universelle [10]. Le problème ouvert présenté par Canetti et al. a été abordé dans [11] qui proposent un PRE bidirectionnel basé sur la signature ElGamal et Schnorr, ce qui implique l'utilisation d'oracles aléatoires. Sur la base de [11], [12] a proposé le premier PRE IND-CCA unidirectionnel sécurisé sans appariements et donc s'appuyant uniquement sur la signature El-Gamal et Schnorr. [13] trouve une faille dans la preuve de sécurité de la construction de Chow et propose de la corriger. Leur construction est également sécurisée IND-CCA unidirectionnelle dans le modèle oracle aléatoire et ne repose pas sur des appariements.

3. Contribution

3.1. Partage de fichiers

Le but de notre système est de développer une application CSP (Content Service Provider) en mode SaaS (Software As A Service) dédiée au stockage. Elle devrait permettre à terme de s'interfacer à des solutions propriétaires telles que GDrive, DropBox, ICloud, ... Mais notre étude se limitera au protocole WebDAV.

Notre application devra utiliser l'organisation classique d'un système de fichiers, rester compatible avec ce modèle, et uniquement ajouter des informations supplémentaires pour atteindre notre objectif, sans perturber les opérations normales.

Les objectifs du système sont :

- chiffrer toutes les informations stockées,
- ne pas exposer les données par défaut à une personne autre que le propriétaire des données,
- permettre le partage de données cryptées,

Il est de plus nécessaire d'avoir une solution rapide en termes d'utilisation et permettre le partage d'un répertoire avec un autre utilisateur sans le copier.

Pour résoudre la problématique de rapidité, il est nécessaire d'utiliser un algorithme de chiffrement symétrique. De plus, pour simplifier les accès aux fichiers, il faut les chiffrer indépendamment les uns des autres, et donc générer une clé secrète unique pour chiffrer chacun d'entre eux, et stocker ces clés dans des fichiers indépendants.

Chaque clé de chiffrement secrète associée à chaque fichier sera chiffrée avec la clé publique du propriétaire dans un fichier indépendant. Cette solution permet une granularité plus fine, car elle permet également de ne traiter que les fichiers chiffrés demandés individuellement. Le partage d'un répertoire nécessite l'accès à toutes les clés secrètes associées à tous les fichiers chiffrés du répertoire lorsque l'utilisateur destinataire demande l'accès.

Afin de protéger les accès aux autres données qui ne sont pas partagées (par le destinataire ou par le CSP), l'idée est de ne pas utiliser directement la paire de clés du propriétaire, mais de générer une paire de clés publique/privée dédiée pour chaque répertoire . Ainsi, comme le montre la "Fig. 1", pour un fichier F1 situé dans un répertoire D1, une clé secrète unique K_{F1} sera générée pour chiffrer le fichier. Cette clé sera elle-même chiffrée avec la clé publique du répertoire Pk_{D1}.

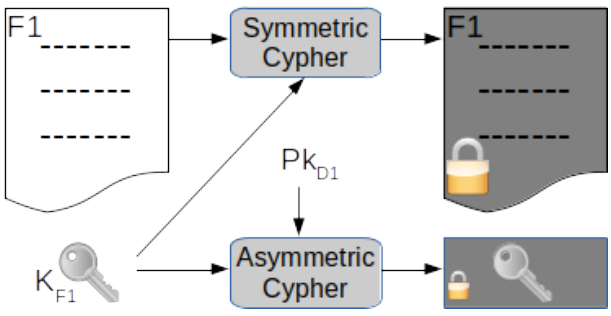

FIGURE 1 – Fichier chiffré et clé de chiffrement.

Ainsi, lorsque le destinataire demande l'accès à un fichier partagé, le CSP transforme la clé secrète de chiffrement associée au fichier en une clé de re-chiffrement pour la rendre accessible

au destinataire utilisant le PRE.
Pour résumer :

- Chaque utilisateur possède sa propre paire de clés générées lors de la création de son compte, pour l'utilisateur A : Pk_a/Sk_a,
- (1) Si l'utilisateur A crée un répertoire D1, une paire de clés correspondantes est générée côté client (Pk_{D1}/Sk_{D1}),
- (2) Si l'utilisateur A crée un fichier F1, une clé secrète K_{F1} est générée, le fichier est chiffré avec cette clé,
- (3) Cette clé secrète est elle-même chiffrée avec la clé publique Pk_{D1} du répertoire et envoyée au serveur pour être placée dans un fichier d'index I_{F1},
- (4) si le propriétaire A veut partager ce répertoire avec l'utilisateur B, il génère côté client une clé de re-chiffrement $Rk_{D1\to B}$ qui autorise la transformation de données chiffrées pour A en données chiffrées pour B, puis envoie cette clé au CSP,
- (5) L'utilisateur B demande l'accès au fichier F1 partagé par A, le CSP lui envoie le fichier chiffré avec la clé K_{F1},
- (6) Le CSP re-chiffre le fichier d'index I_{F1} qui contient la clé secrète K_F1 en utilisant la clé de re-chiffrement $Rk_{D1\to B}$ et l'envoie à l'utilisateur B,
- (7) L'utilisateur B qui possède la clé privée Sk_b peut alors déchiffrer le fichier d'index I_{F1} qui contient la clé secrète, et déchiffrer le fichier F1,

Une clé secrète dérivée du mot de passe de l'utilisateur est générée côté client pour permettre de chiffrer toutes les clés privées et les stocker sur le serveur. Le CSP est donc composé de deux services distants, le PREaaS et le serveur de fichiers. De plus, le composant principal de l'application est situé côté client et s'exécute directement dans le navigateur Internet.

3.2. Proxy Re-Encryption

Un PRE peut être défini comme un tuple ζ : $\{Setup, KeyGen, ReKeyGen, \xi^{asym}, ReEnc, Dec\}$.
Où :

- $Setup(1^k) = params$: prend en entrée un paramètre de sécurité k et génère les paramètres système qui définissent généralement la longueur recommandée des messages et des clés.
- $KeyGen(params) = (Pk, Sk)$: est la fonction qui génère la paire de clés publique/privée Pk and Sk.
- $RekeyGen(Sk_a, Pk_b) = Rk_{a\to b}$: dans le cas du PRE unidirectionnel, il faut indiquer la clé privée de a et la clé publique de b pour générer la clé de re-chiffrement.
- $\xi^{asym}_{Pk_a}(M) = C_a$: est la fonction de chiffrement. Elle prend en entrée le message en clair M et la clé publique Pk .
- $ReEnc(C_a, Rk_{a\to b}) = C_b$: est la fonction de re-chiffrement. Elle prend en entrée un message chiffré et une clé de re-chiffrement.
- $Dec(C, Sk) = M$: est la fonction de déchiffrement. Elle prend en entrée un message chiffré ou re-chiffré et retourne le message en clair correspondant.

Dans certains cas, on peut trouver deux autres fonctions utilisées pour le chiffrement et le déchiffrement, de sorte que le message chiffré créé par cette fonction de chiffrement (aussi appelée chiffrement non transformable) ne puisse pas être re-chiffré et que seul le propriétaire de la clé privée puisse le déchiffrer.
En étudiant l'article [17] qui compare différents PRE, nous avons constaté que celui de Chow est le meilleur. Cependant dans [13] les auteurs trouvent une faille dans la preuve de sécurité de la construction de Chow et proposent de la corriger, c'est donc l'algorithme de Selvi et al. que nous utiliserons car il atteint le niveau de sécurité IND-CCA

Nous avons d'abord implémenté l'algorithme en utilisant simplement un groupe générique d'ordre premier. Cependant le temps nécessaire à l'utilisation des primitives cryptographiques rend le système difficilement utilisable en condition réelle.
Nous avons ensuite étudié les autres possibilités ainsi que les autres algorithmes PRE existants mais sans succès. Nous avons alors eu l'idée de changer de méthode, et d'adapter l'algorithme de [13] pour l'utiliser avec des courbes elliptiques.
Pour plus d'informations sur l'implémentation de l'algorithme de Selvi et al. avec les courbes elliptiques, le lecteur pourra se référer à l'article de Anass et al. [16].

3.3. Implémentation

Afin de pouvoir exécuter les programmes autant côté client que côté serveur, nous avons choisi d'utiliser le langage JavaScript. En effet, côté client ce langage nous assure d'avoir une application utilisable quelque soit le périphérique ou le système. Pour les tests, nous avons utilisé un Intel Core i7 à 2,5 GHz, avec 16 Go de RAM.

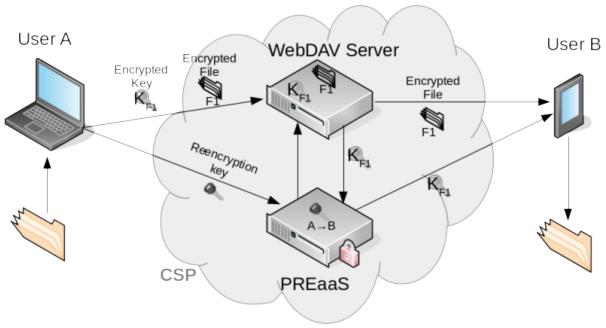

FIGURE 2 – Tasks distribution between different environments.

Le but de l'application est d'utiliser un système de fichiers réseau existant et il doit pouvoir fonctionner comme une couche supplémentaire de chiffrement. De cette façon, tous les systèmes de stockage pourraient être utilisés. Dans un premier temps, nous avons décidé de tester notre application en utilisant le partage de fichiers basé sur le protocole WebDAV. En effet, ce protocole est largement supporté par un grand nombre de logiciels clients, il est léger, simple d'utilisation, et nous permet d'effectuer les opérations dont nous avons besoin :

- Création de répertoires,
- Envoi de fichiers,
- Téléchargement de fichiers,
- Suppression d'éléments distants,

La plus grande partie de l'application s'exécutant en JavaScript sur le client, la majorité des opérations sont effectuées par ce dernier. C'est aussi le client qui télécharge directement depuis le serveur de fichiers sans passer par le PREaaS. Il se contente de re-chiffrer et de transmettre les URL. On peut voir sur la "Fig. 2" le principe d'interaction entre le PREaaS et le serveur de fichiers WebDAV.
Pour l'implémentation de l'algorithme de Selvi, deux instances ont été testées. D'abord, nous utilisons un groupe générique avec une longueur d'ordre premier de 3072 bits, et ensuite la norme NIST ECC (Elliptic Curves Cryptography) p-256 [18] grâce à SJCL (Standford Javascript Crypto Library) [19].

TABLE 1 – Computational time of Selvi's algorithm in (ms)

Function	$Selvi(ms)$	$Selvi(bits)$	$ECC(ms)$	$ECC(bits)$
KeyGen	515	512 & 6144	68	512 & 512
ReKeyGen	502	3584	48	768
Encrypt	1000	6656	95	1024
ReEncrypt	967	6656	89	1024
Decrypt	979	–	78	–

Le tableau 1. montre les ressources utilisées par les différentes fonctions de l'algorithme de Selvi et al. pour les deux implémentations. Nous avons mesuré le temps d'exécution de chacune des fonctions PRE (Keygen, ReKeyGen, ...) sur une série de plusieurs exécutions pour lesquelles nous avons calculé une moyenne.
Concernant la taille correspondant à chaque fonction, elle fait référence aux éléments suivants :

- KeyGen : fait référence à la taille des clés privées $2|\mathbb{Z}_q|$ et des clés publiques $2|\mathbb{G}|$
- ReKeyGen : fait référence à la taille des clés de re-chiffrement $|\mathbb{G}| + |\mathbb{Z}_q| + l_1 + l_2$
- Encrypt : fait référence à la taille des éléments chiffrés originaux $2|\mathbb{G}| + |\mathbb{Z}_q + l_1 + l_2$
- ReEncrypt : fait référence à la taille des éléments re-chiffrés $2|\mathbb{G}| + 2l_1 + 2l_2$

Nous pouvons constater qu'aucune opération n'est contraignante en termes de consommation de temps si nous utilisons ECC. En effet, nous pouvons remarquer que la taille a considérablement diminué et que le temps d'exécution a été réduit d'un rapport d'environ 10 lors de l'utilisation d'une courbe elliptique. Nous pouvons également voir que les fonctions de chiffrement et de re-chiffrement consomment davantage par rapport à la génération de clés et au déchiffrement.

4. Conclusion

Dans cet article nous vous présentons une approche originale pour résoudre le problème du partage de fichiers cryptés. Le fondement de ce travail reste le PRE, cependant, à ce jour, aucune solution ni implémentation pour une utilisation en temps réel n'a été proposée dans la littérature. Nous abordons cela essentiellement sous trois aspects :

- La modification de l'algorithme de Selvi et al. en utilisant les courbes elliptiques et obtenir un gain de vitesse multiplié par 10,
- L'utilisation de Javascript comme langage d'implémentation, permettant une utilisation dans tout terminal mobile et tout système avec un simple navigateur Internet,
- L'utilisation d'un PRE as a Service, permettant de déporter les procédures de re-chiffrement sur un serveur distant, et d'effectuer ces opérations en mode asynchrone à la demande.

Notre application CSP en mode SaaS utilise des protocoles standards pour stocker des fichiers (WebDAV) et pour communiquer (HTTP). Afin de n'exposer aucune donnée, celles-ci sont chiffrées par le navigateur avant l'envoi au serveur. Chaque fichier est chiffré avec une clé secrète indépendante, elle-même chiffrée avec une clé publique propre à chaque répertoire. L'algorithme de chiffrement permet de déléguer les droits de déchiffrement à un autre utilisateur, en générant une clé de re-chiffrement qui est transmise au serveur. Notre application de re-chiffrement fonctionne comme un service que nous appelons PREaaS.

Bibliographie

1. R. A. Popa, N. Zeldovich, and H. Balakrishnan, "Cryptdb : A practical encrypted relational dbms," 2011.
2. S. Kang, B. Veeravalli, and K. M. M. Aung, "Espresso : An encryption as a service for cloud storage systems," in *IFIP International Conference on Autonomous Infrastructure, Management and Security*. Springer, 2014, pp. 15–28.
3. A. Fiat and M. Naor, "Broadcast encryption," in *Annual International Cryptology Conference*. Springer, 1993, pp. 480–491.
4. A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2005, pp. 457–473.
5. D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Annual international cryptology conference*. Springer, 2001, pp. 213–229.
6. M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1998, pp. 127–144.
7. A.-A. Ivan and Y. Dodis, "Proxy cryptography revisited." in *NDSS*, 2003.
8. G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, no. 1, pp. 1–30, 2006.
9. R. Canetti and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," in *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 2007, pp. 185–194.
10. R. Canetti, "Universally composable security : A new paradigm for cryptographic protocols," in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*. IEEE, 2001, pp. 136–145.
11. R. H. Deng, J. Weng, S. Liu, and K. Chen, "Chosen-ciphertext secure proxy re-encryption without pairings," in *International Conference on Cryptology and Network Security*. Springer, 2008, pp. 1–17.
12. S. S. Chow, J. Weng, Y. Yang, and R. H. Deng, "Efficient unidirectional proxy re-encryption," in *International Conference on Cryptology in Africa*. Springer, 2010, pp. 316–332.
13. S. S. D. Selvi, A. Paul, and C. Pandurangan, "A provably-secure unidirectional proxy re-encryption scheme without pairing in the random oracle model," in *International Conference on Cryptology and Network Security*. Springer, 2017, pp. 459–469.
14. A. Jivanyan, R. Yeghiazaryan, A. Darbinyan, and A. Manukyan, "Secure collaboration in public cloud storages," in *CYTED-RITOS International Workshop on Groupware*. Springer, 2015, pp. 190–197.
15. M. Egorov, D. Nuñez, and M. Wilkison, "Nucypher : A proxy re-encryption network to empower privacy in decentralized systems," 2018.
16. A. Sbai, C. Drocourt, and G. Dequen, "Pre as a service within smart grid cities," in *16th International Conference on Security and Cryptography*, 2019.
17. Z. Qin, H. Xiong, S. Wu, and J. Batamuliza, "A survey of proxy re-encryption for secure data sharing in cloud computing," *IEEE Transactions on Services Computing*, 2016.
18. S. Gueron and V. Krasnov, "Fast prime field elliptic-curve cryptography with 256-bit primes," *Journal of Cryptographic Engineering*, vol. 5, no. 2, pp. 141–151, 2015.
19. E. Stark, M. Hamburg, and D. Boneh, "Stanford javascript crypto library," 2013.