Algorithmes Guide

Version 4.0.5-246-g427649bc

Lybero developement team

2019-10-31

Version : 4.0.5-246-g427649bc

Summary

1 Introduction

1.1 Brevet

L'architecture sous-tendant les systèmes de Lybero.net a été breveté en France, en Europe et aux Etats-Unis par Inria. Le brevet est : "Device and method of administration of sequestered digital server". Le brevet français a été délivré sous le numéro FR3033466B1 le 17 février 2017. Le titre du brevet américain est "Device and method for administering a digital escrow server" publié le 5 octobre 2018.

2 Rôles

2.1 Description

Les différents rôles sont les suivants :

- L'administrateur système : il est en charge de l'administration générale de la plateforme. Il peut supprimer des utilisateurs. Il peut créer des groupes à quorum. Il ne peut accéder à aucun partage de fichiers (il n'a pas de clé de chiffrement).
- Les administrateurs de secrets : réunis en groupe à quorum, ils peuvent participer au partage des informations vers un tiers si l'un des groupes à quorum auquel ils appartiennent a été mis en partage d'un dépôt de fichiers.
- Les utilisateurs authentifiés : ils peuvent créer des urls permettant à des utilisateurs non authentifiés de leur envoyé des informations chiffrées. Ils peuvent créer, modifier, partager, participer à des dépôts de fichiers.
- Les utilisateurs non authentifiés : ils peuvent envoyer des informations de manière chiffrée à des utilisateurs authentifiés mais ne peuvent pas y accéder.

La répartition des rôles que nous avons présentée peut être affinée de manière importante pour chaque organisation. En effet, pour chaque utilisateur des droits sont associés, droits qui peuvent être globaux ou limités à un groupe d'objets (de dépôts de fichiers, d'utilisateurs, . . .).

3 Propriétés et contraintes de CryptnDrive

3.1 Dans le navigateur

Toutes les opérations cryptographiques sont faites dans les navigateurs web. Les clés cryptographiques des utilisateurs sont stockées sur le serveur.

Toutes les valeurs aléatoires (clés AES pour chaque dépôt de fichiers, clés asymétriques) sont générées dans le navigateur web de l'utilisateur. Ces opérations sont difficiles à attaquer car distribuées. Si un serveur est hacké, cela ne conduit à la divulgation d'aucun secret.

Dans CryptnDrive la génération des clés est toujours faite au plus proche de l'utilisateur, sur son navigateur à l'inscription. Ensuite, les clés sont stockées de manière centralisée ou non (selon les modes que l'on choisi) avec la clé privée chiffrée en dérivant la clé du mot de passe ou de la phrase de passe choisie. D'un point de vue programmatique, généralement, les bibliothèques de chiffrement séparent strictement la gestion des clés et le chiffrement, en utilisant des outils différents. Dans CryptnDrive, toutes les opérations cryptographiques, y compris la génération des clés, sont toujours faites sur le navigateur client.

4 Opérations simples de chiffrement et déchiffrement

4.1 Création d'un compte utilisateur

Lorsqu'un utilisateur crée un compte, une bi-clef (une clé privée et la clé publique associée) ElGamal est créée dans le navigateur web de la personne. La phrase de passe de la personne est utilisée comme clé de chiffrement symétrique pour chiffrer en AES256 la clé privée qui est ensuite envoyée chiffrée au serveur.

Lorsqu'un utilisateur se connecte au système, une authentification est faite (autonome, via un service d'annuaire ldap, ou Active Folder ou Oauth2). Une double authentification avec une clé Fido U2F est possible. Si l'authentification est faite avec succès, alors la clé privée est récupérée chiffrée du serveur et elle est déchiffrée dans le navigateur, lorsque cela est nécessaire. La règle de «Perfect Forward Secrecy » est respectée sur le serveur.

4.2 Création et chiffrement d'un dépôt

Lors de la création d'un dépôt de fichiers par un utilisateur, le navigateur web génère une clé AES256 aléatoirement *m*.

La clé AES256 *m* est ensuite chiffrée (ElGamal 2048, 4096 ou 8192 selon la configuration) avec la clé publique de l'utilisateur créant le dépôt de fichiers et

stockée sur le serveur. Un sel aléatoire *r* est utilisé.

$$m \rightarrow (\alpha, \beta) = (g^r mod(p), (Y^r . m) mod(p))$$

Le navigateur envoi au serveur les données chiffrées par l'algorithme ElGamal image ainsi que des méta-data d'identification non chiffrées (un numéro d'identifiant du dépôt de fichiers, nom du dépôt, la description, l'identifiant numérique et le login du propriétaire, les identifiants et login de chaque personne partageant, l'identifiant et le nom des groupes à quorum en partage, la date de création, la date de dernière modification et l'identifiant de la dernière personne ayant modifié).

Les fichiers et texte sont ensuite mis à disposition de la page. Lors de la sauvegarde du dépôt de fichiers, les fichiers sont chargés par morceau de 2Mo, chiffrées en AES256 avec la clé *m* dans le navigateur et envoyés au serveur pour stockage. Cette opération est multithreadée plusieurs morceaux de fichiers étant lus, chiffrés et envoyés en parallèle.

Le serveur répond à chaque étape en cas de succès ou d'échec de stockage de toute ou partie des informations.

4.3 Déchiffrement d'un dépôt de fichiers et d'un fichier

Lorsqu'un utilisateur accède à un dépôt de fichiers, la clé *m* AES du dépôt de fichiers chiffré pour l'utilisateur est récupéré, puis la clé *m* est déchiffrée :

$$\frac{\beta}{\alpha^s \bmod(p)} \rightarrow m$$

On peut alors récupérer la liste des fichiers qui est chiffrée AES256 avec la clé *m*. Chaque fichier est lié à un identifiant unique.

Quand on charge un fichier, il est tout d'abord chargé en mémoire, déchiffré AES256 en mémoire avec la clé *m* , puis sauvegardé via le navigateur. Le fichier est stocké par morceau de 2 Mo. Ce sont ces morceaux qui sont déchiffrés puis concaténés pour obtenir le fichier final déchiffré.

Une trace de logs est créée à chaque déchiffrement d'un dépôt de fichiers par un utilisateur, puis lors du chargement ou de la visualisation d'un fichier.

4.4 Création d'un dépôt "anonyme"

Il n'est pas nécessaire d'avoir un compte pour chiffrer pour quelqu'un. Nous utilisons cette propriété pour les dépôts dits "anonymes". Dans ce cas, l'utilisateur disposant d'un compte génère un lien. Ce lien est transmis (mail, tchat, slack,

. . .) à l'expéditeur. Il clique sur le lien est arrive sur une page web lui permettant de déposer des fichiers et d'écrire un message. Lorsque les fichiers sont partagés, une clé AES 256 est générée, les fichiers et le texte est chiffré avec cette clé et transmis au serveur. Puis la clé AES256 est chiffré avec la clé publique de la personne qui a fourni le lien, et cette clé chiffrée est transmise au serveur. Grâce à ce mécanisme, l'expéditeur n'a pas besoin de compte, ni de mot de passe pour chiffrer pour un destinataire.

4.5 Chiffrement pour un tiers n'ayant pas de clé

En théorie, il n'est pas possible de chiffrer des informations si l'on a pas la clé publique de l'interlocuteur. Donc il n'est pas possible de chiffrer pour un interlocuteur non inscrit. Nous utilisons un mécanisme particulier pour faire cela :

- Lorsqu'un dépôt de fichiers est créé, sa clé symétrique est chiffrée avec la clé publique du créateur du dépôt,
- lors du partage avec un tiers non connu, un lien spécifique d'inscription lui est envoyé.
- Lorsque l'invité clique dessus, il peut s'inscrire (choisir un mot de passe en fait). Lors du choix du mot de passe, une bi-clé ElGamal est créée pour l'utilisateur invité.
- La clé privée est stockée sur le serveur en utilisant le mot de passe comme clé de chiffrement AES 256 (avec un algorithme de dérivation de clé pkbf2). Le serveur fourni à l'utilisateur invité la clé AES 256 **chiffrée avec la clé publique de l'utilisateur invitant**.
- Il surchiffre avec sa clé publique ce secret et le transmet à l'utilisateur invitant. Après vérification de l'identité de l'invité (lien entre l'identité virtuelle et l'identité réelle), ce dernier déchiffre cet ensemble avec sa clé privée puis stocke la valeur obtenue. Cette valeur est la clé AES 256 chiffrée pour l'utilisateur invité.

Ceci est possible car en ElGamal, lorsque l'on procéde au chiffrement successif d'une valeur avec des clés publiques différentes, on peut déchiffrer successivement dans n'importe quel ordre. Il n'est pas nécessaire de "dépiler" les déchiffrements dans l'ordre inverse des chiffrements. Dans notre cas, l'invitant A chiffre, l'invité B chiffre le résultat, l'invitant A déchiffre et finalement l'invité B déchiffre, le résultat final est déchiffré.

5 Algorithmes associé aux groupes à quorum

5.1 Remarque préliminaire

Les calculs suivant sont inspirés en grande partie de l'article :

*Distributed ElGamal à la Pedersen - Application to Helios*par Véronique Cortier (CNRS/LORIA, France), David Galindo (CNRS/LORIA, France), Stéphane Glondu (INRIA Nancy Grand-Est, France) et Malika Izabachène (CNRS/LORIA, France) publié dans WPES 2013 - Proceedings of the 12th ACM workshop on privacy in the electronic society - 2013, Nov 2013, Berlin, Germany. ACM, pp.131-142, 2013.

5.2 Initialisation d'un groupe à quorum

5.2.1 Étape 0: création du quorum

Un droit de création de groupe à quorum spécifique existe. Au minimum l'administrateur du site possède ce droit.

Avant de pouvoir exploiter le groupe à quorum, les paramètres de configuration suivants doivent être choisis :

- le nom du groupe à quorum,
- la liste des administrateurs de secrets,
- le quorum associé (le nombre d'administrateurs de secrets nécessaire pour que le recouvrement soit autorisé avec ce groupe à quorum).

Tous ces paramètres sont envoyés au serveur.

5.2.2 Étape 1 : generation des bi-clés pour chaque administrateur de secrets

Si les différents administrateurs n'ont pas encore de compte et donc de bi-clés ElGamal, ils doivent créer un compte afin d'avoir une bi-clé ElGamal.

Une fois que tous les administrateurs de secrets du groupe à quorum ont une bi-clé, l'initialisation du groupe à quorum peut continuer.

5.2.3 Étape 2 : échange des paramètres cryptographiques

Lors de la connexion de l'administrateur de secrets n°*rank*, il récupère les informations classiques (numéro d'identification, nom de login, bi-clé) ainsi que son numéro de rang *rank* dans le groupe à quorum.

Chaque administrateur génère ensuite un polynôme de degré quorum diminué de 1 dans un espace entier discret :

$$f_{rank}(x) = \sum_{j=0}^{quorum-1} a_{rank,j}\, mod(q).x^j$$

Une partie de la clé publique du groupe à quorum et aussi calculé par chaque membre du groupe :

$$A_{0(rank)} = g^{(a_{rank,0})} mod(p) = g^{f_{rank}(0)} mod(p)$$

(*p* est fixe pour un groupe à quorum).

Chaque administrateur *rank* de secrets calcule pour chaque autre administrateur *i* la valeur $S_{rank,i} = f_{rank}(i)$. Il stocke ensuite l'information suivante : $A_{0(rank)}$

Et

La liste des valeurs $S_{rank,i}$ chiffrée avec la clé publique de l'administrateur de secrets correspondant (de rang *i*), notée $enc_i(S_{rank,i})$.

5.2.4 Étape 3 : génération de la clé publique du groupe à quorum

Une fois que toutes les valeurs image et image sont reçues par le serveur, les valeurs sont échangées avec les différents administrateurs de secrets. Ces derniers calculent la clé publique du groupe à quorum en déchiffrant avec leur clé privée lesimage :

$$Y = (\prod_{i=1}^{max} A_{0(i)}) mod(p)$$

Le serveur fourni cette clé publique du groupe à quorum aussitôt que demandé. Lorsqu'un chiffrement est nécessaire pour le groupe à quorum, cette clé publique est utilisée telle que décrit précédemment.

5.3 Recouvrement d'un secret

Le recouvrement d'un secret est fait grâce à un quorum d'administrateurs de secrets opérant indépendamment les uns des autres, mais dans une même fenêtre temporelle configurable (par exemple 1 heure).

5.3.1 Étape 0 : déchiffrement partiel d'un administrateur de secret

Le navigateur de l'administrateur de secret de rang *rank* récupère toutes les informations concernant le groupe à quorum c'est-à-dire toutes les clés publiques des administrateurs de secrets et tous les $enc_{rank}(S_{i,rank})$ pour tous les administrateurs de secret *i*.

Grâce à sa clé privée, il peut déchiffrer et obtenir tous les $S_{i,rank}$ fourni par tous les administrateurs *i*. Il peut alors calculer $\sum_{j=1}^{max} S_{j,rank}\, mod(q)$. *S* est le secret pour l'administrateur de secret courant *rank*.

5.3.2 Étape 1 : récupération des informations chiffrées

L'administrateur de secrets récupère toutes les informations du secret actuellement déchiffré. Ceci inclus α .

5.3.3 Étape 2 : sur-chiffrement

Le navigateur de l'administrateur de rang *rank* calcule $C_{rank} = (\alpha_{rank})^S mod(p)$.

Il envoie C_{rank} au serveur, chiffré pour chaque administrateur en utilisant la clé publique de l'administrateur concerné. Le serveur stocke tous les $enc_i\ (C_{rank}\ )$. Chaque administrateur participant au quorum attend ces valeurs des autres administrateurs. On note l'ensemble des administrateurs participant au quorum image et l'ensemble des indices correspondant : $\tau = \{i_1, \ldots, i_{quorum}\}$.

Une fois le quorum atteint, le serveur renvoie à chaque administrateur i qui a participé au quorum les valeurs $enc_i\ (C_{rank}\ )$ (avec *rank* $\in \tau$), β et les données chiffrées avec la clé de session *m*.

5.3.4 Étape 3 : déchiffrement local

Le navigateur de l'administrateur de rang *j* calcule la clé de session

$$m = \beta.(\prod_{j\in\tau}(C_j)^{\lambda_j^\tau}\, mod(p))^{-1}$$

$\lambda_j{}^{\tau}$ est le coefficient *j* du polynôme de Lagrange: $\lambda_j{}^{\tau} = \prod_{k \in \tau \setminus \{j\}} \frac{k}{k-j} mod(q)$
m est la clé de session nécessaire pour déchiffrer les données chiffrées. Une fois déchiffrée les données peuvent être mises à disposition de l'administrateur de secret.

Dans la description précédente, nous avons considéré que la personne demandant un recouvrement est un des administrateurs de secrets. Cependant, ce peut être une personne tierce. Dans ce cas, image est chiffré avec la clé publique de la personne faisant la demande, et le demandeur fait le déchiffrement final (étape 3 plus haut). Les administrateurs de secrets ne peuvent pas eux déchiffrer les données.