# SÉQUESTRE DE CLÉS :
## MISE EN PRATIQUE DU « SHAMIR'S SECRET-SHARING SCHEME »

Philippe Teuwen - phil@teuwen.org

***mots-clés : ?????????????????????***

*Ou comment livrer vos secrets à la postérité tout en vous prémunissant des faux frères grâce à un astucieux partage de clé secrète de Shamir exécuté depuis un live CD et accompagné d'un chouia de procédures.*

## 1 Introduction

Nos pauvres petits neurones conservent tant bien que mal toujours plus de sésames. Mais nous ne sommes pas éternels et il serait dommage qu'avec nous disparaissent les accès à un héritage numérique, par exemple le mot de passe d'un disque chiffré ou d'un hypothétique site web commençant par « F » ou « G » qui contiendrait nos courriels, notre agenda, nos photos, nos fichiers, voire des musiques, films et jeux chèrement acquis.

Au sein d'une PME, la perte subite d'un collaborateur pourrait même devenir dommageable à l'entreprise peu prévoyante qui verrait alors la continuité de ses activités (*business continuity*) mise à mal. Nous continuerons à utiliser l'exemple de la PME par la suite, mais les principes de base restent transposables à un cercle familial ou d'amis.

Une réponse à cette problématique est ce qu'on appelle le séquestre de clé (*key escrow*), qui permet à un tiers « sous certaines conditions » d'accéder à ces clés. Mais quel tiers, sous quelles conditions, et comment lui accorder notre confiance morale mais aussi technique ? L'autorité de séquestre doit en effet être en mesure de garantir en toute sécurité la confidentialité des clés sous séquestre.

## 2 Clé de séquestre asymétrique

La première pièce du puzzle sera assez logiquement la création d'une paire de clés publique et privée propre à l'autorité de séquestre (entité qui reste à définir pour l'instant). Ainsi, toute personne désireuse de confier ses sésames n'aura qu'à les chiffrer elle-même avec cette clé publique et à les mettre à disposition voire à les rendre publiquement accessibles vu que seule l'autorité possède la clé privée correspondante.

La clé privée de séquestre sera protégée par une *passphrase* sur laquelle repose donc toute la sécurité du système. Mais qui sera le détenteur de cet ultime secret ?

## 3 Règle des deux hommes

Cette règle (*two-man rule*) est utilisée dans des domaines sensibles tels la commande d'envoi de missiles nucléaires afin d'éviter tout dérapage accidentel ou malicieux. En cryptographie, les Américains emploient la locution « two-person integrity » (TPI) quand il s'agit d'empêcher qu'une personne seule ait accès aux clés cryptographiques assurant la sécurité des communications (*COMSEC*).

Voici donc un concept intéressant qui nous aiderait à résoudre ces problèmes de confiance et de sécurité envers l'autorité de séquestre. En exigeant que deux individus collaborent pour révéler les données sous séquestre, nous nous mettons à l'abri d'un acte malveillant isolé, voire d'une faille de sécurité isolée.

Divisons donc la passphrase de la clé de séquestre et remettons les morceaux à un groupe de personnes de confiance constituant ce que nous appellerons dorénavant la *business continuity team* (BCT).

Comment morceler cette passphrase ? Répartir simplement $N$ morceaux entre les $N$ membres de la BCT les oblige à se réunir tous ensemble pour pouvoir



miscHS6_protectionLogiciels.indd 56 21/09/2012 11:22:24

utiliser la clé privée de séquestre, mais si l'un d'eux est indisponible ou est justement le collaborateur disparu dont on souhaite récupérer les sésames, comment faire ?

## 4 Shamir's Secret-Sharing Scheme

Nous allons donc être amenés à répartir la passphrase de la clé privée de séquestre entre les personnes de confiance qui constituent la BCT, mais de sorte qu'il suffisse de réunir une fraction d'entre elles pour accéder à nouveau à la clé privée.

A priori, le problème n'est pas aisé. Pourtant, il a été résolu élégamment par Adi Shamir dès 1979 **[SHAMIR]** et la solution peut s'illustrer par une simple parabole.

Non, non, il ne s'agit pas d'allégorie, mais bien de mathématiques et de polynômes.

Deux points sont suffisants pour définir une ligne, trois pour définir une parabole, quatre pour une cubique, etc. Si maintenant nous voulons partager un secret, disons la valeur 1234, entre six individus et que trois d'entre eux soient nécessaires pour retrouver le secret, nous choisirons au hasard une parabole parmi celles passant par le point (0, 1234) et nous donnerons les coordonnées de six de ses points à ces six individus (cf. Figure 1).

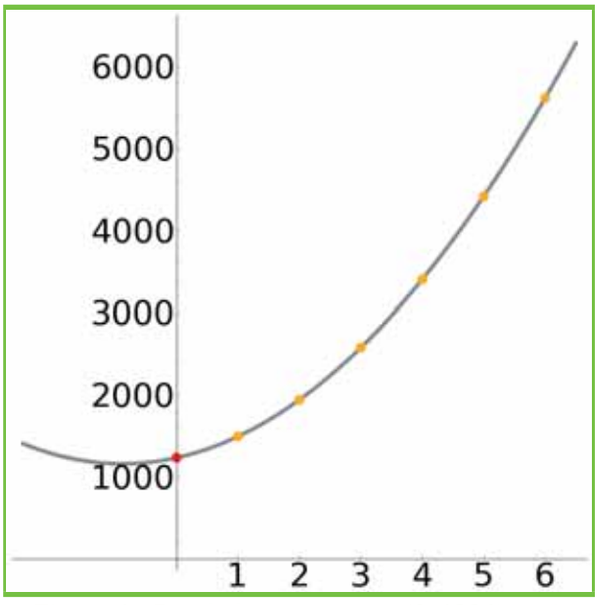


*Figure 1 : Parabole passant par (0, 1234) et six de ses points.*

Si seulement deux d'entre eux, les n° 2 et 4, venaient à partager leurs coordonnées, ils ne pourraient retrouver la parabole originale et donc la valeur du point secret en $x=0$ (cf. Figure 2).

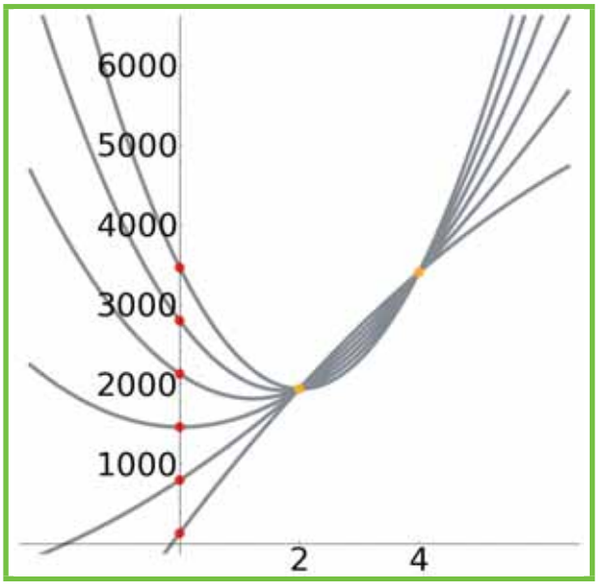


*Figure 2 : Paraboles passant par les points des participants n°2 et 4.*

Il faut donc bien qu'un troisième individu accepte de partager ses coordonnées pour parvenir à définir une et une seule parabole et révéler la valeur secrète 1234 (cf. Figure 3).

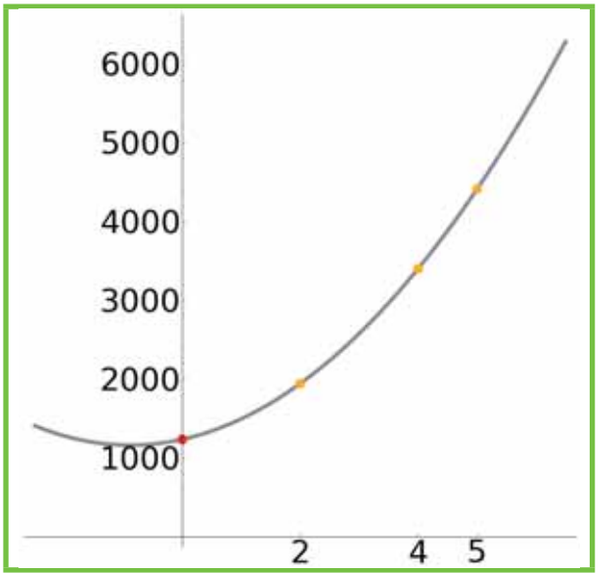


*Figure 3 : Une seule parabole peut passer par les points des participants n°2, 4 et 5.*

Les amateurs retrouveront les équations correspondant à cet exemple sur Wikipédia **[WIKIPEDIA]**.

Cet exemple illustré donne une idée du principe de ce protocole qui utilise en réalité des polynômes dans un corps fini, non représentable dans le plan.

Une implémentation libre existe **[SSSS]** et est disponible dans les principales distributions, notamment dans Debian/Ubuntu par un simple `apt-get install ssss`.



miscHS6_protectionLogiciels.indd 57 21/09/2012 11:22:24

Deux commandes sont alors disponibles : **ssss-split** et **ssss-combine**.

Pour partager le secret « 1234 » de l'exemple précédent en six morceaux dont trois sont nécessaires à sa reconstruction, on exécutera **ssss-split** avec les options **-n 6** (six morceaux) et **-t 3** (dont trois nécessaires) :

```
$ ssss-split -t 3 -n 6
Generating shares using a (3,6) scheme with dynamic security
level.
Enter the secret, at most 128 ASCII characters: 1234
1-44eb879e
2-36abde94
3-43726a38
4-8eabfa4b
5-fb724ef5
6-893217db
```

Et pour le reconstituer, on utilisera **ssss-combine** avec les morceaux numéros 2, 4 et 5 :

```
$ ssss-combine -t 3
Enter 3 shares separated by newlines:
Share [1/3]: 2-36abde94
Share [2/3]: 4-8eabfa4b
Share [3/3]: 5-fb724ef5
Resulting secret: 1234
```

Mais pour mettre le tout en musique, il nous faut encore quelques procédures et moyens techniques garantissant qu'à aucun moment quelqu'un n'ait un accès intégral à la clé privée de séquestre ou à sa passphrase.

# 5 Assemblage

Afin d'éviter toute fuite, l'ensemble des opérations sensibles s'effectue sur un ordinateur sans unité de stockage ni réseau qui démarre sur un live CD contenant les quelques scripts nécessaires. Toutes les communications digitales se feront via des mémoires flash USB labellisées USBPUBLIC, USBSECURE1 et USBSECURE2. Ces dernières sont dédoublées pour offrir une redondance matérielle et ne pourront jamais être utilisées que via le live CD.

## 5.1 Génération de la clé de séquestre

Les clés publiques des membres de la BCT sont copiées sur USBPUBLIC après vérification de leurs signatures ou de leur empreinte. Ensuite, le live CD est démarré, les trois mémoires flash sont insérées et les clés publiques sont copiées de USBPUBLIC vers les deux mémoires USBSECURE.

Une passphrase aléatoire est générée et partagée en quatre secrets de Shamir (**ssss-split -t 2 -n 4**), un par membre de la BCT. Chaque membre chiffre son secret partagé de Shamir avec un mot de passe de son choix. Les secrets chiffrés sont alors sauvés sur les mémoires USBSECURE tandis que les mots de passe sont chiffrés à l'aide des clés publiques des membres correspondants et sauvés sur USBPUBLIC et USBSECURE.

La paire de clés de séquestre peut enfin être créée et la clé privée protégée par la passphrase aléatoire. La paire de clés est sauvée sur les USBSECURE et la clé publique sur USBPUBLIC.

L'empreinte (*fingerprint*) de la clé publique de séquestre est recopiée manuellement sur un papier et un certificat de révocation est généré et sauvé sur les trois mémoires.

Toutes ces étapes sont bien entendu semi-automatisées par un script idoine, les interactions avec les membres étant limitées à la saisie des mots de passe et l'affichage de l'empreinte (cf. Figure 4).

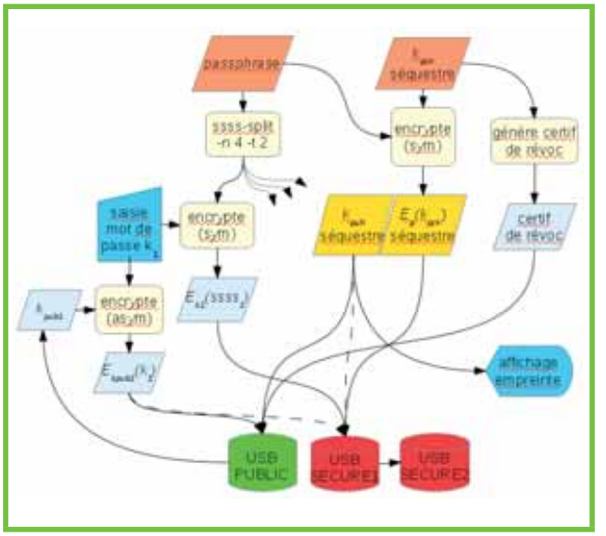


*Figure 4 : Les étapes initiales de la création de la clé de séquestre*

Le live CD est arrêté et USBPUBLIC est insérée dans un autre ordinateur. La clé publique de séquestre en est récupérée, son empreinte extraite, imprimée quatre fois et comparée à la transcription manuelle. Les copies imprimées sont remises à chaque membre de la BCT pour qu'il puisse signer en toute confiance cette nouvelle clé. Les mots de passe chiffrés sont expédiés aux membres correspondants. Le certificat de révocation est imprimé deux fois après quoi USBPUBLIC est effacée de manière sécurisée.

Deux copies du live CD, les mémoires flash USBSECURE1 et USBSECURE2 et les certificats de révocation sont répartis dans deux coffres dont les accès doivent être réglementés par une procédure adéquate.



miscHS6_protectionLogiciels.indd 58 21/09/2012 11:22:24

La clé publique de séquestre est signée par l'ensemble des membres de la BCT et mise à disposition de toute personne qui en aurait besoin.

## 5.2 Mise sous séquestre

L'individu désireux de placer sous séquestre une information récupère la clé publique de séquestre.

Il la valide en vérifiant que la clé est bien signée par les membres de la BCT et qu'il existe une chaîne de confiance entre ces membres et sa propre clé.

```
$ gpg --check-sigs --list-options show-uid-validity escrow@
ma_petite_entreprise.com
```

La sortie doit indiquer **[marginal]** (voire **[full]** s'il est lui-même membre de la BCT) devant l'uid.

S'il ne possède pas de clé OpenPGP ou que l'entreprise n'est pas adepte du *Web-of-Trust*, il peut toujours vérifier la clé de séquestre par son empreinte.

```
$ gpg --fingerprint escrow@ma_petite_entreprise.com
```

Il peut alors chiffrer en toute confiance l'information souhaitée et la placer là où la BCT l'aura convenu.

Les détails de la mise sous séquestre de chaque type d'information sensible (clé OpenPGP, certificat Outlook, mot de passe Truecrypt, *slot* d'une partition LUKS, etc.) doivent être documentés et expliqués aux utilisateurs ainsi que les conventions de nommage et de stockage choisies. Si certaines informations doivent d'abord être déchiffrées avant séquestre, il faut être très prudent : ne pas laisser de trace sur les disques, privilégier les *pipes* en ligne de commandes et s'il est inévitable de créer un fichier temporaire, l'effacer de manière sécurisée (avec wipe, shred, srm ou encore Heidi Eraser sous Windows).

Nous approfondirons le cas des clés OpenPGP, mais voyons d'abord comment se déroule l'opération inverse.

## 5.3 Récupération de données mises sous séquestre

La donnée à récupérer, donc chiffrée avec la clé de séquestre, sera placée sur la mémoire flash USBPUBLIC ainsi qu'éventuellement la clé publique du récipiendaire de cette donnée si on veut éviter que la donnée ne circule en clair.

Le live CD est démarré sur un ordinateur dépourvu d'unité de stockage et de réseau. La clé privée de séquestre et les secrets de Shamir chiffrés sont lus depuis une des deux mémoires USBSECURE. Deux membres de la BCT introduisent tour à tour le mot de passe qu'ils avaient choisi lors de la génération de la clé de séquestre. S'ils l'ont oublié, ils peuvent toujours se rafraîchir la mémoire en consultant la copie chiffrée qu'ils avaient reçue avant de venir participer à la procédure de récupération. La passphrase de la clé de séquestre est reconstituée et la donnée à récupérer est déchiffrée. Si nécessaire, elle sera re-chiffrée avec la clé publique du récipiendaire avant d'être copiée sur USBPUBLIC.

## 5.4 Particularités de la mise sous séquestre d'une clé OpenPGP

Si vous êtes amené à utiliser des clés OpenPGP dans un contexte professionnel avec une mise sous séquestre, il est préférable de ne pas utiliser votre clé personnelle mais de créer une nouvelle paire réservée à cet effet, voire deux paires : une pour le chiffrement et une pour la signature. Seule la clé de chiffrement est mise sous séquestre, et ce, sans passphrase. Ainsi, en cas de nécessité, vos communications chiffrées seront déchiffrées sans que personne n'apprenne votre passphrase ni ne soit en mesure de signer des messages en votre nom.

La mise sous séquestre se fera de la sorte :

```
$ gpg --export-secret-subkey --armor --export-options \
export-reset-subkey-passwd export-minimal alice@ma_petite_entreprise.com | \
gpg --encrypt --armor --recipient escrow@ma_petite_entreprise.com
--trust-model always > \
gnupg_private_yourname_pgpID_date.gpg
```

Si la clé devait être extraite du séquestre et rechiffrée avec la clé publique de la personne habilitée, celle-ci exécuterait :

```
$ gpg --decrypt gnupg_private.gpg |gpg --import --allow-secret-key-import
```

N'importe quel message se déchiffre alors le plus simplement du monde, sans passphrase :

```
$ gpg --decrypt anyfile.gpg
```

À noter que si vous êtes obligé par exemple par un tribunal de fournir le moyen de déchiffrer un message OpenPGP précis, ne donnez jamais votre clé secrète, mais bien la clé de session symétrique de ce message :

```
$ gpg --show-session-key --output /dev/null message.gpg
You need a passphrase to unlock the secret key for
user: "Philippe Teuwen (Doegox) <phil@teuwen.org>"
2048-bit ELG-E key, ID 9A4A59B9, created 2002-05-05 (main key ID 9AD7E3DB)

gpg: encrypted with 2048-bit ELG-E key, ID 9A4A59B9, created 2002-05-05
      "Philippe Teuwen (Doegox) <phil@teuwen.org>"
gpg: session key:
`9:8B4AD21DBF4AAACF3166DC0449211A2D523FAA2ED1F9C3EEBF7EC332C5B18A60'
```



miscHS6_protectionLogiciels.indd 59 21/09/2012 11:22:24

La chaîne obtenue est alors utilisée pour déchiffrer directement le message :

```
$ gpg -override-session-key \
'9:8B4AD21DBF4AAACF3166DC0449211A2D523FAA2ED1F9C3EEBF7EC332C5B18A60' \
message.gpg
```

## 5.5 Cas particuliers

La procédure et les scripts doivent également couvrir certains aspects de maintenance.

Un changement de constitution de la BCT amène à la création d'un nouveau partage et à l'invalidation de l'ancien, le tout en deux temps pour éviter qu'une erreur sur le nouveau partage n'efface à jamais l'accès à la passphrase : seule la mémoire USBSECURE1 sera modifiée et la procédure de récupération de données testée, et en cas de succès, le contenu de USBSECURE2 remplacera celui de USBSECURE1. L'effacement sécurisé des anciens secrets de Shamir sur les USBSECURE suffit à invalider l'ancienne BCT puisqu'il n'y a pas d'autre copie disponible.

D'autres cas particuliers sont par exemple la perte du mot de passe chiffré d'un des membres de la BCT dont la version chiffrée est alors simplement récupérée d'un des USBSECURE et recopiée sur USBPUBLIC.

Enfin, il est sage de tester régulièrement les différentes procédures, par exemple des essais de récupération de données de test mises sous séquestre.

# Conclusions et derniers conseils

Le séquestre de clé en entreprise est avant tout destiné à assurer la continuité des activités, il serait regrettable qu'une fausse manœuvre, un support défectueux ou un bug ruine ce bel échafaudage. En pratique, chaque étape intégrera la redondance et les validations nécessaires pour éviter toute perte ou corruption de données. En outre, il est intéressant de prévoir une trace de l'ensemble des opérations effectuées sur le live CD qui sera retranscrite sur chacune des mémoires USB en fin de session.

La création d'un tel séquestre n'est pas qu'une affaire d'ingénieurs, de crypto et de live CD. C'est avant tout un élément à intégrer à la vie de l'entreprise et à faire accepter aux collaborateurs. Sur ces aspects une documentation claire et exhaustive des procédures à l'usage des membres de la BCT mais aussi de tout un chacun est primordiale et comme on dit : « le diable se cache dans les détails ».

Quelle information placer sous séquestre, quand, sous quelle forme, avec quelle étiquette, à quel endroit ? Qui peut activer la BCT pour récupérer une donnée ? En quelles circonstances ? Quelles procédures appliquer à l'accès physique aux live CD et USBSECURE si on souhaite garantir la règle des deux hommes? Autant de questions qui dépendent de l'environnement dans lequel le déploiement est envisagé.

Nous voici à la fin de cet article qui a tenté de vous montrer une solution de séquestre dont l'autorité est décentralisée ou du moins oligarchique grâce à un partage de Shamir qui devrait gagner la confiance de ses utilisateurs tout en garantissant une certaine résilience quant à son exécution.

Les scripts du live CD sont à disposition dans un projet Google **[ESSSSCROW]** créé pour l'occasion. Toute collaboration ou retour d'expérience sont les bienvenus . ■

### Note sur les générateurs d'aléa :

**Un ordinateur dépourvu d'unité de stockage et de réseau n'a que peu de sources d'entropie à sa disposition s'il n'a pas de composant matériel spécifique (TRNG) pour créer la clé de séquestre. Il peut donc être utile d'intégrer au live CD le paquet haveged. Il s'agit d'une implémentation de l'algorithme HAVEGE (*HArdware Volatile Entropy Gathering and Expansion*) [HAVEGE] qui collecte un maximum d'entropie, notamment par une mesure du temps d'exécution d'une séquence d'instructions, sensible aux prédictions de branchement, à l'état des caches et de divers états internes hautement volatiles du processeur.**

## RÉFÉRENCES

**[SHAMIR] Shamir, Adi (1979), «How to share a secret», Communications of the ACM 22 (11): 612–613**

**[WIKIPEDIA] https://fr.wikipedia.org/wiki/Partage_de_clé_secrète_de_Shamir**

**[SSSS] http://point-at-infinity.org/ssss/**

**[HAVEGE] http://www.irisa.fr/caps/projects/hipsor/**

**[ESSSSCROW] https://code.google.com/p/essssscrow/**



miscHS6_protectionLogiciels.indd 60 21/09/2012 11:22:24