La cryptographie pratique avec GPG

Rémi Oudin

Séminaire Technique du Cr@ns

22 Novembre 2016

Introduction

- Communiquer de façon sûre.
- Quels moyens a-t-on de se protéger ?
- Être sûr de notre interlocuteur.
- Protéger des fichiers sensibles.

Le hachage

But Identifier *rapidement* une donnée initiale.

Le hachage

But Identifier *rapidement* une donnée initiale.

En pratique Une fonction de l'ensemble des données possibles dans l'ensemble des hashs (les résultats), qui est difficilement inversible.

Le hachage

But Identifier *rapidement* une donnée initiale.

En pratique Une fonction de l'ensemble des données possibles dans l'ensemble des hashs (les résultats), qui est difficilement inversible.

Quelques exemples MD5 (cassé), SHA1, SHA256

Le hachage

But Identifier *rapidement* une donnée initiale.

En pratique Une fonction de l'ensemble des données possibles dans l'ensemble des hashs (les résultats), qui est difficilement inversible.

Quelques exemples MD5 (cassé), SHA1, SHA256

Quelles utilisations Mots de passe, salage, checksums

Le checksum, c'est quoi ?

Le *checksum*, ou somme de contrôle, est le hash de donneés transmises. Elle permet de s'assurer que les données n'ont pas été corrompues pendant le transfert.

Un exemple

- Je télécharge un fichier avec des données importantes.
- Je veux m'assurer qu'il n'a pas été corrompu ou modifié pendant le téléchargement.
- La personne en face me transmet le fichier et son checksum.
- Je peux vérifier que le checksum du fichier reçu est bien celui du fichier envoyé.

Quelques exemples de commandes Linux

md5sum, sha256sum, sha512sum

Quelques notions de chiffrement

But : Communiquer de manière sûre avec un interlocuteur connu.

Comment : Transformer un message clair en un message incompréhensible pour toute personne ne possédant pas la clef qui permette de le déchiffrer.

Méthodes :

- Chiffrement simple → Code de César
- Cryptographie symétrique → AES, DES, Blowfish. Une unique clef pour chiffrer et déchiffrer. Autant de clefs que de correspondants.
- Cryptographie asymétrique → RSA (River Shamir Adleman), DSA (Digital Signature Algorithm), ECDSA, ElGamal. 2 clefs par utilisateur. Une clef publique, partageable aux autres, une clef privée.

Le chiffrement asymétrique : le principe

Tout le monde possède un couple de clefs (*publique*, *prive*) :

Clef publique Une clef publique. Tout le monde la possède, elle permet de chiffrer des messages uniquement déchiffrables par la clef privée associée.

Clef privée Gardée secrètement. Elle permet de déchiffrer les messages reçus.

En pratique, on génère une clef symétrique, on la chiffre avec la clef aymétrique, on l'envoie, et on continue avec du chiffrement symétrique.
C'est plus rapide dans la plupart des cas, et c'est plus efficace si on veut chiffrer vers plusieurs destinataires d'un coup.

Un peu de culture : le chiffrement RSA

On parle énormément de RSA. Il est utilisé dans les cartes bleues, dans les clefs SSH,...
Nous allons voir comment ça fonctionne.

- On choisit deux grands entiers premiers distincts p et q

Un peu de culture : le chiffrement RSA

On parle énormément de RSA. Il est utilisé dans les cartes bleues, dans les clefs SSH,...
Nous allons voir comment ça fonctionne.

- On choisit deux grands entiers premiers distincts p et q
- Soit $n = pq$

Un peu de culture : le chiffrement RSA

On parle énormément de RSA. Il est utilisé dans les cartes bleues, dans les clefs SSH,...
Nous allons voir comment ça fonctionne.

- On choisit deux grands entiers premiers distincts p et q
- Soit $n = pq$
- On pose $\varphi(n) = (p-1)(n-1)$

Un peu de culture : le chiffrement RSA

On parle énormément de RSA. Il est utilisé dans les cartes bleues, dans les clefs SSH,...
Nous allons voir comment ça fonctionne.

- On choisit deux grands entiers premiers distincts p et q
- Soit $n = pq$
- On pose $\varphi(n) = (p-1)(n-1)$
- On prend e un entier tel que $e \wedge \varphi(n) = 1$ et $e \leq n$

Un peu de culture : le chiffrement RSA

On parle énormément de RSA. Il est utilisé dans les cartes bleues, dans les clefs SSH,...
Nous allons voir comment ça fonctionne.

- On choisit deux grands entiers premiers distincts p et q
- Soit $n = pq$
- On pose $\varphi(n) = (p-1)(n-1)$
- On prend e un entier tel que $e \wedge \varphi(n) = 1$ et $e \leq n$
- Soit $d \equiv e^{-1} \mod \varphi(n)$

Un peu de culture : le chiffrement RSA

On parle énormément de RSA. Il est utilisé dans les cartes bleues, dans les clefs SSH,...
Nous allons voir comment ça fonctionne.

- On choisit deux grands entiers premiers distincts p et q
- Soit $n = pq$
- On pose $\varphi(n) = (p-1)(n-1)$
- On prend e un entier tel que $e \wedge \varphi(n) = 1$ et $e \leq n$
- Soit $d \equiv e^{-1} \mod \varphi(n)$
- Si M est entier, alors $M^{e \cdot d} \equiv$

Un peu de culture : le chiffrement RSA

On parle énormément de RSA. Il est utilisé dans les cartes bleues, dans les clefs SSH,...
Nous allons voir comment ça fonctionne.

- On choisit deux grands entiers premiers distincts p et q
- Soit $n = pq$
- On pose $\varphi(n) = (p-1)(n-1)$
- On prend e un entier tel que $e \wedge \varphi(n) = 1$ et $e \leq n$
- Soit $d \equiv e^{-1} \mod \varphi(n)$
- Si M est entier, alors $M^{e \cdot d} \equiv$

Un peu de culture : le chiffrement RSA

On parle énormément de RSA. Il est utilisé dans les cartes bleues, dans les clefs SSH,...
Nous allons voir comment ça fonctionne.

- On choisit deux grands entiers premiers distincts p et q
- Soit $n = pq$
- On pose $\varphi(n) = (p-1)(n-1)$
- On prend e un entier tel que $e \wedge \varphi(n) = 1$ et $e \leq n$
- Soit $d \equiv e^{-1} \mod \varphi(n)$
- Si M est entier, alors $M^{e \cdot d} \equiv M \mod n$

Ainsi on a notre couple de clefs !

Clef publique (n, e)

Clef privée d

La signature

But	Prouver son identité aux autres
Clef privée	Elle permet de signer des messages.
Clef publique	Elle permet de vérifier les signatures émises par la clef privée.
Algorithems	DSA, RSA, ElGamal, ECDSA

En pratique, on signe le hash du message, ce qui est plus rapide.

Comment s'échanger ses clefs ?

Jusque maintenant, on a pas abordé l'échange des clefs, on faisait l'hypothèse que les clefs étaient connues des interlocuteurs.
Comment s'échanger les clefs publiques ?

- Rencontrer la personne en vrai. Un peu long, une clef se compte en milliers de caractères.
- L'échanger grâce à une clef USB, au réseau local, par mail,...
- La télécharger sur Internet. Ça implique de vérifier que c'est bien la bonne clef avec l'interlocuteur. Par exemple, confirmer la fingerprint (l'empreinte) de la clef.
- Demander à un.e/des intermédiaires de confiance (de moi et de lui) d'attester l'authenticité de la clef
  $\rightarrow$ C'est le principe des réseaux de confiance.

C'est quoi la différence ?

OpenPGP C'est un protocole et un format de cryptographie. Il définit le format des messagees, des signatures et des certificats que peuvent s'envoyer des logiciels.

GPG (ou GnuPG) C'est le *GNU Privacy Guard*, une des implémentations de OpenPGP, sous licence GPL. Très souvent inclus d'origine sur les OS GNU/Linux, bientôt remplacé par gpg2. Noms des paquets sous debian : gnupg, gnupg2.

Pour l'installer si ce n'est pas le cas : `apt install gnupg`

Générer un couple de clefs

`gpg --gen-key (ou --full-gen-key sous stretch)`

- 4 possibilités de clefs. RSA et RSA est le défaut.
- La taille est un facteur déterminant.
- Elle va de 2048 à 4096 octets (Norme ANSSI $\rightarrow$ 4096 bits pour un usage au delà de 2020).
- Plus la taille est élevée, plus la clef est lourde est longue à générer.
- Pendant la génération de la clef, il est conseillé de générer de l'entropie pour accélérer. On peut copier son dossier de musiques, de séries, taper sur le clavier, ou alors utiliser `haveged`

Un paquet utile : haveged

Il permet de générer de l'entropie dans votre système pour réduire le temps de création de la clef.

Générer un couple de clefs

Indentité C'est le nom et l'email qui apparaîtront aux autres utilisateurs. Ne pas mettre n'importe quoi.

Remarque Soit un moyen de vous identifier de manière sûre, une info importante, soit ne rien mettre.

Date d'expiration Elle peut être repoussée après la génération. Ceci permet de ne pas lui donner une vie trop longue à chaque fois, mais de ne pas changer de clefs tous les 3 mois. Une durée standard est 1 an, prolongeable tous les ans.

Passphrase C'est la seule protection du système en cas de vol de la clef privée. Une passphrase compliquée est plus lente à casser, et permet de gagner du temps pour révoquer la clef avant un usage malhonnête. Quelques conseils :

- Plus la clef est longue, plus elle sera dure à casser.
- Elle devrait utiliser des caractères numériques et spéciaux
- Elle devrait mélanger la casse des caractères alphabétiques.
- Une commande pour générer des mots de passe : `mkpasswd`.

Générer un certificat de révocation

Le certificat de révocation est un fichier qui permet de signifier aux autres de ne plus utiliser la clef publique. Il en existe 2 types différents :

Certicat de perte En cas d'oubli du mot de passe, de perte de la clef.

Certificat de compromission Lorsque la clef privée est compromise.

`gpg --output <fichier>.asc --gen-revoke <key id>` permet de générer un certificat de révocation dans le fichier `<fichier>.asc` Il peut être généré à l'avance, puis stocké sur un support externe (à garder précieusement, toute personne le possédant révoquer votre clef avec).

Lister les informations d'une clef

Le format d'une clef publique est le suivant.

- La portion publique de la clef principale de signature
- Des identifiants utilisés pour associer la clef à l'utilisateur : Nom, commentaire, adresse mail, date de création d'expiration,...Depuis peu, on peut mettre une photo sur une clef pour pouvoir identifier encore mieux une personne. Attention, cette photo est publique.
- Des sous-clefs (chiffrement, authentification, signature, certificat)
- Des signatures.
- Des clefs cryptographiques privées associées si c'est votre clef.

On utilise `gpg --list-key <key_id>` pour afficher les 3 premiers, et de `gpg --list-sigs <key_id>` pour lister les signatures.

Publier sa clef

Il y a deux moyens.

- Exporter sa clef et la publier sur sa page personnelle, par mail,. . .On utilise
  `gpg [--output <file> --armour --export <key_id>`
- La publier sur un serveur de clef !
- On utilise
  `gpg [--keyserver <server url>] --send-key <key_id>`

Récupérer une clef

- La récupérer sur une page perso, par mail...
- `gpg --import <key_file>`
- La récupérer sur un serveur de clef.
- `gpg --recv-key <key_id>` quand on connaît le key id.
- `gpg --search-keys <mail ou nom>` sinon.
- Le serveur affiche les informations publiques des clefs qu'il a trouvé.

Attention !

Il n'y a *a priori* aucune raison d'avoir confiance en cette clef. Elle aurait pu être corrompue pendant le transfert, modifiée par un utilisateur extérieur,...On va voir comment faire pour vérifier qu'une clef appartient bien à son propriétaire.

Comment vérifier le propriétaire d'une clef ?

Comme tout à l'heure, il y a plusieurs moyens.

- Recontrer la personne en vrai et échanger physiquement la clef avec lui.
- La télécharger, et vérifier le hash IRL.

Dans tous les cas, il faut rencontrer la personne physiquement.

L'intérêt de la signature

- Permet d'attester aux autres de l'identité d'une clef, la sienne ou celle d'un autre.

L'intérêt de la signature

- Permet d'attester aux autres de l'identité d'une clef, la sienne ou celle d'un autre.
- C'est quoi une identité ? Un nom sur un passeport ou une carte d'identité ? Ce n'est pas une question facile !

L'intérêt de la signature

- Permet d'attester aux autres de l'identité d'une clef, la sienne ou celle d'un autre.
- C'est quoi une identité ? Un nom sur un passeport ou une carte d'identité ? Ce n'est pas une question facile !
- En général, on utilise un document délivré par l'État. Passeport, carte d'identité, permis,...

L'intérêt de la signature

- Permet d'attester aux autres de l'identité d'une clef, la sienne ou celle d'un autre.
- C'est quoi une identité ? Un nom sur un passeport ou une carte d'identité ? Ce n'est pas une question facile !
- En général, on utilise un document délivré par l'État. Passeport, carte d'identité, permis,...
- Ne pas signer mal ou trop rapidement. La manière dont on signe peut influencer la confiance qu'on accorde à une clef.

Signer en pratique

Pour signer une clef, il faut tout d'abord l'avoir sur son ordinateur.

- On passe en mode édition : `gpg --edit-key <key_id>`
- On vérifie la fingerprint : `fpr` et demander à l'autre de dicter sa fingerprint.
- `sign` On modifie la clef publique pour dire qu'on l'a signée.
- `trust` On donne un niveau de confiance (local) à la clef. Il va de 1 à 5.
  - 5 ⇒ ultime. N'utiliser que pour vos clefs !
  - 4 ⇒ Je fais entièrement confiance. À n'utiliser qu'avec les personnes en lesquelles vous avez le plus confiance. S'ils signent des personnes, ceci peut avoir une incidence sur votre clef !
  - 3 ⇒ normal. Le niveau par défaut qu'on donne à une personne qu'on connaît
  - 2 ⇒ Je ne fais pas confaince.
  - 1 ⇒ Je n'ai pas d'avis

Et enfin, on renvoie la clef qu'on vient de signer afin de mettre le serveur à jour. `gpg --send-key <key_id>`

D'autres commandes utiles

- Mettre à jour les clefs de son trousseau (ses clefs, les clefs qu'on a téléchargé, et les niveaux de confiance) : `gpg --refresh-keys`
- Mettre à jour sa base de confiance : `gpg --update-trust-db`
- Afficher la fingerprint d'une clef : `gpg --fingerprint <key_id>`

Chiffrer, signer un document

- `gpg --armour -e -r <id_name> <fichier>`, pour chiffer le fichier pour le nom donné. Nécessite une confiance suffisante en la clef en face.

Chiffrer, signer un document

- `gpg --armour -e -r <id_name> <fichier>`, pour chiffer le fichier pour le nom donné. Nécessite une confiance suffisante en la clef en face.
- `gpg -d <fichier>` pour déchiffer un fichier. Cela nécessite d'avoir la clef privée associée.

Chiffrer, signer un document

- `gpg --armour -e -r <id_name> <fichier>`, pour chiffer le fichier pour le nom donné. Nécessite une confiance suffisante en la clef en face.
- `gpg -d <fichier>` pour déchiffer un fichier. Cela nécessite d'avoir la clef privée associée.
- `gpg --armour -s <fichier` pour signer un fichier.

Chiffrer, signer un document

- `gpg --armour -e -r <id_name> <fichier>`, pour chiffer le fichier pour le nom donné. Nécessite une confiance suffisante en la clef en face.
- `gpg -d <fichier>` pour déchiffer un fichier. Cela nécessite d'avoir la clef privée associée.
- `gpg --armour -s <fichier` pour signer un fichier.
- `gpg --clearsign <fichier>` pour signer un fichier et le préfixer du fichier en clair. Plus facile lors de l'utilisation de mails par exemple.

Chiffrer, signer un document

- `gpg --armour -e -r <id_name> <fichier>`, pour chiffer le fichier pour le nom donné. Nécessite une confiance suffisante en la clef en face.
- `gpg -d <fichier>` pour déchiffer un fichier. Cela nécessite d'avoir la clef privée associée.
- `gpg --armour -s <fichier` pour signer un fichier.
- `gpg --clearsign <fichier>` pour signer un fichier et le préfixer du fichier en clair. Plus facile lors de l'utilisation de mails par exemple.
- `gpg --verify <fichier>` Vérifier une signature.

Envoyer des mails

La boucle est bouclée, on va apprendre à envoyer des mails !
De nombreux clients ont un add-on pour utiliser gpg.

- Enigmail existe pour Thunderbird.
- Evolution
- mutt qui a un support total de GPG
- Claws mail, qui supporte nativement GPG

Cranspasswords

- Projet développé au Crans pour partager les mots de passe.
- Plusieurs rôles existent, avec différents accès.
- Plusieurs fichiers qui contiennent chacun un mot de passe.
- Il faut être signé par la personne qui chiffre les mots de passe.
- Il y a un serveur qui possède les fichiers chiffrés, et les clients qui les récupèrent via ssh et les déchiffrent en local.

Conclusion

On a déjà vu pas mal de trucs, les bases du chiffrement principalement. Cependant, parmis tous les points qu'on a pas vu :

- Garder sa clef maîtresse à l'abri
- Créer des sous-clefs ou des multiples identités.
- Mettre à jour la durée de vie de sa clef
- Utiliser une smartcard : la clef est sur une puce spéciale, et pour l'utiliser on insère cette puce dans le pc.
- Gérer son réseau de confiance.

N'hésitez pas à lire la documentation sur gnupg, elle est bien faite !

- Les slides des années précédentes.
- http://doc.ubuntu-fr.org/gnupg
- http://www.gnupg.org/gph/fr/manual.html
- http://matrix.samizdat.net/crypto.gpg_intro
- https://wiki.debian.org/subkeys