Introduction à PGP

Chiffrer, déchiffrer et signer

Alexandre IOOSS, *erdnaxe*

CR@NS

Mardi 13 novembre 2018

Section 1

Primitives cryptographiques

Hachage

But du hachage

Créer un *hash* qui identifie *rapidement* une donnée initiale.

En pratique

Fonction de l'ensemble des données (infini) dans l'ensemble des hash (fini), difficilement inversible.

Hachage

But du hachage

Créer un *hash* qui identifie *rapidement* une donnée initiale.

En pratique

Fonction de l'ensemble des données (infini) dans l'ensemble des hash (fini), difficilement inversible.

Exemples d'algorithmes

MD5, SHA1, SHA256...

Hachage

But du hachage

Créer un *hash* qui identifie *rapidement* une donnée initiale.

En pratique

Fonction de l'ensemble des données (infini) dans l'ensemble des hash (fini), difficilement inversible.

Exemples d'algorithmes

MD5, SHA1, SHA256...

Utilisations

Mot de passe, salage, sûreté d'un téléchargement.

Hachage : À vous de jouer ☺

Utilisez md5sum, sha1sum et sha512sum pour hasher “J’aime le crans”.

Hachage : À vous de jouer ☺

Utilisez md5sum, sha1sum et sha512sum pour hasher “J’aime le crans”.

```
echo "J'aime le crans" > mon_fichier
md5sum mon_fichier
sha1sum mon_fichier
sha512sum mon_fichier
```

Allez sur http://releases.ubuntu.com/18.04.1/.

Chiffrement symétrique

But du chiffrement symétrique

Communiquer de manière sûre avec un interlocuteur connu.
Il faut que le *secret* ne soit connu que de ces deux *pairs*.
Un *plaintext* s'échange en étant chiffrer en *cyphertext* puis déchiffrer.

Chiffrement symétrique

But du chiffrement symétrique

Communiquer de manière sûre avec un interlocuteur connu.
Il faut que le *secret* ne soit connu que de ces deux *pairs*.
Un *plaintext* s'échange en étant chiffrer en *cyphertext* puis déchiffrer.

Exemples d'algorithmes

AES, DES, 3DES, Blowfish, XOR...

Chiffrement symétrique

But du chiffrement symétrique

Communiquer de manière sûre avec un interlocuteur connu.
Il faut que le *secret* ne soit connu que de ces deux *pairs*.
Un *plaintext* s'échange en étant chiffrer en *cyphertext* puis déchiffrer.

Exemples d'algorithmes

AES, DES, 3DES, Blowfish, XOR...
Cas réel : Minecraft utilise AES/CFB8 pour jouer sur des serveurs *online*.

Chiffrement asymétrique

Fonctionnement

Clé publique : Tout le monde la possède. Sert à chiffrer des messages qui ne sont pas déchiffrables sans clé privée.
Clé privée : gardée *jalousement*, elle permet de déchiffrer les messages.

Chiffrement asymétrique

Fonctionnement

Clé publique : Tout le monde la possède. Sert à chiffrer des messages qui ne sont pas déchiffrables sans clé privée.
Clé privée : gardée *jalousement*, elle permet de déchiffrer les messages.

Exemples d'algorithmes

RSA, ElGamal...

Chiffrement asymétrique

Fonctionnement

Clé publique : Tout le monde la possède. Sert à chiffrer des messages qui ne sont pas déchiffrables sans clé privée.
Clé privée : gardée *jalousement*, elle permet de déchiffrer les messages.

Exemples d'algorithmes

RSA, ElGamal...

Utilisations

En pratique, on préfère générer une clé symétrique, la chiffrer et l'envoyer, puis continuer avec la communication avec du chiffrement symétrique.
Souvent plus rapide. Plus efficace si l'on chiffre pour plusieurs destinataires à la fois.
SSH, SSL...

Illustration

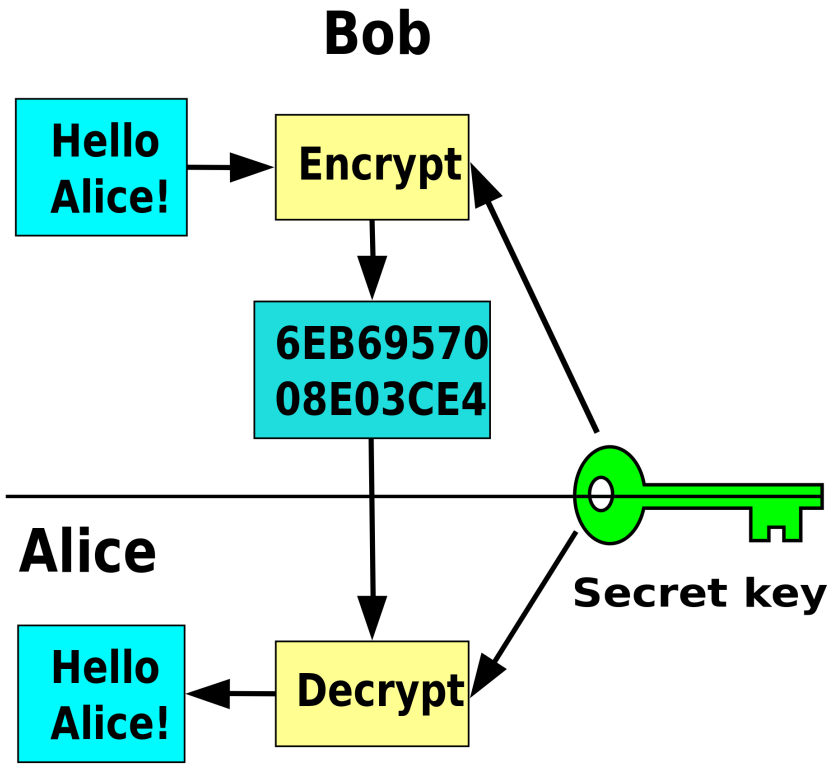


FIG. 1: Chiffrement symétrique

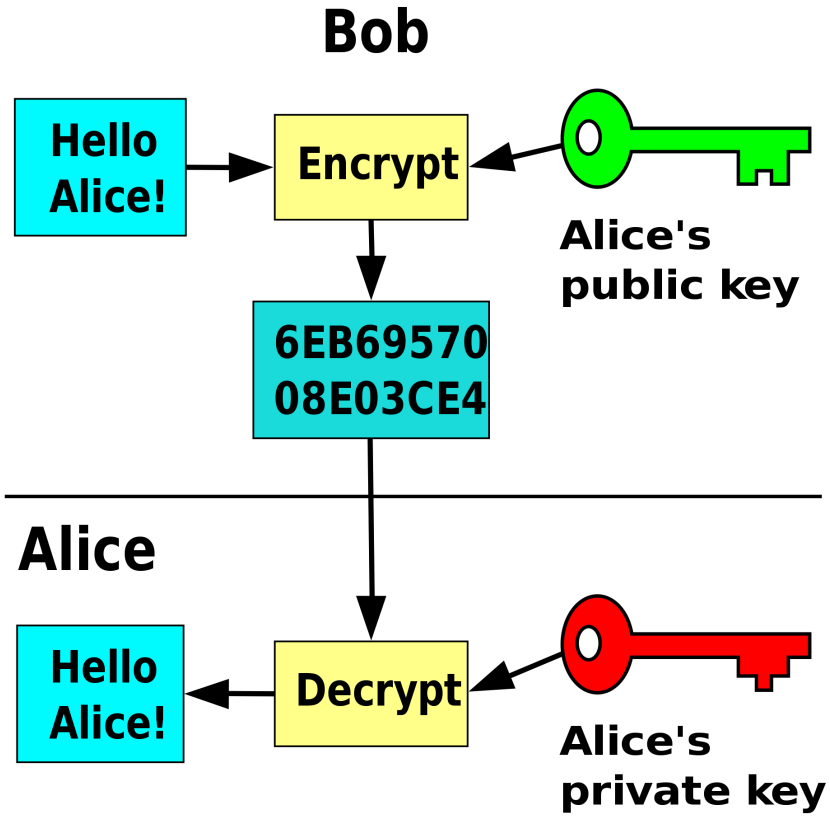


FIG. 2: Chiffrement asymétrique

Pour la culture : principe de RSA

- On choisit deux (*grands*) entiers p et q premiers et distincts
- On pose $n = p \cdot q$
- On calcule $\varphi(n) =$

Pour la culture : principe de RSA

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

Pour la culture : principe de RSA

- On choisit deux (*grands*) entiers p et q premiers et distincts
- On pose $n = p \cdot q$
- On calcule $\varphi(n) = (p-1)(q-1)$
- Soit e entier inférieur à, et premier avec, $\varphi(n)$

Pour la culture : principe de RSA

- On choisit deux (*grands*) entiers p et q premiers et distincts
- On pose $n = p \cdot q$
- On calcule $\varphi(n) = (p-1)(q-1)$
- Soit e entier inférieur à, et premier avec, $\varphi(n)$
- Calculer $d \equiv e^{-1} \mod \varphi(n)$
- On remarque que pour tout M entier, $M^{e \cdot d} \equiv$

Pour la culture : principe de RSA

- On choisit deux (*grands*) entiers p et q premiers et distincts
- On pose $n = p \cdot q$
- On calcule $\varphi(n) = (p-1)(q-1)$
- Soit e entier inférieur à, et premier avec, $\varphi(n)$
- Calculer $d \equiv e^{-1} \mod \varphi(n)$
- On remarque que pour tout M entier, $M^{e \cdot d} \equiv M \mod n$.

Pour la culture : principe de RSA

- On choisit deux (*grands*) entiers p et q premiers et distincts
- On pose $n = p \cdot q$
- On calcule $\varphi(n) = (p-1)(q-1)$
- Soit e entier inférieur à, et premier avec, $\varphi(n)$
- Calculer $d \equiv e^{-1} \mod \varphi(n)$
- On remarque que pour tout M entier, $M^{e \cdot d} \equiv M \mod n$.

Clef publique : (n, e) Clef privée : d.

Signature

But de la signature

Opération inverse : prouver son identité.

Clé privée : gardée jalousement, elle permet d'émettre (signer) des messages.

Clé publique : Tout le monde la possède. Sert à vérifier les signatures, émises par la clé privée.

Signature

But de la signature

Opération inverse : prouver son identité.

Clé privée : gardée jalousement, elle permet d'émettre (signer) des messages.

Clé publique : Tout le monde la possède. Sert à vérifier les signatures, émises par la clé privée.

Exemples d'algorithmes

DSA, RSA, ElGamal, courbes elliptiques.

Signature

But de la signature

Opération inverse : prouver son identité.
Clé privée : gardée jalousement, elle permet d'émettre (signer) des messages.
Clé publique : Tout le monde la possède. Sert à vérifier les signatures, émises par la clé privée.

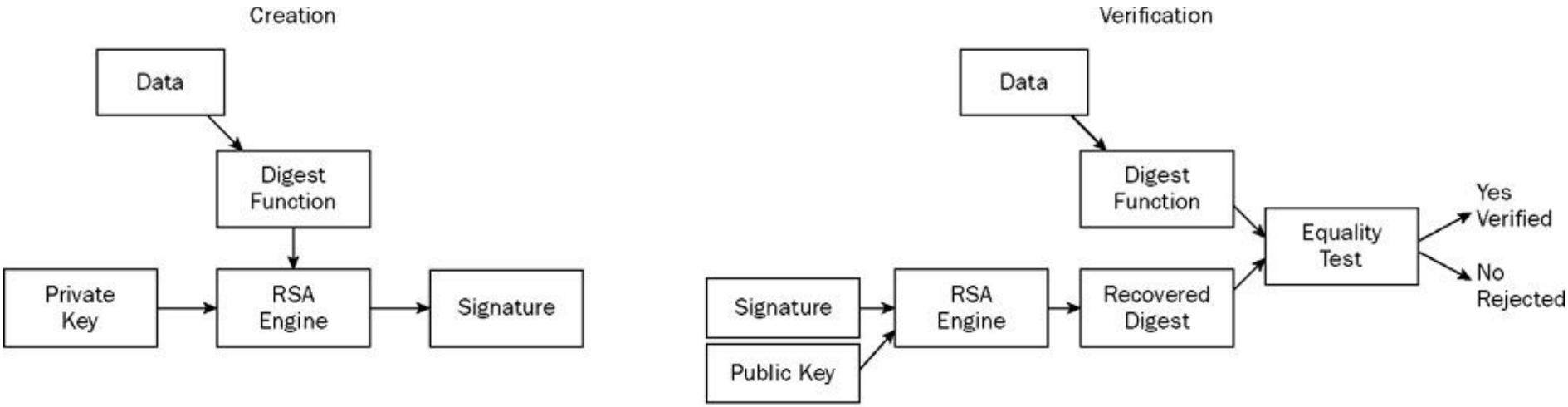
Exemples d'algorithmes

DSA, RSA, ElGamal, courbes elliptiques.

Application

Signer des mails, signer des paquets logiciels.
En pratique : on préfère signer le hash du message.

Illustration



RSA Signature Processes

Introduction aux réseaux de confiance

Comment s'échanger les clés publiques initialement ?

- Rencontrer la personne en vrai ;

Introduction aux réseaux de confiance

Comment s'échanger les clés publiques initialement ?

- Rencontrer la personne en vrai ;
- Une clé c'est long : l'échanger grâce à une clé, au réseau local...

Introduction aux réseaux de confiance

Comment s'échanger les clés publiques initialement ?

- Rencontrer la personne en vrai ;
- Une clé c'est long : l'échanger grâce à une clé, au réseau local…
- La télécharger sur Internet, mais demander à son interlocuteur, dans la vraie vie, de confirmer le hash (empreinte) ;

Introduction aux réseaux de confiance

Comment s'échanger les clés publiques initialement ?

- Rencontrer la personne en vrai ;
- Une clé c'est long : l'échanger grâce à une clé, au réseau local…
- La télécharger sur Internet, mais demander à son interlocuteur, dans la vraie vie, de confirmer le hash (empreinte) ;
- Sinon : demander à un (des) intermédiaire(s) de confiance d'attester de l'authenticité de la clé (signatures).

Section 2

GPG

GPG ou PGP ?

OpenPGP

Format et protocole de cryptographie.

GPG ou PGP ?

OpenPGP

Format et protocole de cryptographie.

GnuPG (GPG)

Une des implémentations de OpenPGP, sous licence GNU GPL.

`apt install gnupg` (souvent pré-installé)

Générer une clef

Pour générer une clef PGP

Avec `gpg --gen-key`

- Taille de clé : entre 2048 et 4096 ;
- Plus la clef est longue, plus elle est dure à casser ;
- ... mais également plus lourde ;
- ... mais également plus longue à générer.

Recommandation pour le RSA par l'*ANSSI* : 4096 bits (pour > 2020).

Générer une clef

Pour générer une clef PGP

Avec `gpg --gen-key`

- Taille de clé : entre 2048 et 4096 ;
- Plus la clef est longue, plus elle est dure à casser ;
- ... mais également plus lourde ;
- ... mais également plus longue à générer.

Recommandation pour le RSA par l'*ANSSI* : 4096 bits (pour > 2020).

Remarques sur la génération de clef

- **La date d'expiration** peut-être changée après génération ;
- **Identité** : c'est le nom et le mail qui apparaîtra pour les autres ;
- **Passphrase** : dernière protection en cas de vol de la clef privée.

Générer une clef

Générer un certificat de révocation

Avec `gpg --gen-revoke`

Exemple : `gpg --output mon_certif_de_revocation.asc --gen-revoke 6E1C820B`

Générer une clef

Générer un certificat de révocation

Avec `gpg --gen-revoke`

Exemple : `gpg --output mon_certif_de_revocation.asc --gen-revoke 6E1C820B`

Une clef PGP est composée de :

`gpg --list-keys 6E1C820B`

- **Une clé cryptographique** publique de signature ;
- **Des identités** : nom et mail ;
- **Des sous-clefs** (chiffrement, signature, authentification, certificat) ;
- **Des signatures** (`gpg --list-sigs 6E1C820B`) ;
- Les clefs cryptographiques privées associées (si c'est la votre).

Échange de clefs

Publier sa clef PGP

- `gpg --armour --export 6E1C820B`
- Sur sa page perso, par mail, etc
- Ou alors...

Échange de clefs

Publier sa clef PGP

- `gpg --armour --export 6E1C820B`
- Sur sa page perso, par mail, etc
- Ou alors... utiliser un serveur de clé !
- `gpg --send-key 6E1C820B`

Échange de clefs

Publier sa clef PGP

- `gpg --armour --export 6E1C820B`
- Sur sa page perso, par mail, etc
- Ou alors... utiliser un serveur de clé !
- `gpg --send-key 6E1C820B`

Récupérer la clé PGP de quelqu'un

- Sur sa page perso, par mail, etc
- `gpg --import`
- Ou alors...

Échange de clefs

Publier sa clef PGP

- `gpg --armour --export 6E1C820B`
- Sur sa page perso, par mail, etc
- Ou alors... utiliser un serveur de clé !
- `gpg --send-key 6E1C820B`

Récupérer la clé PGP de quelqu'un

- Sur sa page perso, par mail, etc
- `gpg --import`
- Ou alors... utiliser un serveur de clé !
- `gpg --recv-key 6E1C820B` (hkp ://keys.gnupg.net)

Échange de clefs

Publier sa clef PGP

- `gpg --armour --export 6E1C820B`
- Sur sa page perso, par mail, etc
- Ou alors... utiliser un serveur de clé !
- `gpg --send-key 6E1C820B`

Récupérer la clé PGP de quelqu'un

- Sur sa page perso, par mail, etc
- `gpg --import`
- Ou alors... utiliser un serveur de clé !
- `gpg --recv-key 6E1C820B` (hkp ://keys.gnupg.net)
- `gpg --search-keys erdnaxe` (à vous d'essayez !)

Échange de clefs

Publier sa clef PGP

- `gpg --armour --export 6E1C820B`
- Sur sa page perso, par mail, etc
- Ou alors... utiliser un serveur de clé !
- `gpg --send-key 6E1C820B`

Récupérer la clé PGP de quelqu'un

- Sur sa page perso, par mail, etc
- `gpg --import`
- Ou alors... utiliser un serveur de clé !
- `gpg --recv-key 6E1C820B` (hkp ://keys.gnupg.net)
- `gpg --search-keys erdnaxe` (à vous d'essayez !)

Attention : Vous n'avez a priori aucune raison d'avoir confiance en cette clef !

Signer

Comment vérifier qu'une clef appartient bien à son propriétaire ?

- Rencontrer la personne en vrai, et échanger physiquement la clef ;

Signer

Comment vérifier qu'une clef appartient bien à son propriétaire ?

- Rencontrer la personne en vrai, et échanger physiquement la clef ;
- La télécharger (Internet) et échanger physiquement le hash (empreinte) ;

Signer

Comment vérifier qu'une clef appartient bien à son propriétaire ?

- Rencontrer la personne en vrai, et échanger physiquement la clef ;
- La télécharger (Internet) et échanger physiquement le hash (empreinte) ;

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

Signer

Comment vérifier qu'une clef appartient bien à son propriétaire ?

- Rencontrer la personne en vrai, et échanger physiquement la clef ;
- La télécharger (Internet) et échanger physiquement le hash (empreinte) ;

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

Attention : ne pas confondre ID et empreinte de clef !

Signer

Comment vérifier qu'une clef appartient bien à son propriétaire ?

- Rencontrer la personne en vrai, et échanger physiquement la clef ;
- La télécharger (Internet) et échanger physiquement le hash (empreinte) ;

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

Attention : ne pas confondre ID et empreinte de clef !

Signature de signatures, pourquoi ?

- Attester aux autres de l'identité d'une clef ;

Signer

Comment vérifier qu'une clef appartient bien à son propriétaire ?

- Rencontrer la personne en vrai, et échanger physiquement la clef ;
- La télécharger (Internet) et échanger physiquement le hash (empreinte) ;

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

Attention : ne pas confondre ID et empreinte de clef !

Signature de signatures, pourquoi ?

- Attester aux autres de l'identité d'une clef ;
- Mais au fond, qu'est-ce qu'une identité ?

Signer

Comment vérifier qu'une clef appartient bien à son propriétaire ?

- Rencontrer la personne en vrai, et échanger physiquement la clef ;
- La télécharger (Internet) et échanger physiquement le hash (empreinte) ;

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

Attention : ne pas confondre ID et empreinte de clef !

Signature de signatures, pourquoi ?

- Attester aux autres de l'identité d'une clef ;
- Mais au fond, qu'est-ce qu'une identité ? Vérification souvent d'une pièce d'identité officielle.

Signer

Comment vérifier qu'une clef appartient bien à son propriétaire ?

- Rencontrer la personne en vrai, et échanger physiquement la clef ;
- La télécharger (Internet) et échanger physiquement le hash (empreinte) ;

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

Attention : ne pas confondre ID et empreinte de clef !

Signature de signatures, pourquoi ?

- Attester aux autres de l'identité d'une clef ;
- Mais au fond, qu'est-ce qu'une identité ? Vérification souvent d'une pièce d'identité officielle.
- La manière dont vous signez une personne peut influencer la confiance qu'elle accordera à votre clé !

Signer

Une procédure de signature

On passe en mode édition avec `gpg --edit-key [user]`

- `fpr` (se faire dicter la clef)
- Vérifier une pièce d'identité
- `sign`
- `trust`
- `save`

Signer

Une procédure de signature

On passe en mode édition avec `gpg --edit-key [user]`

- `fpr` (se faire dicter la clef)
- Vérifier une pièce d'identité
- `sign`
- `trust`
- `save`

Renvoyer la clef signée : `gpg --send-key 6E1C820B` (ou `gpg --export 6E1C820B`).

Signer

Une procédure de signature

On passe en mode édition avec `gpg --edit-key [user]`

- `fpr` (se faire dicter la clef)
- Vérifier une pièce d'identité
- `sign`
- `trust`
- `save`

Renvoyer la clef signée : `gpg --send-key 6E1C820B` (ou `gpg --export 6E1C820B`).

NB : d'autres méthodes de signature sont envisageables quand de nombreuses personnes veulent s'entre-signer.

Trousseau de clefs

- Situé dans `~/.gnupg/` ;
- Contient les clefs privées ;
- Contient les clés téléchargées ;

Trousseau de clefs

- Situé dans `~/.gnupg/` ;
- Contient les clefs privées ;
- Contient les clés téléchargées ;
- Et les niveau de confiance !

Pensez à mettre à jour :

- Les clefs de votre trousseau : `gpg --refresh-keys` ;
- Sa base de confiance : `gpg --update-trustdb`

Section 3

Applications

Chiffrer & signer

- `gpg --armour -e -r erdnaxe [fichier]` nécessite une confiance suffisante en la clé destination ;

Chiffrer & signer

- `gpg --armour -e -r erdnaxe [fichier]` nécessite une confiance suffisante en la clé destination ;
- `gpg -d [fichier]` nécessite de posséder la clef privée associée ;

Chiffrer & signer

- `gpg --armour -e -r erdnaxe [fichier]` nécessite une confiance suffisante en la clé destination ;
- `gpg -d [fichier]` nécessite de posséder la clef privée associée ;
- Signer : `gpg --armour -s [fichier]` ;
- Vérifier une signature : `gpg --verify [fichier]` ;

Chiffrer & signer

- `gpg --armour -e -r erdnaxe [fichier]` nécessite une confiance suffisante en la clé destination ;
- `gpg -d [fichier]` nécessite de posséder la clef privée associée ;
- Signer : `gpg --armour -s [fichier]` ;
- Vérifier une signature : `gpg --verify [fichier]` ;
- Version claire : `gpg --clearsign [fichier]`.

Envoyer des mails

- Thunderbird (Enigmail) ;
- GMail web (FlowCrypt) ;
- etc...

CransPasswords (instant pub)

- Projet Cranseux de partage de mot de passe
- Plusieurs *roles* : nounous, apprentis, ca
- Plusieurs fichiers
- Il faut être signé par la personne qui chiffre les mots de passe

Tout est sur le Gitlab du Crans !

Section 4

Conclusion

Ce dont on n'a pas parlé (liste non exhaustive)

- Comment garder sa clé maîtresse à l'abris ;
- Créer des sous-clés de chiffrement ;
- Mettre à jour les dates d'expiration ;
- Batch signer plusieurs personnes et valider leur mail ;
- Utiliser une smartcard ;
- Comment gérer/personaliser son réseau de confiance.

Quelques liens

- Voir le PDF sur le Wiki !
- http://doc.ubuntu-fr.org/gnupg
- http://fr.wikibooks.org/wiki/GPG
- http://www.gnupg.org/gph/fr/manual.html
- http://www.legifrance.gouv.fr/
- http://fr.wikipedia.org/wiki/Cryptographie
- http://security.stackexchange.com/questions/5096/
- http://www.linuxquestions.org/questions/linux-security-4/
- http://wiki.debian.org/Subkeys
- https://www.dcode.fr/

À vous

- `gpg --search-key erdnaxe`
- `gpg --recv-key A8B68FA13E865318`
- `gpg --keyserver keyserver.ubuntu.com --recv 25B9AD73BBFBD54E`
- `gpg --edit-key`
- `gpg --armour --export A8B68FA13E865318`