

Codage Cryptographie

E. Jeandel

But du cours

Expliquer comment utiliser la cryptographie en pratique

1 Outils

Cryptographie symétrique

- On a deux fonctions $D(k, x)$ et $E(k, x)$ qui permettent respectivement de coder et de décoder le message x avec la clé k .
- $D(k, (E(k, x))) = x$
- On notera plutôt D_k et E_k .
- Exemple : DES.

Cryptographie symétrique

Propriétés voulues :

- Il est “facile” de calculer $D_k(x)$ (resp $E_k(x)$) connaissant k et x .
- Il est “difficile” de calculer x connaissant $E_k(x)$.
- Il est “difficile” de générer un y tel que $E_k(x) = y$.
- Il est “difficile” de calculer k connaissant $E_k(x)$ et x .
- La sécurité de l’algorithme réside dans la clé k .

Cryptographie à clé publique

- Deux clés k_1 et k_2 . k_1 est connue de tout le monde (clé publique), k_2 est secrète (clé privée).
- $D_{k_2}(E_{k_1}(x)) = x$
- Tout le monde peut encrypter, seul ceux connaissant k_2 peuvent décrypter.
- Exemple : RSA.

Variantes :

- $D_{k_1}(E_{k_2}(x)) = x$
- Tout le monde peut décrypter, seul ceux connaissant k_2 peuvent encrypter.

Autre variante : les deux à la fois.

Cryptographie à clé publique

Propriétés voulues :

- Il est "facile" d'encrypter (de calculer $E_{k_1}(x)$) connaissant k_1 et x .
- Il est "difficile" de trouver x connaissant $E_{k_1}(x)$.
- Il est "difficile" de trouver k_2 connaissant k_1 .
- La sécurité de l'algorithme réside dans la clé privée k_2 .

Fonctions unidirectionnelles (de hachage)

- Une fonction f qui envoie tout mot sur un mot de n bits (pour un certain n).
- Exemple : MD5, SHA.

Fonctions unidirectionnelles

- Il est facile de calculer $f(x)$ connaissant x
- Il est difficile de trouver un y tel que $f(y) = x$.
- Il est difficile de trouver x, y tel que $f(y) = f(x)$.

Exercice

- Combien de chaînes y faut-il tester avant d'en trouver une (avec probabilité 0.5) telle que $f(y) = x$?
- Combien de chaînes faut-il tester avant d'en trouver deux x, y telles que $f(y) = f(x)$?

2 Protocoles

Protocoles

- Deux personnes, Alice et Bob, essaient de communiquer

Deux types d'espions

- Un espion qui se contente d'écouter la conversation
- Un espion qui peut prendre part à la conversation (changer des messages, en émettre de nouveaux, etc)

Alice et Bob veulent se protéger des espions.

2.1 Signer

Signer

Alice veut signer un document électronique de sorte que

- Personne ne peut signer à sa place sans qu'on s'en rende compte.
- On se rend compte qu'elle a signé
- La signature n'est pas déplaçable ou supprimable.
- On ne peut pas modifier le document après signature

Signer - Protocole à clé publique 1

Soit x le document.

- Alice calcule $y = E_{k_2}(x)$.
- Le résultat est le document y .

Caractéristiques ?

- Personne ne peut signer à sa place sans qu'on s'en rende compte ?
- On se rend compte qu'elle a signé ?
- La signature n'est pas déplaçable ou supprimable ?
- On ne peut pas modifier le document après signature ?

Signer - Protocole à clé publique 2

Soit x le document.

- Alice calcule $y = E_{k_2}(f(x))$
- Le résultat est le document x , adossé à y

Ajout important : une date de validité du document (pour éviter de l'utiliser plusieurs fois)

2.2 Communiquer

Communiquer

- Alice et Bob veulent communiquer.
- Pour cela, Alice va transférer une clé k à Bob, puis ils vont tous les deux communiquer en utilisant k .

Protocole 1

- Bob envoie sa clé publique k_1 à Alice.
- Alice envoie k à Bob encrypté : $E_{k_1}(k)$
- Bob décrypte la réponse d'Alice, ensuite ils discutent.

Problème ?

Protocole 1 - Attaque (Man in the Middle)

- Bob envoie sa clé publique k_1 à Alice.
- Evil intercepte l'envoi, et envoie sa clé publique k'_1 à Alice.
- Alice envoie k à Bob encrypté : $E_{k'_1}(k)$
- Evil intercepte l'envoi, décrypte le résultat, obtient k
- Evil envoie $E_{k_1}(k)$ à Bob.
- Bob décrypte la réponse d'Evil, ensuite ils discutent.

Résister à l'attaque

- Alice envoie sa clé publique k_A à Bob.
- Bob envoie sa clé publique k_B à Alice.
- Alice prépare un message m et l'encrypte avec k_B pour obtenir m' .
- Bob prépare un message n et l'encrypte avec k_A pour obtenir n' .
- Alice envoie la moitié du message m' .
- Bob réceptionne la moitié de m' et envoie la moitié de n' .
- Alice réceptionne la première moitié de n' et envoie la deuxième moitié de m' .
- Bob réceptionne la deuxième moitié de m' , décode pour obtenir m et envoie la deuxième moitié de n' .
- Alice réceptionne la deuxième moitié de n' , décode pour obtenir n .

2.3 Identifier

Authentification

- Alice veut se connecter sur une machine.

Méthode(s) dangereuse(s)

- Alice envoie son mot de passe x .
- La machine vérifie que x est bien le mot de passe d'Alice
- Alice envoie $f(x)$.
- La machine vérifie que $f(x)$ est bien le résultat de f sur le mot de passe d'Alice

Méthode un peu plus sûre

- Le serveur envoie un nombre r aléatoire.
- Alice envoie $f(xr)$ où x est son mot de passe.
- Le serveur vérifie.

Variante : Le serveur a la clé publique d'Alice.

- Le serveur envoie un nombre r aléatoire.
- Alice encrypte r avec sa clé privée.
- Le serveur vérifie.

Danger : le serveur peut faire encrypter n'importe quoi à Alice.

2.4 Divers

Paris

- Alice veut montrer qu'elle sait prédire l'avenir
- Alice écrit dans une enveloppe qui va gagner le match OM-Benfica
- Bob (Damir Skomina), arbitre du match, ouvre l'enveloppe le lendemain et vérifie qu'Alice avait raison
- Alice ne doit pas pouvoir revenir sur sa décision une fois le match fini
- Bob ne doit pas savoir ce qui est écrit dans l'enveloppe avant la fin du match

Protocole

Avant le match

- Alice "devine" x , et l'encrypte avec une clé k , pour obtenir $y = E_k(x)$
- Alice envoie y à Bob.

Après le match

- Alice envoie k à Bob.

Problème : Alice pourrait trouver un k' à envoyer à Bob après le match. Comme il n'y a que trois possibilités (OM gagne, Benfica gagne, match nul), elle peut trouver facilement trois clés différentes qui donne le même y pour les trois possibilités.

Protocole rectifié

Avant le match

- Bob envoie un nombre aléatoire r à Alice
- Alice "devine" x
- Alice encrypte avec une clé k , pour obtenir $y = E_k(xr)$
- Alice envoie y à Bob.

Après le match

- Alice envoie k à Bob.

Protocole sans intervention de Bob

Avant le match

- Alice "devine" x
- Alice prend deux chaînes aléatoires r_1, r_2 .
- Alice envoie $f(xr_1r_2)$ et r_1 à Bob.

Après le match

- Alice envoie r_2 et x à Bob.

Pile ou Face

- Alice et Bob veulent tirer à Pile ou Face.
- Alice lance la pièce
- Bob répond
- Alice dit si Bob a gagné.

Autre variante

- Les protocoles pour les paris marchent encore.
- Bob envoie un nombre aléatoire r à Alice
- Alice lance la pièce, obtient x
- Alice encrypte avec une clé k , pour obtenir $y = E_k(xr)$
- Alice envoie y à Bob.
- Bob dit pile ou face.
- Alice envoie k

Problème du protocole : Alice connaît le résultat avant Bob.

Algorithmes qui commutent

- Supposons qu'on ait un algorithme de cryptage qui commute, c'est à dire : crypter par k_1 puis par k'_1 permet d'obtenir le même résultat que crypter par k'_1 puis k_1 .
- Métaphore des cadenas : Crypter par k_1 revient à mettre un cadenas sur une valise, crypter par k'_1 à mettre un autre cadenas.

Algorithmes qui commutent - Utilisation

- Transmettre le message x entre Alice et Bob.
- Alice crypte x par k_1 et l'envoie à Bob.
- Bob crypte le résultat par k_2 et l'envoie à Alice.
- Alice décrypte avec k_1 et envoie le résultat à Bob.
- Bob décrypte avec k_2 et lit le résultat.

Algorithmes qui commutent - Pile ou Face

- Alice génère deux messages, un signifiant pile, l'autre face qui sont imprévisibles.
- Alice encode les deux messages avec k_1 et les envoie à Bob.
- Bob en choisit un, le crypte avec k_2 et l'envoie à Alice.
- Alice le décrypte avec k_1 et le renvoie à Bob.
- Bob le décrypte avec k_2 et lit le résultat.
- Alice communique k_1 à Bob qui fait de même avec k_2 .

Signer sans savoir ce qu'on signe

- On suppose avoir un procédé qui permet de signer un document sans savoir ce que l'on signe

Exemple : On a une fonction b qui brouille un message, la fonction s qui le signe, et s et b commutent (lorsqu'on signe un message brouillé, et qu'on débrouille le résultat, on obtient le message original signé).

Immunité diplomatique

- Un pays veut donner une fausse identité et l'immunité diplomatique à un espion X .
- X ne veut pas que le pays connaisse sa nouvelle fausse identité.
- Le pays veut être sûr qu'il n'est pas en train de signer n'importe quoi

Immunité diplomatique - Protocole

- X écrit 200 fois "je, soussigné le pays, donne l'immunité diplomatique à monsieur Machin" pour 200 valeurs de Machin différentes.
- X brouille les 200 messages.
- Le pays en choisit 199 parmi les 200 et demande à X de les débrouiller, qui le fait.
- Le pays vérifie que les 199 messages lui conviennent, et signe le dernier sans regarder
- X débrouille le dernier message

Argent

- Alice veut acheter un député, sans que quiconque s'en rende compte.
- Alice écrit 200 fois : “je, soussigné la banque, donne 135000 euros”.
- Alice brouille les 200 messages.
- Alice va voir sa banque, qui choisit 199 messages, demande à Alice de les débrouiller et les vérifie.
- La banque signe le dernier message, et supprime 135000 euros du compte en banque d’Alice.
- Alice débrouille le dernier message, et s’en va dépenser ses 135000 euros.

Voter

- Les gens veulent voter.
- On veut s’assurer que tout le monde ne vote qu’une fois
- Le vote doit rester anonyme

Protocole

- Chaque voteur signe son vote, puis l’encrypte avec la clé publique du centre de vote.

Problèmes ?

Protocole

- Chaque voteur génère deux messages M_1 et M_2 , et les brouille puis les signe
- Le centre de vote reçoit les deux messages, vérifie que la personne qui les a envoyés a le droit de voter, n’a pas envoyé des messages avant, et les signe
- Le voteur débrouille les deux messages et en envoie un, crypté avec la clé du centre de vote.

Zero-Knowledge

- Alice veut montrer à Bob qu’elle sait résoudre un problème précis, sans donner la solution du problème à Bob.
- Exemple : Alice veut montrer qu’elle sait comment aller de n’importe quel point d’un graphe à n’importe quel autre point en seulement 10 arêtes.

Protocole

- Alice prend le graphe, et permute aléatoirement les sommets (mais retient la permutation)
- Alice donne le nouveau graphe à Bob
- Bob demande à Alice
 - Soit de lui prouver que ce graphe est bien une permutation du graphe d’origine
 - Soit, étant donné deux sommets quelconques, comment aller de l’un à l’autre en moins de 10 arêtes.
- Dans les deux cas, Alice répond
- On recommence jusqu’à ce que Bob soit convaincu.