CRYPTOGRAPHIE (1)

INTRODUCTION

AU PROGRAMME

- Fondamentaux d’un crypto-système
- Types de crypto-systèmes :
  - cryptographie symétrique
  - cryptographie asymétrique
  - hachage (*hashing*)

CRYPTOGRAPHIE

- *Cryptos* + *graphia* : « écrits cachés »
- **Chiffrer** : transformer un message pour **cacher** son contenu
- **Déchiffrer** : processus inverse (*nécessite la clé de déchiffement*)
- **Décrypter** : idem mais *sans avoir la clé*
- ~~Crypter~~ : on ne peut pas chiffrer *sans une clé*
- **Message** : *n'importe quoi* dans sa forme non chiffrée (un fichier)

CLÉ (*KEY*)

- **Clé** : paramètre des algorithmes de cryptographie
  - ⇒ clé différente produit message chiffré différent
- Une fois les données chiffrées, elles sont protégées (confidentialité, contrôle d'intégrité)
- Seulement **du moment que la clé est correctement protégée**
- La gestion des clés devient l'élément primordial par lequel la protection est assurée

ALGORITHME DE CRYPTOGRAPHIE

- **Entrées** :
  - Message original (Mo) et clé (K) = données binaires
- **Sortie** :
  - Message chiffré (Mc) = données binaires
- **Chiffrement** : opération sur les entrées qui produit la sortie
  - addition : 3 + 5 = 8
  - chiffrement : **Mo + K = Mc**

CHIFFREMENT DE CÉSAR

- Rotation de chaque lettre du message original en fonction de la clé
- Exemple de clé : 3
  - A ⇒ D ; B ⇒ E ; C ⇒ F ; …

CHIFFREMENT DE CÉSAR - EXEMPLE

LEA AIME ALLAN

OHD DLPH DOODQ

CHIFFREMENT DE VIGENÈRE

- Plus élaboré :
  - la clé est un mot plutôt qu'un simple nombre
  - chaque lettre représente une rotation (ex: B = 2)
  - à chaque lettre du Mo correspond une rotation dans la clé

CHIFFREMENT DE VIGENÈRE - EXEMPLE

Ici, clé = TARTE (répétée autant de fois que nécessaire)

```
Mo : LEA AIME ALLAN

clé  : TAR TETA RTETA

Mc : FFS UNGF SFQUO
```

PRINCIPE DE KERCKHOFF

- « L'adversaire connaît le système »
- Les algorithmes de cryptographie sont **publics**
  - y compris les plus élaborés, utilisés tous les jours dans le monde
- **La sécurité repose sur la force et la non-divulgation de la *clé***

FORCE DES CLÉS

- Les clés peuvent être de taille quelconque
- Toute clé est cassable par **force brute** *en théorie*
- MAIS : la force de la crypto augmente très vite avec la taille de la clé
- Ex. : clés de 128 bits vs. 40 bits
  - protection des milliards de fois supérieure
- Aujourd'hui : crypto considérée sécurisée avec **clés 256 bits**

PROTECTION DES CLÉS

1. **Où est la clé ?**
2. **Comment est-elle protégée ?**
3. **Qui y a accès ?**

Un **serveur de clés** répond à ces trois questions.

LE CHALLENGE DE LA CRYPTOGRAPHIE

1. **Protéger les données stockées**
2. **Protéger les données en transit**
3. **Protéger les clés**

Échec sur l’un des trois ⇒ *Game Over*

CRYPTO-SYSTÈMES

- Assurer **confidentialité**
  - Algorithmes symétriques (DES, 3DES, IDEA, ARS, RC4, RC5)
- Assurer **authentification**
  - Algorithmes asymétriques (RSA, E1 Gamel, ECC)
- Assurer **intégrité**
  - Algorithmes de hachage (MD5, SHA, RIPEMD, HMAC)
- Assurer **non-répudiation**
  - Signatures numériques (algos asymétriques + hachage)

TECHNIQUES DE CHIFFREMENT SYMÉTRIQUES

- Substitution
  - XOR
  - Rotation
  - Substitution aléatoire
- Permutation
- Techniques hybrides des précédentes

XOR

- XOR = **OU EXCLUSIF**
- Opération binaire : `A XOR B` vaut 1 si A vaut 1 *ou bien* B vaut 1 (et pas les deux)
- Autrement dit : si `A == B` alors `A XOR B` vaut 0 (sinon 1)
- Algorithme de chiffrement/déchiffrement naturel :
  - `Mo XOR K = Mc`
  - `Mc XOR K = Mo`

ROTATION

- Substitution caractère par caractère
- Rotation de l'alphabet de N caractères (N est la clé)
- On appelle ROT-N l'algorithme de clé N
  - `ABCDE` ⇒ `DEFGH` (ici N = 3 ROT-3)
  - `MESSAGE` ⇒ `UMAAOM` (ROT-8)
- Chiffrement de César = ROT-3
- Usenet utilise ROT-13

SUBSTITUTION ALÉATOIRE

- Substitution caractère par caractère
- Chaque caractère est associé à un caractère aléatoire
  - `A ⇒ X ; B ⇒ T ; C ⇒ M ; D ⇒ P ;` ...
  - `BABA ⇒ TXTX`

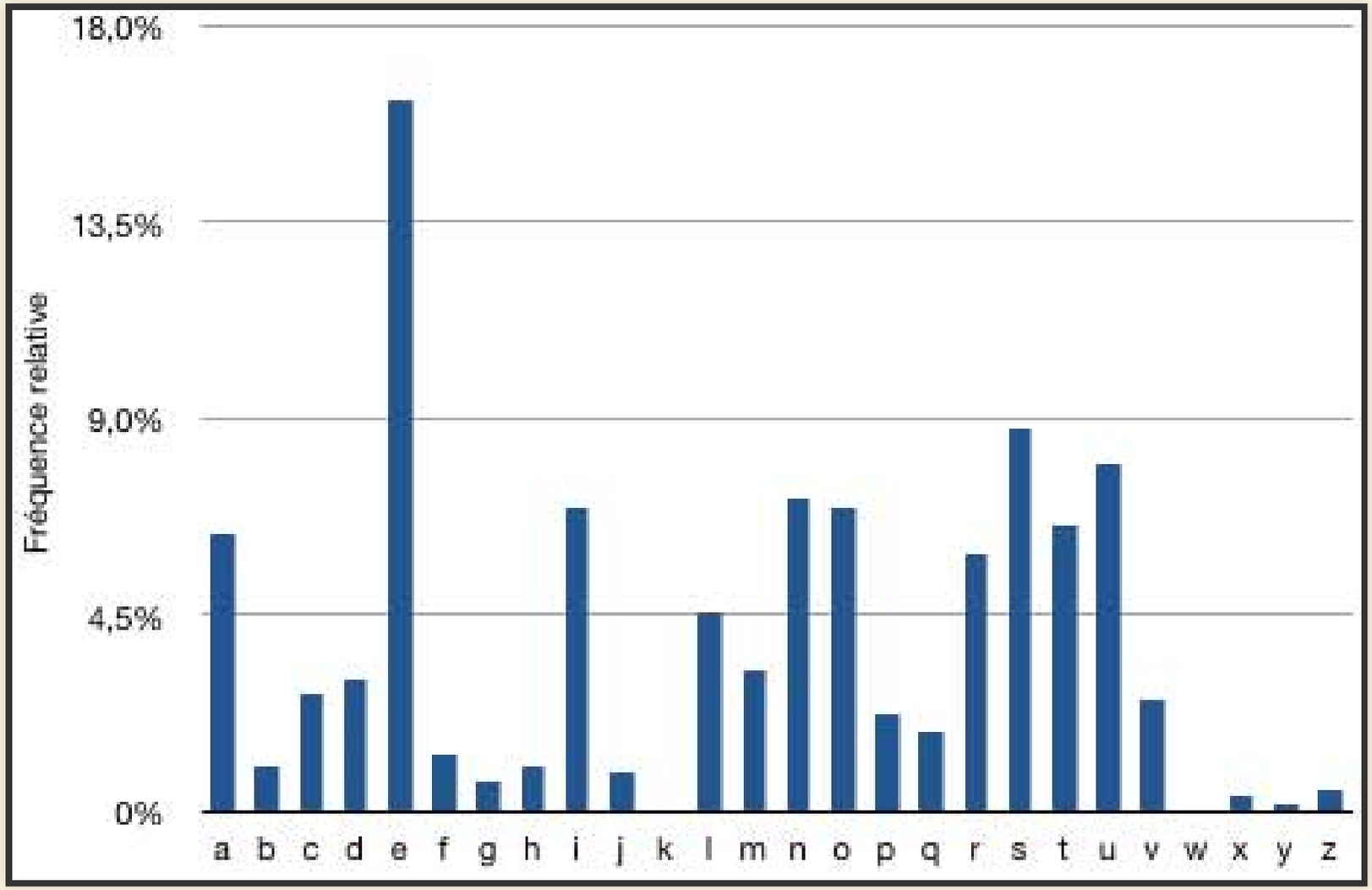
PERMUTATION

- On garde les mêmes lettres exactement, mais on change leur position dans le message
  - `MESSAGE ⇒ ASGMESE`

ANALYSE DE FRÉQUENCES DE CARACTÈRES

Individuellement, toutes ces techniques de substitution (XOR, Rotation, Substitution aléatoire, permutation) sont facilement cassables par **analyse de fréquences de caractères**.

ANALYSE DE FRÉQUENCES DE CARACTÈRES



TECHNIQUES HYBRIDES

- Les bons algorithmes de **chiffrement symétrique** utilisent simplement un panaché des techniques précédentes, en les répétant un grand nombre de fois
- Comme chacune de ces étapes est réversible, on peut déchiffrer en appliquant exactement la même séquence *en sens inverse*
- Ce sont ces **techniques hybrides** qui permettent de rendre le message chiffré très difficile à analyser, si la clé est suffisamment forte

RETOUR SUR LES CATÉGORIES DE CRYPTO-SYSTÈMES

- Algorithmes symétriques
  - 1 clé (clé secrète)
- Algorithmes asymétriques
  - 2 clés (clé publique, clé privée)
- Algorithmes de hachage
  - transformation en sens unique (non-réversible)
  - données (taille quelconque) ⇒ *hash* (petite taille fixe)

CRYPTOGRAPHIE SYMÉTRIQUE

CLÉ UNIQUE

- **Une seule clé** : la clé secrète
  - utilisée pour **chiffrer** le message original
  - et pour **déchiffrer** le message chiffré
- (Très) **rapide**
- Utilisée pour assurer la **confidentialité**
- (Gros) problème : gestion de la clé entre Alice et Bob
  - Requiert un **canal de distribution sécurisé de la clé**

SÉCURISATION DE L'ÉCHANGE DE LA CLÉ

- Trois possibilités :
  - **clé pré-partagée** (échange **en présence**, par exemple pour configuration VPN site-à-site)
  - utilisation du **chiffrement asymétrique** juste pour l'échange de clé
  - échange de clé **Diffie-Hellman** (non étudié ici)
- Pourquoi ne pas utiliser ces canaux sécurisés pour faire transiter directement le message ?
  - ⇒ pas pratique, lent, inefficace

CRYPTOGRAPHIE ASYMÉTRIQUE

DEUX CLÉS AU LIEU D'UNE SEULE

- **Paire de clés** pour une entité :
  - **clé publique** accessible à tout le monde (**Kpe**)
  - **clé privée** connue uniquement de l'entité (**Kve**)

PRINCIPE

Tout ce qui est chiffré avec l'une des deux clés ne peut être déchiffré qu'avec l'autre clé

APPLICATIONS

- (Très) **lente**
- Donc impraticable pour les larges volumes de données
- Deux applications principales :
  1. Canal sécurisé pour l'**échange de clé symétrique**
  2. **Signatures numériques** (authentification/non-répudiation)

APPLICATION 1 - ÉCHANGE DE CLÉ DE CHIFFREMENT

- Alice veut partager sa clé symétrique **`Ks`** avec Bob
- Alice chiffre la clé **`Ks`** avec la clé publique **`Kpb`** de Bob
- Elle obtient le message chiffré **`Mc`**
- Bob reçoit **`Mc`** et le déchiffre avec sa clé privée **`Kvb`** (et récupère **`Ks`**)
- **Bob peut maintenant échanger avec Alice par cryptographie symétrique en utilisant la clé unique et commune `Ks`**
- Technique utilisée aussi par TLS (SSH, HTTPS…), email, chiffrement de disque dur…

APPLICATION 2 - AUTHENTIFICATION

- Alice chiffre **`Mo`** avec sa clé privée **`Kva`** (obtient **`Mc`**)
- Alice envoie **`Mc`** à Bob
- Bob utilise la clé publique **`Kpa`** d'Alice pour déchiffrer **`Mc`**
- Si déchiffrement OK, on **sait** que le message vient bien d'Alice (seule à avoir pu chiffrer ce message avec **`Kva`**)
  - *(pour peu que l'on ait un moyen de certifier que Kpa est la vraie clé publique d'Alice)*

AUTHENTIFICATION - PROBLÈME

- **Rien n’empêche ici Eve de récupérer le message et de le déchiffrer !**
  - On peut juste prouver que le message *vient bien d’Alice*
- Cette technique, seule, n’est pas là pour assurer la confidentialité
  - Mais couplée avec le hachage, elle est la base de la **signature numérique**

CRYPTO ASYM - POINTS CRUCIAUX

1. **Canal sécurisé** (Alice ⇒ Bob)
    - Alice chiffre avec **Kpb**
    - Bob déchiffre avec **Kvb**
2. **Authentification** (Alice est-elle bien émettrice ?)
    - Alice chiffre avec **Kva**
    - Bob déchiffre avec **Kpa**

HACHAGE (*HASHING*)

LE HACHAGE N'EST PAS DU CHIFFREMENT !

- Le hachage d'un fichier permet d'obtenir une **empreinte** de celui-ci
  - ⇒ l'empreinte peut être recalculée et certifie ainsi qu'il s'agit exactement du même fichier non altéré

PRINCIPE DU HACHAGE

- **Pas de clé du tout** (pas du chiffrement)
- Transformation en sens unique ⇒ **non-réversible**
- Message (taille quelconque) passe par une *fonction de hachage*
- On obtient le **condensat** (empreinte, *hash*) du message (petite taille fixe)
- Utilisation principale : **intégrité** (si le ***hash* est identique** au départ et à l'arrivée, le **message n'a pas été modifié**)

TAILLE DU HASH ET ALGORITHMES

- Comme pour la taille des clés, la taille du *hash* résultant est important pour la sécurité du hachage
- Contrairement au chiffrement, la taille du message va en général être beaucoup plus grande que le *hash*
- Cela induit forcément la présence de **collisions**
- **Plus la taille du *hash* est grande, moins on a de collisions**
- Algorithmes : MD2, MD4, MD5, RIPEMD, SHA-1, SHA-2, HMAC

COLLISIONS

- Si deux messages différents produisent le **même *hash*** ⇒ **collision**
- **Inévitable** par définition du hachage
- MAIS **risque acceptable** (pas grave si c'est une coïncidence)
- CAR si la fonction de hachage est bonne, elle garantit deux choses qui rendent le concept sécurisé

GARANTIES D'UN BON HACHAGE

1. **Deux messages proches ne seront pas en collision**
    - la moindre modification du message entraînera un condensat complètement différent
2. Il est **impossible de prédire**, par force brute, quand une **collision** va arriver
    - on ne pourra pas forger un message qui produit le même condensat

INTÉGRITÉ

- Alice hache **Mo**, elle obtient un condensat **H**
- Alice envoie à Bob **Mo + H**
- Bob hache le message reçu et obtient **Hb**
- Bob compare **Hb** au condensat reçu
- Si les condensats correspondent, l'intégrité de **Mo** a pu être vérifiée ?
- NON : Mallory peut intercepter **Mo + H** et les remplacer par **Mm + Hm** qu'elle aura elle-même construits

SOLUTION : SIGNATURE NUMÉRIQUE

- **Signature numérique** : permet de garantir la provenance d'un message
- Garantit donc l'**authenticité** et la **non-répudiation**
- Basée sur **chiffrement asymétrique + hachage**
- Le message est « signé » en chiffrant son condensat avec la clé privée de l'émetteur

SIGNATURE NUMÉRIQUE

- Alice hache **`Mo`**, elle obtient son condensat **`H`**
- Alice chiffre **`H`** avec **`Kva`**, elle obtient **`sign`**
- Alice envoie à Bob **`Mo + sign`**
- Bob hache le message reçu et obtient **`Hb1`**
- Bob déchiffre **`sign`** avec **`Kpa`** et obtient **`Hb2`**
- Bob compare **`Hb1`** et **`Hb2`**
- Cette fois, si les deux *hash* correspondent, Bob est **certain que le message vient bien d'Alice**

CRYPTOGRAPHIE - RÉSUMÉ

- Assurer **confidentialité**
  - Algorithmes symétriques (DES, 3DES, IDEA, ARS, RC4, RC5)
- Assurer **authentification**
  - Algorithmes asymétriques (RSA, E1 Gamel, ECC)
- Assurer **intégrité**
  - Algorithmes de hachage (MD5, SHA, RIPEMD, HMAC)
- Assurer **non-répudiation**
  - Signatures numériques (algos asymétriques + hachage)