UE Parcours Recherche : "Certification d'un algorithme de chiffrement"

Dhia ZNAIDI
Encadrant : Hervé GRALL

IMT Atlantique

22/06/2022

Plan de l'exposé

- Enigma est une machine électromécanique servant au chiffrement et au déchiffrement de l'information.
- Utilisée principalement par l'Allemagne nazie pendant la Seconde Guerre mondiale.

Le chiffrement

Définition

Le chiffrement consiste à convertir les données afin que seules les personnes pourvues d'une clé secrète ou d'un mot de passe soient en mesure de les lire.

Le chiffrement des données

- Une manière efficace pour garantir la sécurité des données transmises d'une entité à une autre.
- Il est aujourd'hui utilisé dans plusieurs domaines de la société :
  - Le protocole HTTPS (Hyper Text Transfer Protocol Secure)
  - Les applications de messageries comme WhatsApp ou Telegram

Le chiffrement des données

Qualités d'un système de chiffrement

- Confidentialité
- Intégrité des données
- Authentification
- Non-répudiation qui se décompose en trois :
  - non-répudiation d'origine
  - non-répudiation de réception
  - non-répudiation de transmission

Chiffrement

Types de chiffrement

On distingue deux types de chiffrement :

- **Le chiffrement asymétrique** : deux clés sont utilisées : une clé publique (peut être partagée avec n'importe qui) et une clé privée (doit impérativement être protégée).
- **Le chiffrement symétrique** : la même clé est utilisée pour le chiffrement et le déchiffrement.

$\Longrightarrow$ Ces deux types diffèrent dans la façon dont les données sont déchiffrées.

Masque jetable ou chiffrement de Vernam

- Algorithme de cryptographie inventé par Gilbert Vernam en 1917
- Perfectionné par Joseph Mauborgne $\Longrightarrow$ la notion de clé aléatoire.
- Caractéristiques :
  - Simple,
  - Facile
  - Rapide, tant pour le codage que pour le décodage
  - Théoriquement impossible à casser

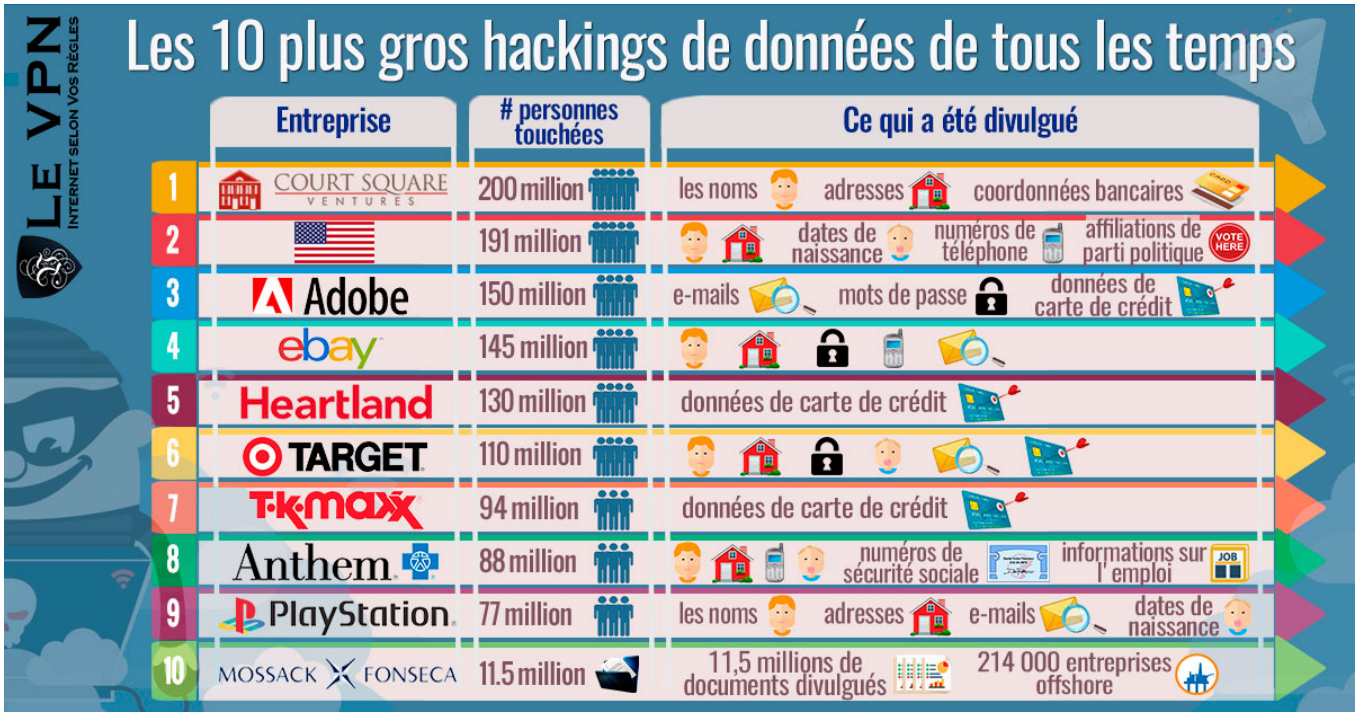
Masque jetable ou chiffrement de Vernam

Principe

- Combiner le message en clair avec une clé de chiffrement
- Caractéristiques de la clé :
  - une suite de caractères au moins aussi longue que le message à chiffrer,
  - Choix aléatoire des caractères de la clé
  - La clé n'est utilisée qu'**une seule fois**

Le chiffrement des données

Les menaces



Source : https ://www.le-vpn.com/fr/10-plus-grands-hacking-de-donnees-de-temps/

FIGURE – Les dix des plus grands vols de données de tous les temps.

Problématique

Concevoir et assurer la fiabilité d'un algorithme de chiffrement majeur demeure une tâche délicate.

- Comment assurer la correction d'un algorithme de chiffrement ?
- Le chiffrement est-il bien fait ? et si c'est bien fait est-il bien codé ?

Coq

De quoi s'agit-il ? Programmer & Démontrer

- L'assistant à la démonstration Coq est un outil complet réunissant un langage de programmation & de démonstration au sein d'un environnement de développement moderne pour l'interaction.

Coq

De quoi s'agit-il ? Programmer & Démontrer

- Coq est un *assistant à la preuve*, il permet de :
  - vérifier des preuves formelles
  - aider à l'élaboration des preuves
  - générer automatiquement des preuves
- La correspondance de Curry-Howard est profondément intégrée à Coq
  - les formules $\iff$ les types
  - les preuves $\iff$ les programmes

Coq

De quoi s'agit-il ? Programmer & Démontrer

- On peut donc dire que dans ce formalisme, toutes les preuves sont des programmes.
- Mais, dans ces programmes, on peut distinguer des parties purement logiques et des parties vraiment informatives.

Exemple : dans une proposition existentielle « $\exists x, P(x)$ », $P(x)$: x est premier

- on peut être intéressé par la manière dont x est construit
- Pour $P(x)$, on a juste besoin de savoir qu'il est vrai, mais la plupart du temps on ne veut pas vraiment savoir pourquoi.

Coq

Certification des programmes

Les assistants de preuves permettent de prouver formellement :

- des théorèmes mathématiques usuels
- des propriétés sur des programmes et des systèmes informatiques en tout genre.

Coq

Certification des programmes

Plusieurs approches existent à la certification de programmes :

- chercher à prouver ***a posteriori*** des propriétés sur des programmes existants en analysant
  - le code source
  - le code machine
  - n'importe quelle version intermédiaire que la compilation peut produire.
- exprimer le comportement souhaité dans un formalisme plus abstrait que les langages de programmation usuels, puis en dériver automatiquement des programmes satisfaisant ***a priori*** les propriétés voulues.

Coq Proof Assistant

Certification des programmes

Voici quelques exemples typiques de ce qui a été réalisé avec l'assistant de preuve Coq :

- Leroy et al. ont développé dans Coq un compilateur optimisant certifié pour C.
- Barthe et al. ont utilisé Coq pour développer Certicrypt, un environnement de preuves formelles pour la cryptographie numérique.

Coq permet également l'extraction de programmes vers des langages comme Ocaml.

Schéma général du chiffrement

$$msg\_clair \longrightarrow \left.\begin{array}{c} msg\_clair \\ + \\ cle\_chiffrement \end{array}\right\} \longrightarrow msg\_chiffre$$

$$msg\_chiffre \longrightarrow \left.\begin{array}{c} msg\_chiffre \\ + \\ cle\_dechiffrement \\ \uparrow \\ cle\_chiffrement \end{array}\right\} \longrightarrow msg\_clair$$

Outils de travail mis en oeuvre

Notion de groupe

- Un ensemble G et une loi de composition interne $\star$ vérifiant 3 propriétés :
  - Associativité :

$$\forall x, y, z \in G \quad x \star (y \star z) = (x \star y) \star z$$

  - Élément neutre e :

$$\forall x \in G \quad e \star x = x \star e = x$$

  - L'inverse :

$$\forall x \in G, \exists y \in G \quad x \star y = y \star x = e$$

Outils de travail mis en oeuvre

Quotient

- Définition de Nat (l'ensemble des entiers naturels) dans Coq :
  `Inductive nat : Set := O : nat | S : nat -> nat`
- Quotienter cet ensemble par la relation d'équivalence $\mathcal{R}$:

$$\forall a, b \; a \mathcal{R} b \iff a \equiv b \quad (mod T A)$$

Outils de travail mis en oeuvre

Quotient

- Structure algébrique de '`Modulo TA`' :
  - algèbre sur la signature (0, S) :
    - une constante correspondant à '0',
    - une fonction correspondant au successeur 'S'.
  - groupe additif commutatif :
    - loi additive associative et commutative,
    - élément neutre '0',
    - opposé (de valeur égale au reste du successeur du complémentaire).

Outils de travail mis en oeuvre

Correction de l'algorithme de chiffrement

- C'est ici où intervient Coq, cette preuve en fait trouve son essence dans les notions abstraites de l'algèbre générale :

```
Lemma neutralite : forall (m c : Modulo TA) ,
        modulo_somme (modulo_somme m c)
          (modulo_opposeSomme c) = m.
```

- Démonstration théorique :

$$\begin{aligned} & (m+c) + (-c) \\ = \; & m + (c+(-c)) \\ = \; & m + 0 \\ = \; & m \end{aligned}$$

Outils de travail mis en oeuvre

Vecteurs ou listes ?

- Deux choix étaient envisageables : listes ou vecteurs.
- L'utilisation des listes posait plusieurs problèmes de logique :
  - Il faut toujours s'assurer que les deux listes ont une même longueur :

```
P : length l = length l1
```

  - le principal défaut : on ne peut pas définir un produit cartésien de listes :

```
Lemma combine_length : forall (l:list A)(l':list B), length
      (combine l l') = min (length l) (length l').
```

$\Longrightarrow$ Choix de travailler avec les vecteurs

Outils de travail mis en oeuvre

Les fonctions adoptées

- f : `(Modulo TA)*(Modulo TA)` $\rightarrow$ `modulo TA`
  $\Longrightarrow$ C'est la fonction de chiffrement :

$$f(msg, c) = msg\_c$$

- g : `(Modulo TA)*(Modulo TA)` $\rightarrow$ `modulo TA`
  $\Longrightarrow$ C'est la fonction de déchiffrement.

$$g(msg\_c, c) = msg$$

Outils de travail mis en oeuvre

Les foncteurs

- Un foncteur $F : \mathcal{C} \to \mathcal{D}$ est la donnée d'une fonction qui :
  - à tout objet X de $\mathcal{C}$ associe un objet $F(X)$ de $\mathcal{D}$
  - à tout morphisme $f : X \to Y$ de $\mathcal{C}$, associe un morphisme $F(f) : F(X) \to F(Y)$ de $\mathcal{D}$

$\Longrightarrow$ Dans notre cas , le foncteur choisi est `Vecteur _ n`

Outils de travail mis en oeuvre

Implémtation du foncteur `Vecteur _ n`

- `chiffrement : (Vecteur (Modulo TA) n) * (Vecteur (Modulo TA) n) -> (Vecteur (Modulo TA) n)`
  $\Longrightarrow$ Une implémentation de f pour les vecteurs.
- `dechiffrement : (Vecteur (Modulo TA) n) * (Vecteur (Modulo TA) n) -> (Vecteur (Modulo TA) n)`
  $\Longrightarrow$ Une implémentation de g pour les vecteurs.

Outils de travail mis en oeuvre

Correction de l'algorithme de chiffrement

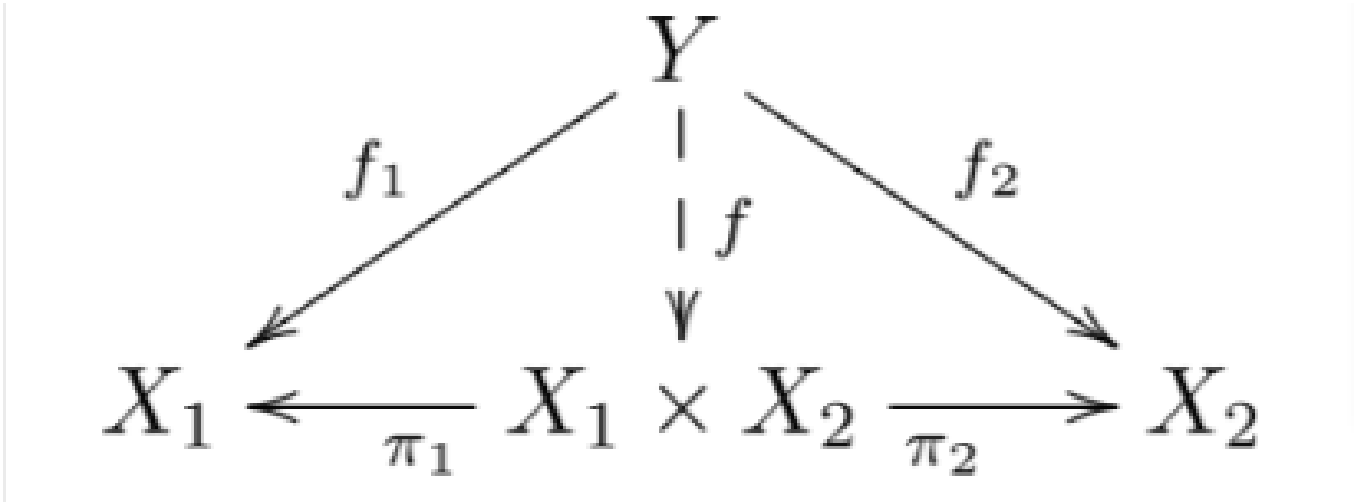
- C'est ici où intervient Coq :

```
Lemma correction : forall (n : nat) (msg: Vecteur
    (Modulo TA) n) (c : Vecteur (Modulo TA) n),
      (masquage ;; demasquage) (msg,c) = msg.
```

$\Longrightarrow$ Un problème : $\not{f \circ g}$?

Outils de travail mis en oeuvre

Produit dans le cadre de la théorie des catégories



Outils de travail mis en oeuvre

Produit dans le cadre de la théorie des catégories

- $(f, \pi_2)(msg, c) = (msg\_c, c)$
- $g(msg\_c, c) = msg$

$$\Longrightarrow \; g \circ (f, \pi_2)(msg, c) = msg$$

Tests sur des exemples

- Les propriétés qu'on cherche à tester :
  - `Theorem testGenerique_verification : forall n c m, ((length m = n) /\ (length c = n)) -> testGenerique n c m.`
  - `Theorem testErreurGenerique_verification : forall n c m, ((length m = n) /\ (length c = n)) -> testErreurGenerique n c m.`

Tests sur des exemples

- `Definition testGenerique (n : nat)(c m : string) : Prop := frontend_composition_sequence (@chiffrement TA n) (@dechiffrement TA n) (m, c) = Some m.`
- `Definition testErreurGenerique (n : nat)(c m : string) : Prop := chiffrer n c m = None.`

Tests sur des exemples

```
Example test5 :
testGenerique 10
"0123456789" "voiliers f".
compute.
reflexivity.
Qed.
```

```
Example tesE5 :
testErreurGenerique 10
"0123456789" "voiliers fg".
compute.
reflexivity.
Qed.
```