Sécurité

Julien BERNARD

Université de Franche-Comté – UFR Sciences et Techniques
Licence Informatique – 3è année

2017 – 2018

Première partie

Introduction

Plan de ce cours

Plan

Votre enseignant
Qui suis-je ?

Qui suis-je ?

Julien BERNARD, Maître de Conférence (enseignant-chercheur)
julien.bernard@univ-fcomte.fr, Bureau 426C

Enseignement

- Responsable du semestre 1 (Starter) de la licence Informatique
- Cours : Bases de la programmation (L1), Publication scientifique (L1), Algorithmique (L2), Sécurité (L3), Théorie des Langages (L3), Analyse Syntaxique (L3)

Recherche

Optimisation dans les réseaux de capteurs

Plan

UE Sécurité
Organisation

Équipe pédagogique

- Julien BERNARD (julien.bernard@univ-fcomte.fr) : Cours, TD
- Lydia YATAGHENE (lydia.yataghene@univ-fcomte.fr) : TP
- Vincent PRÊTRE (vincent.pretre@gmail.com) : Cours, TD, TP
- Régis TISSOT (regis.tissot@rtissot.fr) : TD, TP

Volume

- Cours : 12 x 1h30
- TD : 12 x 1h30
- TP : 12 x 1h30

Évaluation

- deux devoirs surveillés
- un challenge TP
- un devoir TD

UE Sécurité
Comment ça marche ?

Mode d'emploi

1. Prenez des notes ! Posez des questions ! N'attendez pas du tout-cuit !
2. Comprendre plutôt qu'apprendre.
3. Le but de cette UE n'est pas d'avoir une note !

Niveau d'importance des transparents

	trivial	pour votre culture
$\star$	intéressant	pour votre compréhension
$\star\star$	important	pour votre savoir
$\star\star\star$	vital	pour votre survie

Note : les contrôles portent sur *tous* les transparents !

UE Sécurité
Contenu pédagogique

Objectif

Se familiariser avec les problématiques de sécurité

- Codes alphabétiques
- Cryptographie symétrique
- Cryptographie asymétrique
- Fonctions de hachage cryptographiques
- Protocoles cryptographiques
- Sécurité des systèmes d'information
- Politique de sécurité

UE Sécurité
Bibliographie

Jean-Guillaume Dumas, Jean-Louis Roch, Éric Tannier, Sébastien Varette.

Théorie des codes. Compression, cryptage, correction.

1è édition, 2008, Dunod

Bruce Schneier

Cryptographie appliquée. Algorithmes, protocoles et codes source en C.

2è édition, 2001, Vuibert

Damien Vergnaud

Exercices et problèmes de cryptographie.

1è édition, 2012, Dunod

Plan

Qu'est-ce que la sécurité ?

De très nombreux sujets...

De très nombreux sujets...

- Crytographie (symétrique, asymétrique, ...)
- Cryptanalyse (différentielle, linéaire, ...)
- Protocoles de communication
- Défaillance de programmes (buffer overflow, ...)
- Logiciels malveillants (virus, vers, troyens, ...)
- Attaques locale et distantes (DoS, ...)
- Détection d'intrusion
- Politique de sécurité (droits d'accès, ...)

Qu'est-ce que la sécurité ?

. . . utilisant un large spectre de connaissances

. . . utilisant un large spectre de connaissances

- Mathématiques (arithmétique, probabilités, corps finis, . . .)
- Calculabilité, complexité
- Modélisation (logique, automates, . . .)
- Réseaux
- Architecture (assembleur)
- Système

Objectifs de la sécurité

★★★

Quelques objectifs élémentaires

- Confidentialité
  - Informations non-accessibles
  - Communications non-espionnables
- Intégrité
  - Pas de modification des informations
- Authentification
  - Chaque action a un responsable identifiable
- Non-répudiation
  - Un accord ne peut être remis en cause par une des parties
- Disponibilité
  - Le système est toujours utilisable

Folklore

Cas d'étude classique

Alice envoie un message à Bob...

Les figures classiques de la cryptologie

- Alice envoie un message à Bob
- Bob reçoit un message d'Alice
- Charly communique avec Alice et Bob si besoin d'un troisième
- Eve, l'espionne, scrute les messages mais ne peut pas les modifier
- Mallory, la malicieuse, scrute les messages et peut les modifier
- Oscar, l'opposant, comme Mallory
- Trent, le tiers de confiance

Exemple de code secret ⋆

Exemple de code secret

Idée : Une table à double entrée unité du texte en clair / unité correspondante du code secret (clef).

Le sous marin est attendu à	Jean
10 heures	est là
20 heures	n'est pas là

Constats

- Taille de la clé nécessairement supérieure ou égale à celle du message.
- Confidentialité de l'information fondée sur le fait que seules les personnes autorisées connaisse la clef.
- Type d'information échangée très limité

Plan

La stéganographie
Définition et historique

★★

Définition (Stéganographie)

La stéganographie («steganos» couvert, «graphein» écriture) est l'art de dissimuler, c'est-à-dire d'utiliser des techniques pour cacher un message.

Historique

- Vers 600 avant J.C. : Nabuchodonosor utilisait des crânes rasés
- Vers 480 avant J.C. : Dematarus (Grec) isolé en Perse utilise des tablettes de cires pour prévenir son pays des projets de Xerxes (Perse)
- 1er siècle : Apparition de l'encre invisible

La stéganographie
Exemple et inconvénient

★★

Un exemple classique : dissimulation dans une image

- Les couleurs sont généralement codés sur 3 octets (RGB)
- Changer les 2 bits de poids faibles de chaque octet change très peu la couleur, modification «invisible» à l'œil nu
- Exemple :
  $\{00101011, 11001101, 11110011\} \oplus 011001 \rightarrow \{00101\mathbf{01}, 11001\mathbf{10}, 11110\mathbf{01}\}$

Problème majeur

- La sécurité repose sur le secret de la méthode de chiffrement

$\rightarrow$ Si la méthode est découverte, le secret l'est également

- Stéganographie = «Enterrer son argent dans son jardin»

Un exemple littéraire

Correspondance entre George Sand et Alfred de Musset

De George Sand à Alfred de Musset

Je suis très émue de vous dire que j'ai bien compris, l'autre jour, que vous avez toujours une envie folle de me faire danser. Je garde un souvenir de votre baiser et je voudrais que ce soit là une preuve que je puisse être aimée par vous. Je suis prête à vous montrer mon affection toute désinterressée et sans calcul. Si vous voulez me voir ainsi. . .

Un exemple littéraire

Correspondance entre George Sand et Alfred de Musset

Lequel lui répond

Quand je vous jure, hélas, un éternel hommage
Voulez-vous qu'un instant je change de langage
Que ne puis-je, avec vous, goûter le vrai bonheur
Je vous aime, ô ma belle, et ma plume en délire
Couche sur le papier ce que je n'ose dire
Avec soin, de mes vers, lisez le premier mot
Vous saurez quel remède apporter à mes maux.

Un exemple littéraire

Correspondance entre George Sand et Alfred de Musset

Et enfin, Georges Sand conclut par cette missive

Cette grande faveur que votre ardeur réclame
Nuit peut-être à l'honneur mais répond à ma flamme.

Un exemple littéraire

Regardons de plus près. . .

De Georges Sand à Alfred de Musset

Je suis très émue de vous dire que j'ai bien compris, l'autre jour, que vous avez toujours une envie folle de me faire danser. Je garde un souvenir de votre baiser et je voudrais que ce soit là une preuve que je puisse être aimée par vous. Je suis prête à vous montrer mon affection toute désinterressée et sans calcul. Si vous voulez me voir ainsi. . .

Un exemple littéraire

Regardons de plus près. . .

De Georges Sand à Alfred de Musset

Je suis très émue de vous dire que j'ai bien compris, l'autre jour, que vous avez toujours une envie folle de me faire danser. Je garde un souvenir de votre baiser et je voudrais que ce soit là une preuve que je puisse être aimée par vous. Je suis prête à vous montrer mon affection toute désinterressée et sans calcul. Si vous voulez me voir ainsi. . .

Un exemple littéraire

Regardons de plus près. . .

Lequel lui répond

Quand je vous jure, hélas, un éternel hommage
Voulez-vous qu'un instant je change de langage
Que ne puis-je, avec vous, goûter le vrai bonheur
Je vous aime, ô ma belle, et ma plume en délire
Couche sur le papier ce que je n'ose dire
Avec soin, de mes vers, lisez le premier mot
Vous saurez quel remède apporter à mes maux.

Un exemple littéraire

Regardons de plus près. . .

Lequel lui répond

Quand je vous jure, hélas, un éternel hommage
Voulez-vous qu'un instant je change de langage
Que ne puis-je, avec vous, goûter le vrai bonheur
Je vous aime, ô ma belle, et ma plume en délire
Couche sur le papier ce que je n'ose dire
Avec soin, de mes vers, lisez le premier mot
Vous saurez quel remède apporter à mes maux.

Un exemple littéraire

Regardons de plus près. . .

Et enfin, Georges Sand conclut par cette missive

Cette grande faveur que votre ardeur réclame
Nuit peut-être à l'honneur mais répond à ma flamme.

Un exemple littéraire

Regardons de plus près. . .

Et enfin, Georges Sand conclut par cette missive

Cette grande faveur que votre ardeur réclame
Nuit peut-être à l'honneur mais répond à ma flamme.

La cryptographie
Définition et historique

Définition (Cryptographie)

La cryptographie («kryptos» caché, «graphein» écriture) est l'ensemble des techniques (algorithmes et protocoles) permettant de protéger un message.

Historique : un art ancien et une science nouvelle

- 5 siècle avant J.C., Spartiates : Utilisation de scytales
- Chiffre de César : Décalage alphabétique
- 1580 : Chiffre par substitution de symboles – Marie Stuart
- 1586 : Traité des chiffres – Chiffre de Vigenère
- 1930 : Enigma
- Fin 20ème : Apparition du chiffrement à clé publique – RSA en 1977

La cryptographie
De nos jours. . .

La cryptographie moderne

- La cryptographie symétrique
- La cryptographie asymétrique
- Les fonctions de hachage cryptographiques
- Les protocoles cryptographiques

Avantages

- **Utilisation d'algorithmes associés à des clefs**
- La sécurité repose sur le secret de la clef
- Cryptographie = «Mettre son argent dans un coffre-fort»

La cryptographie
Formalisation

★★★

Notations

Le message en clair M est transformé en message chiffré C (*ciphertext*) via la fonction de chiffrement E (*encrypt*) et la clef K (*key*).

$$C = E(M, K)$$

Le message chiffré C est transformé en message en clair M via la fonction de déchiffrement D (*decrypt*) et la clef K^{-1} (qui peut être égale à K).

$$M = D(C, K^{-1})$$

La cryptanalyse
Définition

Définition (Cryptanalyse)

La cryptanalyse (analyse du secret) est la science qui consiste à tenter de trouver un message ayant été protégé par une technique de cryptographie.

Des techniques diverses et variées

- Analyse fréquentielle
- Attaque par force brute
- Cryptanalyse linéaire
- Cryptanalyse différentielle
- ...

La cryptanalyse
Classes d'attaques

★★★

Classes d'attaques

- Texte chiffré connu (*ciphertext-only*) :
  Oscar connaît uniquement un message chiffré
- Texte clair connu (*known-plaintext*) :
  Oscar connaît un message clair et le message chiffré associé
- Texte clair choisi (*chosen-plaintext*) :
  Oscar choisit un message clair et obtient le message chiffré associé
- Texte chiffré choisi (*chosen-ciphertext*) :
  Oscar choisit un message chiffré et obtient le message clair associé

La cryptologie
Définition

Définition (Cryptologie)

La cryptologie (science du secret) est l'ensemble constitué de la cryptographie et de la cryptanalyse.

Chiffre, chiffrement, déchiffrement ⋆⋆

Définition (Chiffre)

Un chiffre est un code secret permettant de transcrire un texte clair (libellé) en un texte chiffré (cryptogramme), généralement à l'aide d'une clef de chiffrement.

Définition (Chiffrement)

Le chiffrement est l'opération qui réalise cette transcription. Le verbe associé est chiffrer.

Définition (Déchiffrement)

Le déchiffrement est l'opération inverse du chiffrement. Le verbe associé est déchiffrer.

Décryptage

Définition (Décryptage)

Le décryptage est l'opération qui consiste à trouver un texte clair à partir d'un texte chiffré, sans la clef de chiffrement. Le verbe associé est décrypter.

Attention !

Cryptage et *crypter* n'existent pas !

Plan

Principe de Kerckhoffs

Principe de Kerckhoffs

1. Une information codée ne doit en aucun cas pouvoir être déchiffrée sans la connaissance de sa clé.
2. Les interlocuteurs ne doivent pas subir de dégâts au cas où le système de codage serait dévoilé.
3. La clé doit être simple et modifiable à souhait.
4. Les cryptogrammes doivent être transportables, c'est-à-dire télégraphiables.
5. L'appareil de codage et les documents doivent être transportables.
6. Le système doit être simple d'utilisation.
7. Le système de chiffrage doit être au préalable examiné par des experts.

La cryptographie militaire dans *Journal des sciences militaires*, Auguste Kerckhoffs, 1883

Maxime de Shannon

Maxime de Shannon

«L'adversaire connaît le système.»

Dit autrement...

- La sécurité repose sur le secret de la clef et non sur le secret de l'algorithme
- Le déchiffrement sans la clef doit être impossible
- Trouver la clef à partir du texte clair et du texte chiffré est impossible

Confusion et diffusion

Confusion et diffusion

- **Confusion** : la relation entre la clef de chiffrement et le texte chiffré doit être la plus complexe possible
- **Diffusion** : la redondance statistique dans un texte en clair est dissipée dans les statistiques du texte chiffré. Un bit changé en entrée doit changer chaque bit en sortie avec une probabilité $\frac{1}{2}$ $\rightarrow$ effet d'avalanche

Communication Theory of Secrecy Systems, Claude Shannon, 1949

Ordres de grandeur ⋆

Vitesse des ordinateurs

Les ordinateurs modernes ($1\text{GHz} = 10^9\text{Hz} \approx 2^{30}\text{Hz}$) calculent à la vitesse de la lumière ($3.10^8 m.s^{-1}$).

Temps (secondes)

2^{10}	15 minutes	
2^{20}	10 jours	
2^{30}	30 ans	
2^{40}	30000 ans	découverte du feu
2^{50}	30M années	extinction des dinosaures [1]
2^{57}	4Md années	âge de la Terre
2^{59}	13Md années	âge de l'Univers

1. 65M années

Deuxième partie

Codes alphabétiques

Plan de ce cours

Plan

Chiffrement par décalage

Principe

★★

Chiffrement par décalage

Le chiffrement monoalphabétique par décalage utilise une permutation circulaire de l'ensemble des lettres de l'alphabet.

- On numérote les lettres de 0 (pour A) à 25 (pour Z)
- On choisit une clef $K \in \{1, \ldots, 25\}$
- Pour chaque lettre $\sigma \in \{0, \ldots, 25\}$:

$$E(\sigma, K) = (\sigma + K) \text{ modulo } 26$$

- Pour chaque lettre $\bar{\sigma} \in \{0, \ldots, 25\}$:

$$D(\bar{\sigma}, K) = (\bar{\sigma} - K) \text{ modulo } 26$$

Chiffrement par décalage

Exemple de chiffrement

Exemple (Chiffre du message ZORRO avec $K = 3$)

- $E(Z, 3) = E(25, 3) = (25 + 3) \text{ modulo } 26 = 28 \text{ modulo } 26 = 2 = C$
- $E(O, 3) = E(14, 3) = (14 + 3) \text{ modulo } 26 = 17 \text{ modulo } 26 = 17 = R$
- $E(R, 3) = E(17, 3) = (17 + 3) \text{ modulo } 26 = 20 \text{ modulo } 26 = 20 = U$

$$\text{ZORRO} \rightarrow \text{CRUUR}$$

Chiffrement par décalage
Inconvénients

★★

Inconvénients du chiffrement par décalage

- Nombre de clefs petit : 25 (taille de l'alphabet - 1)
- Attaque par force brute possible
- Attaque par analyse fréquentielle

Chiffrement par décalage

Cas particuliers

Le chiffre de César

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

- $K = 3$
- Utilisé par Jules César dans ses correspondances militaires

ROT13

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M

- $K = 13$
- Utilisé dans UNIX à une époque
- Clef identique pour le chiffrement et le déchiffrement

Plan

Chiffrement par substitution

Principe

★★

Chiffrement par substitution

Le chiffrement monoalphabétique par substitution utilise une permutation arbitraire de l'ensemble des lettres de l'alphabet

- Étend le principe du chiffrement par décalage
- Nombre clefs possibles : $26! \approx 2^{88}$

Inconvénient

- La clef est la table de permutation dont la taille est assez importante
  $\rightarrow$ Méthode de construction de clef

Chiffrement par substitution

Construction de clef

Comment construire une table de substitution à partir d'un mot-clef ?

1. Établir une clef (ex : SECURITE)
2. Supprimer les lettres doubles (ex : SECURITE $\rightarrow$ SECURIT)
3. Faire correspondre les premières lettres de l'alphabet aux premières lettres de la clef
4. Compléter la table en reprenant l'alphabet à partir de :
   - la dernière lettre de la clef nettoyée, ou
   - la première lettre de l'alphabet

   et en supprimant les lettres présentes dans la clef

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
S	E	C	U	R	I	T	V	W	X	Y	Z	A	B	D	F	G	H	J	K	L	M	N	O	P	Q

Chiffrement par substitution

Cryptanalyse

★★

Faiblesses du chiffrement par substitution

- Toutes les occurrences d'une même lettre ou groupe de lettres sont chiffrées de la même manière

→ Analyse fréquentielle (E < A < S < I < N < T < ...)

Lettre	Fréquence	Lettre	Fréquence	Lettre	Fréquence
A	8.40%	J	0.31%	S	8.08%
B	1.06%	K	0.05%	T	7.07%
C	3.03%	L	6.01%	U	5.74%
D	4.18%	M	2.96%	V	1.32%
E	17.26%	N	7.13%	W	0.04%
F	1.12%	O	5.26%	X	0.45%
G	1.27%	P	3.01%	Y	0.30%
H	0.92%	Q	0.99%	Z	0.12%
I	7.34%	R	6.55%		

Plan

Les chiffrements polyalphabétiques

Raison d'être

★★

Les chiffrements polyalphabétiques

But : masquer autant que possible la structure du texte clair, c'est-à-dire :

- répétitions des caractères
- répétitions des groupes de caractères

$\rightarrow$ Une approche est d'appliquer un décalage fonction de la position du caractère dans le texte

Chiffre de Vigenère
Principe

★★

Le chiffre de Vigenère

Le chiffre de Vigenère utilise une substitution alphabétique multiple par décalage

- On choisit un mot-clef
  Exemple : DECEPTION
- Le rang de chaque lettre de la clef définit un décalage à appliquer
  Exemple : 3 4 2 4 15 19 8 14 13 puis à nouveau 3 4 . . .

Exemple

Chiffrer le texte NOUS SOMMES DECOUVERTS grâce à la clef DECEPTION avec le carré de Vigenère

Chiffre de Vigenère

Utilisation du carré de Vigenère

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
...																									
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
...																									
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
...																									
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
...																									

Exemple

Texte clair	N	O	U	S		S	O	M	M	E	S		D	E	C	O	U	V	E	R	T	S
Clef	D	E	C	E		P	T	I	O	N	D		E	C	E	P	T	I	O	N	D	E
Texte chiffré	Q	S	W	W		H	H	U	A	R	V		H	G	G	D	N	D	S	E	W	W

Chiffre de Vigenère
Formalisation

Chiffre de Vigenère

Soit M un message de taille m et K une clef de taille k. Le message chiffré C est obtenu de la manière suivante :

$$\forall i \in \{0, \ldots, m-1\}, C[i] = (M[i] + K[i \text{ modulo } k]) \text{ modulo } 26$$

Chiffre de Vigenère

Retour sur l'exemple

Exemple

	N	O	U	S		S	O	M	M	E	S		D	E	C	O	U	V	E	R	T	S
M	13	14	20	18		18	14	12	12	4	18		3	4	2	14	20	21	4	17	19	18
	D	E	C	E		P	T	I	O	N	D		E	C	E	P	T	I	O	N	D	E
K	3	4	2	4		15	19	8	14	13	3		4	2	4	15	19	8	14	13	3	4
$+$	16	18	22	22		33	33	20	26	17	21		7	6	6	29	39	29	18	30	22	22
	Q	S	W	W		H	H	U	A	R	V		H	G	G	D	N	D	S	E	W	W
C	16	18	22	22		7	7	20	0	17	21		7	6	6	3	13	3	18	4	22	22

Indice de coïncidence

Un outil pour la cryptanalyse

Indice de coïncidence (1920, William Friedman)

- Probabilité que deux lettres choisies aléatoirement dans un texte soient identiques
- Propriétés :
  - Caractéristique d'une langue (≈ 0.075 pour le français)
  - Invariant par substitution
  - Permet de déterminer si le chiffrement est monoalphabétique ou polyalphabétique
- Soit n le nombre de lettre d'un texte et n_σ le nombre de fois où la lettre σ apparaît, alors l'indice de coïncidence est :

$$IC = \sum_{\sigma=A}^{Z} \frac{n_\sigma(n_\sigma - 1)}{n(n-1)}$$

Chiffre de Vigenère
Cryptanalyse

★★

Cryptanalyse du chiffre de Vigenère

1. Trouver la taille de la clef par des suppositions successives avec le calcul l'indice de coïncidence
   - Avec une taille 1, on calcule l'indice de coïncidence sur la totalité du texte
   - Avec une taille k, on établit k messages $M_1, \ldots, M_k$ à partir de l'original M où on prend une lettre sur k en commençant à la i^{e}
   - On calcule l'indice de coïncidence sur chaque message M_i et si on est proche de l'indice recherché, on a probablement trouvé la taille de la clef
2. Faire une analyse statistique pour déterminer chacune des lettres de la clef

Plan

Limites des code alphabétiques

Limites

- L'alphabet est limité : uniquement 26 lettres
- Oblige à avoir des fonctions de conversion alphabet/décimal

Solution

→ Chiffrer au niveau des bits

- S'applique à n'importe quel type de données
- Pas de conversion

Chiffre XOR
Principe

Rappel sur XOR

XOR est l'opération bit-à-bit *ou exclusif*, notée $\oplus$. Elle est associative, commutative. On peut la voir comme une addition modulo 2.

$\oplus$	0	1
0	0	1
1	1	0

Le chiffre XOR

- Choisir une clef K binaire de taille k
- Découper le texte M en blocs M_i de taille k
- Appliquer XOR entre chaque bloc M_i et la clef K : $C_i = M_i \oplus K$

Le chiffre XOR
Exemple

Exemple

$M = 00001111$ et $K = 0110$

M	0000	1111
K	0110	0110
C	0110	1001

Problèmes

- Et si la taille du message n'est pas un multiple de la taille de la clef ?
- Comment déchiffrer ?

Remplissage (*padding*) ⋆⋆⋆

Définition (Remplissage)

Le remplissage (ou *padding*) permet d'aligner la taille du message sur un multiple de la taille de la clef en remplissant la fin du message par des bits qui ne font pas partie du message. Le remplissage est nécessaire pour tous les algorithmes qui travaillent par bloc (cryptographie symétrique, fonctions de hachages cryptographiques). Il existe plusieurs méthodes de remplissage, parmi lesquels :

- Ajouter des 0
- Ajouter un 1 puis des 0, suivi de la taille du message
- Ajouter une suite alternant 0 et 1
- Ajouter des bits aléatoires

Chiffre XOR
Déchiffrement

Déchiffrement

Appliquer l'opération $\oplus$ entre chaque bloc C_i et la clef K :

$$C_i \oplus K = (M_i \oplus K) \oplus K = M_i \oplus (K \oplus K) = M_i$$

Démonstration

Vient des propriétés de $\oplus$:

1. Tout message est son propre opposé : $\forall x \in \{0,1\}, x \oplus x = 0$
2. 0 est l'élément neutre de $\oplus$: $\forall x \in \{0,1\}, x \oplus 0 = x$

Plan

Chiffre de Vernam

Principe

★★★

Chiffre de Vernam

Le chiffre de Vernam, appelé aussi masque jetable (*one-time pad*), est le seul chiffre inconditionnellement sûr. On dit aussi que c'est un chiffrement parfait.

- Soit un message M
- On choisit une clef K parfaitement aléatoire de la même taille que M
- Le message chiffré C est :

$$C = M \oplus K$$

- On jette la clef K

Chiffre de Vernam

Preuve de la perfection

Idée de la preuve

- Comme la clef est parfaitement aléatoire, toute clef est équiprobable, et donc elle n'apporte aucune information au message
- La relation entre le texte clair, le texte chiffré et la clef ne permet pas d'obtenir d'information
- On en déduit, que la connaissance du texte chiffré n'apporte aucune information sur le texte clair

Conséquence

Même une attaque par force brute ne permettra pas de trouver le texte original !

Chiffre de Vernam
Limites

Limites

- La clef doit être jetée
- La transmission de la clef doit être physique
- Il est très difficile d'engendrer une clef parfaitement aléatoire

Utilisations historiques

- Téléphone rouge
- Che Guevara et Fidel Castro

Troisième partie

Cryptographie symétrique

Plan de ce cours

Plan

Cryptographie symétrique
Principe

★★

Cryptographie symétrique

En *cryptographie symétrique* ou *cryptographie à clef secrète*, la clef K est partagée entre Alice et Bob. Elle est utilisée à la fois pour chiffrer et déchiffrer.

$$C = E_K(M), M = D_K(C)$$

Bestiaire

- Chiffres alphabétiques
  - Substitution (César, Vigenère)
  - Transposition
- Chiffres binaires
  - Chiffrement par flot (A5/1, RC4)
  - Chiffrement par bloc (DES, AES)

Plan

Retour sur le chiffre de Vernam ⋆

Chiffre de Vernam

Le chiffre de Vernam, appelé aussi masque jetable (*one-time pad*), est le seul chiffre inconditionnellement sûr. On dit aussi que c'est un chiffrement parfait.

- Soit un message M
- On choisit une clef K parfaitement aléatoire de la même taille que M
- Le message chiffré C est : $C = M \oplus K$
- On jette la clef K

Idée

L'idée du chiffrement par flot est de générer une suite aléatoire de bits (à partir d'une clef K) pour mimer le chiffre de Vernam.

Générateur aléatoire cryptographiquement sûr ⋆⋆

Générateur aléatoire cryptographiquement sûr

Un générateur aléatoire cryptographiquement sûr doit avoir deux propriétés :

- Aléatoire du point de vue statistique : il a l'air aléatoire, ses sorties réussissent tous les tests statistiques connus
- Aléatoire du point de vue cryptographique : il est imprévisible, il doit être difficile de prédire le prochain bit, étant donné l'algorithme connu et tous les bits déjà générés

Clef

Une graine (*seed*) est utilisée pour initialiser le générateur, elle sert de clef pour les chiffrement par flot

Générateur congruentiel linéaire ⋆⋆

Générateur congruentiel linéaire

Les nombres pseudo-aléatoire forment une suite :

$$u_{n+1} = a * u_n + c \mod m$$

où a est le multiplicateur, c est l'incrément et m est le module.

Période maximale

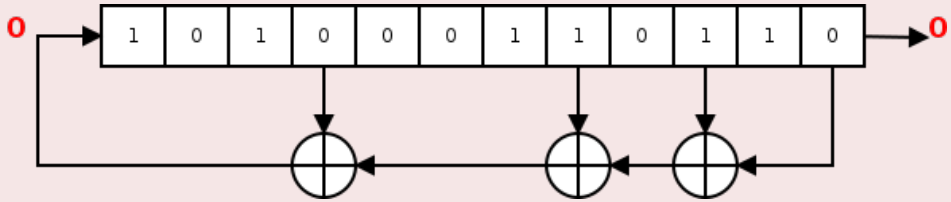
La période de ce générateur est maximale (égale à m) si et seulement si :

- c est premier avec m
- Pour chaque premier p divisant m, $(a-1)$ est un multiple de p
- $(a-1)$ est un multiple de 4 si m en est un

Registre à décalage rétroactif linéaire ⋆⋆

Registre à décalage rétroactif linéaire

Un registre à décalage rétroactif linéaire, ou *linear feedback shift register* (LFSR) est un registre de taille k dont certains bits sont combinés en un bit qui est réinséré dans le registre, les autres bits étant décalé.



On peut les voir comme des suites binaires (dans $\mathbb{Z}/2\mathbb{Z}$) :

$$u_{n+k} = \alpha_1 u_{n+k-1} + \ldots + \alpha_{k-1} u_{n+1} + \alpha_k u_n$$

La période est maximale $(2^k - 1)$ si et seulement si le polynôme $P(X) = 1 + \alpha_1 X + \ldots + \alpha_k X^k$ (appelé polynôme de rétroaction) est primitif dans l'anneau des polynômes à coefficient dans $\mathbb{Z}/2\mathbb{Z}$.

Limite des LFSR ⋆⋆⋆

Définition (Complexité linéaire)

Étant donné une suite binaire de longueur m, la complexité linéaire de la suite est définie comme étant la taille k du plus petit LFSR générant cette suite. Nécessairement, $k \leq m$.

Algorithme de Berlekamp-Massey

Si on arrive à observer L termes de la suite $u_t, \ldots, u_{t+L-1}$, avec L supérieur ou égal à deux fois la complexité linéraire des u_i, alors on peut retrouver le polynôme de rétroaction et l'initialisation du registre.

Conséquences pratiques

- Il est impossible d'utiliser un générateur congruentiel linéaire ou un LFSR pour générer des bits aléatoires sûrs.
- Pratiquement, on peut combiner plusieurs LFSR.

ISAAC

Un générateur aléatoire sûr cryptographiquement (?)

ISAAC

ISAAC (*Indirection, Shift, Accumulate, Add, and Count*) est un générateur aléatoire que son auteur, Bob Jenkins, prétend sûr cryptographiquement. Il en existe deux versions :

- Une version 32 bits qui a une période garantie minimale de 2^{40} et une période moyenne de 2^{8295}.
- Une version 64-bits qui a une période garantie minimale de 2^{72} et une période moyenne de 2^{16583}

Blum Blum Shub

Blum Blum Shub

- Proposée en 1986
- On choisit p et q deux grands nombres premiers congru à 3 modulo 4
- On calcule $n = pq$
- On considère la suite :

$$u_{n+1} = u_n^2 \text{ modulo } n$$

- On prend le bit de poids faible de u_n (en fait, on peut prendre les $\log(\log n)$ derniers bits)
- La sécurité repose sur la difficulté de la factorisation de n
- Très lent !

Plan

A5/1
Historique

Historique d'A5/1

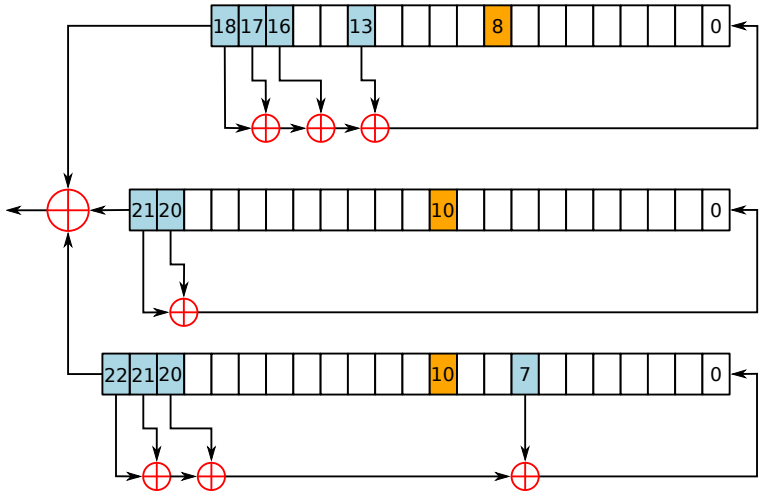
A5/1 est un algorithme de chiffrement par flot utilisé dans le GSM pour chiffrer les communications entre le mobile et l'antenne.

- Années 80 : Des tractations ont lieu à l'OTAN pour savoir si le chiffrement du futur GSM devrait être fort ou pas.
- 1987 : Spécifications d'A5/1 mais tenues secrètes
- 1994 : Fuite des spécifications
- 1997–2000 : Premières attaques à texte clair connu (Golic, puis Biryukov-Shamir-Wagner)
- 2003 : Première attaque à texte chiffré seulement (Barkan-Biham-Keller)
- 2007–2009 : Plusieurs projets visent à décrypter A5/1 en temps réel

A5/1
Description

Description d'A5/1

- 3 LFSR de taille 19, 22 et 23
- La sortie est combinée par un XOR
- Les registres sont décalés si leur bit du milieu est majoritaire
- Clef de 64 bits mais ramené à 54 bits dans GSM



Plan

RC4
Historique

Historique de RC4

- Inventé par Rivest en 1987
- Ron's Code
- Tenu secret, mais retrouvé par rétro-ingénierie : ARC4
- Utilisé dans WEP, WPA, TLS

RC4
Fonctionnement

Fonctionnement de RC4

- État : tableau T de 256 octets représentant une permutation
- Une phase d'initialisation, pour i de 0 à 255 ($j = 0$ au départ) :

$$j \leftarrow j + T[i] + K[i \text{ modulo } |K|] \text{ modulo } 256$$

  Et on échange $T[i]$ et $T[j]$
- La phase de génération ($i = 0$ et $j = 0$ au départ) :

$$\begin{cases} i \leftarrow i + 1 \\ j \leftarrow j + T[i] \text{ modulo } 256 \\ \text{Echanger}(T[i], T[j]) \\ K_i \leftarrow T[(T[i] + T[j]) \text{ modulo } 256] \end{cases}$$

Plan

Chiffrement par bloc

Principe

★★★

Chiffrement par bloc

- Le message M est découpé en bloc de taille égale M_i
- Une fonction E_K permet de chiffrer un bloc avec la clef K
- Une fonction D_K pour déchiffrer un bloc avec la clef K
- Il existe différents modes d'opération pour combiner la clef K, l'entrée et la sortie de E_K et D_K

Réseau de Feistel

Définition

★★★

Réseau de Feistel

On considère les éléments suivants :

- Les blocs sont partagées en deux sous-bloc R et L de taille b
- Un nombre n de rondes
- $K_0, \ldots, K_{n-1}$, n sous-clef de taille k déduit de la clef originale
- Une fonction f prenant en paramètre un bloc de taille b et un bloc de taille k et renvoyant un bloc de taille b

Alors, pour j de 1 à n, on a :

$$\begin{cases} L_{j+1} = R_j \\ R_{j+1} = L_j \oplus f(R_j, K_j) \end{cases}$$

Avec (L_0, R_0) le bloc en clair et (L_n, R_n) le bloc chiffré

Réseau de Feistel
Déchiffrement et propriétés

Déchiffrement à l'aide d'un réseau de Feistel

Pour j de n à 1, on a :

$$\begin{cases} R_j = L_{j+1} \\ L_j = R_{j+1} \oplus f(L_{j+1}, K_j) \end{cases}$$

Propriété d'un réseau de Feistel

- La sécurité repose sur f et le nombre de rondes
- Utilisé dans DES et d'autres algorithmes
- Vulnérable à la cryptanalyse linéraire et à la cryptanalyse différentielle

Plan

DES : Data Encryption Standard

Historique

Historique de DES

- Mai 1973 : le National Bureau of Standards (NBS, actuellement National Institute of Standards and Technology, NIST) demande la création d'un algorithme de chiffrement
- IBM propose Lucifer, créé en 1971
- La NSA retouche Lucifer qui devient alors DES
- Novembre 1976 : DES devient un standard
- Algorithme de chiffrement symétrique le plus utilisé (`crypt(3)`, banques, SSL)

Caractéristiques

- Utilise un réseau de Feistel à 16 rondes
- Utilise une clef K de 56 bits étendue à 64 avec des bits de parité

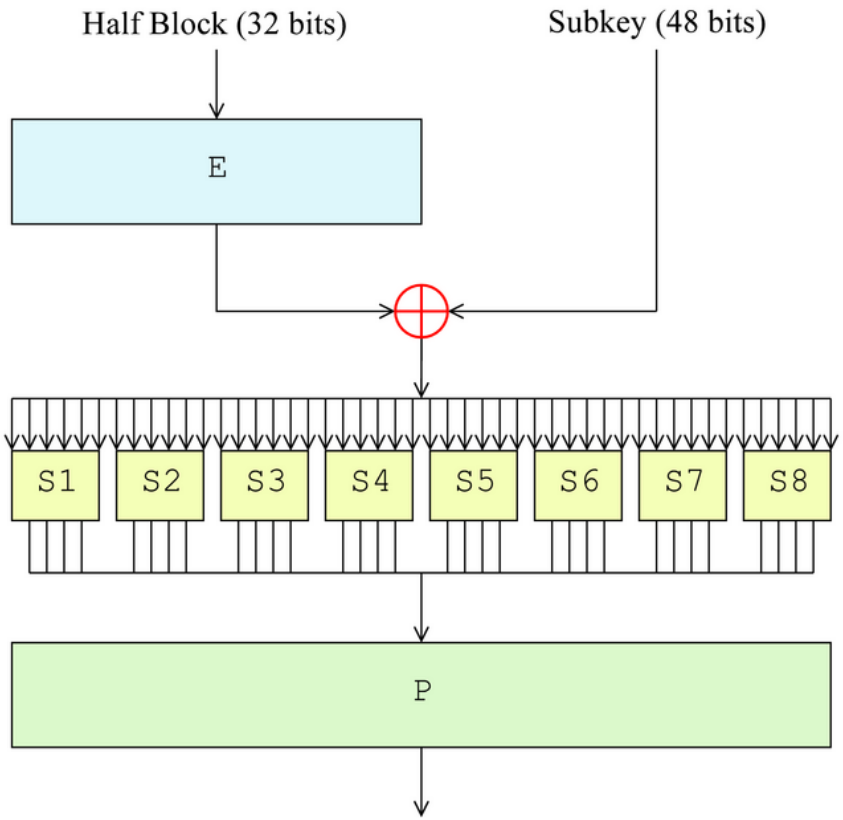
DES : Data Encryption Standard

Fonctionnement

★★

La fonction f de DES

- Permutation expansive
- Mélange avec la clef
- Substitution avec des S-Box
- Permutation



DES : Data Encryption Standard

Génération des clefs

★★

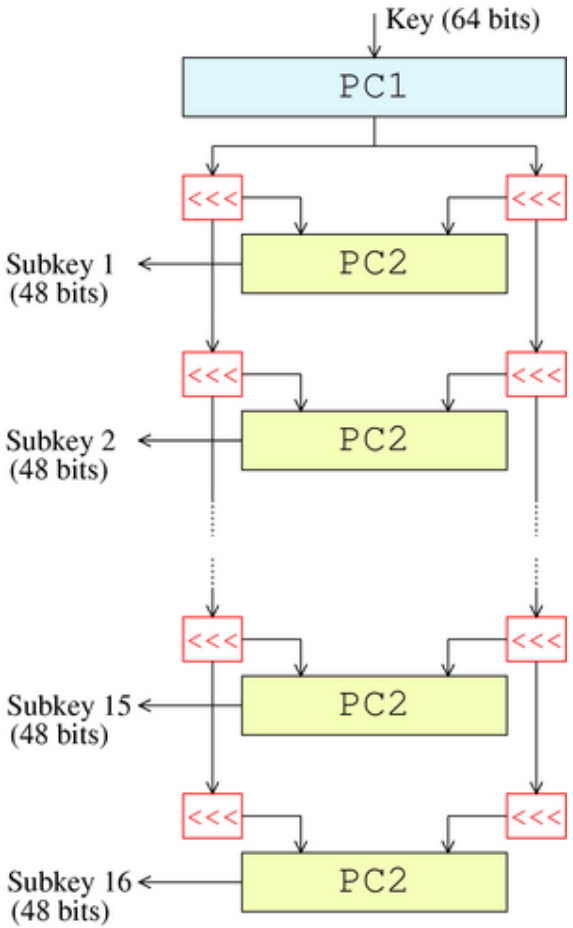
Génération des clefs de DES

Au début :

- Permutation compressive (PC-1)
- Découpe en deux moitiés

Puis à chaque ronde :

- Chaque moitié est décalée
- Permutation compressive (PC-2)
- Fournit une sous-clef de 48 bits



DES : Data Encryption Standard

Sécurité

★★

Sécurité de DES

- Les S-Box
  - La sécurité repose sur les S-Box (non-linéarité)
  - Mais choisie par la NSA !
  - 1990 : Cryptanalyse différentielle $\rightarrow$ les S-Box sont sûres !
- Attaque par force brute
  - La taille de la clef (56 bits) le permet !
  - Pratiquement, c'est la meilleure attaque
  - 1998 : EFF DES Cracker (2 jours, 250000$)
  - 2006 : COPACABANA (120 FPGA, 10000$)
  - 2008 : COPACABANA RIVYERA (125 FPGA, $<$ 1 jour)
- Apparition de nouveaux types d'attaques :
  - Cryptanalyse différentielle
  - Cryptanalyse linéaire

Cryptanalyse différentielle ⋆⋆

Cryptanalyse différentielle

- 1990 : Biham et Shamir notamment pour DES
- Attaque à textes clairs choisis
- Étude de $\Delta_Y = S(X) \oplus S(X \oplus \Delta_X)$
- On en déduit des bits de la clef avec une certaine probabilité
- En choisissant bien Δ_X, on peut faire mieux que la force brute
- Avec 2^{47} textes clair choisis, on casse DES !

Cryptanalyse linéaire ⋆⋆

Cryptanalyse linéaire

- 1993 : Matsui
- Attaque à texte clairs connus
- Faire une approximation linéaire de l'algorithme
- Plus on a de textes, meilleure est l'approximation
- Avec 2^{43} textes clairs connus, on casse DES !

Double-DES

Double-DES

- On choisit deux clefs K_1 et K_2 et on applique deux fois DES :

$$C = E_{K_2}(E_{K_1}(M))$$

- Attaque de type rencontre au milieu (*meet in the middle*)
  - On dispose de M_0 et C_0 une paire de texte
  - Pour toutes les clefs K, on calcule $E_{K_1}(M_0)$
  - Pour toutes les clefs K, on calcule $D_{K_2}(C_0)$
  - Dès qu'on trouve un des textes précalculés, c'est fini !
  - Complexité : 2^{57} !
- Équivalent à une clef de 57 bits (en fait $\approx$ 64 bits)
- À ne pas utiliser !

Triple-DES

Triple-DES

- 2 modes :
  - EEE : $C = E_{K_3}(E_{K_2}(E_{K_1}(M)))$ (≈ 120 bits)
  - EDE : $C = E_{K_3}(D_{K_2}(E_{K_1}(M)))$
    3 options pour les clefs :
    - 3 clefs indépendantes (≈ 112 bits)
    - $K_1 = K_3$, mieux de Double-DES (≈ 80 bits)
    - $K_1 = K_2 = K_3$, équivalent de DES
- Inconvénient : très lent !

Plan

AES : Advanced Encryption Standard

Historique

Historique d'AES

- 1997 : Appel à contribution du NIST pour remplacer DES
  - Taille de bloc de 128 bits
  - Clef de 128, 192 et 256 bits
  - Implémentable facilement en matériel
  - Résistant à la cryptanalyse différentielle
  - Résistant à la cryptanalyse linéaire
- 1999 : 5 finalistes (MARS, RC6, Rijndael, Serpent, et Twofish)
- 2001 : Rijndael, par Daemen et Rijmen, devient AES

AES : Advanced Encryption Standard

Fonctionnement

★★

Fonctionnement d'AES

- Réseau de substitution-permutation ($\neq$ réseau de Feistel)
- Plusieurs rondes
- Basé les corps finis et en particulier $\mathbb{F}_{256}$
- 4 opérations de base :
  - `SubBytes` : substitution non-linéaire basée sur l'inversion dans $\mathbb{F}_{256}$
  - `ShiftRows` : décalage des données vues dans une matrice 4×4
  - `MixColumns` : transformation linéaire, multiplication par le polynôme $3X^3 + X^2 + X + 2$ modulo $X^4 + 1$ dans $\mathbb{F}_{256}$
  - `AddRoundKey` : XOR entre les données et une sous-clef

AES : Advanced Encryption Standard

Cryptanalyse

Cryptanalyse d'AES

- À ce jour, la seule attaque disponible contre AES est la force brute
- La NSA le recommande pour les données classifiées TOP-SECRET
- Mais attaques par canaux auxiliaires

Plan

Modes d'opération

Modes d'opération

- 4 modes définis en 1981 pour DES : ECB, CBC, CFB, OFB
- Modes étendus à AES et un cinquième mode ajouté en 2001 : CTR
- À l'origine pour le chiffrement et l'authentification
- Utilisation d'un vecteur d'initialisation (*initialization vector*, IV)
  - Aléatoire (mais pas cryptographiquement sûr)
  - Envoyé avec le message chiffré
  - Permet d'utiliser la même clef avec le même texte en clair

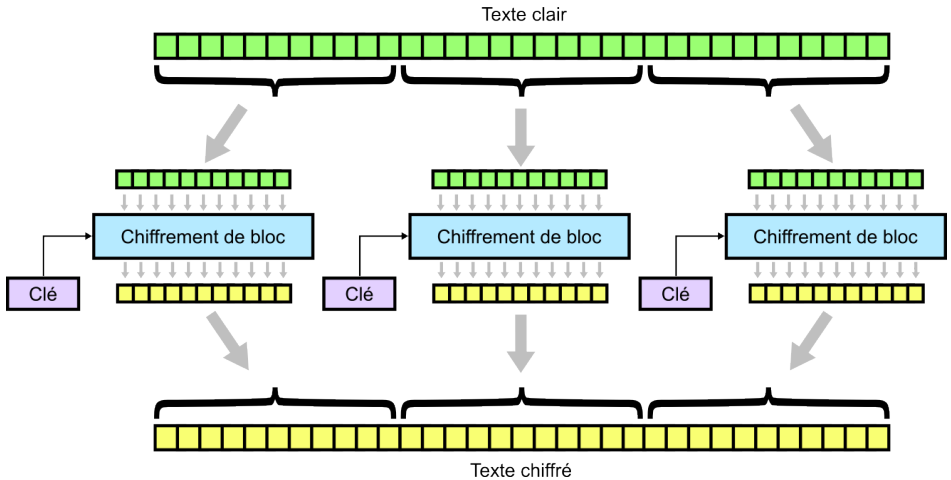
Electronic CodeBook (ECB)

ECB

- Chaque bloc est chiffré séparément

$$C_i = E_K(M_i), M_i = D_K(C_i)$$

- Avantage : on peut déchiffrer une partie seulement du texte chiffré
- Inconvénient : deux morceaux identiques de texte clair sont chiffrés de la même manière



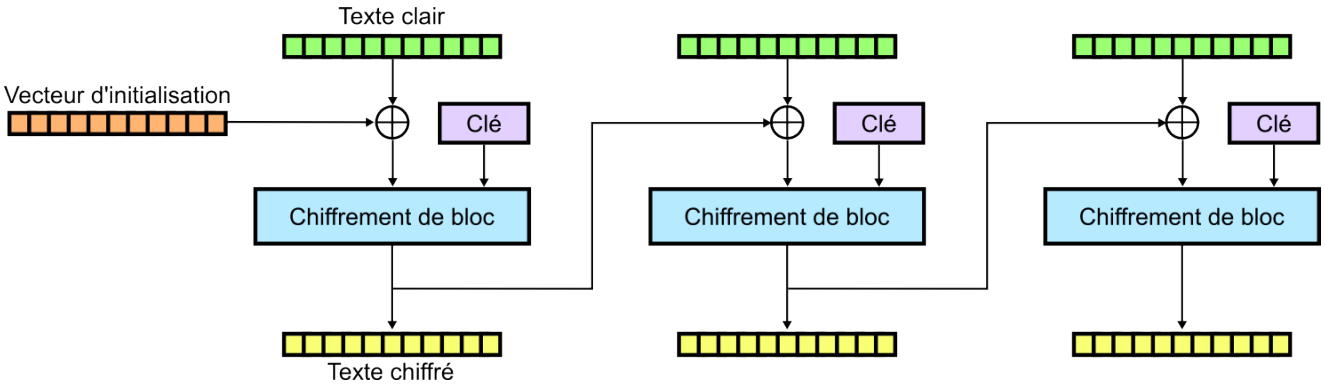
Cipher Block Chaining (CBC)

CBC

- Un bloc en clair est mélangé avec le blocs chiffré précédent

$$C_0 = IV, C_i = E_K(M_i \oplus C_{i-1}), M_i = D_K(C_i) \oplus C_{i-1}$$

- Avantage : meilleure dissimulation
- Inconvénient : le chiffrement est séquentiel (mais pas le déchiffrement)



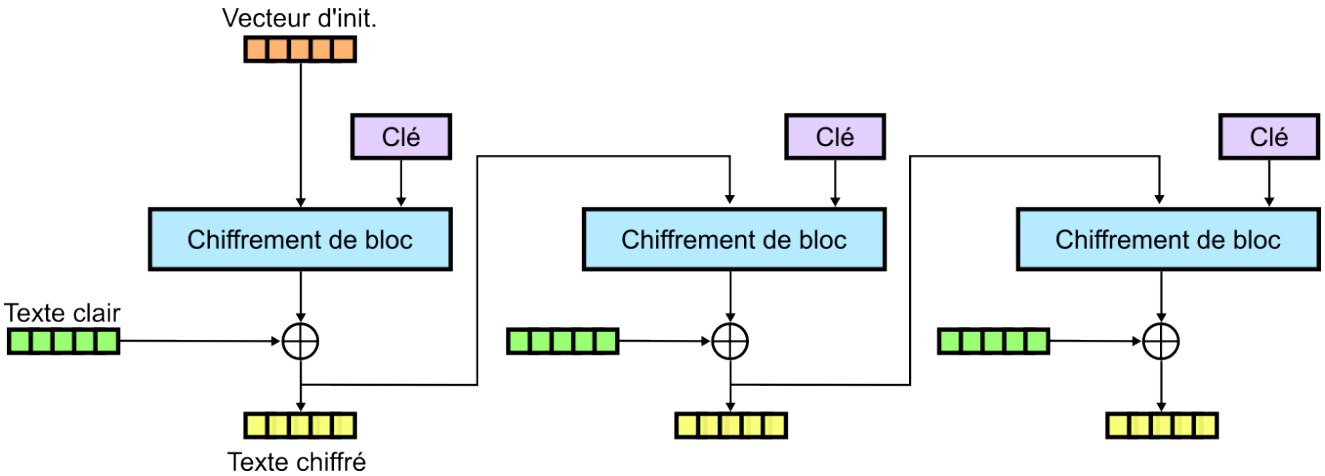
Cipher FeedBack (CFB)

CFB

- Similaire à un chiffrement par flux

$$C_0 = IV, C_i = E_K(C_{i-1}) \oplus M_i, M_i = E_K(C_{i-1}) \oplus C_i$$

- Avantage : ne nécessite que la fonction de chiffrement
- Inconvénient : vulnérable à des modifications contrôlées



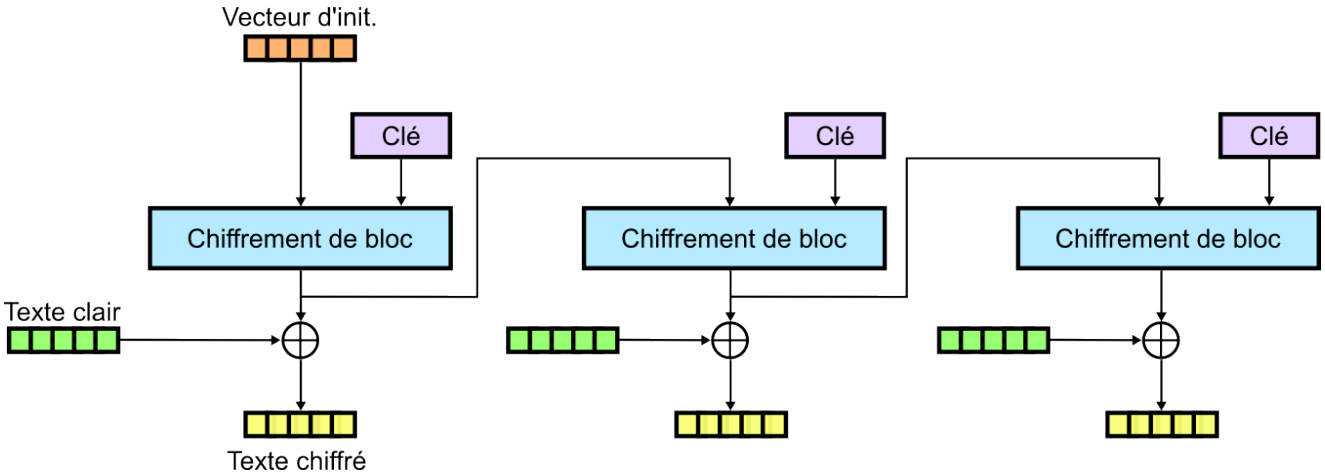
OutputFeedBack (OFB)

OFB

- Vrai chiffrement par flux

$$O_0 = IV, O_i = E_K(O_{i-1}), C_i = M_i \oplus O_i, C_i = M_i \oplus O_i$$

- Avantage : meilleure tolérance aux fautes
- Inconvénient : vulnérable à une attaque à texte clair connu



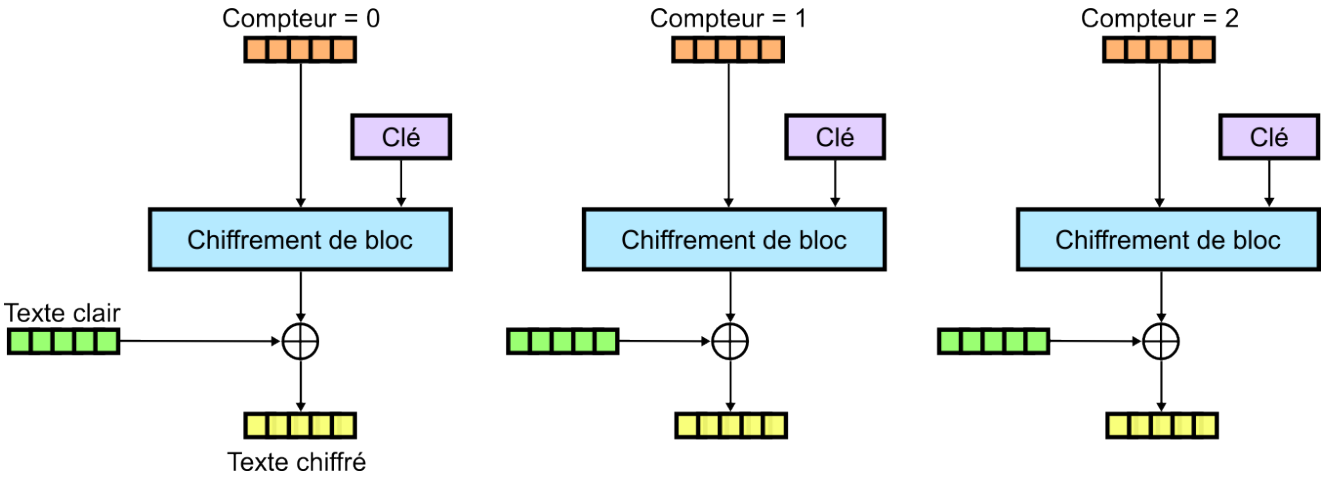
CounTeR (CTR)

CTR

- Vrai chiffrement par flux

$$O_i = E_K(i), C_i = M_i \oplus O_i, C_i = M_i \oplus O_i$$

- Avantage : précalculable, accès aléatoire



Quatrième partie

Cryptographie asymétrique

Plan de ce cours

Plan

Cryptographie asymétrique
Principe

★★

Cryptographie asymétrique

En *cryptographie asymétrique* ou *cryptographie à clef publique*, Alice a un couple de clef :

- K_A, sa clef publique qu'elle distribue
- K_A^{-1}, sa clef privée qu'elle conserve

Il existe deux utilisations principales de ces clefs :

- Le chiffrement
- La signature

Chiffrement

Chiffrement

- Bob veut envoyer un message M à Alice
- Il récupère sa clef publique K_A
- Il envoie $C = E_{K_A}(M)$ à Alice
- Alice utilise alors sa clef privée K_A^{-1}
- Elle calcule $D_{K_A^{-1}}(C) = M$

Propriété obtenue : confidentialité

Signature

Signature

- Alice veut signer un message M pour Bob
- Elle calcule $S = D_{K_A^{-1}}(M)$
- Elle envoie M et S à Bob
- Bob récupère la clef publique d'Alice K_A
- Il calcule $E_{K_A}(S)$ et vérifie que le résultat est égal à M

Propriété obtenue : authentification

Comparaison

Comparaison

- Cryptographie asymétrique
  - Une clef publique K_A
  - Une clef privée K_A^{-1}
  - ☺ Un couple de clef par acteur
  - ☹ Chiffrement lent
- Cryptographie symétrique
  - Une clef secrète K_{AB}
  - ☺ Chiffrement rapide
  - ☹ Une clef par couple d'acteur (C_n^2 clefs)

$\rightarrow$ On utilise les clefs publiques pour établir une clef secrète partagée

Plan

Historique

Historique de RSA

- 1977 : Rivest, Shamir, Adelman
- Basé sur la difficulté de la factorisation
- Utilise le théorème d'Euler

Théorème d'Euler

Théorème (Théorème d'Euler)

Soit n un entier naturel et a un entier premier avec n, alors

$$a^{\varphi(n)} \equiv 1 \mod n$$

où φ est la fonction indicatrice d'Euler.

Rappels sur l'indicatrice d'Euler

$\varphi(n)$ est le nombre d'entiers strictement positifs inférieurs ou égaux à n et premiers avec n. Quelques propriétés :

- Si $n = p^k$ avec p premier et $k \in \mathbb{N}^*$, $\varphi(n) = \left(1 - \frac{1}{p}\right) \times n$
- Si n_1 et n_2 sont premiers entre eux, $\varphi(n_1 \times n_2) = \varphi(n_1) \times \varphi(n_2)$
- Si $n = \prod_{i=1}^{k} p_i^{k_i}$, $\varphi(n) = n \times \prod_{i=1}^{k} \left(1 - \frac{1}{p_i}\right)$

Établissement des clefs ⋆⋆⋆

Établissement des clefs

1. On choisit deux grands nombres premiers p et q
2. On calcule $n = pq$, alors $\varphi(n) = (p-1)(q-1)$
3. On choisit $e < \varphi(n)$ et premier avec $\varphi(n)$
4. On calcule d tel que $ed \equiv 1 \mod \varphi(n)$

Alors, on a :

- $\langle e, n \rangle$ est la clef publique
- $\langle d, n \rangle$ est la clef privée

Chiffrement

Chiffrement

- On chiffre des blocs de k bits, avec $2^k < n \leq 2^k + 1$
- Chaque bloc M est vu comme un entier m écrit sous forme binaire, et donc $m < 2^k$
- m est chiffré de la manière suivante :

$$c \equiv m^e \mod n$$

Déchiffrement

★★★

Déchiffrement

- On déchiffre c de la manière suivante :

$$m \equiv c^d \mod n$$

- Démonstration :
$c^d \equiv (m^e)^d \equiv m^{ed} \equiv m^{1+q\times\varphi(n)} \equiv m \times (m^{\varphi(n)})^q \equiv m \mod n$

Force de RSA

Force de RSA

Comment trouver d en connaissant e ?

- d est l'inverse de e modulo $\varphi(n)$
- Il faut donc pouvoir calculer $\varphi(n) = (p-1)(q-1)$
- Il faut donc connaître p et q
- Il faut donc pouvoir factoriser $n = pq$

$\rightarrow$ «Casser RSA» est équivalent à «Factoriser $n = pq$»

- On utilise n de l'ordre de 1024 bits ou supérieur

Challenge RSA

Challenge RSA

- 1991–2007 : Trouver la factorisation de n en deux grands entiers premiers p et q
- Plusieurs n de taille croissante
- Des récompenses pour ceux qui trouvent p et q

Nom	Dec	Bin	Date
RSA-100	100	330	Avril 1991
RSA-129	129	426	Avril 1994
RSA-155	155	512	Août 1999
RSA-576	174	576	Décembre 2003
RSA-640	193	640	Novembre 2005
RSA-200	200	663	Mai 2005
RSA-768	232	768	Décembre 2009

RSA-768

Factorisation de RSA-768

123018668453011775513049495838496272077285356959533479219732245215172640050726365751874520219978646938995647494277406384592519255732630345373154826850791702612214291346167042921431160222124047927473779408066535141959745985690214341 3

=

334780716989568987860441698482126908177047949837137685689124313889828837938780022876147116525317430877378144679994893

×

367460436667995904282446337996279526322791581643430876426760322838157396665112792333734171433968102700927987363089173

- Première étape : 6 mois sur 80 processeurs
- Deuxième étape : 2 ans sur plusieurs centaines de processeurs
  Équivalent à 1500 ans sur un Opteron 2.2 GHz avec 2 Gio de RAM
- Dernières étapes : quelques semaines sur quelques processeurs

$\rightarrow$ Il faut penser à remplacer les clefs de 1024 bits

Plan

Difficultés pratiques de RSA ⋆⋆

Difficultés pratiques de RSA

- Calculer des exponentiations modulaires
  - → Exponentiation rapide
- Multiplier des grands nombres
  - → Algorithme de Karatsuba
- Trouver des grands nombres premiers
  - → Test de primalité de Miller-Rabin

Exponentiation
Problème

Problème

Soit a, b et n trois entiers, on cherche à calculer $a^b \mod n$

Remarque préliminaire

On ne va pas calculer a^b puis ensuite calculer le modulo n !

Exponentiation
Algorithme naïf

Algorithme naïf

Exp(a,b)

$x \leftarrow 1$
for i from 1 to b **do**
  $x \leftarrow x \times a \mod n$
end for
return x

Coût

$$C(b) = b$$

Exponentiation
Algorithme rapide

Algorithme rapide

Exp(a,b)

if $b = 1$ **then**
  return a
end if
if b pair **then**
  return $\text{Exp}(a^2 \mod n, b/2) \mod n$
else
  return $b \times \text{Exp}(a^2 \mod n, b/2) \mod n$
end if

Coût

$$C(b) = C(b/2) + O(1) = O(\log_2 b)$$

Multiplier des grands nombres

Problème

Problème

Soit x et y deux nombres de n chiffres en base B, on cherche à calculer le plus rapidement possible $x \times y$. Pour tout $m < n$, on peut écrire x et y sous la forme suivante :

$$x = x_1 B^m + x_0,\ y = y_1 B^m + y_0$$

où x_0 et y_0 sont tous les deux plus petits que B^m.

Intérêt

Si $B = 2$ et que n est supérieur à la taille du plus grand registre, on ne peut plus faire la multiplication directement avec le processeur. Il faut alors utiliser des nombres à précision arbitraire.

Multiplier des grands nombres

Algorithme naïf

Algorithme naïf

$$x \times y = (x_1 B^m + x_0)(y_1 B^m + y_0) = z_2 B^{2m} + z_1 B^m + z_0$$

avec :

- $z_2 = x_1 y_1$
- $z_1 = x_1 y_0 + x_0 y_1$
- $z_0 = x_0 y_0$

Coût

Si $m \approx \frac{n}{2}$, alors :

$$C(n) = 4 \times C\left(\frac{n}{2}\right) + O(n) = O(n^2)$$

Multiplier des grands nombres

Algorithme de Karatsuba (1960)

★★

Algorithme de Karatsuba

$$x \times y = (x_1 B^m + x_0)(y_1 B^m + y_0) = z_2 B^{2m} + z_1 B^m + z_0$$

avec :

- $z_2 = x_1 y_1$
- $z_1 = (x_1 + x_0)(y_1 + y_0) - z_2 - z_0$
- $z_0 = x_0 y_0$

Coût

Si $m \approx \frac{n}{2}$, alors :

$$C(n) = 3 \times C\left(\frac{n}{2}\right) + O(n) = O(n^{\log_2 3}) = O(n^{1.585\ldots})$$

Trouver des grands nombres premiers ⋆⋆

Principes

Principes

- On dispose d'un test de primalité qui indique la primalité d'un nombre (éventuellement avec une certaine probabilité)
- On tire un nombre n impair au hasard
- S'il n'est pas premier, on recommence avec $n + 2$
- Comme il y a en moyenne $\frac{n}{\log n}$ premiers inférieurs à n, on en déduit qu'il faut $O(\log n)$ tentatives avant de trouver un premier

Trouver des grands nombres premiers

Petit théorème de Fermat

★★

Théorème (Petit théorème de Fermat)

Si p est un nombre premier et si a est un entier non divisible par p, alors :

$$a^{p-1} - 1 \equiv 0 \mod p$$

Remarques

- Cas particulier du théorème d'Euler
- Il n'y a pas d'équivalence, juste une implication !

Trouver des grands nombres premiers ⋆

Idée du test de Miller-Rabin

Idée du test de Miller-Rabin

- Soit n l'entier impair à tester
- Soient s et t tels que $n-1 = t2^s$
- Pour $a < n$, on a :

$$a^{n-1} - 1 = a^{t2^s} - 1 = (a^t - 1)(a^t + 1)(a^{2t} + 1)\dots(a^{(2^{s-1})t} + 1)$$

- Si n est premier alors, $a^{n-1} - 1 \equiv 0 \mod n$, donc :
  - Soit $a^t - 1 \equiv 0 \mod n$
  - Soit $a^{t2^i} + 1 \equiv 0 \mod n, 0 \leq i < s$

Trouver des grands nombres premiers ⋆

Test de Miller-Rabin

Test de Miller-Rabin

Calculer s et t tels que $n-1 = t \times 2^s$
$a \leftarrow \text{RANDOM}(0, n-1)$
$q \leftarrow a^t \mod n$
if $q = 1$ **then**
  return "probablement premier"
end if
for i from 1 to s **do**
  $q \leftarrow q \times q \mod n$
  if $q = n-1$ **then**
    return "probablement premier"
  end if
end for
return "composé"

Trouver des grands nombres premiers ⋆⋆

Coût du test de Miller-Rabin

Coût du test de Miller-Rabin

- a réussit le test si "composé" est renvoyé
- Si n est impair et non-premier, moins de $\frac{n-1}{4}$ nombres a échouent au test
- La probabilité d'échouer est donc inférieur à $\frac{1}{4}$
- On peut donc tirer k entiers a, s'ils échouent tous, on a une probabilité de $\frac{1}{4^k}$ d'échouer
- Coût total : $O(k \log^3 n)$ pour une probabilité d'erreur de 4^{-k}
- Pour n avec 1024 bits, on a besoin de 44 tests pour avoir une probabilité d'erreur inférieur à $\frac{1}{2^{80}}$ (1 chance sur 10^{24} ; Euromillions : 1 chance sur 10^8)

Plan

Historique

Historique de ElGamal

- 1984 : ElGamal
- Basé sur la difficulté du logarithme discret

Établissement des clefs ⋆

Établissement des clefs

1. On choisit un groupe cyclique G d'ordre q dont g est un générateur
2. On choisit $x \in \{0, \ldots, q-1\}$
3. On calcule $h = g^x$

Alors, on a :

- $\langle h, G, q, g \rangle$ est la clef publique
- $\langle x \rangle$ est la clef privée

Chiffrement

Chiffrement

- Bob choisit $y \in \{0, \ldots, q-1\}$ et calcule $c_1 = g^y$
- Bob calcule $s = h^y$ $(= g^{xy})$
- Bob calcule $c_2 = m \times s$
- Bob envoie (c_1, c_2)

Déchiffrement

Déchiffrement

- Alice calcule $s = c_1^x \ (= (g^y)^x = g^{xy})$
- Alice calcule $m = c_2 \times s^{-1}$
- Démonstration :
  $c_2 \times s^{-1} = m \times s \times s^{-1} = m$

Force de ElGamal

Force de ElGamal

Comment trouver x en connaissant h, G, q, g ?

- $h = g^x$
- Il faut donc trouver x en connaissant h et g
- x est le logarithme discret de h, $x = \log_g(h)$

$\rightarrow$ «Casser ElGamal» équivaut à «Trouver le logarithme discret»

Quel groupe utiliser ?

Quel groupe utiliser ?

- $\mathbb{Z}/p\mathbb{Z}^\star$
- Courbe elliptique sur un corps fini
  - Taille de clef plus petite
  - ECC-160 = RSA-1024

Cinquième partie

Fonctions de hachage cryptographiques

Plan de ce cours

Plan

Fonction de hachage
Définitions

Définition (Fonction de hachage)

Une fonction de hachage H est une fonction qui transforme un message M de taille arbitraire en une valeur h de taille fixe n. h est appelé la somme de contrôle, l'empreinte ou le résumé de M. M est appelé l'antécédent ou la préimage de h.

Définition (Collision)

On parle de collision entre M_1 et M_2 lorsque :

$$\begin{cases} M_1 \neq M_2 \\ H(M_1) = H(M_2) \end{cases}$$

Fonction de hachage
Quelques considérations

Quelques considérations

- Par définition, la fonction H n'est pas injective, c'est-à-dire qu'on peut trouver des collisions.
- Mais on espère que la fonction H est difficile à inverser, c'est-à-dire qu'étant donné une image, il est difficile de trouver un antécédent

$\rightarrow$ On parle de fonction à sens unique

Exemples de fonctions à sens unique

- Le produit de deux grands entiers p et q *vs* la factorisation (RSA)
- L'exponentiation modulaire *vs* le logarithme discret (ElGamal)

Fonction de hachage

Propriétés

★★★

Résistance à la préimage

Étant donné h, il est très difficile de trouver M tel que :

$$h = H(M)$$

Résistance à la seconde préimage

Étant donné M_1, il est très difficile de trouver $M_2 \neq M_1$ tel que :

$$H(M_1) = H(M_2)$$

Résistance aux collisions

Il est très difficile de trouver M_1 et $M_2 \neq M_1$ tels que :

$$H(M_1) = H(M_2)$$

Paradoxe des anniversaires

Principe

★★★

Paradoxe des anniversaires

Quel est la probabilité, en tirant au hasard k valeurs dans un ensemble de N valeurs, de tirer deux fois la même valeur ?

$$p(k) = 1 - \frac{N!}{(N-k)!} \cdot \frac{1}{N^k} \approx 1 - e^{-\frac{k(k-1)}{2N}}$$

Et inversement :

$$k(p) \approx \sqrt{2 \cdot N \ln\left(\frac{1}{1-p}\right)}$$

Application

Combien de personnes sont nécessaires pour qu'il y ait une probabilité supérieure à $\frac{1}{2}$ qu'au moins deux personnes ait leur anniversaire le même jour ? Réponse : 23 !

Paradoxe des anniversaires

Conséquence pour les fonctions de hachage

★★★

Conséquence

Pour une fonction de hachage qui fournit une empreinte de n bits, on a seulement $\frac{n}{2}$ bits de sécurité. C'est-à-dire qu'on peut trouver une collision avec une probabilité supérieure à $\frac{1}{2}$ en $O(\sqrt{2^n}) = O(2^{\frac{n}{2}})$

Fonction de hachage
Application

Applications (cryptographiques) des fonctions de hachage

- Intégrité d'un message par vérification du résumé
- Signature d'un résumé plutôt que du message lui-même
- Vérification d'un mot de passe dont on conserve le résumé
- Identification d'un fichier (dans les gestionnaires de version)

Applications (non-cryptographiques) des fonctions de hachage

- Table de hachage

Plan

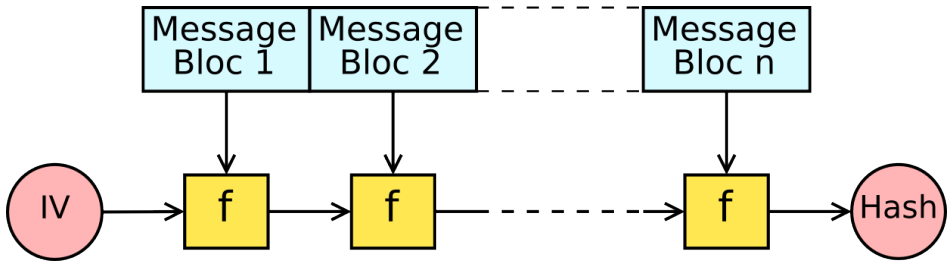
Construction de Merkle-Damgård ⋆⋆

Construction de Merkle-Damgård

- On découpe le message M en blocs de taille égale M_i
- On utilise une fonction de compression f :

$$h_0 = IV, h_{i+1} = f(M_i, h_i)$$

- Propriété : si f est résistante aux collisions, alors H l'est



Construction de Davies–Meyer ⋆

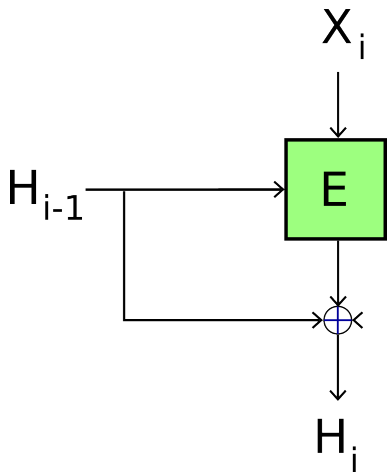
Construction de Davies–Meyer

- Cas particulier de construction de Merkle-Damgård :

$$f(M_i, h_i) = E(M_i, h_i) \oplus h_i$$

- La découverte d'un point fixe permet de trouver des collisions

→ Les fonction utilisant cette construction sont moins robustes



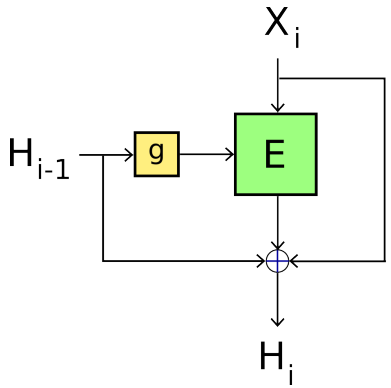
Construction de Miyaguchi-Preneel

Construction de Miyaguchi-Preneel

- Cas particulier de construction de Merkle-Damgård :

$$f(M_i, h_i) = E(M_i, g(h_i)) \oplus h_i \oplus m_i$$

- Amélioration par rapport à Davies-Meyer
- Utilisé dans Whirlpool, avec E dérivée de AES
- Sûre pour l'instant



Plan

Code d'Authentification de Message ⋆⋆

Définition (Code d'Authentification de Message (MAC))

Un Message Authentication Code (MAC) est une association entre une fonction de compression et une clef secrète permettant de vérifier à la fois l'intégrité et l'authenticité des données reçues.

Utilisation

- Alice veut envoyer un message M à Bob
- Elle calcule le MAC avec une clef secrète K qu'elle partage avec Bob
- Elle envoie son message et le MAC à Bob
- Bob recalcule le MAC et vérifie qu'il est identique à celui d'Alice

Remarques

On utilise une fonction de hachage comme fonction de compression, ou un algorithme de cryptographie symétrique avec un mode d'opération

Hash-based Message Authentication Code (HMAC) ⋆

Hash-based Message Authentication Code (HMAC)

- Défini dans la RFC 2104
- Utilisé dans IPSec et TLS
- $\mathrm{HMAC}(K, M) = H\left((K \oplus opad) \,||\, H\left((K \oplus ipad) \,||\, M\right)\right)$
  - H est une fonction de hachage cryptographique
  - K est la clef secrète
  - $opad = 0\text{x}5\text{c}5\text{c}5\text{c}\ldots5\text{c}5\text{c}$
  - $ipad = 0\text{x}363636\ldots3636$
  - $||$ est la concaténation
- Plusieurs variantes suivant H : HMAC-MD5, HMAC-SHA-1

Plan

Hachage de mots de passe

Hachage de mots de passe

Le hachage de mots de passe est un cas particulier très usuel d'utilisation des fonctions de hachage. L'idée est la suivante :

1. l'utilisateur définit son mot de passe ;
2. le système enregistre une empreinte du mot de passe ;
3. quand l'utilisateur se connecte, il donne son mot de passe ;
4. le système calcule l'empreinte du mot de passe et la compare à l'empreinte stokée.

Remarque importante

On ne stocke jamais un mot de passe en clair !

Hachage de mots de passe

Hachage de mots de passe

Un attaquant qui récupère une base de données de mot de passe peut essayer de trouver les mots de passe des utilisateurs en utilisant la fonction de hachage. Trois problèmes se posent :

- l'espace de recherche est-il suffisamment grand ?
- deux utilisateurs qui ont le même mot de passe ont-il la même empreinte ?
- la fonction de hachage est-elle trop rapide ?

Espace de recherche

Espace de recherche

Ce paramètre dépend de l'utilisateur. Si l'utilisateur utilise un mot du dictionnaire, l'espace de recherche est ridiculement faible !

- mots du dictionnaire : $130000 \approx 2^{17}$
- deux mots du dictionnaire : 2^{34}
- trois mots du dictionnaire : 2^{51}

Espace de recherche

Espace de recherche

Longueur	6	7	8	9	10
a-z (26)	2^{28}	2^{32}	2^{37}	2^{42}	2^{47}
a-zA-Z (52)	2^{34}	2^{39}	2^{45}	2^{51}	2^{57}
a-zA-Z0-9 (62)	2^{35}	2^{41}	2^{47}	2^{53}	2^{59}
+ [punct] (94)	2^{39}	2^{45}	2^{52}	2^{58}	2^{65}
ENT	-	-	2^{48}	2^{55}	2^{61}

Règles en vigueur sur l'ENT

- Au moins 8 caractères
- Au moins un caractère spécial (parmi 26)
- Au moins 2 chiffres (mais pas que des chiffres)
- Au moins une lettre

Salage

Définition (Salage)

Le salage consiste à ajouter une chaîne de caractère aléatoire (appelé le sel) au mot de passe avant hachage.

Conséquences

- Deux mots de passe identiques auront un hachage différent avec deux sels différents.
- Il est nécessaire de stocker le sel à côté du mot de passe haché, de manière à pouvoir reconstruire l'empreinte.
- Le salage ne protège pas un mot de passe particulier mais un ensemble de mots de passe.
- Le salage protège d'une attaque par table arc-en-ciel *(rainbow table)*.

Vitesse de la fonction de hachage

Vitesse de la fonction de hachage

- Une fonction de hachage est normalement destinée à être rapide pour être capable de hacher de grandes quantité de données.
- Pour limiter la portée d'une attaque sur des mots de passe, on a intérêt à ce que la fonction de hachage soit la plus lente possible.

$\rightarrow$ On n'utilise pas les mêmes fonctions de hachages pour les mots de passe et pour les autres utilisations cryptographiques

Hachage cryptographique pour les mots de passe

Généralités

Dans les fonctions suivantes, on peut souvent régler :

- le temps d'exécution
- l'espace mémoire utilisé

Exemples (Hachage cryptographique pour les mots de passe)

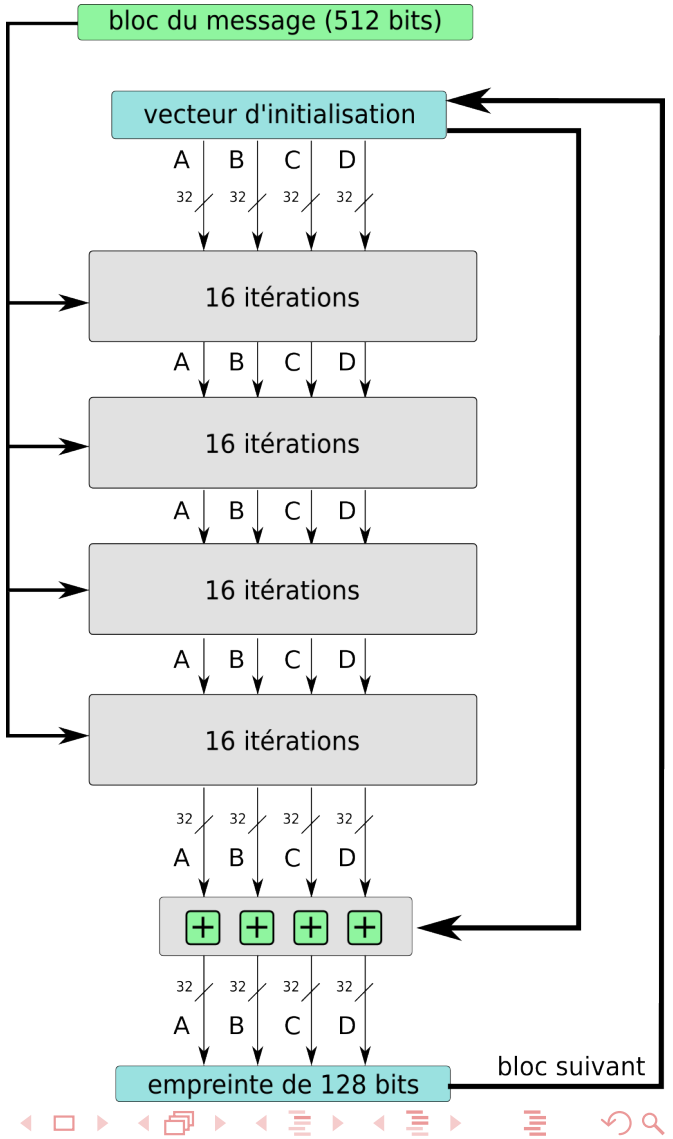
- bcrypt, 1999, basé sur Blowfish
- PBKDF2, 2000, RFC 2898
- scrypt, 2012, RFC 7914
- Argon2, 2015, gagnant du Password Hashing Competition

Plan

MD5
Généralités

Généralités

- Inventé par Ronald Rivest en 1991
- Spécifié dans la RFC 1321
- Amélioration de MD4
- Bloc de 512 bits
- Empreinte de 128 bits
- 64 rondes (ou itérations)



MD5
Padding

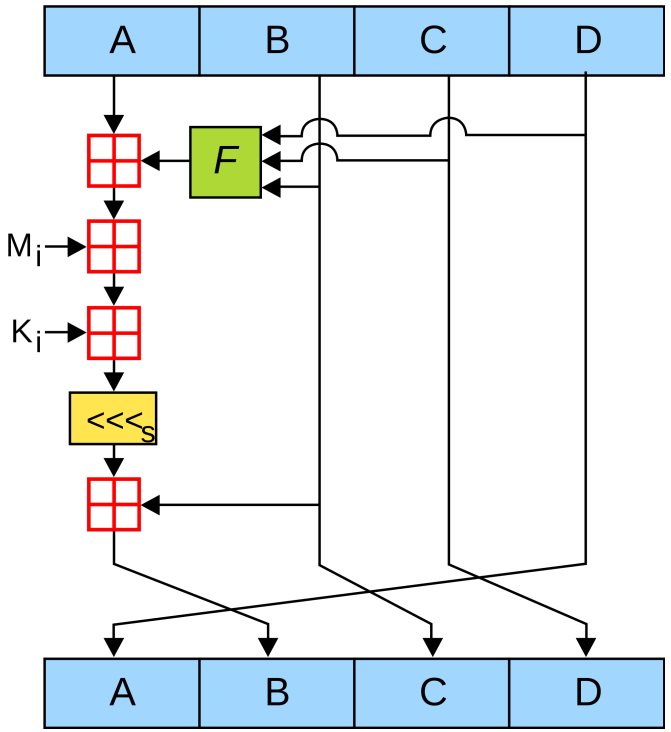
Padding

- On veut obtenir une taille multiple de 512 bits
- On ajoute un 1 puis des 0, sauf les 64 derniers bits
- On code la taille du message sur 64 bits en *little endian*

MD5
Description d'une itération

Description d'une itération

- Le message M est découpé en bloc M_i
- Les constantes K_i sont fixées
- État : A, B, C, D
- Fonctions F :
  - $F_1(B, C, D) = (B \wedge C) \vee (\neg B \wedge D)$
  - $F_2(B, C, D) = (B \wedge D) \vee (C \wedge \neg D)$
  - $F_3(B, C, D) = B \oplus C \oplus D$
  - $F_4(B, C, D) = C \oplus (B \vee \neg D)$



MD5
Sécurité

Sécurité

- Résistance aux collisions
  - 1996 : une faille est trouvée dans la fonction de compression
  - 2004 : premières collisions générées
- Résistance à la préimage
  - 2009 : Attaque en $2^{123.4}$
- Attaque par *rainbow table* : compromis temps-mémoire

$\rightarrow$ MD5 n'est plus une fonction sûre cryptographiquement

Plan

Famille SHA

Famille SHA

- Famille de fonction de hachage cryptographique
- Mises au point par la NSA (sauf SHA-3)
- Publiées par le NIST
- Les différentes versions :
  - 1993 : SHA-0
  - 1995 : SHA-1
  - 2001 : SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512)
  - 2012 : SHA-3

SHA-0

SHA-0

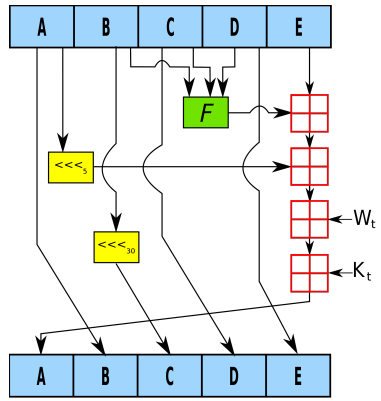
- Empreinte de 160 bits
- Fonctionnement similaire à MD5
- Très vite abandonnée à cause d'une «faille significative» non-dévoilée
- Remplacée par SHA-1
- 2004 : première collision par Joux avec une attaque en 2^{51}
- 2005 : collision avec une attaque en 2^{39}

SHA-1

SHA-1

- Empreinte de 160 bits
- Fonctionnement similaire à MD5
- 80 rondes
- 2005 : collision avec une attaque théorique en 2^{63}
- 2017 : collision pratique (6500 années-CPU + 110 années-GPU)

- $F_1(B, C, D) = Ch(B, C, D) = (B \wedge C) \oplus (\neg B \wedge D)$
- $F_2(B, C, D) = Parity(B, C, D) = B \oplus C \oplus D$
- $F_3(B, C, D) = Maj(B, C, D) = (B \wedge C) \oplus (B \wedge D)$
- $F_4(B, C, D) = Parity(B, C, D) = B \oplus C \oplus D$



SHA-2

SHA-2

- Ensemble de 4 fonctions de hachage :
  SHA-224, SHA-256, SHA-384, SHA-512
- Empreinte de 224, 256, 384, 512 bits
- Fonctionnement similaire à MD5
- 64 rondes (SHA-224/256) ou 80 rondes (SHA-384/512)
- Sûres à l'heure actuelle

SHA-3

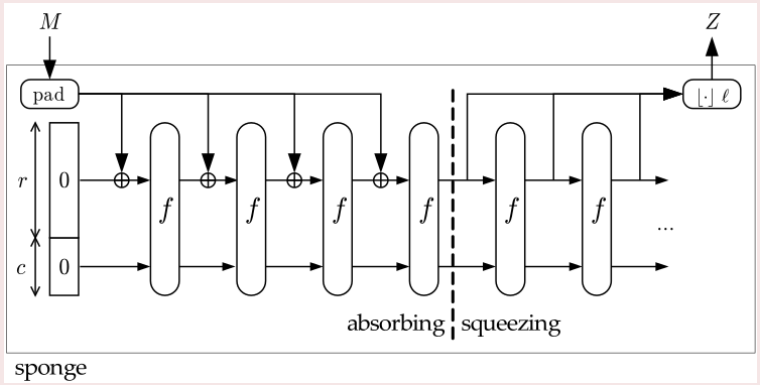
SHA-3

- Concours lancé par le NIST en 2007
- Octobre 2008 : 64 candidats
- Décembre 2008 : Round 1 $\rightarrow$ 51 candidats
- Juillet 2009 : Round 2 $\rightarrow$ 14 candidats
- Décembre 2010 : Round 3 $\rightarrow$ 5 finalistes
  - BLAKE
  - Grøstl
  - JH
  - Keccak (Daemen et al.)
  - Skein (Schneier et al.)
- Octobre 2012 : Annonce du vainqueur, Keccak

Keccak

Keccak

- Utilise une nouvelle construction : la construction éponge
- La fonction f consiste en $12 + 2l$ itérations de 5 sous-fonctions appelées θ, ρ, π, χ et ι.
- Le NIST recommande toujours SHA-2 pour l'instant, mais pense qu'il est préférable d'avoir une fonction de hachage avec des propriétés et des implémentations différentes de SHA-2.



sponge

Sixième partie

Protocoles cryptographiques (1)

Plan de ce cours

Plan

Protocole cryptographique

Définition (Protocole cryptographique)

Un **protocole cryptographique** est une suite d'actions permettant à deux ou plusieurs agents qui communiquent d'assurer des propriétés de sécurité. Un protocole cryptographique utilise des outils cryptographiques.

Hypothèse de cryptographie parfaite ⋆⋆

Hypothèse de cryptographie parfaite

Sont considérés comme sûrs :

- Les algorithmes de cryptographie symétrique
- Les algorithmes de cryptographie asymétrique
- Les fonctions de hachages

Notation

Notation

Soient :

- M, M_1, M_2 des messages
- K une clef
- A et B des agents (participants)

On note :

- M_1, M_2 (parfois $M_1||M_2$) la concaténation de M_1 et M_2
- $\{M\}_K$ le message M chiffré avec la clef K
- $A \rightarrow B : M$: l'envoi par A d'un message M à B
- $I(A) \rightarrow B : M$: l'envoi d'un message M par I se faisant passer pour A

Notation

Pour les chiffrements symétriques

- $K_{AB} = K_{AB}^{-1}$
- Bon rapport $\frac{\text{volume de données à chiffrer}}{\text{temps de chiffrement}}$

Pour les chiffrements asymétriques

- Clef publique de A : K_A
- Clef privée de A : K_A^{-1}
- Propriétés : $\{\{M\}_{K_A}\}_{K_A^{-1}} = \{\{M\}_{K_A^{-1}}\}_{K_A} = M$
- À éviter pour chiffrer un gros volume de données

Pour les fonctions de hachage

- hash(M)

Plan

Objectifs de la sécurité
Propriétés associée à la cryptographie

★★

Propriétés associée à la cryptographie

- Confidentialité
  - $\{$"4976 0974 2373 7788"$\}_{K_B}$
  - $\{$`recette-sushi.ps`$\}_{K_{AB}}$
- Authentification
  - «Le mot de passe est secret», $\{$«Le mot de passe est secret»$\}_{K_A^{-1}}$
  - `securite.pdf`, $\{$hash(`securite.pdf`)$\}_{K_A^{-1}}$
- Intégrité
  - Le contenu M d'un message chiffré $\{M\}_K$ ne peut être modifié sans K
  - `contrat.pdf`, hash(`contrat.pdf`)

Objectifs de la sécurité

Propriétés associée aux protocoles cryptographiques

Propriétés associée aux protocoles cryptographiques

- Secret
- Authentification d'un message
- Authentification d'une entité
- Fraîcheur / Anti-rejeu
- Accord non répudiable
- Équité
- Anonymat

Propriétés des protocoles cryptographiques ⋆⋆⋆

Définition (Secret)

Un protocole assure le **secret d'une donnée** s si un intrus ne peut pas déduire s.

Définition (Authentification de message)

Un protocole permet à un agent A d'**authentifier un message** M si A peut connaître de façon sûre l'émetteur de M.

Définition (Authentification d'entité)

Un protocole permet à un agent A d'**authentifier un agent** B si à la fin d'une session réussie, A a la garantie qu'il a réalisé le protocole avec B.

Définition (Fraîcheur)

Pendant une session, une donnée est **fraîche** si l'on peut garantir qu'elle a été émise spécifiquement pour cette session par un des acteurs.

Propriétés des protocoles cryptographiques ⋆⋆⋆

Définition (Accord non-répudiable)

Un protocole établit un **accord non-répudiable** entre deux agents si chaque agent peut fournir la preuve que l'autre a accepté les termes de l'accord.

Définition (Équité)

Un protocole d'accord non-répudiable entre deux agents A et B est **équitable** si aucun agent ne peut obtenir d'avantage sur l'autre : A n'obtient pas la preuve de l'accord de B avant que B n'ait une preuve de l'accord de A (et vice-versa).

Définition (Anonymat)

Un protocole préserve l'**anonymat** d'un agent A s'il est impossible d'identifier l'agent A à partir des messages échangés.

Plan

Envoi d'un secret de B à A ⋆

Envoi d'un secret

On va établir un protocole entre deux acteurs A et B afin qu'il puisse échanger un message secret M, tout en assurant un certain nombre de propriétés. On fait l'hypothèse qu'ils ont pu échanger des clefs (K_{AB}, K_A, K_B) auparavant.

Envoi d'un secret de B à A

Essai numéro 1

★★

Essai numéro 1

Protocole :

1. $B \rightarrow A : \{M\}_{K_A}$

Propriétés :

- Si K_A et $\{\cdot\}_{K_A}$ sont robustes, seul A peut lire M
- De qui vient le message M ?

Envoi d'un secret de B à A

Essai numéro 2

★★

Essai numéro 2 : tentative d'authentification

Protocole :

1. $B \rightarrow A : \{B, M\}_{K_A}$

Propriétés :

- Facile à détourner par un intrus :
  1. $I(B) \rightarrow A : \{B, M_I\}_{K_A}$

$\rightarrow$ M_I connu de I et accepté par A comme venant de B !

Envoi d'un secret de B à A

Essai numéro 3

★★

Essai numéro 3 : authentification par signature

Protocole :

1. $B \rightarrow A : \{B, M, \{M\}_{K_B^{-1}}\}_{K_A}$

ou avec des clefs symétriques :

1. $B \rightarrow A : B, \{M\}_{K_{AB}}$

Propriétés :

- M secret
- A authentifie M comme venant de B
- I peut faire accepter de nouveau M par **rejeu** :
  1. $I(B) \rightarrow A : \{B, M, \{M\}_{K_B^{-1}}\}_{K_A}$

  ou avec des clefs symétriques :
  1. $I(B) \rightarrow A : B, \{M\}_{K_{AB}}$

Envoi d'un secret de B à A

Essai numéro 4

★★★

Essai numéro 4 : secret, authentification et fraîcheur

Protocole :

1. $A \rightarrow B : \{A, B, N_A\}_{K_B}$ (challenge de A pour B)
2. $B \rightarrow A : \{A, B, N_A, M\}_{K_A}$ (réponse de B au challenge)

où N_A est une valeur aléatoire fraîche : un **nonce** (*number used once*).

Propriétés :

- M secret
- $\{A, B, N_A, M\}_{K_A}$ frais
- A authentifie M comme venant de B
- A authentifie B lors de la session

Envoi d'un secret de B à A

Essai numéro 4

★★

Essai numéro 4 : solution avec une clef symétrique

Protocole :

1. $A \rightarrow B : \{A, B, N_A\}_{K_{AB}}$ (challenge de A pour B)
2. $B \rightarrow A : \{A, B, N_A, M\}_{K_{AB}}$ (réponse de B au challenge)

où N_A est une valeur aléatoire fraîche : un **nonce** (*number used once*).

Propriétés :

- M secret
- $\{A, B, N_A, M\}_{K_{AB}}$ frais
- A authentifie M comme venant de B
- A authentifie B lors de la session

Plan

Login Unix (telnet) ⋆⋆

Login Unix (telnet)

Hypothèses :

- A connaît P (son mot de passe)
- M connaît A et hash(P)

Protocole :

1. $M \rightarrow A$: "login ?"
2. $A \rightarrow M$: A
3. $M \rightarrow A$: "password ?"
4. $A \rightarrow M$: P

M vérifie le mot de passe en calculant hash(P) et compare avec celui stocké dans `/etc/shadow`.

Propriétés :

- Aucune ! (si la communication entre A et M n'est pas sûre)

Login Unix par One Time Passwd (OTP) ⋆⋆

Hypothèses

- A connaît P (son mot de passe)
- M connaît A et $\text{hash}^n(P)$ et k initialisé à $n-1$

Login Unix par OTP (première connexion)

Protocole (première connexion, $k = n-1$) :

1. $M \to A$: "login ?"
2. $A \to M$: A
3. $M \to A$: $n-1$, "password ?"
4. $A \to M$: $\text{hash}^{n-1}(P)$

M calcule $\text{hash}(\text{hash}^{n-1}(P))$ et compare avec $\text{hash}^n(P)$.

Login Unix par One Time Passwd (OTP) ⋆⋆

Login Unix par OTP (deuxième connexion)

Protocole (première connexion, $k = n - 2$) :

1. $M \to A$: "login ?"
2. $A \to M$: A
3. $M \to A$: $n - 2$, "password ?"
4. $A \to M$: $\text{hash}^{n-2}(P)$

M calcule $\text{hash}^2(\text{hash}^{n-2}(P))$ et compare avec $\text{hash}^n(P)$.

Login Unix par One Time Passwd (OTP) ⋆⋆

Login Unix par OTP (i^e connexion)

Protocole (i^e connexion) :

1. $M \to A$: "login ?"
2. $A \to M$: A
3. $M \to A$: $n - i$, "password ?"
4. $A \to M$: $\text{hash}^{n-i}(P)$

M calcule $\text{hash}^i(\text{hash}^{n-i}(P))$ et compare avec $\text{hash}^n(P)$.

Login Unix par One Time Passwd (OTP) ⋆⋆

Login Unix par OTP

Propriétés :

- Secret de P
- Secret de $\text{hash}^i(P)$ avant la connexion i
- Fraîcheur de $\text{hash}^i(P)$ dans la session i
- Authentification de $\text{hash}^i(P)$
- Authentification de A par M

Login SSH par mot de passe ⋆⋆

Login SSH par mot de passe

Hypothèses :

- Idem login Unix

Protocole :

1. $M \rightarrow A : K_M$
2. $A \rightarrow M : \{K_{AM}\}_{K_M}$
3. $M \rightarrow A : \{\text{"login ?"}\}_{K_{AM}}$
4. $A \rightarrow M : \{A\}_{K_{AM}}$
5. $M \rightarrow A : \{\text{"password ?"}\}_{K_{AM}}$
6. $A \rightarrow M : \{P\}_{K_{AM}}$

Propriétés :

- A, P secrets partagés entre A et M
- M authentifie A mais A n'authentifie pas M

Login SSH par clef asymétrique ⋆⋆

Login SSH par clef asymétrique

Hypothèses :

- A a déposé au préalable sa clef publique K_A sur M

Protocole :

1. $M \rightarrow A : K_M$
2. $A \rightarrow M : \{K_{AM}\}_{K_M}$
3. $M \rightarrow A : \{\{N_M\}_{K_A}\}_{K_{AM}}$
4. $A \rightarrow M : \{N_M\}_{K_{AM}}$

Propriétés :

- A, N_M secrets partagés entre A et M
- M authentifie A mais A n'authentifie pas M

Login SSL/TLS (HTTPS) ⋆⋆

Login SSL/TLS (HTTPS)

Hypothèses :

- B (*browser*) connaît A et P (ou A saisit P dans B)
- B connaît K_S où S est une autorité de certification (CA)
- V (*vendor*) a un certificat $\{V, K_V\}_{K_S^{-1}}$ et connaît A et P

Protocole :

1. $V \rightarrow B : \{V, K_V\}_{K_S^{-1}}$
2. $B \rightarrow V : \{K_{BV}\}_{K_V}$
3. $V \rightarrow B : \{\text{"login ?"}\}_{K_{BV}}$
4. $B \rightarrow V : \{A\}_{K_{BV}}$
5. $V \rightarrow B : \{\text{"password ?"}\}_{K_{BV}}$
6. $B \rightarrow V : \{P\}_{K_{BV}}$

Plan

Protocole d'accord non-répudiable

Un exemple rudimentaire

Protocole d'accord non-répudiable

Protocole :

1. $A \rightarrow B : \{A, B, \text{Contrat}\}_{K_A^{-1}}$
2. $B \rightarrow A : \{A, B, \text{Contrat}\}_{K_B^{-1}}$

Propriétés :

- Accord non-répudiable entre A et B sur Contrat.
- Non-équitable car B obtient la signature avant A !

Protocole d'accord non-répudiable

Un exemple rudimentaire

Protocole d'accord non-répudiable et un peu plus équitable

Protocole :

1. $A \rightarrow B : \{A, B, \text{Cont}\}_{K_A^{-1}}$
2. $B \rightarrow A : \{A, B, \text{Cont}\}_{K_B^{-1}}$
3. $A \rightarrow B : \{A, B, \text{rat}\}_{K_A^{-1}}$
4. $B \rightarrow A : \{A, B, \text{rat}\}_{K_B^{-1}}$

Propriétés :

- Accord non-répudiable entre A et B sur Contrat.
- Pas tout à fait équitable...

Accusé de réception équitable avec un tiers de confiance ⋆⋆

Zhou-Gollmann 96

Accusé de réception

Protocole :

1. $A \to B : \{\text{Propose}, B, N_A, \{M\}_K\}_{K_A^{-1}}$
2. $B \to A : \{\text{Ack}, A, N_A, \{M\}_K\}_{K_B^{-1}}$
3. $A \to S : \{\text{Submit}, B, N_A, K\}_{K_A^{-1}}$ (S est le tiers de confiance)
4. $S \to A, B : \{\text{Confirm}, A, B, N_A, K\}_{K_S^{-1}}$

Propriétés :

- M secret pour tous (excepté A) jusqu'en 3
- Les 4 message sont authentifiable (signature K_S^{-1})
- A et B ont en même temps (en 4.) la preuve que :
  - pour A : que le message a été reçu par B
  - pour B : que A a bien envoyé le message

Septième partie

Protocoles cryptographiques (2)

Plan de ce cours

15 Études de protocoles
- Échange et diffusion des clefs
- Paiement électronique

Plan

Échange et diffusion de clefs ⋆⋆

Échange et diffusion de clefs

Jusqu'à présent, on a supposé que les agents avaient échangé leurs clefs au préalable. Maintenant, nous allons voir des protocoles qui permettent un échange ou une diffusion de clefs, en garantissant un certain nombre de propriétés.

Types de clefs

Il existe deux types de clefs :

- Master Key : Clef permanente entre deux agents (généralement un individu et un centre de distribution)
- Session Key : Clef partagée entre deux agents dont la durée de vie est en général la session

Protocole de Diffie-Hellman

Protocole

★★★

Protocole de Diffie-Hellman (DH)

Ce protocole est basé sur la difficulté de calculer le logarithme discret.

1. A choisit un groupe cyclique G d'ordre n dont g est un générateur $G = \{g^i, i \in [1, n]\}$ (par exemple $\mathbb{Z}/p\mathbb{Z}^*$ avec p premier)
2. A tire au hasard $x_A < n$, calcule $M_A = g^{x_A}$
3. $A \rightarrow B : G, g, M_A$
4. B tire au hasard $x_B < n$, calcule $M_B = g^{x_B}$
5. $B \rightarrow A : M_B$
6. A calcule $(M_B)^{x_A} = g^{x_B x_A} = K_{AB}$
7. B calcule $(M_A)^{x_B} = g^{x_A x_B} = K_{AB}$

Protocole de Diffie-Hellman

Sécurité

★★

Sécurité du protocole de Diffie-Hellman

- Pas d'authentification des messages

→ Attaque de l'homme du milieu (*man-in-the-middle*)

Attaque de l'homme du milieu sur DH

1. $A \to I : M_A = g^{x_A}$
2. $I(A) \to B : M_I = g^{x_I}$
3. $B \to I : M_B = g^{x_B}$
4. $I(B) \to A : M_I = g^{x_I}$

Ainsi, Alice et Bob croient ainsi avoir échangé une clé secrète alors qu'en réalité ils ont chacun échangé une clé secrète avec l'homme du milieu.

Needham-Schroeder Symmetric Key Protocol

Protocole

Needham-Schroeder Symmetric Key Protocol (1978)

Protocole avec un tiers de confiance S et des clefs symétriques

1. $A \rightarrow S : A, B, N_A$
2. $S \rightarrow A : \{N_A, K_{AB}, B, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
3. $A \rightarrow B : \{K_{AB}, A\}_{K_{BS}}$
4. $B \rightarrow A : \{N\}_{K_{AB}}$
5. $A \rightarrow B : \{N - 1\}_{K_{AB}}$

Propriétés :

- Secret de K_{AB} partagé entre S, A et B
- Fraîcheur de K_{AB} (pour A mais pas pour B)
- Authentification des messages émis par S pour A et B

Needham-Schroeder Symmetric Key Protocol ⋆⋆

Sécurité

Sécurité du Needham-Schroeder Symmetric Key Protocol (1981)

Si un attaquant a réussi à obtenir K_{AB}, il peut rejouer le protocole en renvoyant $\{K_{AB}, A\}_{K_{BS}}$ à B. La solution :

1. $A \rightarrow B : A$
2. $B \rightarrow A : \{A, N_B\}_{K_{BS}}$
3. $A \rightarrow S : A, B, N_A, \{A, N_B\}_{K_{BS}}$
4. $S \rightarrow A : \{N_A, K_{AB}, B, \{K_{AB}, A, N_B\}_{K_{BS}}\}_{K_{AS}}$
5. ...

Propriété supplémentaire :

- Fraîcheur de K_{AB} pour A et pour B

Version utilisée dans Kerberos

Needham-Schroeder Public Key Protocol

Protocole

★★

Needham-Schroeder Public Key Protocol (1978)

Protocole avec un tiers de confiance S et des clefs symétriques

Distribution des clefs et authentification mutuelle des agents

1. $A \rightarrow S : A, B$
2. $S \rightarrow A : \{K_B, B\}_{K_S^{-1}}$
3. $S \rightarrow B : \{K_A, A\}_{K_S^{-1}}$
4. $A \rightarrow B : \{N_A, A\}_{K_B}$ (je suis Alice, je veux parler à Bob)
5. $B \rightarrow A : \{N_A, N_B\}_{K_A}$ (je te prouve que je suis Bob)
6. $A \rightarrow B : \{N_B\}_{K_B}$ (je te prouve que je suis Alice)

Needham-Schroeder Public Key Protocol ⋆⋆

Propriétés

Propriétés

Pour la distribution des clefs (étape 1–3) :

- Authentification des clefs publiques K_A et K_B
- Pas de secret (nécessaire ?)
- Pas de fraîcheur (nécessaire ?)

Pour l'authentification mutuelle des agents (étape 4–6) :

- Authentification de B par A
- Pas d'authentification de A par B !

$\rightarrow$ Attaque de Lowe

Needham-Schroeder Public Key Protocol

Attaque de Lowe

★★★

Attaque de Lowe (1995)

L'attaquant I persuade A d'initier une communication avec lui et, en relayant les messages à B, persuade B qu'il est en train de parler avec A

1. $A \rightarrow I : \{N_A, A\}_{K_I}$
2. $I(A) \rightarrow B : \{N_A, A\}_{K_B}$
3. $B \rightarrow A : \{N_A, N_B\}_{K_A}$
4. $A \rightarrow I : \{N_B\}_{K_I}$
5. $I(A) \rightarrow B : \{N_B\}_{K_B}$

Needham-Schroeder Public Key Protocol

Attaque de Lowe

★★★

Correction du protocole

1. $A \rightarrow B : \{N_A, A\}_{K_B}$ (je suis Alice, je veux parler à Bob)
2. $B \rightarrow A : \{N_A, N_B, \mathbf{B}\}_{K_A}$ (je te prouve que je suis Bob)
3. $A \rightarrow B : \{N_B\}_{K_B}$ (je te prouve que je suis Alice)

Infrastructure de clefs publiques ⋆⋆

Définition (Infrastructure de clefs publiques)

Une **infrastructure de clefs publiques** (*Public Key Infrastructure*, PKI) est un ensemble de machines, de logiciels, de procédures permettant de gérer un ensemble de clefs publiques (certificats). Elle comprend :

- L'**autorité de certification** (AC ou CA) qui a pour mission de signer les demandes de certificat (CSR : *Certificate Signing Request*) et de signer les listes de révocation (CRL : *Certificate Revocation List*).
- L'**autorité d'enregistrement** (AE ou RA) qui a pour mission de générer les certificats, et d'effectuer les vérifications d'usage sur l'identité de l'utilisateur final.
- L'**autorité de dépôt** (*Repository*) qui a pour mission de stocker les certificats numériques ainsi que les listes de révocation (CRL).
- L'**entité finale** (EE : *End Entity*). L'utilisateur ou le système qui est le sujet d'un certificat.

Plan

Paiement électronique

Les différents types

- Paiement sur Internet
  - Vérification en ligne (ex : SET/CSET)
  - Vérification hors ligne (ex : ECash)
- Paiement sur terminal protégé
  - Vérification en ligne (ex : Carte bancaire)
  - Vérification hors ligne (ex : Carte bancaire à puce)

Secure Electronic Transaction

Description

Secure Electronic Transaction (SET)

- Protocole proposé par Visa et MasterCard en 1997
- Spécification totalement ouverte (!)
- CSET : version française avec carte à puce
- Paiement en ligne entre trois agents connectés :
  - (C)lient
  - (M)archand
  - (B)anque
- N_M et N_C nonces
- Purchamt = montant de la transaction
- Od = «Order Details» = détails de la commande
- Pd = «Payment Details» = détails du règlement

Secure Electronic Transaction

Protocole

Secure Electronic Transaction

1. $C \rightarrow M : C, M$
2. $M \rightarrow C : N_M$
3. $C \rightarrow M : \{\text{Trans}\}_{K_C^{-1}}, \{\text{Od}\}_{K_M}, \{\text{Pd}\}_{K_B}$
4. $M \rightarrow B : \{\text{Trans}\}_{K_C^{-1}}, \{\text{Trans}\}_{K_M^{-1}}, \{\text{Pd}\}_{K_B}$
5. $B \rightarrow M : \{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$
6. $M \rightarrow C : \{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$

avec $\text{Trans} = \{C, M, (N_C, N_M), \text{Purchamt}, \text{hash}(\text{Od}), \text{hash}(\text{Pd})\}$

Secure Electronic Transaction

Protocole

★★

Secure Electronic Transaction

1. $C \rightarrow M : C, M$
2. $M \rightarrow C : N_M$
3. $C \rightarrow M : \{\text{Trans}\}_{K_C^{-1}}, \{\text{Od}\}_{K_M}, \{\text{Pd}\}_{K_B}$
4. $M \rightarrow B : \{\text{Trans}\}_{K_C^{-1}}, \{\text{Trans}\}_{K_M^{-1}}, \{\text{Pd}\}_{K_B}$
5. $B \rightarrow M : \{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$
6. $M \rightarrow C : \{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$

avec $\text{Trans} = \{C, M, (N_C, N_M), \text{Purchamt}, \text{hash}(\text{Od}), \text{hash}(\text{Pd})\}$

Propriétés

Od secret partagé entre C et M
et inconnu à tout autre agent (y compris B)

Secure Electronic Transaction

Protocole

Secure Electronic Transaction

1. $C \rightarrow M : C, M$
2. $M \rightarrow C : N_M$
3. $C \rightarrow M : \{\text{Trans}\}_{K_C^{-1}}, \{\text{Od}\}_{K_M}, \{\text{Pd}\}_{K_B}$
4. $M \rightarrow B : \{\text{Trans}\}_{K_C^{-1}}, \{\text{Trans}\}_{K_M^{-1}}, \{\text{Pd}\}_{K_B}$
5. $B \rightarrow M : \{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$
6. $M \rightarrow C : \{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$

avec $\text{Trans} = \{C, M, (N_C, N_M), \text{Purchamt}, \text{hash}(\text{Od}), \text{hash}(\text{Pd})\}$

Propriétés

Pd secret partagé entre C et B
et inconnu à tout autre agent (y compris M)

Secure Electronic Transaction

Protocole

Secure Electronic Transaction

1. $C \rightarrow M : C, M$
2. $M \rightarrow C : N_M$
3. $C \rightarrow M : \{\text{Trans}\}_{K_C^{-1}}, \{\text{Od}\}_{K_M}, \{\text{Pd}\}_{K_B}$
4. $M \rightarrow B : \{\text{Trans}\}_{K_C^{-1}}, \{\text{Trans}\}_{K_M^{-1}}, \{\text{Pd}\}_{K_B}$
5. $B \rightarrow M : \{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$
6. $M \rightarrow C : \{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$

avec $\text{Trans} = \{C, M, (N_C, N_M), \text{Purchamt}, \text{hash}(\text{Od}), \text{hash}(\text{Pd})\}$

Propriétés

Authentification mutuelle des agents C et M
grâce à $\{\ldots N_M \ldots\}_{K_C^{-1}}$ et $\{\ldots N_C \ldots\}_{K_M^{-1}}$

Secure Electronic Transaction

Protocole

★★

Secure Electronic Transaction

1. $C \to M : C, M$
2. $M \to C : N_M$
3. $C \to M : \{\text{Trans}\}_{K_C^{-1}}, \{\text{Od}\}_{K_M}, \{\text{Pd}\}_{K_B}$
4. $M \to B : \{\text{Trans}\}_{K_C^{-1}}, \{\text{Trans}\}_{K_M^{-1}}, \{\text{Pd}\}_{K_B}$
5. $B \to M : \{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$
6. $M \to C : \{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$

avec $\text{Trans} = \{C, M, (N_C, N_M), \text{Purchamt}, \text{hash}(\text{Od}), \text{hash}(\text{Pd})\}$

Propriétés

Authentification de l'entité B par C et M
grâce à $\{\ldots \text{hash}(\ldots N_C, N_M \ldots) \ldots\}_{K_B^{-1}}$

Secure Electronic Transaction

Protocole

★★

Secure Electronic Transaction

1. $C \rightarrow M : C, M$
2. $M \rightarrow C : N_M$
3. $C \rightarrow M : \{\text{Trans}\}_{K_C^{-1}}, \{\text{Od}\}_{K_M}, \{\text{Pd}\}_{K_B}$
4. $M \rightarrow B : \{\text{Trans}\}_{K_C^{-1}}, \{\text{Trans}\}_{K_M^{-1}}, \{\text{Pd}\}_{K_B}$
5. $B \rightarrow M : \{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$
6. $M \rightarrow C : \{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$

avec $\text{Trans} = \{C, M, (N_C, N_M), \text{Purchamt}, \text{hash}(\text{Od}), \text{hash}(\text{Pd})\}$

Propriétés

Authentification par tous les agents des trois messages grâce à $\{\text{Trans}\}_{K_C^{-1}}$, $\{\text{Trans}\}_{K_M^{-1}}$, $\{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$

Secure Electronic Transaction

Protocole

★★

Secure Electronic Transaction

1. $C \rightarrow M : C, M$
2. $M \rightarrow C : N_M$
3. $C \rightarrow M : \{\text{Trans}\}_{K_C^{-1}}, \{\text{Od}\}_{K_M}, \{\text{Pd}\}_{K_B}$
4. $M \rightarrow B : \{\text{Trans}\}_{K_C^{-1}}, \{\text{Trans}\}_{K_M^{-1}}, \{\text{Pd}\}_{K_B}$
5. $B \rightarrow M : \{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$
6. $M \rightarrow C : \{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$

avec $\text{Trans} = \{C, M, (N_C, N_M), \text{Purchamt}, \text{hash}(\text{Od}), \text{hash}(\text{Pd})\}$

Propriétés

Accord tripartite C, M, B non-répudiable
grâce à $\{\text{Trans}\}_{K_C^{-1}}$, $\{\text{Trans}\}_{K_M^{-1}}$, $\{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$

Secure Electronic Transaction

Protocole

★★

Secure Electronic Transaction

1. $C \rightarrow M : C, M$
2. $M \rightarrow C : N_M$
3. $C \rightarrow M : \{\text{Trans}\}_{K_C^{-1}}, \{\text{Od}\}_{K_M}, \{\text{Pd}\}_{K_B}$
4. $M \rightarrow B : \{\text{Trans}\}_{K_C^{-1}}, \{\text{Trans}\}_{K_M^{-1}}, \{\text{Pd}\}_{K_B}$
5. $B \rightarrow M : \{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$
6. $M \rightarrow C : \{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$

avec $\text{Trans} = \{C, M, (N_C, N_M), \text{Purchamt}, \text{hash}(\text{Od}), \text{hash}(\text{Pd})\}$

Propriétés

Fraîcheur de la transaction vérifiée par B avec N_C et N_M

Secure Electronic Transaction

Protocole

★★

Secure Electronic Transaction

1. $C \to M : C, M$
2. $M \to C : N_M$
3. $C \to M : \{\text{Trans}\}_{K_C^{-1}}, \{\text{Od}\}_{K_M}, \{\text{Pd}\}_{K_B}$
4. $M \to B : \{\text{Trans}\}_{K_C^{-1}}, \{\text{Trans}\}_{K_M^{-1}}, \{\text{Pd}\}_{K_B}$
5. $B \to M : \{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$
6. $M \to C : \{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$

avec $\text{Trans} = \{C, M, (N_C, N_M), \text{Purchamt}, \text{hash}(\text{Od}), \text{hash}(\text{Pd})\}$

Propriétés

Pas d'anonymat : B sait que C a acheté chez M !

Secure Electronic Transaction

Protocole

★★

Secure Electronic Transaction

1. $C \rightarrow M : C, M$
2. $M \rightarrow C : N_M$
3. $C \rightarrow M : \{\text{Trans}\}_{K_C^{-1}}, \{\text{Od}\}_{K_M}, \{\text{Pd}\}_{K_B}$
4. $M \rightarrow B : \{\text{Trans}\}_{K_C^{-1}}, \{\text{Trans}\}_{K_M^{-1}}, \{\text{Pd}\}_{K_B}$
5. $B \rightarrow M : \{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$
6. $M \rightarrow C : \{\text{Results}, \text{hash}(\text{Trans})\}_{K_B^{-1}}$

avec $\text{Trans} = \{C, M, (N_C, N_M), \text{Purchamt}, \text{hash}(\text{Od}), \text{hash}(\text{Pd})\}$

Propriétés

Pas d'équité : C signe avant M

Protocole des cartes bancaires ⋆

Protocole des cartes bancaires

Un protocole d'utilisation d'une carte bancaire doit fournir les garanties suivantes :

- La carte a bien été émise par une autorité légitime
- Une carte légitimement émise n'est pas utilisée frauduleusement

Mise en garde !

Art. 67-1. (L. n° 91-1382 du 30 décembre 1991)

Seront punis d'un emprisonnement d'un an à sept ans et d'une amende de 3.600 F à 5.000.000 F ou de l'une de ces deux peines seulement :

- Ceux qui auront contrefait ou falsifié une carte de paiement ou de retrait ;
- Ceux qui, en connaissance de cause, auront fait usage d'une carte de paiement ou de retrait contrefaite ou falsifiée ;
- Ceux qui, en connaissance de cause, auront accepté de recevoir un paiement au moyen d'une carte de paiement contrefaite ou falsifiée.

Bruce Schneier

«Il est certainement plus simple d'implémenter de la mauvaise sécurité et de rendre hors la loi tous ceux qui le font remarquer que d'implémenter de la bonne sécurité.»

Construction d'un code PIN ⋆⋆

Construction d'un code PIN

1. PIN généré aléatoirement
2. Chiffrement par un triple DES avec une clef K_{CB} du PIN et du numéro de compte : $3DES_{K_{CB}}(PIN, N)$
3. Application d'une fonction à sens unique pour obtenir le **PIN Offset** : $\text{hash}(3DES_{K_{CB}}(PIN, N))$
4. Enregistrement sur la carte du numéro de compte et du PIN Offset
5. Envoi par la Poste du code PIN et suppression de la mémoire du PIN : seul le possesseur de la carte connaît le code PIN

Construction de la carte ⋆⋆

Construction de la carte

La carte contient :

- PIN Offset
- Data = { Nom, Prénom, Numéro de la Carte, Date de Validité }
- $\{\text{hash(Data)}\}_{K_B^{-1}}$ où K_B^{-1} est la clef privée de la banque.
- La clef secrète DES K_{CB}

La banque, le terminal et l'utilisateur ⋆⋆

La banque, le terminal et l'utilisateur

- L'utilisateur connaît le code PIN
- La banque possède :
  - La clef publique K_B
  - La clef privée K_B^{-1}
  - La clef secrète K_{CB}
- Le terminal possède :
  - La fonction hash
  - La clef publique K_B

Validation d'une transaction ⋆⋆

Validation d'une transaction

Le terminal (T) authentifie la carte (C) :

1. $C \rightarrow T$: Data, $\{\text{hash(Data)}\}_{K_B^{-1}}$

L'utilisateur (U) donne son code à la carte (C) :

2. $T \rightarrow U$: «code ?»
3. $U \rightarrow C$: PIN
4. $C \rightarrow T$: «OK»

Avant de dire «OK», la carte (C) :

- Calcule $\text{3DES}_{K_{CB}}(PIN, N)$
- Applique la fonction à sens unique hash
- Compare le résultat obtenu à ce qu'elle connaît
- Valide la transaction si les deux informations sont identiques

Validation d'une transaction

Si le montant est supérieur à 100€

★★

Validation d'une transaction

Le terminal (T) demande à la banque (B) l'autorisation pour la carte (C) :

5. $T \rightarrow B$: «Auth»

La banque (B) réalise l'authentification de la carte (C) via le terminal (T) :

6. $B \rightarrow T : N_B$
7. $T \rightarrow C : N_B$
8. $C \rightarrow T : U, \{N_B\}_{K_{CB}}$
9. $T \rightarrow B : U, \{N_B\}_{K_{CB}}$

La banque (B) donne l'autorisation :

10. $B \rightarrow T$: «OK»

Failles de la carte bancaire ⋆⋆

Failles de la carte bancaire

Initialement, la sécurité du système reposait sur :

- La non-réplicabilité de la carte
- Le secret autour des clefs employées, du protocole

Mais

- Faille cryptographique : les clefs RSA 320 bits utilisées ne sont plus sûres (1988)
- Faille logique du protocole : pas de lien entre PIN et authentification
- Faille physique : réplicabilité, YesCard (Humpich, 1998)

Failles de la carte bancaire ⋆⋆

YesCard

Exploitation des trois failles :

1. $C \rightarrow T$: ZZZZ, $\{\text{hash}(\text{ZZZZ})\}_{K_B^{-1}}$ (fausse signature RSA)
2. $T \rightarrow U$: «code ?»
3. $U \rightarrow C$: 0000
4. $C \rightarrow T$: «OK» (la carte dit toujours oui)

C'est tout pour le moment...

Des questions ?