

Sécurité

*Une introduction à la cryptographie
et aux protocoles cryptographiques*

Roberto Amadio

Master – Université Paris Diderot (Paris 7)

2011-2012

1

3. *Travail pratique*. A titre indicatif :

- (a) exercice élémentaire de crypto-analyse,
- (b) utilisation de bibliothèques (PGP, SSL),
- (c) programmation d'algorithmes cryptographiques en Java,
- (d) analyse de protocoles (avec AVISPA).

3

Objectifs du cours

1. Introduction à la *cryptographie* (pré-requis : mathématiques élémentaires).
2. Application de concepts cryptographiques au projet de protocoles. Par exemple, les *protocoles d'authentification*, les protocoles à connaissance zéro,...

2

Cryptographie et sécurité

Citation 1 Dans sa présentation pour le prix Turing, Shamir prédit que la sécurité basée sur autre chose que les principes cryptographiques restera un bazar (*will remain a mess*).

4

Citation 2 Si vous pensez que la cryptographie est la solution à votre problème alors vous ne comprenez pas la cryptographie et vous ne comprenez pas votre problème (attribué à Lampson et/ou Needham).

5

Remarques

- Les ponts étaient construits bien avant la découverte du calcul différentiel (certains sont encore debout!).
- Aujourd’hui, nombreuses personnes qui travaillent à la construction de ponts ignorent le calcul différentiel.
- Cependant, certaines personnes travaillant à la construction de ponts doivent avoir une bonne compréhension du calcul différentiel.

7

Point de vue de l’enseignant

Cryptographie : Sécurité d’un Système Informatique
=
Équations Différentiels : Construction de Ponts

6

Ceci n’est pas un cours sur...

- ... l’administration et la gestion de réseaux.
- ... la description détaillée de protocoles SSL, SSH, IPSEC,...

D’autres cours du Master sont dédiés à ces sujets.

8

Contrôle des connaissances

- 50% *travaux pratiques avec contrôle continu*,
- 50% *examen final écrit*.

NB Les questions posées à l'examen écrit seront des simples variations d'exercices proposés dans la cours. Il est donc conseillé de résoudre ces exercices !

9

De quoi s'agit-il ?

La *cryptologie* se décompose en deux activités :

- *Crypto-graphie* : l'art de chiffrer les communications.
- *Crypto-analyse* : l'art d'analyser les communications chiffrées.

NB En pratique, on utilise le terme *cryptographie* pour se référer aux deux activités.

11

Introduction à la Cryptographie

10

- Une longue histoire et beaucoup d'histoires amusantes.
- Pendant longtemps un art et une activité mystérieuse.
- Facteurs de changements : introduction des ordinateurs, intérêt commercial, consolidation mathématique.
- Une variété d'ingrédients techniques : théorie des nombres, probabilités, théorie de l'information et codage, théorie de la complexité du calcul.

12

Références bibliographiques

- *Introduction à la cryptographie*, par Johannes A. Buchmann, publié par Dunod. Jolies mathématiques mais perspective étroite.
- *Cryptographie : Théorie et pratique*, deuxième édition, par Douglas Stinson, publié par Eyrolles. Un peu moins systématique côté math mais avec une perspective plus large (pas vraiment pratique).
- *Modern cryptography, theory and practice*, Wenbo Mao, Prentice-Hall. Couvre une grande partie du cours, parfois un peu prolix.

13

- *Handbook of applied cryptography*, Disponible en ligne. <http://www.cacr.math.uwaterloo.ca/hac/>. Encyclopédique.
- *Applied Cryptography : Protocols, Algorithms, and Source Code in C*, Second Edition, par Bruce Schneier, publié par John Wiley & Sons (disponible aussi en Français). Flou sur les fondements mais avec tous les détails de mise en oeuvre.
- *Security Engineering*, Ross Anderson, Wiley. Disponible en ligne. <http://www.cl.cam.ac.uk/~rja14/book.html> Un survol à haut-niveau sur l'état de l'art en sécurité. Utile pour réaliser qu'une fois que le protocole cryptographique est au point, il y a encore des attaques dont il faut se soucier.

14

Quelles propriétés la cryptographie peut-elle garantir ?

Voici une *tâche élémentaire* :

- A souhaite envoyer un *message* à B sur un *canal* où un attaquant C peut *écouter* (*eavesdrop*).
- A et B se mettent d'accord sur une méthode pour *chiffrer* et *déchiffrer* le message.
- A chiffre le message avant de l'envoyer.
- B déchiffre le message après l'avoir reçu.

NB Il n'est pas si simple de préciser ce que veut dire que l'attaquant comprend/obtient l'information contenue dans le message.

15

Plusieurs variations sont possibles :

- **Attaquant actif** : C peut intercepter et remplacer des messages.
- **Authentification** : B veut être sûr que le message qu'il reçoit vient de A .
- **Anonymat** : A et B veulent voter sans révéler leur identité.
- **Non-répudiation** : B peut prouver que A a effectué une commande.

16

Une définition de système cryptographique

Un vecteur $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, où :

– $\mathcal{P}, \mathcal{C}, \mathcal{K}$ sont des *ensembles finis*.

– $\mathcal{E} = \{E_k \mid k \in \mathcal{K}\}, \mathcal{D} = \{D_k \mid k \in \mathcal{K}\}$: familles indexées sur les clefs telles que :

$$E_k : \mathcal{P} \rightarrow \mathcal{C} \quad D_k : \mathcal{C} \rightarrow \mathcal{P} \\ \forall k_1 \exists k_2 D_{k_2} \circ E_{k_1} = id_{\mathcal{P}}$$

NB E_k est toujours *injective*.

17

Terminologie

Français	English
Texte clair	Plaintext
Texte chiffré	Ciphertext
Clef	Key
Déchiffrement	Decryption
Chiffrement	Encryption
Système cryptographique	Cryptographic system
Texte chiffré connu	Known cipher text
Texte clair connu	Known plain text
Texte clair choisi	Chosen plain text
Texte chiffré choisi	Chosen cipher text

19

Remarques sur les systèmes cryptographiques

- Tous les espaces sont supposés *finis*.
- Pour traiter les *messages de longueur arbitraire*, il faut les couper en segments de taille bornée.
- L'attaquant C comprend *quand un texte clair est significatif*.
- Les algorithmes de chiffrement et déchiffrement sont *publics*.

18

Systèmes symétriques

- Il est facile de calculer la clef de déchiffrement à partir de la clef de chiffrement (et vice versa).
- Utilisation :
 1. A génère de façon aléatoire une nouvelle clef k .
 2. A communique la nouvelle clef k à B en utilisant un canal sûr.
 3. A et B échangent des messages en utilisant la clef k .

20

- Exemples : DES, AES,...
- Avantages : Chiffrement et déchiffrement rapide. Petites Clefs.
- Inconvénients : Un canal sûr est nécessaire pour initialiser la communication. Pour chaque correspondant, il faut maintenir une clef secrète.

21

- Exemples : *RSA*, El Gamal, ...
- Avantages : il ne faut pas se soucier du secret de la clef (mais il faut quand même être sûr que la clef est authentique).
- Inconvénient : Moins efficace que les systèmes symétriques (au moins, actuellement).

NB Plusieurs protocoles utilisent les deux systèmes. Par exemple, on utilise une clef asymétrique avec *une longue durée de vie* pour déterminer une clef symétrique pour *une session seulement*.

23

Systèmes asymétriques

- Les clefs de chiffrement et déchiffrement sont différentes et il est difficile de calculer l'une à partir de l'autre.
- Utilisation :
 1. B génère un couple de clefs (k_1, k_2) telles que $D_{k_2} \circ E_{k_1} = id_{\mathcal{P}}$.
 2. B rend k_1 publique et maintient k_2 secrète.
 3. A peut apprendre la clef publique k_1 , chiffrer des messages avec elle et les envoyer à B . Dans ce cas, seulement B sera capable de déchiffrer ces messages en utilisant la clef secrète k_2 .

22

Cryptographie, avant l'avent de l'ordinateur

- Texte clair comme un mot de l'alphabet $\Sigma = \{A, B, \dots, Z\}$.
- De façon équivalente $\mathbf{Z}_{26} = \{0, 1, \dots, 25\}$.
- Espaces, ponctuations, lettres minuscules et majuscules sont ignorées.
I wake up early devient *IWAKEUPEARLY*
- Les langages naturels sont redondants.

24

Système de César (système à décalage)

– $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbf{Z}_{26}$.

– Pour $k \in \mathbf{Z}_{26}$ on définit :

$$E_k(p) = (p + k) \bmod 26 \quad D_k(c) = (c - k) \bmod 26 .$$

25

Attaque par énumération

– Hypothèse : l'attaquant sait quand un texte suffisamment long est un texte significatif.

– Étant donné un message m

$$m = c_1 \dots c_n$$

– On essaye les 26 clefs et on calcule

$$(c_1 - k) \bmod 26 \dots (c_n - k) \bmod 26 .$$

– Morale : l'espace des clefs est trop petit.

27

– On vérifie :

$$\begin{aligned} D_k(E_k(p)) &= ((p + k) \bmod 26 - k) \bmod 26 \\ &= (p + k - k) \bmod 26 \\ &= p \end{aligned}$$

– Le système de César est un exemple simple de système symétrique.

– Le décalage est appliqué lettre par lettre.

26

Exercice (sur les système de César)

Le texte chiffré VHFUHW a été généré avec le système de César. Calculer la clef et le texte clair (qu'on sait être en Anglais).

28

Système de substitution

On élargit l'espace des clefs.

– Soit

$$\mathcal{P} = \mathcal{C} = \mathbf{Z}_{26} \quad \mathcal{K} = \{\pi : \mathbf{Z}_{26} \rightarrow \mathbf{Z}_{26} \mid \pi \text{ permutation}\}$$

– Étant donnée une permutation π , on définit :

$$E_{\pi}(p) = \pi(p) \quad D_{\pi}(c) = \pi(c) .$$

– Si π^{-1} est l'inverse de la permutation alors :

$$D_{\pi^{-1}}(E_{\pi}(p)) = (\pi^{-1} \circ \pi)(p) = p .$$

29

– Il y a un nombre fini de langages naturels.

– Étant donné un texte suffisamment long, la lettre qui est présente le plus souvent correspond probablement à e , et ainsi de suite...

– Comme les langages naturels sont redondants, des qu'un nombre limité de substitutions ont été déterminées, il est possible de deviner celles qui restent.

– Il y a des statistiques aussi sur la fréquence de digrammes, tri-grammes,... Par exemple, THE est un tri-gramme qu'on trouve souvent en Anglais.

31

Attaque par analyse de fréquence

L'espace des clefs est maintenant $26!$, ce qui est considérable.

Cependant...

– La fréquence des lettres dans les langages naturels a été bien étudiée. Par exemple,

Lettre	—	English	—	Français
e	—	12.7	—	17.8
a	—	8.2	—	8.2
o	—	7.5	—	5.7
i	—	7.0	—	7.3
t	—	9.1	—	7.3
s	—	6.3	—	8.3

30

Masque jetable (one time pad)

– Un message est maintenant représenté par une séquence de bits. Par exemple, codes ASCII de caractères du clavier.

– Si le texte clair a longueur n , on prend :

$$\mathcal{P} = \mathcal{C} = \mathcal{K} = \{0, 1\}^n .$$

– Ces ensembles dépendent de la longueur du message qu'on veut chiffrer !

32

Fonction xor

– Le xor (ou exclusif) $\oplus : \{0, 1\}^2 \rightarrow \{0, 1\}$

x_1	x_2	$ \oplus$
0	0	0
0	1	1
1	1	0
1	0	1

– On étend à $\{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$ point à point.

33

Masque jetable (suite)

– Étant donné, $k \in \mathcal{K} = \{0, 1\}^n$, on définit :

$$E_k : \{0, 1\}^n \rightarrow \{0, 1\}^n \quad E_k(p) = p \oplus k$$

$$D_k : \{0, 1\}^n \rightarrow \{0, 1\}^n \quad D_k(c) = c \oplus k$$

– On observe que \oplus est associatif, $k \oplus k = 0$ et $p \oplus 0 = p$.

– Donc $D_k(E_k(p)) = (p \oplus k) \oplus k = p$.

34

Espace de probabilité

On rappelle certains faits. Un *espace de probabilité* (Ω, \mathcal{A}, P) est un triplet où :

– Ce système est attribué à Gilbert Vernam (1917).

– Il s’agit d’un système symétrique où la longueur de la clef est supérieure ou égale à la longueur du message envoyé.

– On dit que les communications ‘top secret’ entre USA et URSS étaient chiffrées en utilisant ce système.

– Ω est un ensemble non-vidé.

– \mathcal{A} est un sous-ensemble non vide d’*événements* de 2^Ω , stable par complément et unions (et intersections) dénombrables.

– $P : \mathcal{A} \rightarrow [0, 1]$ est une fonction qu’on appelle *mesure de probabilité* telle que $P(\Omega) = 1$ et telle que P est additive par rapport à des unions dénombrables d’événements disjoints entre eux :

$$P\left(\bigcup_{i \geq 0} A_i\right) = \sum_{i \geq 0} P(A_i)$$

35

36

Probabilité conditionnelle

Soient A, B deux événements sur un espace de probabilité (Ω, \mathcal{A}, P) tel que $P(A) > 0$. Alors

– $P(B|A) = P(B \cap A)/P(A)$ est la *probabilité conditionnelle* de B étant donné A .

– On remarquera que si $P(A) = 0$ alors $P(B|A)$ n'est pas définie.

– *Formule de Bayes* : en supposant $P(A), P(B) > 0$:

$$P(B)P(A|B) = P(A)P(B|A)$$

37

Analyse de la masque jetable

– Cette analyse est due à *Claude Shannon* (1949), le père de la théorie de l'information.

– Il s'agit du premier fait non-trivial que vous apprendrez dans ce cours !

– On suppose que l'espace de probabilité est défini sur $\Omega = \mathcal{P} \times \mathcal{K}$.

39

Indépendance

– Deux événements A, B sont *indépendants* si

$$P(A \cap B) = P(A)P(B).$$

– En particulier, ceci implique que $P(A|B) = P(A)$ si $P(B) > 0$.

38

Événements à considérer

– $(p, k) \equiv \{(p, k)\}$ est l'événement où le texte clair p est chiffré avec la clef k .

– $p \equiv \{(p, k) \mid k \in \mathcal{K}\}$ est l'événement où le texte clair p est chiffré.

– $k \equiv \{(p, k) \mid p \in \mathcal{P}\}$ est l'événement où la clef k est utilisée pour chiffrer un texte clair.

– $c \equiv \{(p, k) \mid E_k(p) = c\}$ est l'événement où le texte chiffré est c .

40

Hypothèses

- Les événements p et k sont *indépendants*, c'est-à-dire

$$P(p \cap k) = P(p, k) = P(p)P(k) .$$

- L'attaquant connaît $P(p)$, pour tout $p \in \mathcal{P}$.

NB Ceci est en faveur de l'attaquant qui a une connaissance parfaite de la fréquence de tout texte clair de longueur n !

Exemple

- $\mathcal{P} = \{0, 1\}$, $\mathcal{K} = \{A, B\}$, $\mathcal{C} = \{a, b\}$.

- Les fonctions de chiffrement sont définies par

p	k	$E_k(p)$
0	A	a
1	A	b
0	B	b
1	B	a

Sécurité parfaite

- On dit qu'un système cryptographique est *parfaitement sûr* si le fait d'observer un texte chiffré c ne fournit aucune information (probabiliste) sur le texte clair correspondant p .

- En d'autres termes, le système est parfaitement sûr si pour tout $p \in \mathcal{P}$ et $c \in \mathcal{C}$ avec $P(c) > 0$

$$P(p | c) = P(p)$$

- On remarquera que ceci implique que p et c sont indépendants car

$$P(p \cap c)/P(c) = P(p) \Rightarrow P(p \cap c) = P(p)P(c) .$$

- Les probabilités sont les suivantes :

p	P	k	P
0	1/4	A	1/4
1	3/4	B	3/4

- La probabilité dérivée du texte chiffré c :

c	P
a	$10/16$
b	$6/16$

- Non !
- Par exemple, supposons que l'attaquant voit le texte chiffré a .
- La probabilité que le texte clair est 1 est $9/10$.
- Ce qui est supérieur à $3/4$ qui est la probabilité qu'un texte clair est 1.

- Les autres probabilités dérivées :

p	c	$P(p \cap c)$	$P(p \cap c)/P(c)$
0	a	$1/16$	$1/10$
1	a	$9/16$	$9/10$
0	b	$3/16$	$1/2$
1	b	$3/16$	$1/2$

- Ce système est-il parfaitement sûr ?

Théorème

Soit donné un système cryptographique tel que :

- $\#\mathcal{P} = \#\mathcal{C} = \#\mathcal{K}$ et

- pour tout $p \in \mathcal{P}$ et $c \in \mathcal{C}$, $P(p), P(c) > 0$.

Alors le système est parfaitement sûr ssi

1. Les clés sont sélectionnées avec une probabilité uniforme $P(k) = 1/\#\mathcal{K}$.
2. $\forall p, c \exists !k (E_k(p) = c)$.

Preuve : Parfaitement sûr implique 1 et 2

– Pour tout texte clair p , on doit avoir que

$$\#\{E_k(p) \mid k \in \mathcal{K}\} = \#\mathcal{C}$$

– S’il y avait un texte chiffré c qui diffère de $E_k(p)$ pour tout k , alors

$$P(c \mid p) = 0 \neq P(c) > 0 .$$

ce qui contredit l’hypothèse de sécurité parfaite. Pourquoi ?

– Ainsi pour tout texte clair p et texte chiffré c il existe unique $k(p, c) \in \mathcal{K}$ tel que $E_{k(p,c)}(p) = c$.

Preuve : 1 et 2 impliquent la sécurité parfaite

Il suffit d’observer que :

$$\begin{aligned} P(p \mid c) &= P(p)P(c \mid p)/P(c) && \text{(par Bayes)} \\ &= P(p)P(k(p, c))/\sum_{q \in \mathcal{P}} P(q)P(k(q, c)) && \text{(par hyp. (2))} \\ &= (P(p)1/\#\mathcal{K})/(1/\#\mathcal{K}(\sum_{q \in \mathcal{P}} P(q))) && \text{(par hyp. (1))} \\ &= P(p) . \end{aligned}$$

– On remarque que $\mathcal{K} = \{k(p, c) \mid p \in \mathcal{P}\}$.

– Maintenant, on démontre que $P(k(p, c))$ est constante par rapport à p , ce qui implique que $P(k(p, c)) = 1/\#\mathcal{K}$.

– Pour démontrer cette propriété, on calcule :

$$\begin{aligned} P(p) &= P(p \mid c) && \text{(par sécurité parfaite)} \\ &= P(c \mid p)P(p)/P(c) && \text{(par Bayes)} \\ &= P(k(p, c))P(p)/P(c) && \text{(par définition).} \end{aligned}$$

– Ainsi $P(k(p, c)) = P(c)$ pour tout texte clair p .

Corollaire

Le système de la masque jetable avec choix uniforme des clefs est parfaitement sûr.

– On vérifie sur $\{0, 1\}$ que $\forall c, p \exists! k (p \oplus k = c)$.

– Ceci se généralise de suite à $\{0, 1\}^n$.

Sommaire

Nous avons analysé deux systèmes ‘extrêmes’ :

- Systèmes qu’un élève de primaire peut utiliser et casser (décalage, substitution).
- Systèmes qui sont incassables (en théorie!) mais dont l’utilisation n’est pas très pratique.

53

Survol

- Les ‘fonctions cryptographiques’ ont plusieurs applications qui vont bien au delà de la simple tâche de protéger l’information que nous avons considéré pour l’instant.
- Comme exemple simple, nous considérons le problème de *jouer à pile ou face au téléphone*.
- Le protocole considéré se base sur le concept cryptographique de *fonction à sens unique*.

NB Ceci est un aperçu de la part du cours qui traite de protocoles cryptographiques.

55

Un exemple simple de protocole cryptographique

54

Fonctions à sens unique

Soit $f : D \rightarrow D$ une fonction avec un (grand) domaine D . On dit que f est à sens unique (*one way function*) si :

1. Étant donné x , il est *facile* de calculer $f(x)$.
2. Étant donné y , il est *difficile* de trouver x tel que $f(x) = y$ (Inversion).
3. Il est *difficile* de trouver $x_1 \neq x_2$ tels que $f(x_1) = f(x_2)$ (Collision).

NB On analysera ces notions plus tard quand on étudiera les *fonctions de hachage*.

56

- Bien sûr, on a *pas* donné une définition : facile et difficile n'ont pas de signification précise.
- Une possibilité est d'établir que difficile signifie *pas en temps probabiliste polynomial*.
- Sous cette hypothèse, l'existence d'une fonction à sens unique est un problème ouvert majeur de la cryptographie.

NB Il s'agit d'une caractéristique du domaine : plusieurs protocoles se basent sur la conjecture que certaines fonctions sont *difficiles* à calculer (par exemple, la fonction qui calcule les facteurs premiers d'un nombre).

Un peu de notation

- $(\nu x \in D)$: affecte à x une valeur aléatoire du domaine D . Par exemple, $D = \mathbf{Z}_m = \{0, \dots, m - 1\}$.
- $A!exp$: envoie à A un message qui résulte de l'évaluation de l'expression exp .
- $B?x$: on reçoit de B un message dont la valeur est affectée à la variable x .

Hypothèses pour le protocole

- A et B veulent jouer à pile ou face au téléphone.
- La ligne de téléphone est 'sécurisée' : A et B sont sûrs de parler entre eux.
- A et B se mettent d'accord pour utiliser une fonction f à sens unique.
- A et B ne se font pas confiance (il viennent de divorcer et doivent décider qui garde la maison...).

Une description du protocole

A	B
(1) $(\nu x_1 \in \mathbf{Z}_m)B!f(x_1)$	$\rightarrow A?y_1.$
(2) $B?x_2.$	$\leftarrow (\nu y_2 \in \{0, 1\})A!y_2$
(3) $B!x_1.$	$\rightarrow A?y_3.$
(4) if $x_1 \bmod 2 = x_2$	if $f(y_3) = y_1$ then
then LOSE	if $y_3 \bmod 2 = y_2$ then WIN
else WIN	else LOSE

Remarques

- Chaque participant au protocole *exécute en parallèle* un certain nombre d'instructions.
- Dans le protocole en question, la communication est *fiable* et *synchrone* (handshaking).
- En général, la communication est *asynchrone* (Internet...) et *non fiable*.
- Ici on suppose que *A commence le protocole*, qu'il y a seulement *deux participants* et que le protocole est exécuté *une fois*.
- En général, on peut avoir *plusieurs participants*, le protocole peut être *exécuté plusieurs fois* (éventuellement en parallèle) et chaque participant peut *jouer différents rôles* (celui qui démarre, celui qui répond, ...). En plus, certains participants peuvent être *malhonnêtes*.

61

Exercice (sur jouer à pile ou face)

1. On suppose que *B* a la capacité d'inverser *f*. Comment *B* peut-il tricher (en effectuant un calcul additionnel)?
2. On suppose que *A* a la capacité de trouver des collisions dans *f*. Comment *A* peut-il tricher (en effectuant un calcul additionnel)?
3. On suppose que *f* est à sens unique. Pourquoi *B* devrait-il avoir confiance en *A*?
4. On suppose que le générateur aléatoire de *A* est biaisé alors que celui de *B* est équitable. Par exemple, *A* choisit plus souvent un nombre pair. Les chances que *A* gagne sont-elles affectées? Quid si *B* est biaisé et *A* équitable?
5. Maintenant on suppose que les deux générateurs aléatoires de *A* et *B* sont biaisés. Les chances que *A* gagne sont-elles affectées?

62

Exercice

On considère la fonction $f : (\mathbf{Z}_2)^{200} \rightarrow (\mathbf{Z}_2)^{200}$ définie par :

$$f(x_1, \dots, x_{200}) = (x_1 \oplus x_{101}, \dots, x_{100} \oplus x_{200}, 0, \dots, 0)$$

La fonction en question pourrait-elle être à sens unique?

63

Arithmétique Modulaire

64

- Pour apprécier les algorithmes cryptographiques il est nécessaire de rappeler des notions d'*arithmétique modulaire*.
- Probablement, vous connaissez déjà les notions introduites. Les faits faciles sont laissés comme *exercices*.
- *Attention* : vous aurez besoin de ces notions pour comprendre le premier travail pratique.

65

Terminologie

- Par défaut, on travaille sur les nombres entiers \mathbf{Z} .
- Si $x \in \mathbf{R}$ alors $\lfloor x \rfloor = \max\{a \in \mathbf{Z} \mid a \leq x\}$.
NB $\lfloor 3.5 \rfloor = 3$ et $\lfloor -3.5 \rfloor = -4$.
- On dit que a divise b et on écrit $a \mid b$ si $\exists c \in \mathbf{Z} \ ca = b$.

67

Synthèse des résultats principaux

Quotient et reste Soient $a, b \in \mathbf{Z}$ avec $b > 0$. Alors

$$\exists !q, r \ (a = qb + r, 0 \leq r < b).$$

Théorème de Bezout Soient $a, b \in \mathbf{Z}$. Alors

$$a\mathbf{Z} + b\mathbf{Z} = \text{pgcd}(a, b)\mathbf{Z}$$

Algorithme d'Euclide Soient $a, b \in \mathbf{Z}$. Alors on peut calculer x, y tels que

$$\text{pgcd}(a, b) = ax + by$$

Inverse dans \mathbf{Z}_m Soient $a, m \in \mathbf{Z}$, $m \geq 2$. Alors $(ax \equiv 1) \text{ mod } m$ a une solution ssi $\text{pgcd}(a, m) = 1$ (et dans ce cas l'algorithme d'Euclide permet de calculer x).

66

Exercice (propriétés de la relation 'divise')

1. $a \mid 0$.
2. $0 \mid a$ implique $a = 0$.
3. Si $a \mid b$ et $b \mid c$ alors $a \mid c$.
4. Si $c \mid a$ et $c \mid b$ alors pour tout d, e ($c \mid da + eb$).
5. Si $a \mid b$ et $b \neq 0$ alors $|a| \leq |b|$.
6. Si $a \mid b$ et $b \mid a$ alors $|a| = |b|$.

68

Quotient et reste

Soient $a, b \in \mathbf{Z}$ avec $b > 0$. Alors

$$\exists! q, r (a = qb + r, 0 \leq r < b).$$

Notation on écrit $a \bmod b$ pour le reste r .

Existence

- Soit $r = a - \lfloor a/b \rfloor b$.
- On vérifie $a = \lfloor a/b \rfloor b + (a - \lfloor a/b \rfloor b)$ et $0 \leq r < b$.

69

- Ceci implique $\lfloor a/b \rfloor = q$ et q ne dépend pas de r .
- Donc si $a = q_1 b + r_1 = q_2 b + r_2$, avec $0 \leq r_1, r_2 < b$.
- Alors $q_1 = q_2 = \lfloor a/b \rfloor$.
- En conséquence $r_1 = r_2$.

71

Unicité

- On suppose $0 \leq r < b$ et $q \in \mathbf{Z}$ tel que $a = qb + r$.
- Alors $r = (a - qb)$ et $0 \leq (a - qb) < b$.
- Comme $b > 0$, on dérive $0 \leq a/b - q < 1$.
- C'est-à-dire $q \leq a/b < q + 1$ avec $q \in \mathbf{Z}$ et $a/b \in \mathbf{Q}$.

70

Plus grand commun diviseur (pgcd)

- Si $c \mid a$ et $c \mid b$ on dit que c est un diviseur commun de a et b .
- Si $a \neq 0$ ou $b \neq 0$ alors on sait que les diviseurs de a et b sont bornés en valeur absolue par $\max(|a|, |b|)$.

- Donc on peut définir le plus grand commun diviseur (pgcd) comme

$$\text{pgcd}(a, b) = \max\{c \mid (c \mid a) \text{ et } (c \mid b)\}.$$

- Si $a = b = 0$ alors le plus grand commun diviseur n'existe pas car $\forall c \ c \mid 0$ et alors *par convention* on établit que $\text{pgcd}(0, 0) = 0$.

72

Théorème de Bezout

Par convention, soit

- $k\mathbf{Z} = \{kc \mid c \in \mathbf{Z}\}$.
- $X + Y = \{x + y \mid x \in X \text{ et } y \in Y\}$.

Le théorème suivant caractérise les combinaisons linéaires de deux nombres comme les multiples du pgcd.

Théorème Soient $a, b \in \mathbf{Z}$. Alors

$$a\mathbf{Z} + b\mathbf{Z} = \text{pgcd}(a, b)\mathbf{Z}$$

73

- D'autre part, soit $c = ak_1 + bk_2 \neq 0$.

- On sait que $\exists q, r \ c = qg + r, 0 \leq r < g$.

- Donc

$$\begin{aligned} r &= ak_1 + bk_2 - q(ax + by) \\ &= a(k_1 - qx) + b(k_2 - qy) \in a\mathbf{Z} + b\mathbf{Z} \end{aligned}$$

- Comme g est le plus petit entier positif, on doit avoir que $r = 0$.

Donc $c = qg \in g\mathbf{Z}$ et $a\mathbf{Z} + b\mathbf{Z} \subseteq g\mathbf{Z}$.

75

Preuve du théorème

- Soient $A = a\mathbf{Z} + b\mathbf{Z}$ et $I = \text{pgcd}(a, b)\mathbf{Z}$.

- Si $a = b = 0$ alors $A = \{0\} = I$.

- Donc on suppose $a \neq 0$ ou $b \neq 0$ et on définit

$$g = \min\{xa + yb \mid x, y \in \mathbf{Z}, xa + yb > 0\}$$

- Clairement, $g\mathbf{Z} \subseteq a\mathbf{Z} + b\mathbf{Z}$.

74

- Ensuite, on montre que $g = \text{pgcd}(a, b)$.

- Comme $a, b \in g\mathbf{Z}$, il suit que $g \mid a$ et $g \mid b$.

- D'autre part, si $d \mid a$ et $d \mid b$ alors $d \mid ax + by = g$.

- Comme $g \neq 0$, on sait que $|d| \leq g$, donc g est le plus grand commun diviseur.

76

Exercice (pgcd)

1. Montrez que l'équation $ax + by = c$ a une solution en x, y ssi $\text{pgcd}(a, b) \mid c$.
2. Pour tout $a, b \in \mathbf{Z}$, ils existent $x, y \in \mathbf{Z}$ tels que $\text{pgcd}(a, b) = ax + by$.
3. Soient $a, b \in \mathbf{Z}$ avec $\text{pgcd}(a, b) > 0$. Alors $d \mid a$ et $d \mid b$ implique que $d \mid \text{pgcd}(a, b)$.

77

3. Donc on dérive l'algorithme suivant pour $a, b \geq 0$:

$$\text{pgcd}(a, b) = \begin{cases} a & \text{if } b = 0 \text{ then } a \\ \text{pgcd}(b, a \bmod b) & \text{else } \end{cases}$$

Calculez $\text{pgcd}(-100, -35)$.

79

Exercice (algorithme d'Euclide)

Soient $a, b \in \mathbf{Z}$. Montrez que :

1. Si $b = 0$ alors $\text{pgcd}(a, b) = |a|$.
2. Autrement, on suppose $b \neq 0$. Par le théorème du quotient et reste on sait :

$$a = q|b| + (a \bmod |b|)$$

Montrez que :

- d divise a et b ssi il divise $|b|$ et $a \bmod |b|$.
- $\text{pgcd}(a, b) = \text{pgcd}(|b|, a \bmod |b|)$.

78

Algorithme d'Euclide modifié

On modifie l'algorithme pour qu'il produise les coefficients x, y tels que $\text{pgcd}(a, b) = ax + by$.

Pas 0	Pas 1	Pas $k \geq 1$ for $r_k \neq 0$
$r_0 = a $	$r_1 = b $	$r_{k+1} = r_{k-1} \bmod r_k$
	$q_1 = \lfloor r_0/r_1 \rfloor$	$q_{k+1} = \lfloor r_k/r_{k+1} \rfloor$
$x_0 = 1$	$x_1 = 0$	$x_{k+1} = q_k x_k + x_{k-1}$
$y_0 = 0$	$y_1 = 1$	$y_{k+1} = q_k y_k + y_{k-1}$

Si $r_k = 0$ alors on retourne :

$$\text{pgcd}(a, b) = r_{k-1}, \quad x = (-1)^{k-1} x_{k-1}, \quad y = (-1)^k y_{k-1}$$

80

Exemple : Euclide modifié, initialisation

L'application des règles à 100 et 35 donne :

k	0	1
r_k	100	35
q_k	-	2
x_k	1	0
y_k	0	1

81

Exemple : Euclide modifié, pas 2

k	0	1	2
r_k	100	35	30
q_k	-	2	1
x_k	1	0	1
y_k	0	1	2

82

Exemple : Euclide modifié, pas 3

k	0	1	2	3
r_k	100	35	30	5
q_k	-	2	1	6
x_k	1	0	1	1
y_k	0	1	2	3

83

Exemple : Euclide modifié, pas 4 et fin

k	0	1	2	3	4
r_k	100	35	30	5	0
q_k	-	2	1	6	
x_k	1	0	1	1	
y_k	0	1	2	3	

Donc $\text{pgcd}(100, 35) = 5 = -1 \cdot 100 + 3 \cdot 35$.

84

Exercice (Euclide modifié)

Appliquez l'algorithme pour déterminer x, y tels que $\text{pgcd}(91, 143) = 91x + 143y$.

85

Nombres premiers

Un nombre naturel $p > 1$ est *premier* s'il admet comme seules diviseurs *positifs* 1 et p .

87

Suggestions sur la preuve d'Euclide modifié

Pour démontrer la correction, on doit vérifier par récurrence que :

$$r_k = (-1)^k x_k a + (-1)^{k+1} y_k b$$

NB Il n'est pas trop difficile de montrer que l'algorithme dérivé termine en $O(|a||b|)$.

86

Exercice (nombres premiers)

Montrez que :

1. Chaque nombre $a > 1$ a un diviseur premier.
2. Si un premier divise un produit de nombres alors il divise au moins un des facteurs.
3. Si un nombre premier p divise un produit de nombres premiers $q_1 \cdots q_k$ alors il est égal au moins à un des facteurs.
4. Chaque entier $a > 1$ peut être écrit comme un produit de nombres premiers. Cette représentation est unique à permutation de facteurs près.

88

Congruences

On dit que a est congru à b modulo m et on écrit $(a \equiv b) \bmod m$ si $m \mid (a - b)$ (ou de façon équivalente $m \mid (b - a)$).

89

Notation

- L'ensemble des classes de résidus modulo m est dénotée par $\mathbf{Z}/m\mathbf{Z}$.
- Il a m éléments.
- On travaillera avec un représentant par classe $\mathbf{Z}_m = \{0, \dots, m - 1\}$.

91

Exercice (congruence)

1. Montrez que la relation $(\cdot \equiv \cdot) \bmod m$ est une relation d'équivalence.
2. Montrez que les conditions suivantes sont équivalentes :
 - (a) $(a \equiv b) \bmod m$,
 - (b) $a = b + km$, et
 - (c) $a \bmod m = b \bmod m$.

90

Exercice (arithmétique dans \mathbf{Z}_m)

Montrez que si $(a \equiv b) \bmod m$ et $(c \equiv d) \bmod m$ alors :

1. $(-a \equiv -b) \bmod m$,
2. $(a + c \equiv b + d) \bmod m$, et
3. $(ac \equiv bd) \bmod m$.

92

Exercice (structure d'anneau dans \mathbf{Z}_m)

1. On considère la structure $(\mathbf{Z}_m, 0, +, (-)^{-1})$ où $+$ est l'addition modulaire $(a + b) \bmod m$ et $a^{-1} = m - a$. Montrez que ceci est un *groupe abélien*.
2. On considère le produit modulaire $(ab) \bmod m$. Montrez que ceci donne un *monïde commutatif* avec unité 1.
3. Par rapport à l'addition et multiplication modulaire, montrez que $a(b + c) = (ab) + (ac)$ in \mathbf{Z}_m .

NB Conclure que $(\mathbf{Z}_m, +, \cdot)$ est un *anneau*.

93

Si la congruence a une solution alors a et m sont premiers entre eux

- Soit $(ax \equiv 1) \bmod m$, c'est-à-dire $(ax - 1) = km$.
- Soit $g = \text{pgcd}(a, m)$.
- Comme $1 = km - ax$, $g \mid m$ et $g \mid a$, on dérive $g \mid 1$, c'est-à-dire $g = 1$.

95

Quand \mathbf{Z}_m est-il un corps ?

En d'autres termes, dans quel cas chaque $a \neq 0$ admet un x tel que $(ax \equiv 1) \bmod m$?

Théorème La congruence $(ax \equiv 1) \bmod m$ a une solution ssi $\text{pgcd}(a, m) = 1$, c'est-à-dire, a et m sont premiers entre eux. En plus, la solution, si elle existe, est unique.

94

Si a et m sont premiers entre eux alors la congruence a une solution

- Si $g = 1$ alors $\exists x, y$ $1 = ax + my$.
- Donc x est une solution de la congruence.

96

La solution est unique

- Soit $(av \equiv 1) \pmod{m}$ une autre solution.
- Alors $(ax \equiv av) \pmod{m}$.
- C'est-à-dire $m \mid (a(x - v))$. Comme $m \nmid a$, il suit que $m \mid (x - v)$.
- Donc $(x \equiv v) \pmod{m}$.

97

Exercice (sur le chiffrement affine)

On considère les méthodes de chiffrement suivantes où $\Sigma = \mathbf{Z}_{26}$:

Méthode 1 Chaque lettre $a \in \Sigma$ est remplacée par $ka \pmod{26}$, pour $k \in \{1, \dots, 26\}$ et k impair.

Méthode 2 Chaque lettre $a \in \Sigma$ est remplacée par $ka \pmod{26}$ où $k \in \{1, \dots, 26\}$ et $\text{pgcd}(k, 26) = 1$.

1. De quelle méthode peut-on dériver un système cryptographique ?
2. Quels sont les espaces de textes clairs, de textes chiffrés et de clefs ?

99

Exercice (inverse dans \mathbf{Z}_m)

1. Montrez que l'algorithme d'Euclide modifié peut être utilisé pour calculer l'inverse.
2. Montrez que l'anneau \mathbf{Z}_m est un corps ssi m est premier.

98

Exercice (révision)

On considère les fonctions $E_k : \mathbf{Z}_{143} \rightarrow \mathbf{Z}_{143}$ définies par

$$E_k(x) = (k \cdot x) \pmod{143}$$

pour $k = 15$ et $k = 65$.

1. Pour quel k (s'il existe) E_k est susceptible de représenter une fonction de chiffrement ?
2. Si E_k est une fonction de chiffrement, quel est la fonction de déchiffrement D_k qui lui correspond ?

100

Chiffrement à bloc

101

Electronic codebook mode (ECB)

- Étant donné un chiffrement à bloc sur Σ^n et un document sur Σ , on décompose le document en blocs de longueur n .
- Si nécessaire, on complète le document avec des symboles choisis de façon aléatoire.
- Étant donnée une clef k , l'émetteur applique la même fonction de chiffrement E_k à chaque bloc et le récepteur applique la fonction de déchiffrement correspondante D_k .

103

Chiffrement à bloc

- Un système cryptographique est un *chiffrement à bloc* si

$$\mathcal{P} = \mathcal{C} = \Sigma^n$$

- Le système de César et le système par substitution sont des chiffrements à bloc où $n = 1$ et $\Sigma = \mathbf{Z}_{26}$.
- On remarquera que les fonctions de chiffrement dans les chiffrements à bloc sont toujours des permutations.
- En pratique, on cherche des classes spéciales de permutations.

102

Faiblesses du ECB

- A chaque bloc de texte clair correspond un bloc unique de texte chiffré. Il est donc possible de reconnaître un motif du texte clair en regardant le texte chiffré.
- Un attaquant actif peut substituer un bloc de texte chiffré avec un autre bloc de texte chiffré produit avec la même clef (un problème d'authentification de blocs).

Ces problèmes sont résolus par la méthode suivante (CBC).

104

Cipherblock chaining mode (CBC)

- On suppose que $\Sigma = \{0, 1\}$.
- On fixe un vecteur d'initialisation $IV \in \{0, 1\}^n$ qui peut être rendu publique.
- Le chiffrement c_1, \dots, c_t de blocs m_1, \dots, m_t est maintenant défini itérativement comme suit :

$$\begin{aligned}c_0 &= IV \\c_{j+1} &= E_k(c_j \oplus m_{j+1}) \quad j = 0, \dots, t-1\end{aligned}$$

105

Exercice

A utilise un chiffrement à bloc $E : 2^s \rightarrow 2^s$ en mode CBC et avec un vecteur initial $IV \in 2^s$. Pour chiffrer un texte clair $(x_1, x_2, \dots, x_N) \in (2^s)^N$, A procède comme suit :

$$c_0 = IV \quad c_1 = E(x_1 \oplus c_0) \quad \dots \quad c_i = E(x_i \oplus c_{i-1}) \quad \dots$$

Ensuite le message chiffré (c_0, c_1, \dots, c_N) est envoyé à B qui connaît la fonction de déchiffrement $D = E^{-1} : 2^s \rightarrow 2^s$.

107

- Le récepteur déchiffre en appliquant la définition suivante :

$$\begin{aligned}n_0 &= IV \\n_{j+1} &= D_k(c_{j+1}) \oplus c_j \quad j = 0, \dots, t-1\end{aligned}$$

- Nous avons $n_j = m_j$, car :

$$n_{j+1} = D_k(E_k(m_{j+1} \oplus c_j)) \oplus c_j = m_{j+1} \oplus c_j \oplus c_j = m_{j+1} .$$

106

1. Une erreur se présente pendant la transmission du premier bloc c_1 , ainsi B reçoit un bloc c'_1 plutôt que c_1 . Quels blocs seront déchiffrés correctement par B ?

108

2. Maintenant, on suppose que A fait une erreur dans le chiffrement du deuxième bloc, ainsi il calcule un bloc c'_2 plutôt que c_2 . Quels blocs chiffrés seront affectés par cette erreur ? Quels blocs de texte clair pourront être calculés par B , en supposant qu'il n'y ait pas d'erreur de transmission ?

Systemes lineaires affines

109

Systeme affine

Considérons un chiffrement à bloc où la longueur du bloc est $n = 1$.

- Fixons un entier positif m .
- L'espace des clefs est donné par le couple $(a, b) \in \mathbf{Z}_m^2$ tel que a et m sont premiers entre eux.
- Pour $k = (a, b)$, la fonction de chiffrement est donnée par

$$E_k(x) = (ax + b) \bmod m$$

111

110

- Si a^{-1} est l'inverse de a dans \mathbf{Z}_m (qu'on peut déterminer avec l'algorithme d'Euclide) alors

$$D_k(x) = (a^{-1}(x - b)) \bmod m$$

- On remarque que le système affine (comme le système par décalage) est un cas particulier de système par substitution.

112

- Pour $m = 26$, l'espace des clefs contient $\phi(m) * m = 312$ éléments, où $\phi(m)$ est le nombre d'éléments dans \mathbf{Z}_m premiers avec m .
- Évidemment, cet espace peut être analysé par une recherche complète.
- Une autre faiblesse : on peut utiliser une attaque par texte clair connu qui exploite la linéarité de la fonction de chiffrement. La connaissance de la fonction en deux points peut être suffisante à déterminer la clef.

113

Système de Vigenère

- Attribué à Vigenère (16^{ème} siècle).
- Généralisation du système de César où le décalage dépend de façon périodique de la position de la lettre dans le texte clair.
- On fixe une période $n \geq 1$.
- On prend

$$\mathcal{P} = \mathcal{C} = \mathcal{K} = (\mathbf{Z}_{26})^n$$

115

Exercice (attaque sur texte clair connu)

On sait qu'un système affine avec clef (a, b) envoie E sur R et S sur H.

1. Dérivez de ces hypothèses un système composé de deux congruences.
2. Essayez de résoudre le système et d'obtenir la valeur de la clef.

114

- On pose :

$$E_{\vec{k}}(\vec{p}) = (\vec{p} + \vec{k}) \bmod 26 \quad D_{\vec{k}}(\vec{c}) = (\vec{c} - \vec{k}) \bmod 26$$

- Évidemment, la fonction de chiffrement est affine.

116

- Le système de Vigenère est un premier exemple de *système poly-alphabétique* : un caractère peut être transformé dans des caractères différents selon sa position dans le texte.

- Par opposition, dans le système par substitution chaque caractère est transformé dans un seul caractère, de façon indépendante de sa position. Dans ce cas, on parle de *système mono-alphabétique*.

NB Dans un système mono-alphabétique, le diagramme de distribution des caractères aura des *pics* comme dans les langages naturels (6 lettres constituent environ moitié du texte).

Attaque par texte chiffré

- Le nombre de clefs est maintenant $(26)^n$, ce qui est déjà considérable pour $n = 10$. Ainsi une attaque par énumération semble inefficace.

- Si on connaît la période n , alors on peut disposer le texte sur n colonnes. Par exemple, pour $n = 5$

X	U	O	L	M
A	B	D	H	J
K	Z	W	J	M . . .

Exercice (Vigenère)

On suppose que $n = 6$ et que la clef est CIPHER, c.-à.-d. (2, 8, 15, 7, 4, 17). Déchiffrer le texte :

VPXZGIAXIVWPUBTTMJPWIZITWZT

- On sait que les lettres sur la même colonne dépendent du même décalage.

- Analyse de fréquence : la lettre la plus fréquente dans une colonne est probablement la lettre la plus fréquente dans le langage naturel.

- On calcule le décalage correspondant.

Comment déterminer la période ?

La système a été considéré inattaquable pendant 3 siècles jusqu'à Kasinski (1860). Il propose la méthode suivante pour déterminer la période.

1. On cherche dans le texte des répétitions du même mot (le plus long le mieux).
2. On calcule les distances entre les occurrences de chaque mot répété. Si un mot répété vient du chiffrement du même mot à partir de la même position alors les distances doivent être des multiples de la période n .
3. On calcule le *pgcd* des distances de mots (longs) répétés.
4. On applique l'analyse de fréquence à *chaque colonne*.

121

Algèbre linéaire sur les anneaux

On généralise le système affine aux espaces de dimension supérieure. D'abord on rappelle certaines propriétés d'algèbre linéaire sur un anneau.

- Soit R un anneau commutatif avec un élément 0 comme unité de l'addition et un élément 1 comme unité de la multiplication.
- Par exemple, on peut penser à \mathbf{Z}_m .
- Soit $R[m, n]$ la collection de matrices sur R avec m lignes et n colonnes, avec éléments génériques A, B, \dots

123

Indice de coïncidence

Une autre méthode pour calculer la période.

1. On compte le **nombre** n_i **d'occurrences** de la lettre i dans un texte avec n lettres.
2. La **probabilité** que 2 lettres soient i est $C(n_i, 2)/C(n, 2)$.
3. L'**indice de coïncidence** (IC) est obtenu en additionnant cette probabilité pour toutes les lettres.
4. L'indice est **invariant par substitution** (donc par décalage).
5. On **calcule les IC par colonnes** jusqu'à trouver des IC proches du langage naturel.

<i>Langage</i>	<i>Anglais</i>	<i>Français</i>	<i>Allemand</i>
<i>IC</i>	0,065	0,074	0,072

122

- L'addition et la multiplication de matrices est définie de façon standard.
- $R[n, n]$ possède une structure d'anneau avec élément unité I .
- Si A est une matrice alors on dénote avec $A[i, j]$ l'élément avec coordonnées (i, j) .

124

Déterminant

Le *déterminant* d'une matrice $A \in R[n, n]$ est défini de façon inductive par

– si $n = 1$ alors $\det(A) = A[1, 1]$.

– Autrement,

$$\det(A) = \sum_{j=1, \dots, n} (-1)^{i+j} A[i, j] \det(A_{i,j})$$

où $A_{i,j} \in R[n-1, n-1]$ est la matrice qui résulte de A en effaçant la ligne i et la colonne j .

– On sait que le déterminant ne dépend pas du choix de la ligne (ou colonne) $i \in \{1, \dots, n\}$.

125

Exercice (déterminant 3×3)

On suppose

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}$$

Donnez une formule pour calculer $\det(A)$.

127

Exemple (déterminant 2×2)

On suppose

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix}$$

Alors en calculant par rapport à la première ligne :

$$A_{1,1} = [a_{2,2}] \quad A_{1,2} = [a_{2,1}]$$

Donc

$$\begin{aligned} \det(A) &= (-1)^{1+1} a_{1,1} A_{1,1} + (-1)^{1+2} a_{1,2} A_{1,2} \\ &= a_{1,1} a_{2,2} - a_{1,2} a_{2,1} \end{aligned}$$

126

Adjoint et Inverse

– Une matrice $A \in R[n, n]$ a une *inverse multiplicative* si et seulement si $\det(A)$ a une inverse dans R .

– Pour calculer l'inverse, on définit la *matrice adjointe* (notez l'inversion des indices) :

$$\text{adj}(A)[i, j] = (-1)^{i+j} \det(A_{j,i}) .$$

– Alors

$$A^{-1} = (1/\det(A)) \text{adj}(A)$$

– Il y a des algorithmes *polynomiaux en temps* pour calculer déterminant et inverse basés sur l'élimination de Gauss.

128

Exercice (calculer une inverse)

On travaille sur \mathbf{Z}_{11} . On suppose

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

1. Calculez $\det(A)$.
2. Déterminez son inverse.
3. Ensuite calculez la matrice adjointe.
4. Enfin, calculez la matrice inverse.

129

Exercice

On considère un chiffrement à bloc linéaire avec blocs de longueur 2 et alphabet $\{0, 1\}^2$. L'espace des clefs est constitué de matrices $K \in \mathbf{Z}_2[2, 2]$ telle que $(\det(K) \equiv 1) \pmod{2}$, choisies avec une distribution uniforme. La distribution du texte clair est aussi uniforme. Le chiffrement est $E_K(x) = (K \cdot x) \pmod{2}$. Ce système est-il parfaitement sûr ?

131

Exercice (inverse dans \mathbf{Z}_2)

Déterminez l'inverse de la matrice suivante dans \mathbf{Z}_2 .

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

130

Fonctions linéaires affines

- **Fait :** Une *fonction linéaire affine* $f : R^n \rightarrow R^n$ est déterminée par une matrice $A \in R[n, n]$ et un vecteur $b \in R[n]$ tels que

$$f(x) = Ax + b$$

- Si $b = 0$ alors on parle d'une fonction *linéaire*.
- Une fonction linéaire affine est *bijective* si et seulement si $\det(A)$ a une inverse dans R .
- Plusieurs anciens systèmes sont déterminés par une fonction linéaire affine. Par exemple celui de Vigenère. D'autres exemples suivent.

132

Permutation (ou transposition) comme un chiffrement à bloc

- Une permutation $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ induit un chiffrement à bloc sur Σ^n où

$$\begin{aligned} E_\pi(x_1, \dots, x_n) &= (x_{\pi(1)}, \dots, x_{\pi(n)}) \\ D_{\pi^{-1}}(x_1, \dots, x_n) &= (x_{\pi^{-1}(1)}, \dots, x_{\pi^{-1}(n)}) \end{aligned}$$

NB Ceci est *différent* d'un système par substitution.

133

Exemple (matrice de permutation)

- On suppose une permutation π telle que

$$\pi(1) = 2 \quad \pi(2) = 3 \quad \pi(3) = 1$$

- La matrice I_π correspondante est :

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

135

- Avec chaque permutation π , on associe une matrice $I_\pi \in R[n, n]$, obtenue de la matrice identité en permutant les lignes de la matrice identité selon la permutation π .

- Alors $E_\pi(\vec{x}) = I_\pi \cdot \vec{x}$.

- Ainsi, le chiffrement par permutation peut être vue comme une fonction linéaire.

134

Remarques

- La fréquence des lettres dans le texte chiffré est la même que dans le texte clair. Ainsi il est facile de voir si un texte chiffré a été chiffré avec un système par permutation.
- Pour réduire l'espace des clefs, on peut sélectionner certains digrammes ou tri-grammes du langage du texte clair et ensuite chercher des permutations qui maximisent leur fréquence dans le texte chiffré.
- Aussi si un texte clair est connu, on peut chercher son anagramme dans le texte chiffré.

136

Système de Hill

- Attribué à Hill (1929).
- La fonction de chiffrement est déterminée par une *fonction linéaire inversible* sur $(\mathbf{Z}_m)^n$.
- En d'autres termes une matrice $A \in \mathbf{Z}_m[n, n]$ avec un déterminant inversible.
- $E_A(\vec{x}) = (A\vec{x}) \bmod m$.

137

1. Soit W une matrice $n * n$ dont les colonnes sont les différences $(w_i - w_0) \bmod m$, pour $i = 1, \dots, n$.
2. Soit C une matrice $n * n$ dont les colonnes sont les différences $(c_i - c_0) \bmod m$, pour $i = 1, \dots, n$.

139

Attaque par texte clair connu sur les systèmes linéaires affines

Les systèmes linéaires affines avec alphabet \mathbf{Z}_m et longueur du bloc n sont vulnérables aux *attaques par texte clair connu*.

- On suppose que la fonction de chiffrement est

$$E : (\mathbf{Z}_m)^n \rightarrow (\mathbf{Z}_m)^n, \quad E(\vec{x}) = (A\vec{x} + b) \bmod m$$

- Si l'attaquant connaît $n + 1$ textes clairs w_i et les textes chiffrés correspondants $c_i = (Aw_i + b) \bmod m$, $i = 0, \dots, n$ alors il y a une méthode pour calculer la clef (A, b) .

138

3. On remarque que $c_i - c_0 \equiv A(w_i - w_0) \bmod m$. Ainsi on a $(AW \equiv C) \bmod m$.
4. Si, avec un peu de chance, $\det(W)$ est premier avec m , alors soit w' son inverse et soit $W^{-1} = w' \text{adj}(W)$. Alors

$$(A \equiv CW^{-1}) \bmod m \quad \text{et} \quad (b \equiv (c_0 - Aw_0)) \bmod m .$$

140

Exercice (attaque par texte clair connu)

On suppose savoir qu'un système de Hill sur $(\mathbf{Z}_{26})^2$ chiffre HAND avec FOOT. Est-ce suffisant pour déterminer la clef?

141

Schéma de Feistel, DES, AES, . . .

143

Sommaire

Les fonctions linéaires affines $(\mathbf{Z}_m)^n$ expliquent plusieurs systèmes classiques. Tous ces systèmes sont vulnérables aux attaques par texte clair connu.

142

Survol

La linéarité est une faiblesse. Règle de bon sens : une bonne fonction de chiffrement devrait être *non-linéaire*.

- Une méthode générale est de considérer le *produit* de systèmes cryptographiques élémentaires.

144

– A la base, les méthodes utilisées sont une combinaison de *substitutions* et de *permutations* :

1. Schéma de Feistel.
2. Data Encryption Standard (DES).
3. Advanced Encryption Standard (AES).

145

Produit de systèmes cryptographiques

- Pour simplicité, on suppose $\mathcal{P} = \mathcal{C}$.
- On suppose aussi $S_i = (\mathcal{P}, \mathcal{P}, \mathcal{K}_i, \mathcal{E}_i, \mathcal{D}_i)$ pour $i = 1, 2$.
- On définit le produit comme suit :

$$S_1 \times S_2 = (\mathcal{P}, \mathcal{P}, \mathcal{K}_1 \times \mathcal{K}_2, \mathcal{E}, \mathcal{D})$$

avec

$$E_{(k_1, k_2)}(p) = E_{k_2}(E_{k_1}(p)) \quad D_{(k_1, k_2)}(c) = D_{k_1}(D_{k_2}(c))$$

147

Combiner les systèmes cryptographiques

Pour améliorer la sécurité, une idée naturelle est de *combiner* les systèmes cryptographiques.

- Comment les combiner ?
- La combinaison est-elle utile ?

146

Exemple

Soit S un système par décalage (César) et L un système linéaire défini comme un système affine avec constante additive $b = 0$. Alors le système affine peut être vu comme $L \times S$:

$$E_{(a,b)}(x) = ((ax) \bmod m + b) \bmod m = (ax + b) \bmod m$$

où $\text{pgcd}(a, m) = 1$.

148

Exercice

On considère le système produit $S \times L$. Le système résultat est-il équivalent à un système connu ?

149

Double chiffrement

- Soit S un chiffrement à bloc avec clefs sur 2^n .
- Maintenant on considère le produit $S \times S$. On suppose qu'étant données les clefs k_1, k_2 il est 'improbable' qu'il y ait une clef k telle que k :

$$E_{k_2} \circ E_{k_1} = E_k$$

- On pourrait espérer qu'une *attaque par recherche complète* contre le système produit demanderait 2^{2n} essais ...

151

Exercice (sur le produit)

1. Montrez que le produit de systèmes cryptographiques est associatif.
2. Montrez que le système par décalage et le système linéaire sont idempotents, c'est-à-dire $S \times S$ est équivalent à S et $L \times L$ à L .
3. Montrez que si deux systèmes sont idempotents et si leur produit commute alors leur produit est aussi idempotent.
4. Conclure qu'un système affine est idempotent.

150

Attaque au milieu (meet in the middle attack)

En connaissant un certain nombre de couples texte clair-texte chiffré, on peut réduire l'espace de recherche.

- On suppose connaître :

$$c_i = E_{k_2}(E_{k_1}(p_i)), \quad i = 1, 2$$

- Alors, il faut que :

$$E_{k_1}(p_i) = D_{k_2}(c_i), \quad i = 1, 2$$

152

- Pour toutes les clefs k , on calcule (on a besoin d'une mémoire de taille 2^n) :

$$E_k(p_1)$$

- Pour toutes les clefs possibles k' , on calcule :

$$D_{k'}(c_1)$$

et on vérifie si $E_k(p_1) = D_{k'}(c_1)$.

- Alors on peut tomber sur $k = k_1$ et $k' = k_2$. Pour améliorer la confiance dans le résultat, on vérifie :

$$E_k(p_2) = D_{k'}(c_2)$$

Évidemment, le plus de couples texte clair-texte chiffré le mieux.

153

Règles de bon sens pour le chiffrement symétrique

Confusion Il faut cacher la relation entre le texte clair et le texte chiffré. On utilise des substitutions non-linéaires sur des sous-blocs.

Diffusion Il faut cacher la redondance du message et diffuser un changement de 1-bit dans le texte clair à tout le texte chiffré. On utilise une opération de permutation (linéaire) sur la totalité du bloc.

Clef de tour La clef génère un vecteur de clefs de tour qui sont combinées avec les données à l'aide de la fonction xor.

Pseudo-aléatoire Une condition 'idéale' est que le texte chiffré 'ressemble' à une séquence de bits générés de façon aléatoire.

155

Chiffrement triple

- A cause de l'attaque au milieu, le chiffrement double n'est pas utilisé.

- Une approche différente qui évite l'attaque est de *chiffrer-déchiffrer-chiffrer*. On définit :

$$c = E_{k_1}(D_{k_2}(E_{k_1}(p))) \quad p = D_{k_1}(E_{k_2}(D_{k_1}(c)))$$

- Ceci est connu comme *chiffrement triple* et il s'agit de la méthode standard pour 'améliorer' la sécurité de DES sans changer l'algorithme.

NB Si $k_1 = k_2$ alors le chiffrement triple est la même chose que le chiffrement standard.

154

Attention

- Nous allons décrire des schémas de chiffrement.
- Nous n'essayons pas d'expliquer *pourquoi* on suppose que ces schémas sont sûrs.
- La théorie qui considère cette question est compliquée et incomplète.

156

- Pour empirer les choses, les schémas comme DES ont été proposés *sans explication*.
- La théorie développée après DES semble être une justification partielle du système (par exemple, l'analyse différentielle).
- La situation est un peu meilleure pour le nouveau standard AES qui est le résultat d'une compétition publique.

Clef Une clef K est aussi sur 16 bits. La clef K_i du tour i pour $i = 1, 2, 3, 4$ est obtenue en effectuant $4 * (i - 1)$ décalages circulaires vers la droite de la clef K . Ainsi si $K = x_1 \cdot x_2 \cdot x_3 \cdot x_4$ et $x_i \in 2^4$ pour $i = 1, 2, 3, 4$, on aura :

$$\begin{aligned} K_1 &= x_1 \cdot x_2 \cdot x_3 \cdot x_4 \\ K_2 &= x_4 \cdot x_1 \cdot x_2 \cdot x_3 \\ K_3 &= x_3 \cdot x_4 \cdot x_1 \cdot x_2 \\ K_4 &= x_2 \cdot x_3 \cdot x_4 \cdot x_1 \end{aligned}$$

Exemple (un réseaux de substitution permutation RSP)

On veut définir une fonction de chiffrement $E_k : 2^{16} \rightarrow 2^{16}$.

Substitution On utilise la première ligne de la première boîte de DES (à venir) qui est une fonction $S : 2^4 \rightarrow 2^4$ spécifiée par le tableau suivant :

Entrée :	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Sortie :	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

Permutation Comme permutation on utilise une fonction

$P : \{1, \dots, 16\} \rightarrow \{1, \dots, 16\}$ qui est spécifiée par le tableau suivant :

Entrée :	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Sortie :	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

Tour Le chiffrement se décompose en 4 tours. Le tour i , pour $i = 1, 2, 3, 4$ consiste à transformer un bloc $x = (x_1 \cdot x_2 \cdot x_3 \cdot x_4)$, où $x_i \in 2^4$ comme suit :

$$\begin{aligned} (y_1 \cdot y_2 \cdot y_3 \cdot y_4) &= (x \oplus K_i) && \text{(XOR avec la clef)} \\ (w_1 \cdot w_2 \cdot w_3 \cdot w_4) &= (S(y_1) \cdot S(y_2) \cdot S(y_3) \cdot S(y_4)) && \text{(Substitution)} \\ z &= P(w_1 \cdot w_2 \cdot w_3 \cdot w_4) && \text{(Permutation)} \end{aligned}$$

Exercice Expliquer comment déchiffrer.

Schéma de Feistel (ingrédients de base)

- Alphabet $\Sigma = \{0, 1\}$.
- Un chiffrement à bloc de longueur t . Si k est une clef, f_k est la fonction de chiffrement correspondante.
- Un nombre de tours r .
- Une méthode pour générer d'une clef k , r clefs de tour k_1, \dots, k_r .

161

Déchiffrement

- Pour $L_i R_i = R_{i-1}(L_{i-1} \oplus f_{k_i}(R_{i-1}))$ on dérive

$$\begin{aligned} R_{i-1} L_{i-1} &= L_i(R_i \oplus f_{k_i}(R_{i-1})) \\ &= L_i(R_i \oplus f_{k_i}(L_i)) \end{aligned}$$

- Étant donné un bloc de texte chiffré c , on le divise en $L_r R_r$ et on calcule $L_i R_i$ pour $i = r - 1, \dots, 0$ en utilisant les clefs k_r, \dots, k_1 .
- Alors

$$D_k(c) = L_0 R_0$$

163

Schéma de Feistel

Il s'agit d'un chiffrement qui opère sur des blocs de longueur $2t$ en r tours.

Chiffrement

- On divise le texte clair p en deux moitiés $L_0 R_0$.
- On définit pour $i = 1, \dots, r$:

$$L_i R_i = R_{i-1}(L_{i-1} \oplus f_{k_i}(R_{i-1}))$$

- On pose

$$E_k(p) = L_r R_r$$

162

Exercice (Feistel faible)

On utilise un schéma de Feistel où la clef de tour est toujours la même et la fonction de codage f_k est le xor avec k . On a donc :

$$\begin{aligned} L_{i+1} &= R_i \\ R_{i+1} &= L_i \oplus (R_i \oplus k) \end{aligned}$$

Quelles sont les faiblesses de ce schéma en fonction du nombre de tours ?

164

Data encryption standard (DES)

Il s'agit de l'exemple le plus connu de schéma de Feistel.

- Soit $\mathcal{P} = \mathcal{C} = 2^{64}$.
- Soit $\mathcal{K} = 2^{56}$. Une clef est complétée à 64 bits en insérant 8 bits aux positions

8, 16, 24, 32, 40, 48, 56, 64

Ces bits sont utilisés pour détecter et corriger 1 erreur.

- Il y a $r = 16$ tours.

165

La fonction d'expansion $Ex : 2^{32} \rightarrow 2^{48}$

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

NB *Redondance* en première et sixième colonne.

167

DES : fonctionnement interne

- Ici $t = 64/2 = 32$ bits.
- Les clefs de tour sont sur 48 bits.
- Il y a une fonction d'expansion $Ex : 2^{32} \rightarrow 2^{48}$.
- Il y a une série de *S-boites* (S-boxes) qui sont des fonctions (non-affines) $S_i : 2^6 \rightarrow 2^4, i = 1, \dots, 8$.

166

Exemple : la boite S_1

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]
[0]	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
[1]	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
[2]	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
[3]	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

NB Les valeurs dans chaque ligne sont *différentes*. La composition de l'expansion et des S-boites est bien une *permutation* sur 2^{32} .

168

- Soit $B = b_1b_2b_3b_4b_5b_6$.
- $i = (b_1b_6)_2 \in \{0, 1, 2, 3\}$ détermine l'indice de la ligne.
- $j = (b_2b_3b_4b_5)_2 \in \{0, \dots, 15\}$ détermine l'indice de la colonne.
- Le résultat est la valeur de coordonnées (i, j) exprimée en base 2.

169

Exercice (substitution non-linéaire)

- La boîte S_1 est une fonction qui transforme 6 bits en 4 bits. Chaque ligne du tableau est une fonction de type $(\mathbf{Z}_2)^4 \rightarrow (\mathbf{Z}_2)^4$ qui correspond à une permutation $\pi : \mathbf{Z}_{16} \rightarrow \mathbf{Z}_{16}$.
- Considérez, par exemple, la deuxième ligne 0, 15, 7, ... et montrez que la permutation π ne s'exprime pas comme une fonction affine sur $(\mathbf{Z}_2)^4$. Ceci veut dire qu'on ne peut pas trouver une matrice A , 4×4 , et un vecteur b , à coefficients dans \mathbf{Z}_2 , tels que

$$(A(x_3, x_2, x_1, x_0)^T + b)_2 = \pi((x_3, x_2, x_1, x_0)_2) \quad \text{pour tout } (x_3, x_2, x_1, x_0) \in (\mathbf{Z}_2)^4$$

Notation On dénote avec $(y_3, y_2, y_1, y_0)_2$ la valeur en base 2 de la suite $y_3y_2y_1y_0$ et avec $(x_3, x_2, x_1, x_0)^T$ la transposée de (x_3, x_2, x_1, x_0) .

171

- Par exemple, pour calculer $S_1(001011)$:
 1. $i = (01)_2 = 1$.
 2. $j = (0101)_2 = 5$.
 3. $S_1[1, 5] = 2$.
 4. $S_1(001011) = 0010$.

170

Calcul de $f_{k_i}(R_{i-1})$

$$R' := Ex(R_{i-1}) \quad (\text{Expansion } R_i \text{ à 48 bits})$$

$$B_1 \cdots B_8 := R' \oplus k_i \quad (B_i \text{ a 6 bits})$$

$$C_i := S_i(B_i), i = 1, \dots, 8 \quad (C_i \text{ a 4 bits})$$

$$P(C_1 \cdots C_8) \quad (P \text{ pour une permutation sur 32 bits})$$

NB Avant de commencer les 16 tours, DES applique une permutation initiale à ne pas confondre avec la permutation P .

172

La permutation P

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

173

DES : génération des clefs de tour

- Les fonctions suivantes sont données :

$$f_1 : 2^{64} \rightarrow 2^{28} \times 2^{28}$$

$$f_2 : 2^{28} \times 2^{28} \rightarrow 2^{48}$$

et les fonctions de décalage :

$$\text{shift}_i \text{ pour } i = 1, \dots, 16$$

- Alors, étant donné une clef k , on calcule :

$$C_0 D_0 := f_1(k)$$

$$C_i := \text{shift}_i(C_{i-1}) \quad D_i := \text{shift}_i(D_{i-1}) \quad i = 1, \dots, 16$$

$$k_i := f_2(C_i D_i)$$

175

Remarques

- Permutation, expansion et S -boites sont spécifiées par certaines tables fixées.
- Si on veut implémenter DES, il faut faire référence à ces tables.
- Autrement, l'intérêt de ces tables est limité...

174

Attaques

- Les techniques développées (analyses linéaires et différentielles) marchent seulement dans des situations simplifiées.
 - Linéaire** On utilise une approximation linéaire des S -boites.
 - Différentielles** On compare le \oplus de l'entrée et de la sortie.
- Une recherche complète est maintenant possible.

176

Successeur : Advanced Encryption Standard (AES)

Il a une histoire intéressante :

- Spécification du AES en 1997 :
 - Sécurité (blocs de longueur 128, longueur de la clef 128, 192, 256).
 - Coût (vitesse et mémoire).
 - Implémentation simple (matériel)
- Compétition internationale ouverte (par opposition au DES).
- Le gagnant : Rijandel (2000).

177

Encore des résultats sur l'arithmétique modulaire

179

Sommaire

- Un schéma intéressant : Feistel.
- Peut être spécialisé (DES, AES, ...).
- Le diable est dans les détails... et les détails sont difficiles à justifier.

178

Motivation

- Avant d'introduire la cryptographie à *clef publique* (public key) on a besoin d'introduire certains résultats standards d'arithmétique modulaire.
- Ces résultats ont un intérêt *indépendant de la cryptographie* et d'ailleurs ils étaient bien connus avant qu'on leur trouve des applications cryptographiques.
- **Morale** : il n'y a rien de plus pratique qu'une bonne théorie...

180

Sommaire

Petit théorème de Fermat

$$\text{pgcd}(a, m) = 1 \Rightarrow (a^{\phi(m)} \equiv 1) \text{ mod } m$$

Exponentiation rapide Calcul de $(a^n) \text{ mod } m$ en temps polynomial.

Restes Chinois Soient m_1, \dots, m_n premiers entre eux et $M = m_1 \cdots m_n$. Alors la solution du système

$$(x \equiv a_i) \text{ mod } m_i \text{ pour } i = 1, \dots, n$$

existe et est unique modulo M .

181

Fonction d'Euler et petit théorème de Fermat

– La *fonction d'Euler* ϕ est définie par :

$$\phi(m) = \#\{a \mid \text{pgcd}(a, m) = 1 \text{ et } 0 \leq a < m\}$$

– Le résultat qu'on vise s'appelle *petit théorème de Fermat*.

Si $\text{pgcd}(a, m) = 1$ alors $(a^{\phi(m)} \equiv 1) \text{ mod } m$.

183

Groupe multiplicatif des résidus

182

Le groupe multiplicatif

On dénote par $(\mathbf{Z}_m)^*$ l'ensemble des $n \in \mathbf{Z}_m$ qui ont une inverse multiplicative. En d'autres termes :

$$(\mathbf{Z}_m)^* = \{a \in \mathbf{Z}_m \mid \text{pgcd}(a, m) = 1\}$$

On remarque que $\phi(m) = \#(\mathbf{Z}_m)^*$.

184

Exercice (sur $(\mathbf{Z}_m)^*$ et ϕ)

1. Montrez que $(\mathbf{Z}_m)^*$ est un groupe abélien par rapport à la multiplication dans \mathbf{Z}_m .
2. Calculez $(\mathbf{Z}_{12})^*$.
3. Calculez $\phi(m)$ pour $m = 1, \dots, 15$.
4. Quelle est la valeur de $\phi(p)$ pour p premier ?

185

Cardinalité de groupes et ordre d'éléments

- Soit G un *groupe fini abélien* avec unité 1.
- Si $g \in G$ alors g^{-1} dénote son inverse.
- La cardinalité de G est dénotée par $\#G$.

186

- Étant donné $g \in G$ et $a \in \mathbf{Z}$, l'*exposant* est défini par :

$$g^0 = 1 \quad g^a = gg^{(a-1)} \quad g^{-a} = (g^{-1})^a \text{ où } a > 0 .$$

- L'*ordre* de $g \in G$, écrit $ord(g)$, est le plus petit nombre *positif* tel que $g^e = 1$ (s'il existe), autrement on dit que $ord(g)$ est infini.

187

Exercice (ordre de l'élément d'un groupe)

Déterminez l'ordre de 2 dans \mathbf{Z}_{13} .

188

Ordre et divisibilité

Soit $g \in G$ et $e \in \mathbf{Z}$. Alors $g^e = 1$ ssi $\text{ord}(g) \mid e$.

(\Leftarrow) On suppose $n = \text{ord}(g)$ et $e = kn$. Alors

$$g^e = g^{kn} = (g^n)^k = 1$$

189

Exercice (exposant et congruence)

Soit $g \in G$ et $k, l \in \mathbf{Z}$. Alors $g^l = g^k$ ssi $(l \equiv k) \text{ mod } \text{ord}(g)$.

191

(\Rightarrow)

– On suppose $g^e = 1$ et $e = qn + r$ avec $0 \leq r < n$ et $n = \text{ord}(g)$.

– Alors

$$g^r = g^{e-qn} = g^e(g^n)^{-q} = 1$$

– Mais comme n est le plus petit entier *positif* tel que $g^n = 1$, il faut que $r = 0$.

190

Sous-groupe généré

– Le *sous-groupe généré* par g est dénoté par

$$\langle g \rangle = \{g^a \mid a \in \mathbf{Z}\}$$

– Par l'exercice précédent : $\text{ord}(g) = \# \langle g \rangle$.

192

Cardinalité de sous-groupes (Lagrange)

Soit H un sous-groupe d'un groupe fini G . Alors $\#H \mid \#G$.

- Soit H un sous-groupe.
- On écrit $a \sim b$ si $ab^{-1} \in H$.
- On vérifie que \sim est une relation d'équivalence.
- On vérifie que si $c \in H$ alors $ca \sim a$.

193

Preuve du théorème de Fermat

On suppose que G est un groupe fini.

- Soit $g \in G$.
- $\langle g \rangle$ est un sous-groupe de G .
- Ainsi par la proposition de Lagrange :

$$\# \langle g \rangle \mid \#G$$

195

- Si $a \in G$ alors $[a] = \{c \mid ca^{-1} \in H\}$ dénote sa classe d'équivalence.

- On remarque que $[1] = H$.

- Toutes les classes d'équivalence ont la même cardinalité car la fonction $f : [1] \rightarrow [a]$ définie par $f(c) = ca$ est bijective. Par exemple, pour la surjectivité :

$$\begin{aligned} b \in [a] &\Rightarrow a \sim b \\ &\Rightarrow ab^{-1} \in H \\ &\Rightarrow ba^{-1} \in H \\ &\Rightarrow f(ba^{-1}) = b \end{aligned}$$

- Ainsi la cardinalité de H divise la cardinalité de G .

194

- On rappelle que $\# \langle g \rangle = \text{ord}(g)$. Ainsi :

$$g^{\#G} = 1$$

196

– En particulier, considérons un élément $a \in (\mathbf{Z}_m)^*$ (c'est-à-dire avec $\text{pgcd}(a, m) = 1$).

– On rappelle que $\#(\mathbf{Z}_m)^* = \phi(m)$.

– Donc le résultat précédent donne :

$$(a^{\phi(m)} \equiv 1) \text{ mod } m$$

197

Nombre de générateurs

Il y a une relation surprenante entre la fonction d'Euler et le nombre de générateurs.

Fait Soit G un groupe fini cyclique. Alors G a exactement $\phi(\#G)$ générateurs (d'ordre $\#G$).

199

Cyclicité

On dit qu'un groupe est *cyclique* s'il y a un élément g qui génère le groupe, c'est-à-dire tel que $G = \langle g \rangle$.

198

Groupe multiplicatif modulo un premier

Cyclicité et primalité sont connectées...

Fait Si p est premier alors $(\mathbf{Z}_p)^*$ est cyclique et par le fait précédent il admet exactement $\phi(p - 1)$ générateurs.

200

Exercice (comptage des générateurs)

Trouvez tous les générateurs de $(\mathbf{Z}_{13})^*$.

201

Motivation

- Le petit théorème de Fermat donne une autre méthode pour calculer l'inverse multiplicative dans \mathbf{Z}_m . A savoir on calcule $a^{\phi(m)-1}$.
- Normalement, l'exponentiation est une opération coûteuse.
- Cependant, en arithmétique modulaire elle peut être calculée de façon efficace. Ainsi l'exponentiation modulaire est une méthode alternative (à Euclide) pour calculer l'inverse.

203

Exponentiation en arithmétique modulaire

202

Exponentiation rapide

- Soit G un monïde et e un entier positif.

- Soit

$$e = \sum_{i=0, \dots, k} e_i 2^i$$

l'expansion binaire de e .

- Les coefficients e_i appartiennent à l'ensemble $\{0, 1\}$ avec $e_k = 1$ si $k > 0$.

204

– On observe :

$$\begin{aligned}g^e &= g^{\sum_{i=0, \dots, k} e_i 2^i} \\ &= \prod_{i=0, \dots, k} (g^{2^i})^{e_i} \\ &= \prod_{0 \leq i \leq k, e_i=1} (g^{2^i}).\end{aligned}$$

205

- Quand on exécute les opérations dans \mathbf{Z}_m , la taille des données est *bornée* par la taille de m .
- On en déduit que l'exposant $(a^e) \bmod m$ avec $a \in \mathbf{Z}_m$ peut être calculé en temps polynomial dans la taille de e et m .

207

– On suit la procédure suivante :

1. On calcule g^{2^i} pour $0 \leq i \leq k$. En remarquant que

$$g^{2^{i+1}} = (g^{2^i})^2$$

Ainsi $O(k)$ opérations d'élevation au carré sont nécessaires.

2. On détermine g^e comme le produit des g^{2^i} tels que $e_i = 1$. Ainsi $O(k)$ multiplications sont nécessaires.

206

Exercice (calcul de l'exposant)

Déterminez $6^{73} \bmod 100$.

208

Restes Chinois

209

Théorème des restes Chinois

Soient m_1, \dots, m_n des nombres entiers positifs premiers entre eux et soient a_1, \dots, a_n des nombres entiers. Alors le système de congruences :

$$(x \equiv a_i) \pmod{m_i}, i \in \{1, \dots, n\}$$

a une solution x qui est unique modulo $M = m_1 \cdots m_n$.

211

Exercice (système de congruences)

Calculez la solution du système

$$(x \equiv 2) \pmod{4} \quad (x \equiv 1) \pmod{3} \quad (x \equiv 0) \pmod{5}$$

210

Unicité

- Si x et x' sont deux solutions du système alors $(x \equiv x') \pmod{m_i}$ pour $i = 1, \dots, n$, par transitivité de la congruence.
- Comme les m_i sont premiers entre eux, on dérive que $(x \equiv x') \pmod{M}$. Pourquoi ?

212

Existence

– On pose :

$$M_i = M/m_i, \quad i = 1, \dots, n .$$

– On remarque que m_i et M_i sont premiers entre eux.

– Par l'algorithme d'Euclide, on détermine y_i tels que

$$(M_i y_i \equiv 1) \pmod{m_i}, \quad i = 1, \dots, n .$$

– On pose :

$$x = (\sum_{i=1, \dots, n} a_i y_i M_i) \pmod{M}$$

213

Sommaire : Restes Chinois

– Étant donné $(x \equiv a_i) \pmod{m_i}$, m_i premiers entre eux pour $i = 1, \dots, n$.

– On calcule :

1. $M = m_1 \cdots m_n$, $M_i = M/m_i$, $y_i = (M_i^{-1}) \pmod{m_i}$.

2. $x = (\sum_{i=1, \dots, n} a_i y_i M_i) \pmod{M}$.

215

– On dérive

$$(a_i y_i M_i \equiv a_i) \pmod{m_i}$$

– Comme m_i divise M_j pour $i \neq j$ on a $(a_j y_j M_j \equiv 0) \pmod{m_i}$.

– Ainsi

$$(x \equiv a_i y_i M_i + \sum_{j \neq i, j=1, \dots, n} a_j y_j M_j \equiv a_i) \pmod{m_i}, \quad i = 1, \dots, n$$

214

Exercice (solution d'un système de congruences)

Déterminez x tel que

$$(x \equiv 5) \pmod{7} \quad (x \equiv 3) \pmod{11} \quad (x \equiv 10) \pmod{13}$$

216

Exercice (produit d'anneau)

On suppose m_1, \dots, m_n premiers entre eux. Soit $M = m_1 \cdots m_n$.

1. On considère le produit cartésien $\mathbf{Z}_{m_1} \times \cdots \times \mathbf{Z}_{m_n}$. Vérifiez que les opérations d'addition et de multiplication peuvent être définies par composante de façon à obtenir un anneau (qu'on appelle *anneau produit*).
2. Montrez que la fonction

$$i : \mathbf{Z}_M \rightarrow \mathbf{Z}_{m_1} \times \cdots \times \mathbf{Z}_{m_n}$$

définie par

$$i(x) = (x \bmod m_1, \dots, x \bmod m_n)$$

est une bijection (en effet un isomorphisme d'anneau).

3. Conclure que les opérations arithmétiques dans \mathbf{Z}_m peuvent être effectuées sur l'anneau produit.

217

RSA

219

Exercice (sur ϕ)

- On suppose que m_1, \dots, m_n sont premiers entre eux et que $m = m_1 \cdots m_n$. Montrez que :
 1. Les groupes $(\mathbf{Z}_m)^*$ et $(\mathbf{Z}_{m_1})^* \times \cdots \times (\mathbf{Z}_{m_n})^*$ sont isomorphes.
 2. $\phi(m) = \phi(m_1) \cdots \phi(m_n)$.
 3. On suppose $m = pq$ avec p, q premiers et différents. Pouvez-vous calculer $\phi(m)$?

218

Systemes à clef publique

Diffie-Hellman (1976)

- Chaque utilisateur A dispose de deux clefs : une est gardée secrète et l'autre est rendue publique.
- Chaque utilisateur du réseaux dispose de la clef publique du destinataire pour chiffrer un message et d'une clef privée pour déchiffrer les messages reçus.

Rivest-Shamir-Adleman 1978 Première réalisation concrète de cette idée. La sécurité du système dépend de la difficulté à trouver la factorisation du produit de deux grands nombres premiers.

Idée de fonction à sens unique Facile à calculer mais difficile à inverser (et à trouver!).

220

RSA

1. On génère deux grands nombres premiers p, q (on présentera dans la suite une méthode probabiliste pour cette tâche connue comme test de Miller-Rabin).
2. Soit $n = pq$ le *module*.
3. On choisit e *exposant de chiffrement* tel que $1 < e < \phi(n)$ et $\text{pgcd}(e, \phi(n)) = 1$.
4. On calcule d *exposant de déchiffrement* tel que $(ed \equiv 1) \text{ mod } \phi(n)$ (cf. algorithme d'Euclide). Alors :

$$\text{Clef publique} = (n, e) \quad \text{Clef privée} = (n, d)$$

221

Est-ce un système cryptographique ?

On doit vérifier :

$$(m^e)^d \text{ mod } n = m \quad \text{pour } 0 \leq m < n$$

– On sait

$$(ed \equiv 1) \text{ mod } \phi(n)$$

– Ce qui implique

$$ed = 1 + l(p-1)(q-1)$$

– Ainsi

$$(m^e)^d = m^{ed} = m^{1+l(p-1)(q-1)} = m(m^{(p-1)(q-1)})^l$$

223

5. Textes clairs et textes chiffrés :

$$\mathcal{P} = \mathcal{C} = \{m \mid 0 \leq m < n\} = \mathbf{Z}_n$$

6. Fonction de chiffrement :

$$c = E_{(n,e)}(m) = m^e \text{ mod } n$$

on peut utiliser l'exponentiation rapide.

7. Fonction de déchiffrement :

$$D_{(d,n)}(c) = c^d \text{ mod } n$$

on peut utiliser à nouveau l'exponentiation rapide.

222

– Montrons que

$$(m^{ed} \equiv m(m^{(p-1)})^{(q-1)l} \equiv m) \text{ mod } p$$

On considère deux cas :

- Si $p \nmid m$ alors on applique le petit théorème de Fermat.
- D'autre part, si $p \mid m$ alors la congruence ci-dessus est valide car m est un multiple de p .

224

– De la même façon on montre :

$$(m^{ed} \equiv m(m^{(q-1)})^{(p-1)l} \equiv m) \pmod{q}$$

– Ainsi

$$(m^{ed} \equiv m) \pmod{p} \quad (m^{ed} \equiv m) \pmod{q}$$

225

Sommaire : système RSA

1. On génère deux grands nombres premiers (*différents !*) p, q . Soit $n = pq$ le *module*.
2. On choisit e *exposant de chiffrement* tel que $1 < e < \phi(n)$ et $\text{pgcd}(e, \phi(n)) = 1$.
3. Soit d , l'*exposant de déchiffrement*, l'inverse de e modulo $\phi(n)$.

$$\text{Clef Publique} = (n, e) \quad \text{Clef Privée} = (n, d)$$

$$c = E_{(n,e)}(m) = m^e \pmod{n} \quad D_{(d,n)}(c) = c^d \pmod{n} \quad \text{pour } m, c \in \mathbf{Z}_n$$

227

– Mais en général, pour p, q premiers différents on a :

$$(x_1 \equiv x_2) \pmod{p} \text{ et } (x_1 \equiv x_2) \pmod{q} \Rightarrow (x_1 \equiv x_2) \pmod{pq}$$

– C'est-à-dire

$$p \mid (x_1 - x_2) \text{ et } q \mid (x_1 - x_2) \Rightarrow pq \mid (x_1 - x_2)$$

226

Exemple (RSA)

– On choisit comme premiers $p = 11$ et $q = 23$.

– Alors $n = pq = 253$, et

$$\phi(n) = (p-1)(q-1) = 4 \cdot 5 \cdot 11$$

– Le plus petit exposant de chiffrement possible est $e = 3$ (en général, on ne peut pas avoir $e = 2$ car $\phi(n)$ est pair!).

– L'algorithme d'Euclide donne $d = 147$.

228

- L'espace des textes clairs et des textes chiffrés est $\{0, 1, \dots, 252\}$.
- Pour chiffrer le texte clair $m = 165$, on calcule $165^3 \bmod 253 = 110$.
- Pour déchiffrer le texte chiffré 110, on calcule $110^{147} \bmod 253$.

229

Exercice (RSA)

Générez un couple de nombres premiers différents p et q représentable avec au plus 4 bits et tels que le module RSA $n = pq$ est sur 7-bits et la clef publique $e = 5$ ne peut pas être utilisée.

231

Exercice (RSA)

On suppose $p = 101$, $q = 113$.

1. Vérifiez que $e = 3533$ est un exposant admissible.
2. Chiffrez le message 9726.
3. Déterminez l'exposant de déchiffrement.

230

Exercice (RSA)

Générez deux nombres premiers différents p et q représentables avec 8-bits et tels que le module RSA $n = pq$ est un nombre représentable avec 16 bits et la clef publique $e = 5$ peut être utilisée. Calculez la clef privée d correspondante. Chiffrez 27063 avec la clef $e = 5$.

232

Exercice (RSA)

Soit $n = 221$. La clef publique (n, e) est choisie pour que e soit minimal. Déterminez l'exposant de déchiffrement.

233

RSA comme un chiffrement à bloc

235

Exercice (RSA *ppcm*)

Dans le système RSA, l'exposant de déchiffrement d est choisi de façon telle que $(ed \equiv 1) \pmod{(p-1)(q-1)}$. Montrez qu'en choisissant $(ed \equiv 1) \pmod{\text{ppcm}(p-1, q-1)}$ on a toujours un système cryptographique, c'est-à-dire pour $0 \leq m < pq$ nous avons :

$$(m^e)^d \pmod{pq} = m$$

où *ppcm* est le *plus petit commun multiple*.

234

On montre que RSA peut être utilisé (presque) comme un *chiffrement à bloc*.

- Soit $\Sigma = \mathbf{Z}_N$ un alphabet. Par exemple $N = 256$ pour représenter les caractères ASCII.
- Soit $n = pq$ le module.
- Soit $k = \lfloor \log_N(n) \rfloor$.
- On représente un mot $m_0 \cdots m_{k-1} \in \Sigma^k$ comme un nombre en base N :

$$m = \sum_{i=0, \dots, k-1} m_i N^i$$

236

– Comme $k = \lfloor \log_N(n) \rfloor$, on dérive :

$$0 \leq m \leq (N - 1) \sum_{i=0, \dots, k-1} N^i = (N^k - 1)$$

$$(N^k - 1) \leq N^{\log_N(n)} - 1 < n$$

– Alors on calcule $c = (m^e) \bmod n$ où e est l'exposant de chiffrement.

237

Exemple (RSA comme chiffrement à bloc)

– Soit $\Sigma = \{0, 1, 2, 3\}$, le module $n = 253 = 11 \cdot 23$ et $e = 3$.

– Alors $k = \log_4(253) = 3$ est la longueur des blocs de texte clair.

– On suppose qu'on représente le bloc $m_0 m_1 m_2 = 221$. On obtient :

$$2 + 2 \cdot 4 + 1 \cdot 16 = 26$$

239

– Comme

$$0 \leq c < n < N^{k+1}$$

le nombre c peut être représenté en base N avec $k + 1$ chiffres c_0, \dots, c_k et ceci nous donne un bloc chiffré de longueur $k + 1$.

– On a donc une fonction injective qui transforme un bloc de longueur k dans un bloc de longueur $k + 1$. D'après la définition ceci n'est pas un chiffrement à bloc... cependant les modes ECB et CBC peuvent être adaptés.

238

– Le chiffrement est

$$c = 26^3 \bmod 253 = 119$$

– Ensuite on écrit c en base 4

$$c = 3 \cdot 1 + 1 \cdot 4 + 3 \cdot 16 + 1 \cdot 64$$

– Ainsi le bloc chiffré est $c_0 c_1 c_2 c_3 = 3131$.

240

Sécurité de RSA

- Aujourd’hui, on considère que RSA est un système fiable.
- Cependant, il y a un certain nombre de pièges à éviter...

241

Choix de p et q

- Aujourd’hui, on ne sait pas factoriser $n = pq$ si n a 1024 bits et p, q sont deux nombres premiers de 512 bits choisis de façon aléatoire.
- Il y a des algorithmes de factorisation qui marchent mieux pour des choix particuliers de p et q . Par exemple, si $p - 1$ a seulement des petits facteurs premiers. Cependant, la probabilité de sélectionner ces nombres est négligeable.

243

242

Exercice (racines dans \mathbf{Z})

Décrivez un algorithme de type *diviser pour régner* qui étant donnés deux entiers positifs c et e décide s’il existe m tel que $c = m^e$ dans \mathbf{Z} , et dans ce cas calcule m .

244

Choix de e

- Il est tentant de choisir e aussi petit que possible. Le choix le plus petit possible est $e = 3$, car $\phi(n) = (p - 1)(q - 1)$ est pair et e doit être premier avec $\phi(n)$.
- Si e est choisi trop petit alors une attaque est possible.
- On suppose que A envoie le *même* message m à e correspondants en utilisant comme exposant e avec différents modules n_i , $i = 1, \dots, e$ qui sont premiers entre eux. Cette situation est assez probable, par exemple, une banque envoie le même message à tous les clients.
- Un attaquant voit les chiffrements $c_i = m^e \bmod n_i$, pour $i = 1, \dots, e$ et applique l'algorithme suivant.

245

Exercice (attaque sur le petit exposant)

Soit $e = 3$, $n_1 = 51$, $n_2 = 65$, $n_3 = 77$. On suppose $c_1 = (m^e) \bmod n_1 = 23$, $c_2 = (m^e) \bmod n_2 = 60$ et $c_3 = (m^e) \bmod n_3 = 48$. Déterminez m .

247

1. On détermine c tel que $(c \equiv c_i) \bmod n_i$ pour $i = 1, \dots, e$ et $0 \leq c < n_1 \cdots n_e$. Il suffit d'appliquer le théorème des restes chinois.
 2. On détermine m tel que $m^e = c$ dans \mathbf{Z} .
- L'attaque demande que le *même* message soit envoyé. Une contre-attaque possible est d'insérer dans chaque message des *données générées de façon aléatoire*.

246

Exercice (plagiat dans RSA)

On suppose que l'attaquant voit deux textes chiffrés :

$$c_1 = m_1^e \bmod n \quad c_2 = m_2^e \bmod n$$

Sans connaître d , m_1 et m_2 l'attaquant peut construire le chiffrement de $m_1 m_2$. Expliquez comment. Comment peut-on prévenir cette attaque ?

248

Exercice (sur un module partagé)

Si le *même* texte clair est chiffré *deux fois* avec le système RSA en utilisant deux clefs publiques (n, e) et (n, f) qui partagent le *même* module et si $\text{pgcd}(e, f) = 1$ alors le texte clair m peut être calculé à partir des textes chiffrés $c_1 = m^e \bmod n$ et $c_2 = m^f \bmod n$.

Expliquez comment organiser ce calcul.

249

- Ainsi RSA est sûr à condition qu'on ne puisse pas factoriser de façon efficace le produit de deux grands nombres premiers.
- Si on connaît d alors il y a un *algorithme probabiliste* pour factoriser n . Ainsi apprendre d est aussi difficile que factoriser n .
- Cependant, il pourrait y avoir d'autres méthodes pour obtenir de l'information sur le texte clair qui ne nécessitent pas le calcul de d . À l'état de nos connaissances, casser RSA pourrait être *plus facile* que factoriser le produit de deux grands nombres premiers.

251

RSA et factorisation

- L'attaquant connaît (n, e) . S'il peut factoriser $n = pq$, alors :
 1. Il calcule $\phi(n) = (p - 1)(q - 1)$.
 2. Il calcule d tel que $(ed \equiv 1) \bmod \phi(n)$ et il peut déchiffrer chaque message.

250

Test de primalité

252

Motivations et approche

- Plusieurs systèmes cryptographiques nécessitent la génération de *grands nombres premiers* (par exemple, RSA).
- Méthode : on génère un nombre impair de façon *aléatoire* et on *teste* sa primalité.

253

- Avancée récente (2003, prix Gödel 2006) : *PRIME est dans P*.
 - C'est un algorithme *déterministe* qui vérifie si un nombre est premier.
 - L'algorithme n'est pas encore compétitif avec les algorithmes probabilistes.
 - C'est encore un problème ouvert de savoir si la *factorisation* peut être calculée en temps polynomial.

255

- On utilise des méthodes et des arguments *probabilistes*.
 - On a besoin d'un bon *générateur aléatoire*.
 - On doit estimer la *probabilité de tomber sur un nombre premier* en tirant un nombre au hasard.
 - Le test de primalité est basé sur un *algorithme probabiliste*.
 - On doit évaluer la probabilité que le test donne une *réponse correcte*.

254

Un mot sur les algorithmes probabilistes

- Un *algorithme probabiliste* est un algorithme qui peut jouer à pile ou face pour décider quelle instruction exécuter.
- Pour simplicité on suppose qu'on s'intéresse à une réponse *oui/non* et que la *terminaison* est garantie.

256

- Traditionnellement, on distingue deux types d'algorithmes probabilistes :

Type Las Vegas L'algorithme peut rendre trois types de réponses : oui, non, je ne sais pas.

Type Montecarlo L'algorithme peut rendre une réponse oui/non. Cependant la réponse non est correcte seulement avec une certaine probabilité p .

NB Le test de Miller-Rabin qu'on considère dans la suite est un algorithme de Montecarlo où *oui* = *pas premier*.

257

Exercice (factorisation)

Soit $n \geq 2$. Montrez que si n n'est pas premier alors il y a un premier p tel que $p \mid n$ et $p \leq \sqrt{n}$.

259

Factorisation par division par essai

258

Division par essai

- Pour s'assurer que n est premier, on vérifie que

$$p \nmid n \text{ pour } p \text{ premier et } 2 \leq p \leq \sqrt{n}$$

260

- La liste des premiers inférieurs à un n donné peut être générée avec le *crible d'Eratosthène* :

$P[i] := true$ pour $i = 2, \dots, n$

for $i = 2, \dots, n/2$ do

 for $j = 2, \dots, n/i$ do

$P[i \cdot j] := false$

261

- Exemple Eratosthène pour $n = 10$

i	$i \cdot j, j = 2, \dots, 10/i$
2	$2 \cdot 2, 2 \cdot 3, 2 \cdot 4, 2 \cdot 5$
3	$3 \cdot 2, 3 \cdot 3$
4	$4 \cdot 2$
5	$5 \cdot 2$

262

- Exemple : division par essai pour $n = 15413$.

$$\lfloor \sqrt{15413} \rfloor = 124$$

Les premiers inférieurs à 124 sont :

3 5 7 11 13 17 19 23 29
 31 37 41 47 53 59 61 67
 71 73 79 83 97 101 103 109 113

Aucun divise n . Donc n est premier.

263

Division par essai itérée

On peut itérer la division par essai pour trouver une factorisation complète.

1. On trouve p_1 tel que $p_1 \mid n$.
2. Ensuite on trouve p_2 tel que $p_2 \mid n/p_1$.
3. ...

264

Exercice (division par essai)

Trouvez la factorisation en nombres premiers de $n = 476$.

265

- Pour $n = 10^6$, on a environ 10^3 divisions.
- Pour les applications RSA, on a $n = 10^{75}$. Ceci donne 10^{36} divisions qui est beaucoup trop.

NB Des algorithmes de factorisation plus efficaces ont été proposés. Ceci pourrait être le sujet d'un autre cours...

267

Efficacité de la division par essai

- Soit $\pi(n) = \#\{p \mid p \text{ premier et } p \leq n\}$.
- On sait que :
 1. Si $n \geq 17$ alors $\pi(n) > n/\log(n)$.
 2. Si $n \geq 2$ alors $\pi(n) < (1.3)(n/\log(n))$.
- Pour démontrer que n est premier avec la division par essai on a besoin d'au moins $\sqrt{n}/\log(\sqrt{n})$ divisions.

266

Exercice (premier aléatoire)

Estimez la probabilité qu'un nombre binaire sur 512 bits, dont le premier et le dernier bit est 1, est premier.

268

Test de Fermat

– On rappelle que :

$$\text{pgcd}(a, m) = 1 \Rightarrow (a^{\phi(m)} \equiv 1) \text{ mod } m$$

– Si m est premier alors $\phi(m) = m - 1$, ainsi on a :

$$\text{pgcd}(a, m) = 1 \Rightarrow (a^{(m-1)} \equiv 1) \text{ mod } m$$

269

Exercice (test de Fermat)

Utilisez le test de Fermat pour montrer que 1111 n'est pas un nombre premier.

271

– On pourrait essayer le test suivant :

1. Choisir $a \in \{2, \dots, m - 1\}$.
2. Si $\text{pgcd}(a, m) \neq 1$ alors m n'est pas premier. Sinon, calculer $a^{(m-1)}$ par exponentiation rapide.
3. Vérifier si $(a^{(m-1)} \equiv 1) \text{ mod } m$. Sinon, m n'est pas premier.

270

Test de Fermat (suite)

– D'autre part si $(a^{(m-1)} \equiv 1) \text{ mod } m$ alors *on ne peut pas* conclure que m est premier. Par exemple

1. Soit $n = 341$.
2. Il s'agit d'un nombre composé $n = 11 \cdot 31$.
3. Mais en prenant $a = 2$, on a $(2^{340} \equiv 1) \text{ mod } 341$.

Cependant, on remarque que $(3^{340} \equiv 56) \text{ mod } 341$. Est-ce toujours le cas ?

272

Nombres de Carmichael (1910)

- Un nombre de Carmichael est un nombre composé n tel que

$$(a^{(n-1)} \equiv 1) \pmod n$$

pour tous les entiers a tels que $\text{pgcd}(a, n) = 1$.

- Mauvaises nouvelles ! Les nombres de Carmichael existent. Le plus petit est

$$3 \cdot 11 \cdot 17 = 561$$

- En plus il y en a une infinité (même s'ils ne sont pas très fréquents). On sait qu'un nombre de Carmichael a au moins 3 facteurs premiers.
- Soit $n = p_1 \cdots p_k$ nombre de Carmichael avec p_j premier pour $j = 1, \dots, k$. Les seuls témoins potentiels de la non-primauté de n sont les nombres qui contiennent p_j comme facteur. Ces témoins peuvent être trop rares.

273

Le test

- Soit $n \geq 3$ un nombre impair candidat à être premier.
- On calcule k, m tels que $(n - 1) = 2^k m$, m impair.
- On choisit $a \in \{1, \dots, n - 1\}$.
- On vérifie :

$$(a^m \equiv 1) \pmod n \text{ ou } \exists i \in \{0, \dots, k - 1\} (a^{2^i m} \equiv -1) \pmod n$$

275

Test de Miller-Rabin

Une variante du test de Fermat.

Fait Si un nombre n est impair et composé alors l'ensemble $\{1, \dots, n - 1\}$ contient au plus $(n - 1)/4$ nombres qui sont premiers avec n et qui ne *témoignent pas* du fait que n est composé.

- Donc il n'y a pas d'analogue des nombres de Carmichael pour le test de Miller-Rabin.
- En plus la probabilité de trouver un témoin est au moins $3/4$. Ce test est parmi les plus efficaces.

274

Exemple

- On prend le plus petit nombre de Carmichael :
 $n = 561 = 3 \cdot 11 \cdot 17$.
- Alors $n - 1 = 560 = 2^4 \cdot 35$. Donc $k = 4$ et $m = 35$.
- On choisit $a = 2 \in \{1, \dots, 560\}$.
- Alors on calcule :

$$(2^{35} \equiv 263) \pmod{561} \quad (2^{2 \cdot 35} \equiv 166) \pmod{561}$$

$$(2^{4 \cdot 35} \equiv 67) \pmod{561} \quad (2^{8 \cdot 35} \equiv 1) \pmod{561}$$

- Ceci est assez pour conclure que 561 est *composé*.

276

Exercice (Miller-Rabin)

Déterminez tous les éléments $a \in \{2, \dots, 14\}$ premiers avec 15 et qui témoignent du fait que 15 est composé par le biais du test de Miller-Rabin.

277

Exercice (résidu quadratique)

Calculez les résidus quadratiques de \mathbf{Z}_{11} .

279

Résidu Quadratique

Pour montrer que tous les premiers passent le test de Miller-Rabin, on a besoin d'étudier la solution d'une équation quadratique modulo p .

Définition Soit $p \geq 3$ premier et $a \in \mathbf{Z}$ tel que $(a \not\equiv 0) \pmod{p}$. On dit que a est un *résidu quadratique* modulo p si

$$\exists y \ (y^2 \equiv a) \pmod{p}$$

278

Solutions d'une équation quadratique

Proposition On suppose $p \geq 3$ premier et a résidu quadratique modulo p . Alors l'équation

$$((x^2 - a) \equiv 0) \pmod{p}$$

a exactement *deux solutions* modulo p .

280

Preuve

- Comme a est un résidu quadratique, il y a y tel que

$$(y^2 \equiv a) \pmod{p}$$

- Alors $((-y)^2 \equiv a) \pmod{p}$ et en plus $y \not\equiv (-y) \pmod{p}$ car p est impair.

- Ceci montre que l'équation $(x^2 - a) \equiv 0 \pmod{p}$ a au moins deux solutions.

281

Chaque premier satisfait le test de Miller-Rabin

- On procède par contradiction. On suppose que $n \geq 3$ premier échoue le test.

- Ceci veut dire que

$$(a^m \not\equiv 1) \pmod{n} \text{ et } (a^{2^i m} \not\equiv -1) \pmod{n}, i = 0, \dots, k-1$$

- Comme n est premier, on sait que $\phi(n) = n - 1$.

283

- D'autre part, l'équation $(x^2 - a) \equiv 0 \pmod{p}$, peut être ré-écrite

$$((x - y)(x + y) \equiv 0) \pmod{p}$$

- Ceci implique :

$$p \mid (x - y)(x + y)$$

- Comme p est premier :

$$p \mid (x - y) \text{ ou } p \mid (x + y)$$

- C'est-à-dire :

$$(x \equiv y) \pmod{p} \text{ ou } (x \equiv (-y)) \pmod{p}$$

282

- En sachant $(n - 1) = 2^k m$, par le théorème de Fermat :

$$(a^{(n-1)} \equiv a^{2^k m} \equiv 1) \pmod{n}$$

- Il suit que $a^{2^{k-1} m}$ est une racine carrée de 1 modulo n .

- Par ce qu'on sait des résidus quadratiques, il y a exactement deux possibilités : $+1$ ou -1 .

284

– Comme le test échoue, on doit avoir

$$(a^{2^{k-1}m} \equiv 1) \text{ mod } n$$

– Maintenant, $a^{2^{k-2}m}$ est une racine de 1 modulo n . Par un raisonnement similaire on peut conclure que

$$(a^{2^{k-2}m} \equiv 1) \text{ mod } n$$

– On itère cet argument jusqu'à $k = 1$, pour obtenir

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

ce qui contredit l'hypothèse que n échoue le test.

285

Aléa et pseudo-aléa

287

Sommaire : comment générer un nombre premier aléatoire sur k bits

1. On fixe à 1 le premier et le dernier bit.
2. On génère de façon aléatoire $k - 2$ bits. Soit n le nombre impair sur k qui en résulte.
3. On vérifie si n est divisible par un nombre premier $p \leq B$ (où typiquement $B = 10^6$).
4. Sinon, on applique le test de Miller-Rabin t fois (en choisissant des éléments différents). La probabilité d'erreur est bornée par $1/4^t$.
5. Si tous les t tests sont positifs alors on considère que n est premier.

286

Avertissement

- On pourrait passer tout un cours sur la notion d'aléa et pseudo-aléa. . .
- En particulier, la définition mathématique de génération pseudo-aléatoire est basée sur la théorie de la complexité.

288

Certaines méthodes pour la génération aléatoire

Tables RAND En 1955, *Rand corporation* publie un livre qui contient un million de chiffres aléatoires. Les chiffres sont produites en utilisant une sorte de roulette électronique.

Frappe au clavier On mesure le temps entre deux frappes sur le clavier, le temps pour lire un bloc dans le disque dur, ou un autre événement physique. . . et on prend le bit le moins significatif de la mesure (ceci marche si on a besoin de peu de bits aléatoires, par exemple pour générer une clef).

Bruit aléatoire On utilise des dispositifs spécialisés qui mesurent l'affaiblissement radioactif, la radiation thermique, . . .

289

XOR plusieurs bits ensemble

– Par exemple, on suppose

$$P(x = 0) = 1/2 + p \quad 1/2 > p > 0$$

– En prenant le XOR de 2 bits ensemble on a :

$$P(x = 0) = 1/2 + 2 \cdot p^2$$

– en prenant le XOR de 4 bits on a

$$P(x = 0) = 1/2 + 8 \cdot p^4$$

291

Observations biaisées et corrélations

Le problème n'est pas tellement de trouver une source aléatoire mais plutôt d'échantillonner la source de façon à préserver le comportement aléatoire.

290

Réduire la corrélation

– Un problème de ces méthodes est que s'il y a une corrélation entre bits adjacents le filtrage va l'incrémenter.

– Une façon de traiter le problème est de prendre les XOR de bits provenant de deux sources *indépendantes*.

292

Un produit commercial (2007)

<http://www.idquantique.com/products/quantis.htm>

Quantis is a physical random number generator exploiting an elementary quantum optics process. Photons are sent one by one onto a semi-transparent mirror and detected.

The exclusive events (reflection or transmission) are associated to '0' or '1' bit values (...)

Quantis is available as a (...) USB module.

Quantis produces random numbers at a very high rate, up to 16Mbs.

293

Une citation

Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.

J. Von Neumann

295

Générateur pseudo-aléatoire

Une fonction mathématique qui états donné un *germe (seed)* initial produit une séquence de bits qui *semble aléatoire* et *imprévisible* (sans connaître le germe).

294

Cependant il y a des avantages...

- Pas de dispositif spécial.
- Comportement reproductible.
 - En *simulation*, en commençant avec le même germe on peut exécuter exactement la même simulation.
 - En *cryptographie*, deux participants qui partagent le même germe peuvent produire la même séquence de bits pseudo-aléatoires et ils peuvent l'utiliser comme clef pour un système à masque jetable.

296

Avertissement

- Les générateurs pour le test et la simulation ne sont pas adaptés à la cryptographie. Par exemple, les congruences linéaires marchent bien pour la simulation mais sont à proscrire en cryptographie.
- Une grande partie des générateurs pseudo-aléatoires dans les bibliothèques de langages de programmation ne sont pas adaptés aux applications cryptographiques.

297

Un exemple sans preuve : générateur de Blum-Blum-Shub

1. On trouve deux grands nombres premiers p, q tels que $p \equiv q \equiv 3 \pmod{4}$ (ces nombres ont des propriétés spéciales par rapport aux racines carrées. . .)
2. Soit $n = pq$. On choisit x premier avec n .
3. On calcule $x_0 = x^2 \pmod{n}$, ceci est le germe.
4. La séquence de bits est le bit le moins significatif de $x_{i+1} = x_i^2 \pmod{n}$.

NB La sécurité dépend de la difficulté à factoriser n .

299

État de l'art

- Les générateurs pseudo-aléatoires pour les applications cryptographiques peuvent être construits.
- Ces générateurs sont sûrs tant qu'un algorithme pour un certain problème difficile de la théorie des nombres n'a pas été découvert.

298

Systemes basés sur le logarithme discret

300

Logarithme discret

Soit p premier. On rappelle :

Fait $(\mathbf{Z}_p)^*$ est toujours un groupe cyclique (avec $\phi(p-1)$ générateurs).

Définition Soit g un générateur. Pour un entier $A \in \{1, \dots, p-1\}$ le *logarithme discret* en base g est défini par :

$$dlog_g(A) = a \quad \text{si } a \in \{0, \dots, p-2\} \text{ et } (g^a \equiv A) \text{ mod } p$$

NB Comme pour la factorisation, on ne connaît pas une méthode efficace pour calculer le logarithme discret dans \mathbf{Z}_p . Par contre, on connaît des méthodes *guess and test* pour trouver un générateur (cf. primalité).

301

Exercice (logarithme discret dans d'autres groupes)

Le logarithme discret peut être défini dans *tout groupe cyclique*. Selon le groupe choisi, le calcul peut être plus ou moins difficile... Par exemple, on considère le groupe additif \mathbf{Z}_n pour n positif.

1. Déterminez un générateur de \mathbf{Z}_n .
2. Pouvez-vous déterminer tous les générateurs du groupe ?
3. Est-il difficile de calculer le logarithme discret ?

303

Exercice (logarithme discret dans $(\mathbf{Z}_p)^*$)

On considère le groupe multiplicatif $(\mathbf{Z}_{13})^*$.

1. Montrez que 2 est un générateur.
2. Calculez le logarithme discret en base 2 pour tous les éléments de $(\mathbf{Z}_{13})^*$.
3. Déterminez tous les générateurs du groupe.

302

Exercice (trouver un générateur dans $(\mathbf{Z}_p)^*$)

1. Montrez que si $p > 2$ est premier et $a \in (\mathbf{Z}_p)^*$ alors a est un générateur du groupe multiplicatif si et seulement si $(a^{(p-1)/r} \not\equiv 1) \text{ mod } p$ pour tout r premier tel que $r \mid (p-1)$.
2. Concluez que si l'on sait que $(p-1)/2$ est premier alors on peut tester de façon efficace si a est un générateur.

304

Le protocole pour l'échange de clefs

Échange de clefs de Diffie Hellman

- Attention, ce n'est pas un système à clef publique (même si un tel système peut en être dérivé).
- Il s'agit d'un protocole qui permet à deux participants qui partagent seulement de l'information publique d'établir une clef commune.
- Le protocole a été proposé *avant* RSA. On peut le voir comme une autre méthode pour initialiser la communication entre participants.

305

Exercice (échange de clef)

On suppose que $p = 17$, $g = 3$, Héloïse choisit $a = 7$, et Abélard $b = 4$. Quelle est la clef commune générée par le protocole ?

307

1. Héloïse et Abélard se mettent d'accord sur un *grand nombre premier* p et un *générateur* g pour $(\mathbf{Z}_p)^*$ tel que $2 \leq g \leq p - 2$.
2. Héloïse choisit de façon aléatoire $a \in \mathbf{Z}_{p-2}$, et elle calcule et envoie à Abélard $A = (g^a) \bmod p$.
3. Abélard choisit de façon aléatoire $b \in \mathbf{Z}_{p-2}$, il calcule et envoie à Héloïse $B = (g^b) \bmod p$.
4. La *clef commune* $K = g^{ab} \bmod p$ est extraite :

$$K = B^a \bmod p \quad (\text{comme Héloïse calcule } K)$$

$$K = A^b \bmod p \quad (\text{comme Abélard calcule } K)$$

306

Problème de Diffie-Hellman

- Étant donné (p, g, A, B) (mais en ignorant le logarithme discret de A et B) peut-on déterminer K ?
- Clairement, si on peut calculer le logarithme discret de façon efficace alors on peut déterminer K (et résoudre le problème de Diffie-Hellman). Par exemple, on calcule $a = d\log_g(A)$ et $K = (B^a) \bmod p$.
- On ne sait pas si la solution du problème de DH est suffisante pour calculer le logarithme discret de façon efficace (cf. situation RSA).

308

Attaque de l'homme au milieu (man in the middle)

Diffie-Hellman est conçu pour résister à un attaquant *passif*. Que se passe-t-il si l'attaquant est actif, c'est-à-dire s'il peut intercepter et générer des messages ?

- L'attaquant exécute le protocole d'échange de clef avec Héloïse en jouant le rôle d'Abélard et avec Abélard en jouant le rôle d'Héloïse.
- A la fin des deux protocoles, Héloïse et Abélard vont envoyer des messages chiffrés avec deux clefs connues par l'attaquant.
- A remarquer que cette attaque n'est pas basée sur une manipulation de l'information publique (g, p) . Le problème est l'*authentification* du correspondant.

309

Chiffrement Abélard connaît (p, g, A) . Pour chiffrer un message

$m \in \mathbf{Z}_p$ pour Héloïse, il choisit *un exposant aléatoire* $b \in \{0, 1, \dots, p-2\}$ et il calcule (jusqu'ici c'est comme l'échange de clef!) :

$$B = (g^b) \bmod p$$

Pour *chiffrer* m , Abélard calcule (remarquez que A^b était la clef dans DH) :

$$c = A^b m \bmod p$$

Ainsi le code chiffré comprend le produit de la clef DH avec m modulo p :

$$E_{(p,g,A)}(m) = (B, c) .$$

311

ElGamal

Il s'agit d'un système à *clef publique* basé sur le problème de Diffie-Hellman et effectivement utilisé (GPG).

Génération de la Clef

1. Héloïse génère un *grand nombre premier* p et un générateur g pour $(\mathbf{Z}_p)^*$ tel que $2 \leq g \leq p-2$.
2. Ensuite elle choisit de façon aléatoire un *exposant* $a \in \{0, 1, \dots, p-2\}$ et elle calcule :

$$A = g^a \bmod p$$

La *clef publique* est (p, g, A) . La *clef secrète* est l'exposant a .

310

Déchiffrement Héloïse obtient (B, c) et elle connaît a . Pour calculer m , elle divise c par $B^a \bmod p$ comme suit :

1. Calculer (une fois) $x = p-1-a$, $1 \leq x \leq p-1$.
2. Calculer $B^x c \bmod p$ (ceci est égal à m).

312

Générer :

p premier $g \in \{2, \dots, p-2\}$ générateur pour $(\mathbf{Z}_p)^*$ $a \in \mathbf{Z}_{p-1}$

Calculer :

Clef Publique : $(p, g, A = (g^a) \bmod p)$ Clef Privée : a

Chiffrement et Déchiffrement

Texte clair $\mathcal{P} = \mathbf{Z}_p$ Texte chiffré $\mathcal{C} = \mathbf{Z}_p \times \mathbf{Z}_p$

$E_{(p,g,A)}(m) = (\nu b \in \mathbf{Z}_{p-1})(g^b \bmod p, (A^b \cdot m) \bmod p)$

$D_{(p,g,a)}(B, c) = (B^{p-1-a} \cdot c) \bmod p$

NB La chiffrement *n'est pas* une fonction (composante aléatoire).

Vérifions que $B^x c \bmod p = m$ pour $x = p-1-a$:

$$\begin{aligned} B^x c &\equiv g^{b(p-1-a)} A^b m \\ &\equiv (g^{p-1})^b (g^{-ab}) (g^{ab}) m \\ &\equiv (1)^b (1) m \\ &\equiv m \bmod p \end{aligned}$$

313

314

Comparaison de RSA et ElGamal

- Le déchiffrement dans RSA et ElGamal demandent *1 exponentiation modulaire*.
- Le chiffrement dans ElGamal demande *2 exponentiations modulaires* alors que dans RSA 1 suffit.
- Cependant dans ElGamal les exponentiations pour le chiffrement sont *indépendantes* du texte clair et elles pourraient être calculées à l'avance. Sous cette hypothèse le chiffrement dans ElGamal demande seulement *1 multiplication modulaire*, ce qui est plus facile à calculer que 1 exponentiation. D'autre part, les valeurs pré-calculées doivent être gardées secrètes.
- Dans ElGamal le *texte chiffré est deux fois plus long que le texte clair*. Dans RSA il n'est pas nécessaire de doubler le message.
- Dans ElGamal chaque chiffrement dépend d'un choix aléatoire d'un exposant b . Ainsi si le même texte clair m est chiffré deux fois, on choisit deux exposant b et b' de façon aléatoire et donc les deux textes chiffrés sont (très probablement) différents.
- ElGamal peut être implémenté dans chaque groupe cyclique où le logarithme discret est difficile à calculer (autant qu'on sache...). Cas importants : *courbes elliptiques* sur les corps premiers.

315

316

Exercice (ElGamal)

1. On suppose qu'Héloïse choisit $p = 23$, $g = 7$ et $a = 6$. Quelles sont les clefs publique et privée ?
2. On suppose Abélard chiffre $m = 7$ en choisissant $b = 3$. Quel est le texte chiffré ?
3. Comment Héloïse peut-elle obtenir le texte clair à partir du texte chiffré ?

317

Exercice (ElGamal déchiffrement)

Soit $p = 53$, $g = 2$, $A = 30$ la clef publique d' Héloïse dans le système d'ElGamal. Abélard l'utilise pour générer le texte chiffré $(24, 37)$. Déterminez le texte clair correspondant.

319

Exercice (plagiat dans ElGamal)

Montrez que dans le système ElGamal, on peut utiliser deux textes chiffrés pour obtenir le chiffrement d'un texte clair inconnu. Comment prévenir ce type d'attaque ?

318

Hachage

320

Terminologie

- Soit Σ un alphabet donné.
- Une *fonction de hachage* est une fonction $h : \Sigma^* \rightarrow \Sigma^n$.
- Une *fonction de compression* est une fonction $h : \Sigma^m \rightarrow \Sigma^n$ où $m > n$.

321

3. Pour un x donné, il devrait être difficile de trouver un autre x' tel que $h(x) = h(x')$. Si c'est le cas, on dit que h est *faiblement résistante aux collisions*. Exemple :
 - On veut protéger un programme important de toute modification (e.g., le programme qui calcule le chiffrement).
 - On mémorise le hachage du programme dans un endroit sûr (par exemple, une smart card).
 - Avant d'utiliser le programme, on compare toujours le hachage du programme à la valeur mémorisée dans un endroit sûr.Ceci marche si h est faiblement résistante aux collisions.

323

Propriétés souhaitables

Les fonctions de hachage sont utilisées dans plusieurs contextes. En cryptographie, une fonction de hachage doit avoir certaines propriétés particulières (voir le protocole pour jouer à pile ou face au téléphone) :

1. Pour un x donné, il devrait être facile de calculer $h(x)$.
2. Pour un y donné, il devrait être difficile de trouver une *image inverse* x telle que $h(x) = y$, c'est-à-dire h est à *sens unique*. Par exemple, l'exponentiation modulaire est facile à calculer mais le logarithme discret est difficile (pour l'instant!).

322

4. Il devrait être difficile de trouver x, x' tels que $h(x) = h(x')$. Dans ce cas on dit que h est *fortement résistante aux collisions*. Cette propriété est nécessaire par exemple dans les applications aux signatures digitales (à venir).

324

Les fonctions fortement résistantes aux collisions sont à sens unique

Voici l'idée (informelle) :

1. On suppose disposer d'un algorithme d'inversion.
2. On choisit x' de façon aléatoire. En utilisant l'algorithme, on détermine un x tel que $h(x) = h(x')$.
3. (x, x') va être une collision sauf si $x = x'$, ce qui est peu probable.

325

– Tous les événements ayant la même probabilité, on compte le nombre d'événements (d_1, \dots, d_k) tels que $d_i \neq d_j$ si $i \neq j$.

– Ce nombre est :

$$n(n-1) \cdots (n-k+1) = \prod_{i=0, \dots, k-1} (n-i)$$

– Donc la probabilité q qu'il *n'y ait pas deux personnes* avec la même date d'anniversaire est

$$\begin{aligned} q &= (1/n^k) \prod_{i=0, \dots, k-1} (n-i) \\ &= \prod_{i=0, \dots, k-1} (n-i)/n \\ &= \prod_{i=1, \dots, k-1} (1 - i/n) \end{aligned}$$

327

Attaque de l'anniversaire (birthday attack)

On estime l'effort nécessaire pour trouver une collision.

- Combien de personnes faut-il réunir pour que la probabilité de trouver deux personnes avec la même date d'anniversaire soit supérieure ou égale à $1/2$?
- Soit n le nombre de jours dans une année et k le nombre de personnes.
- Un événement est un élément de $\{1, \dots, n\}^k$.

326

– Comme $1 + x \leq e^x$ pour tout nombre réel, on dérive :

$$q \leq \prod_{i=1, \dots, k-1} e^{-i/n} = e^{-\sum_{i=1, \dots, k-1} (i/n)} = e^{-k(k-1)/(2n)}$$

– En appliquant le logarithme et en étudiant l'inégalité quadratique on dérive que pour avoir $q \leq 1/2$, il faut poser :

$$k \geq (1 + (1 + 8n \log(2))^{1/2}/2)$$

– En particulier, pour $n = 365$, il suffit d'avoir $k = 23$.

328

Encore sur les fonctions de hachage !

- Si $\Sigma = \{0, 1\}$, alors le nombre de valeurs de hachage (les dates d'anniversaire) est 2^n et l'inégalité devient :

$$k \geq (1 + (1 + 8 \log(2) 2^n)^{1/2} / 2)$$

- Ceci signifie que si on calcule un peu moins que $2^{n/2}$ valeurs alors la probabilité de trouver une collision est plus grande que $1/2$.
- Actuellement, on recommande $n \geq 160$.

329

Théorème [Merkle-Damgard]

On travaille sur un alphabet binaire $\Sigma = \{0, 1\}$. On suppose que $c : 2^{m+t} \rightarrow 2^m$ est une fonction de compression résistante aux collisions avec $t \geq 1$. Alors on peut construire une fonction de hachage résistante aux collisions :

$$h : \bigcup_{i > m+t} 2^i \rightarrow 2^m$$

En supposant $n = |x|$, le nombre de fois qu'on appelle c pour calculer $h(x)$ est au plus :

$$\begin{array}{ll} 1 + \lceil n/(t-1) \rceil & \text{si } t \geq 2 \\ 2n + 2 & \text{si } t = 1 \end{array}$$

331

Des fonctions de compression aux fonctions de hachage

- On dit qu'une fonction de hachage (ou de compression) est *résistante aux collisions* s'il n'y a pas d'algorithme probabiliste polynomial en temps qui trouve une collision.
- On ne sait pas si de telles fonctions existent. Cependant, on peut montrer que si de telles fonctions de compression existent alors de telles fonctions de hachage existent aussi.

330

Idée de la preuve

- La preuve montre qu'étant donnée une collision (x, x') pour la fonction de hachage, on peut trouver en temps polynomial une collision pour la fonction de compression.
- Il y a deux cas à considérer pour $t \geq 2$ et $t = 1$. On présente le cas $t = 1$. Le cas $t \geq 2$ est assez similaire.

332

Preuve pour $t = 1$

– Soit f une fonction définie par :

$$f(0) = 0 \quad f(1) = 01$$

333

– On remarque deux propriétés importantes de la transformation

$$x \mapsto y = y(x).$$

1. la transformation est *injective* (Pourquoi?).
2. aucun codage est un *suffixe propre* d'un autre codage, c'est-à-dire, $y(x) = zy(x')$ avec $z \neq \epsilon$.

La deuxième propriété suit de l'observation que la suite 11 peut se trouver seulement au début d'un mot.

335

– Voici l'algorithme qui calcule la fonction de hachage (pour $t = 1$) :

1. On suppose $x = x_1 \cdots x_n$. On définit :

$$y = y_1 \cdots y_k = 1 \cdot 1 \cdot f(x_1) \cdots f(x_n) \quad (n+2) \leq k \leq (2n+2)$$

2. On itère $c : 2^{m+1} \rightarrow 2^m$, k fois :

$$g_1 := c(0^m \cdot y_1)$$

$$g_2 := c(g_1 \cdot y_2)$$

... ..

$$g_k := c(g_{k-1} \cdot y_k)$$

3. Alors $h(x) = g_k$.

334

– On suppose avoir trouvé $x \neq x'$ avec $h(x) = h(x')$. On dénote :

$$y(x) = y_1 \cdots y_k \quad y(x') = y'_1 \cdots y'_l$$

– On considère deux cas $k = l$ et $k < l$ pour conclure que ou bien il y a une collision pour c ou bien on tombe sur une contradiction.

336

– Cas $k = l$. On sait :

$$g_k = c(g_{k-1} \cdot y_k) = c(g'_{k-1} \cdot y'_k) = g'_k$$

– Si $y_k \neq y'_k$ on a trouvé une collision pour c .

– Si $y_k = y'_k$ alors on doit avoir $g_{k-1} = g'_{k-1}$ (autrement on a une collision pour c).

337

– Cas $k < l$. On raisonne comme dans le cas précédent et on suppose qu'aucune collision a été trouvée.

– Alors on dérive :

$$y_k = y'_k \quad y_{k-1} = y'_{k-1} \quad \dots \quad y_1 = y'_{l-k+1}$$

– Mais ceci contredit le fait que $y(x)$ n'est pas un suffixe propre de $y(x')$.

Fin de la preuve pour $t = 1$.

339

– En itérant le raisonnement, on dérive :

$$y_k = y'_k \quad y_{k-1} = y'_{k-1} \quad \dots \quad y_1 = y'_1$$

– Donc on a $y(x) = y(x')$ et comme la transformation est injective, on dérive que $x = x'$. Contradiction !

338

Une construction simplifiée pour $t \geq 64$

– Soit $c : 2^{m+t} \rightarrow 2^m$ la fonction de compression avec $t > 1$.

– Soit $x = x_1 \dots x_n$ la suite de bits à hacher.

– On fait le bourrage (*padding*) suivant de x

$$y(x) = x_1 \dots x_n \cdot 0 \dots 0 \cdot (n)_2$$

où $(n)_2$ est la représentation en base 2 de n (la longueur de x) sur 64 bits (ce qui suffit en pratique).

340

- Dans le bourrage, on met assez de '0' pour que la longueur du mot à hacher soit un multiple de t . Ainsi :

$$y(x) = B_1 \cdots B_r \text{ où } B_i \in 2^t$$

- Soit $IV \in 2^m$ un vecteur d'initialisation de m bits.

- Alors on calcule $h(x) = H_r$ où :

$$H_1 = c(IV, B_1)$$

$$H_2 = c(H_1, B_2)$$

$$\cdots = \cdots$$

$$H_r = c(H_{r-1}, B_r)$$

341

- Un schéma populaire pour hacher un mot

$$x = x_1 \cdots x_n \quad x_i \in 2^t \text{ pour } i = 1, \dots, n$$

est d'adapter le mode de *transmission CBC* :

$$y_0 = IV$$

$$y_{i+1} = E_k(y_i \oplus x_i) \quad i = 0, \dots, n-1$$

$$h(x) = y_n$$

343

Fonctions de compression (pragmatique)

- On ne sait pas si les fonctions de compression/hachage 'sûres' existent.
- En pratique, on se contente de construire une fonction de compression à partir d'une fonction de *chiffrement*. Le résultat semble résister aux collisions pour peu que la fonction de chiffrement soit 'sûre'.
- Par exemple, on suppose $\mathcal{P} = \mathcal{C} = 2^t$ avec $t \geq 160$.

342

Des données

Fonction de hachage	longueur du bloc	vitesse relative
MD4	128	1.00
MD5	128	0.68
SHA-1	160	0.28

344

- Une collision pour MD4 peut être trouvée en calculant 2^{20} valeurs de hachage.
- Une collision de la fonction de compression de MD5 a été trouvée aussi.
- Avancée récente (2004) : une équipe chinoise a trouvé une méthode pour trouver une collision dans SHA-1 en 2^{69} essais. Ceci est 2000 fois plus rapide que les 2^{80} essais demandés par une méthode de recherche complète.
- Encore des progrès depuis !

345

Terminologie : Message authentication codes (MAC)

- Il s'agit d'une *famille de fonctions de hachage* $\{h_k \mid k \in \mathcal{K}\}$.
- \mathcal{K} est l'espace des clefs.
- Une fonction de hachage $h : 2^* \rightarrow 2^n$ peut être transformée en MAC très simplement en définissant $\mathcal{K} = 2^n$ et

$$h_k(x) = h(x) \oplus k$$

346

Une fonction de compression arithmétique (Chaum et al.)

- On construit une fonction de compression dont on peut *démontrer* qu'elle résiste aux collisions si le calcul du logarithme discret dans $(\mathbf{Z}_p)^*$ est difficile.
- On suppose que p et $q = (p - 1)/2$ sont premiers.
- On suppose que a est un générateur pour $(\mathbf{Z}_p)^*$ et b choisi de façon aléatoire dans $\{1, \dots, p - 1\}$.

347

- On considère la fonction :

$$c : \mathbf{Z}_q \times \mathbf{Z}_q \rightarrow \mathbf{Z}_p \quad c(x_1, x_2) = a^{x_1} b^{x_2} \text{ mod } p$$

Comme $q = (p - 1)/2$, la taille de (x_1, x_2) est approximativement deux fois la taille de p .

348

Exercice (sur la compression)

On suppose $q = 11$, $p = 23$, $a = 5$, et $b = 4$. Calculez $c(5, 10)$.

349

1. Si $(ax \equiv b) \pmod m$ a une solution alors $g = \text{pgcd}(a, m)$ divise b .
2. Si $\text{pgcd}(a, m) = 1$ alors la congruence $(ax \equiv b) \pmod m$ a une solution unique modulo m .
3. Si $g = \text{pgcd}(a, m)$ divise b alors supposez $a' = a/g$, $b' = b/g$ et $m' = m/g$ et montrez que $(a'x \equiv b') \pmod{m'}$ a une solution unique, disons x' .

351

Exercice (solutions $(ax \equiv b) \pmod m$)

– On sait déjà :

$(ax \equiv 1) \pmod m$ a une solution unique dans \mathbf{Z}_m ssi $\text{pgcd}(a, m) = 1$.

– On considère la congruence plus générale :

$$(ax \equiv b) \pmod m$$

– En particulier, pour montrer comment calculer le logarithme discret à partir d'une collision on a besoin du fait suivant (voir exercice qui suit) :

Si $2 = \text{pgcd}(a, m)$ et 2 divise b alors la congruence $(ax \equiv b) \pmod m$ a exactement deux solutions dans \mathbf{Z}_m .

350

4. Si $g = \text{pgcd}(a, m)$ divise b alors $x' + ym'$ est une solution de $ax \equiv b \pmod m$ pour tout $y \in \mathbf{Z}$.
5. Si $g = \text{pgcd}(a, m)$ divise b et $(ax \equiv b) \pmod m$ alors $(x \equiv x') \pmod{m'}$.
6. Enfin, supposez $2 = \text{pgcd}(a, m)$ divise b . Combien de solutions la congruence $(ax \equiv b) \pmod m$ a-t-elle modulo m ?

352

De la collision au logarithme discret

- Une *collision* est un couple (x_1, x_2) et (x_3, x_4) tel que

$$(a^{x_1} b^{x_2} \equiv a^{x_3} b^{x_4}) \text{ mod } p$$

- Ceci implique :

$$(a^{x_1 - x_3} \equiv b^{x_4 - x_2}) \text{ mod } p$$

353

- La congruence a une solution si et seulement si $d = \text{pgcd}(x_4 - x_2, p - 1)$ divise $x_1 - x_3$ (*voir exercice !*).

- On sait que $|x_4 - x_2| < q$ car $x_2, x_4 \in \mathbf{Z}_q$. Comme $p - 1 = 2q$ et q est premier, on dérive :

$$d \in \{1, 2\}$$

355

- On montre que *trouver une collision* permet de calculer le *logarithme discret* :

$$y = d \log_a(b) \text{ modulo } p$$

- Comme $b = a^y$, la congruence précédente donne :

$$(a^{x_1 - x_3} \equiv a^{y(x_4 - x_2)}) \text{ mod } p$$

- Comme a est un *générateur* pour $(\mathbf{Z}_p)^*$, ceci implique :

$$(x_1 - x_3 \equiv y(x_4 - x_2)) \text{ mod } (p - 1) \text{ et } (p - 1) = 2q$$

354

- Si $d = 1$ alors la solution modulo $p - 1$ de la congruence précédente est unique et peut être calculée. Ceci permet de déterminer le logarithme discret.

- Si $d = 2$ alors la congruence a deux solutions différentes (*voir exercice !*) modulo $p - 1$ et le logarithme discret peut être calculé en essayant les deux.

NB La fonction de hachage de Chaum nécessite une exponentiation modulaire. Il s'agit d'un calcul qui est considéré comme trop lourd pour une fonction de hachage.

356

Une application aux problèmes d'identification

- On appelle *protocoles d'identification* les méthodes qui permettent l'identification d'un participant.
- L'objectif typique d'un protocole d'identification est le *contrôle d'accès*. Par exemple, pour savoir combien d'argent reste sur votre compte en banque, pour lire/modifier vos fichiers, . . .
- Dans un protocole d'identification, Héloïse –le *démonstrateur*– doit convaincre (en temps réel) Abélard –le *vérificateur*– de son identité.

357

Des attaques simples. . .

- Attaque du dictionnaire : obtenir le fichier des mots de passe, calculer $f(w)$ pour tous les mots de passe probables et les comparer aux hachages dans le fichier.
- Modifier le fichier des mots de passe (problème avec l'OS. . .)
- Écouter sur la ligne pour apprendre le mot de passe (en particulier dans une connexion à distance).

359

Exemple : mots de passe et fonctions à sens unique

- Pour accéder à un ordinateur on demande un nom id et un mot de passe w .
- L'ordinateur mémorise le couple $(id, f(w))$, où f est une fonction à sens unique.
- Pour vérifier l'identité, l'ordinateur obtient (id, w') du démonstrateur et vérifie si $f(w') = f(w)$.

358

Hachage : sommaire

1. L'espace d'hachage doit être suffisamment large pour résister à l'attaque de l'anniversaire (il s'agit d'un espace plus grand que pour le chiffrement symétrique).
2. Si vous pouvez compresser alors vous pouvez hacher, en sécurité!
3. Les fonctions de hachage sûres dans un sens théorique existent, mais elles sont considérées comme trop coûteuses.
4. Les fonctions de hachage pratiques sont dérivées de systèmes de chiffrement symétriques.
5. Du progrès du côté des attaquants.

360

Signatures digitales

361

Idée générale

- Héloïse a une *clef publique* e et une *clef privée* d . Pour *signer* un document m elle calcule $s(d, m)$.
- Abélard reçoit le couple $(m, s(d, m))$. En utilisant la *clef publique* e , il peut vérifier que le message a vraiment été signé par Héloïse.

363

Problème

- Abélard connaît la signature d'Héloïse. S'il reçoit une lettre avec la signature d'Héloïse il est confiant que la lettre a été rédigée par Héloïse.
- On souhaite avoir un mécanisme similaire pour les documents électroniques, par exemple contrats électroniques, transactions bancaires, ...
- On verra que les *signatures digitales* se basent sur la *cryptographie à clef publique* et sur les *fonctions de hachage*.

362

Remarque

Dans ce schéma :

- la *clef privée* est utilisée quand on *envoie* le message,
- et la *clef publique* (de l'émetteur) quand on le *reçoit*.
- C'est juste le *contraire* de ce qui se passe dans le système à clef publique !

NB Dans RSA, les clefs de 'chiffrement' et de 'déchiffrement' ont les *mêmes* propriétés (chaque clef est l'inverse de l'autre). Les noms se réfèrent à *l'utilisation envisagée* et non pas à des propriétés mathématiques particulières.

364

Rappel : système RSA

1. On génère deux grands nombres premiers p, q . Soit $n = pq$ le *module*.
2. On choisit e l'*exposant de chiffrement* tel que $1 < e < \phi(n)$ et $\text{pgcd}(e, \phi(n)) = 1$.
3. On prend d , l'*exposant de déchiffrement*, comme l'inverse de e modulo $\phi(n)$.

Clef Publique = (n, e) Clef Privée = (n, d)

$$c = E_{(n,e)}(m) = m^e \text{ mod } n \quad D_{(d,n)}(c) = c^d \text{ mod } n \quad \text{for } m, c \in \mathbf{Z}_n$$

365

Exercice (signature RSA)

On suppose que Héloïse choisit $p = 11$, $q = 23$ et $e = 3$.

1. Déterminez la clef privée et publique.
2. On suppose qu'Héloïse souhaite retirer 111 Euros d'une machine. Comment Héloïse signe-t-elle sa requête ?
3. Comment la machine vérifie-t-elle la requête d'Héloïse ?

367

Signature RSA

- Les clefs publique et privée d'Héloïse sont déterminées comme dans RSA. Héloïse signe le document m en calculant

$$s = s(d, m) = (m^d) \text{ mod } n$$

- Abélard reçoit (m, s) et vérifie si

$$(s^e \equiv m) \text{ mod } n$$

- Un attaquant, devrait être capable de calculer la racine e modulo n du document m . Actuellement, ceci n'est pas considéré faisable.

366

Attaques sur la signature RSA

- Pourquoi Héloïse devrait-elle envoyer (m, s) ? En principe, Abélard pourrait calculer m à partir de s , car

$$m = (s^e) \text{ mod } n$$

368

- Replaçons Abélard par la machine de l'exemple précédent.
- Un attaquant pourrait envoyer à la machine un élément arbitraire $s \in \mathbf{Z}_n$.
- La machine calcule $m = (s^e) \bmod n$ et croit que Héloïse veut m Euros.
- C'est une forme de *plagiat* : l'attaquant peut produire une signature sans connaître le texte clair qui lui correspond.

369

- Comme dans les systèmes à clef publique, il est important que Abélard obtienne réellement la clef publique d'Héloïse.
- Si, à sa place, Abélard reçoit une clef publique produite par un attaquant, alors l'attaquant peut produire des messages arbitraires et faire croire à Abélard qu'ils ont été signés par Héloïse.

371

- La *propriété multiplicative* de RSA que nous avons déjà rencontré, permet d'envisager un autre type de *plagiat* :
 - On suppose que l'attaquant connaît :

$$s_1 = (m_1^d) \bmod n \quad s_2 = (m_2^d) \bmod n$$

- Alors il peut calculer :

$$(s_1 s_2) \bmod n = (m_1 m_2)^d \bmod n = s(d, m_1 m_2)$$

370

Redondance (rappel)

Comme pour le système à clef publique, une contre-attaque standard contre le plagiat consiste à introduire un peu de redondance dans les messages et à s'assurer que cette redondance (très probablement) n'est pas préservée par la multiplication.

372

Signature avec fonction de hachage

Pour l'instant on a supposé que $m \in \mathbf{Z}_n$.

Question Que fait-on si m est un message de longueur arbitraire ?

Réponse On signe $h(m)$ plutôt que m , où h est une fonction de hachage résistante aux collisions.

373

Bénéfices du hachage

Supposons que la fonction de hachage résiste aux collisions.

- Trouver un message x qui correspond à une signature s , revient à trouver x tel que $h(x) = (s^e) \bmod n$, c'est-à-dire, à déterminer l'image inverse de (s^e) .
- Dans le cas de plagiat qui exploite la propriété multiplicative de RSA, il est improbable que $h(m_1 m_2) = h(m_1) h(m_2)$.
- Enfin, on note que remplacer (x, s) par (x', s) revient à trouver x' tel que $h(x) = h(x')$.

375

Signature RSA avec fonction de hachage

- La signature d'Héloïse du message m est

$$s = (h(m)^d) \bmod n$$

On remarque que maintenant le message m *ne peut pas être dérivé* de la signature.

- Abélard reçoit (x, s) et vérifie :

$$(s^e) \bmod n = h(x)$$

374

Rappel : ElGamal

Clefs Soit p premier, g générateur pour $(\mathbf{Z}_p)^*$, $a \in \mathbf{Z}_{p-1}$,

$$A = g^a \bmod p :$$

Clef Publique (p, g, A) Clef Privée a

Chiffrement Soit $m \in \mathbf{Z}_p$. Alors

$$E_{(p,g,A)}(m) = \nu b \in \mathbf{Z}_{p-1} (B, c) \quad \text{avec } B = (g^b) \bmod p, \quad c = A^b m \bmod p$$

Déchiffrement

$$D_a(B, c) = B^x c \bmod p \quad \text{avec } x = p - 1 - a$$

376

Signature d'ElGamal

On suppose une fonction de hachage $h : \{0, 1\}^* \rightarrow \{1, \dots, p-2\}$

- Pour signer x , Héloïse choisit $k \in \{1, \dots, p-2\}$ premier avec $p-1$, détermine son inverse k^{-1} , et calcule (r, s) où :

$$r = g^k \bmod p \quad s = k^{-1}(h(x) - a \cdot r) \bmod (p-1)$$

377

Expliciter la condition de vérification

On vérifie qu'une signature correcte passe le test :

$$A^r r^s \equiv g^{h(x)} \bmod p$$

On suppose $r = g^k \bmod p$ et $(s \equiv k^{-1}(h(x) - ar)) \bmod (p-1)$. On note $(A^r r^s \equiv g^{ar} g^{k \cdot s}) \bmod p$. Donc le test est équivalent à :

$$(g^{k \cdot s} \equiv g^{(h(x) - ar)}) \bmod p$$

Comme g est un générateur, ceci est équivalent à :

$$(k \cdot s \equiv h(x) - ar) \bmod (p-1)$$

ce qui est vrai par définition de s .

379

- Abélard connaît la clef publique d'Héloïse (p, g, A) . Il reçoit un message x et une signature (r, s) . D'abord, il vérifie

$$1 \leq r \leq (p-1)$$

Si la vérification est positive, il vérifie

$$A^r r^s \equiv g^{h(x)} \bmod p$$

et si ce contrôle est aussi positif il accepte.

378

Exercice (signature d'ElGamal)

On suppose $p = 23$, $g = 7$, $a = 6$.

1. Déterminez la clef publique d'Héloïse.
2. Calculez la signature d'Héloïse pour un document x tel que $h(x) = 7$, en supposant qu'elle choisit $k = 5$.
3. Explicitez les vérifications que Abélard doit effectuer.

380

Pratique de la signature d'ElGamal

- Une méthode générale pour attaquer la signature d'ElGamal est de calculer le logarithme discret modulo p . Actuellement, on recommande que p soit sur 768 bits.
- Pour chaque signature, il faut choisir un k nouveau. Autrement, on peut déterminer la clef privée.
- Si aucune fonction de hachage est utilisée la signature d'ElGamal est aussi sujette à des attaques type plagiat.
- La vérification de la condition $1 \leq r \leq p - 1$ est aussi importante. Autrement, il est possible d'utiliser des vieilles signature pour générer des nouvelles.

381

Protocoles d'Authentification

383

Terminologie : Digital signature algorithm (DSA)

- Il s'agit d'une variante plus efficace de la signature d'ElGamal proposée par une agence des USA.
- Le nombre d'exponentiations modulaires dans la vérification est réduit de 3 à 2.
- Le nombre de bits dans l'exposant est 160 plutôt que les 768 recommandés.
- La méthode est implémentée dans GnuPG.

382

État de l'art

- Chaque fois qu'on définit un protocole, on aimerait savoir que sa sécurité dépend d'un problème fondamental (factorisation, logarithme discret, ...).
- On est encore très loin de cet objectif. Le problème est ouvert même pour des protocoles de base comme Diffie-Hellman ou RSA.
- Il y a aussi des exceptions heureuses : le générateur pseudo aléatoire de Blum-Blum-Shub, la fonction de hachage de Chaum, le système à clef publique de Rabin, ... Ces protocoles ne sont pas forcément utilisés en pratique mais ils établissent un (haut) standard.

384

Attaques 'triviales'

- Plusieurs protocoles qui ont été proposés sont sujets à des *attaques triviales*.
- Par exemple, la réutilisation d'information de sessions précédentes, l'utilisation d'information d'un type non attendu,...
- Ces attaques ne demandent pas de connaissances mathématiques profondes, juste un petit peu de logique. . .
- Dans la suite, nous présentons une liste de ces attaques (triviales mais fréquentes).

385

Terminologie

Authentification de l'entité (Entity authentication) Parfois on néglige le *message* et on s'intéresse juste à son *origine*. Par exemple, on veut être sûr que *A* est actif (un *ping* avec authentification).

Authentification de la Clef (Authenticated key establishment)

Parfois le message est une *clef* qu'on utilise dans des communications qui suivent.

387

Protocole d'authentification : définition informelle

Une procédure par laquelle une *entité* (un participant) *A* envoie un *message* à une autre *entité* (participant) *B* de façon à ce que *B* soit sûr que :

Origine le message vient bien de *A*.

Intégrité des Données le message n'a pas été manipulé.

Nouveauté Le message ne provient pas d'une exécution ancienne du protocole.

386

Remarque : Authentification \neq Confidentialité

La confidentialité consiste à garder une donnée *secrète*.

– Un message peut être *secret* sans être *authentique*.

– Un message peut être *authentique* sans être *secret*.

388

Challenge avec chiffrement symétrique

But B veut être sûr que A est actif.

Hypothèse A, B partagent une clef symétrique K_{AB} .

Exécution standard du protocole

1. $B \rightarrow A : N_B$ (un nom (nonce) nouveau).
2. $A \rightarrow B : \{N_B\}_{K_{AB}}$ ($\{-\}_-$ pour chiffrement).
3. B déchiffre le message et vérifie le nom.

Notation N pour un nonce, $\{m\}_K$ pour le chiffrement du message m avec la clef K .

389

Une attaque basée sur la confusion de type

- Le nom N_B est une séquence de bits ‘aléatoires’.
- Dans un protocole par challenge, A ne va pas vérifier N_B .
- Un attaquant B pourrait envoyer comme nom :

$$N_B = \text{hash}(\text{Transfer my money to } B \text{ account } 663280)$$

- Dans ce cas, A *signerait* un virement sans le savoir (si la *même* clef est utilisée pour l’identification et la signature, ce qui est une mauvaise pratique).

391

Challenge avec clef publique

Hypothèses B connaît la clef publique $Pub(A)$ de A .

Exécution standard du protocole

1. $B \rightarrow A : N_B$ (un nom nouveau).
2. $A \rightarrow B : \{N_B\}_{Priv(A)}$ (on signe le nom).
3. B déchiffre le message et vérifie la signature.

NB Notation $Pub(A)$, $Priv(A)$.

390

Étiquetage (tagging)

- Une contre-attaque contre les attaques basées sur la confusion de type est d’*étiqueter explicitement les messages* avec le type de leur contenu.
- Si A croit qu’il est en train de signer un nom alors il émettra $\{nonce, N_B\}_{Priv(A)}$.
- Quand la banque reçoit la signature, elle vérifie que l’étiquette est bien “signature” (et non pas “nonce”).

Notation Il faut penser à $-, _$ comme une façon de concaténer les messages avec un séparateur pour récupérer chaque message.

392

Nécessité d'un modèle

Pour *décrire* les protocoles et les attaques, il faut préciser :

- comment le protocole exécute,
- ce que les attaquants peuvent faire.

393

Opérations sur les données L'activité de base des participants

est de recevoir, analyser, composer et envoyer des messages. En particulier, ils disposent d'opérations pour :

- Coupler (concaténer) et projeter les messages.
- Chiffrer et déchiffrer (symétrique ou asymétrique).
- Générer des noms et des clefs.
- Hacher.
- Comparer.
- Introduire des étiquettes de temps (time stamps),...

395

Modèle de Dolev-Yao

Rôles Chaque protocole comprend un certain nombre de *rôles* : initiateur, serveur, partenaire,...

Participants Le protocole comprend un certain nombre de *participants* qui peuvent jouer des rôles différents, éventuellement en *même temps*.

Mauvais joueurs Certains participants peuvent ne pas respecter les règles. Par exemple, ils peuvent révéler leur clef secrète à un adversaire.

394

Interaction Tous les messages passent par l'*adversaire*. Un adversaire *actif* peut :

- mémoriser,
- analyser (eavesdropping),
- synthétiser,
- remplacer certains (parts de) messages selon sa connaissance courante.

NB La pire des situations est celle où les adversaires coopèrent, ainsi on considère *un adversaire tout puissant*.

396

Toutes les communications passent par l'adversaire

Si on dit :

A envoie à B un message m

on doit comprendre :

(un participant qui joue le rôle) A envoie à l'adversaire un message m et (un participant qui joue le rôle) B *peut* recevoir de l'adversaire un message m' (éventuellement m).

Référence pour le modèle de Dolev-Yao D. Dolev, A. Yao.
On the security of public key protocols, IEEE Trans. on Information Theory, 29(2), 1993.

397

Exemple

- Le protocole devrait permettre à deux *participants* A et B d'établir une *clef de session*.
- Dans ce but, A , B peuvent utiliser un *tiers de confiance* S .

399

Attaques : rejouer les messages

Un attaquant utilise dans la session courante des messages enregistrés dans des sessions précédentes. Par exemple, la *nouveauté* d'une interaction peut être compromise.

398

- Propriétés souhaitables :
 1. Seulement A et B (et éventuellement S) connaissent la *clef de session*.
 2. A (B) est certain que la clef est partagée avec B (A).
 3. La clef de session est *nouvelle*, c'est-à-dire elle ne vient pas d'une session précédente.

400

- Le protocole doit marcher aussi si :
 1. Il est exécuté *plusieurs fois*.
 2. Par plusieurs *participants*, jouant des *rôles* différents, éventuellement en *parallèle*.
 3. Certains participants sont *malhonnêtes*. Dans ce cas, le protocole doit fonctionner pour autant que A, B, S jouent d'après les règles (sont honnêtes).

NB La méthode standard de modéliser un participant malhonnête est de postuler que ses clefs privées sont connues par l'attaquant.

401

- Le protocole précédent est sujet à une *attaque par re-jeu*.
- Le protocole a été publié en *1978*.
- L'attaque a été découverte en *1981*.
- Une variante de l'attaque a été découverte en *1996*.
- Les protocoles cryptographiques ont une description compacte mais ils sont *difficiles à analyser* à cause de l'interlacement non-déterministe des actions des participants et du comportement arbitraire de l'attaquant.
- Un cas d'étude intéressant pour les *méthode formelles*!

403

Hypothèses Chaque participant A partage avec le serveur S une clef secrète symétrique différente K_{AS} .

Exécution standard du protocole

1. A génère un nom N_A et envoie à S : $\langle A, B, N_A \rangle$.
2. S génère une clef K et envoie à A : $\{N_A, K, B, \{K, A\}_{K_{BS}}\}_{K_{AS}}$.
3. A déchiffre, vérifie le nom N_A et l'identité B et envoie à B : $\langle S, \{K, A\}_{K_{BS}} \rangle$.
4. B déchiffre, vérifie l'identité de A , génère un nom N_B et envoie à A : $\{m, N_B\}_K$, où m est un message secret.
5. A réplique en envoyant à B : $\{m', N_B, N_B\}_K$, où m' est un autre message secret.

402

Exercice

Étant donnée la description du protocole, écrivez explicitement les comportements qui correspondent aux *rôles* de l'*initiateur* (joué par A), du *partenaire* (joué par B) et du *serveur* (joué par S).

404

Attaque : Homme au milieu (Man-in-the-Middle)

Idée générale On peut utiliser certains participants comme des *oracles* qui nous aident à résoudre un problème difficile.

Un exemple mémorable Si on peut jouer en parallèle contre Karpov et Kasparov alors on peut certainement améliorer son propre classement comme joueur d'échecs (soit on gagne un match soit on fait deux matches nuls !)

405

Exécution standard du protocole (rappel)

1. Héloïse choisit une valeur aléatoire $a \in \mathbf{Z}_{p-2}$, elle calcule et envoie à Abélard $A = (g^a) \bmod p$.
2. Abélard choisit une valeur aléatoire $b \in \mathbf{Z}_{p-2}$, il calcule et envoie à Héloïse $B = (g^b) \bmod p$.
3. La *clef commune* $K = g^{ab} \bmod p$ est dérivée :

$$K = B^a \bmod p \quad (\text{comme Héloïse calcule } K)$$

$$K = A^b \bmod p \quad (\text{comme Abélard calcule } K)$$

407

Exemple (Diffie-Hellman)

Hypothèses Héloïse et Abélard se mettent d'accord sur un *grand nombre premier* p et un *générateur* g pour $(\mathbf{Z}_p)^*$ tel que $2 \leq g \leq p - 2$.

But Création d'une clef secrète entre Héloïse et Abélard.

406

Exercice

Décrivez une attaque 'homme au milieu' contre ce protocole.

408

Attaque : Sessions parallèles

Deux (ou plusieurs) sessions du même protocole sont exécutées en parallèle sous le contrôle de l'attaquant. Normalement, une (ou plusieurs) session est avortée mais les messages produits avant l'arrêt sont utilisés dans (au moins) une autre session qui complète son exécution normalement.

409

Exécution standard du protocole

1. A envoie à B : A .
2. B génère un nom N_B et envoie à A : N_B .
3. A réplique à B avec : $\langle A, \{N_B\}_{K_{AS}} \rangle$.
4. B reçoit le texte chiffré et envoie à S : $\{A, \{N_B\}_{K_{AS}}\}_{K_{BS}}$.
5. S déchiffre le message avec K_{BS} et avec K_{AS} et envoie à B : $\{N_B\}_{K_{BS}}$.
6. B déchiffre le message et vérifie le nom.

NB On peut concatener à ces messages de l'information en clair pour en déterminer le but.

411

Exemple : protocole de Woo-Lam

Hypothèses Chaque participant A partage une clef symétrique K_{AS} avec le serveur S .

But A s'authentifie auprès de B (même si A et B ne se sont jamais rencontrés auparavant).

410

Exemple (attaque)

On suppose que B est disponible à parler à A et à un participant (malhonnête) M (qui joue aussi le rôle A) au même temps. On décrit une méthode par laquelle M peut induire B à penser que A est en ligne.

1. M commence deux protocoles avec B : un comme A et un comme M .
2. B produit deux nonces : N_B pour A et N'_B pour M .

412

3. M intercepte le nonce pour A , jette le nonce pour lui même et envoie à B : $\langle A, \{N_B\}_{K_{MS}} \rangle$ and $\langle M, \{N_B\}_{K_{MS}} \rangle$ (il n'y a pas de raison pour que B remarque que les deux textes chiffrés sont le même).
4. B envoie à S : $\{A, \{N_B\}_{K_{MS}}\}_{K_{BS}}$ et $\{M, \{N_B\}_{K_{MS}}\}_{K_{BS}}$.
5. Le serveur ne peut pas déchiffrer le premier message mais il réplique au deuxième par $\{N_B\}_{K_{BS}}$.
6. B obtient le nonce N_B et authentifie A tout en refusant M .

413

Exemple : Woo-Lam après “correction”

Au pas 5, on ajoute l'identité de A .

1. A envoie à B : A .
2. B génère un nom N_B et envoie à A : N_B .
3. A réplique à B avec : $\langle A, \{N_B\}_{K_{AS}} \rangle$.
4. B reçoit le texte chiffré et envoie à S : $\{A, \{N_B\}_{K_{AS}}\}_{K_{BS}}$.
5. S déchiffre le message avec K_{BS} et avec K_{AS} et envoie à B : $\{A, N_B\}_{K_{BS}}$.
6. B déchiffre le message et vérifie le nom.

415

Attaque : Réflexion

Un participant A envoie un message qui est intercepté par l'attaquant et retourné au participant A qui ne réalise pas être à l'origine du message.

414

Exercice

1. Expliquez pourquoi l'attaque avec sessions parallèles échoue pour cette variante du protocole.
2. On suppose qu'un participant malhonnête M , en jouant A , réplique au pas 3 avec le nom N_B généré par B (une réflexion). On suppose que B ne peut pas distinguer entre un nom aléatoire et un texte chiffré. Est-il possible de faire croire à B qu'il est en train de parler avec A ?

416

Un autre exemple : Neuman-Stubblebine

Hypothèses Chaque participant partage une clef symétrique secrète avec un tiers de confiance S . T_A dénote une étiquette de temps du participant A .

But A et B veulent s'authentifier mutuellement et établir une *clef* commune en utilisant un tiers de confiance S .

Exécution standard du protocole

1. A envoie à B : $\langle A, N_A \rangle$.
2. B envoie à S : $\langle B, \{A, N_A, T_B\}_{K_{BS}}, N_B \rangle$.
3. S envoie à A : $\langle \{B, N_A, K_{AB}, T_B\}_{K_{AS}}, \{A, K_{AB}, T_B\}_{K_{BS}}, N_B \rangle$.
4. A envoie à B : $\langle \{A, K_{AB}, T_B\}_{K_{BS}}, \{N_B\}_{K_{AB}} \rangle$.

417

418

Attaque : Erreur de type

Un participant confond le type d'un message. Par exemple il confond un nom avec une clef (voir exemple sur l'identification par challenge et la variante 'corrigée' du protocole de Woo-Lam).

Exercice

On suppose qu'un nom et une clef ont le même format et qu'un participant ne peut pas distinguer entre un nom aléatoire et une clef aléatoire. Y-a-t-il une méthode pour l'attaquant de faire croire à B que le nom N_A est la nouvelle clef de session. Comment ?

419

Attaque : Omission d'identité

Parfois l'identité d'un participant peut être déduite du message ou d'information contextuelle. Par ailleurs, il est souhaitable de minimiser la taille des messages dans un souci d'efficacité et d'élégance. Cependant, cette pratique peut s'avérer dangereuse (voir première version de Woo-Lam).

420

Règles de bon sens

A partir des attaques analysées, on peut tirer un certain nombre de *règles* de bon sens :

1. Expliciter l'*identité* des participants.
2. Insérer des témoins de la *nouveauté* d'une donnée (par rapport à la session).
3. Vérifier le *type* des messages.

421

Exemple : variation sur le protocole de Otway-Rees

Hypothèses Chaque participant A partage une clef secrète symétrique K_{AS} avec un tiers de confiance S . On dénote avec R_{AB} un identificateur de sessions entre A et B .

But A et B veulent s'authentifier mutuellement et établir une clef de session K_{AB} .

423

Attaque : Mauvaise utilisation de fonctionnalités cryptographiques

- Les concepteurs de protocoles appliquent ces principes et utilisent certains schéma de communication (par exemple, challenge-réponse) pour avoir certaines fonctionnalités (par exemple, l'identification).
- Parfois ces schémas sont *mal utilisés*. On considère un exemple dans la suite.

422

Exécution standard du protocole

1. A génère un nom N_A et envoie à B :

$\langle R_{AB}, A, B, \{N_A, R_{AB}, A, B\}_{K_{AS}} \rangle$.

2. B envoie à S :

$\langle \langle R_{AB}, A, B, \{N_A, R_{AB}, A, B\}_{K_{AS}} \rangle, \{N_B\}_{K_{BS}}, \{R_{AB}, A, B\}_{K_{BS}} \rangle$

3. S envoie à B : $\langle R_{AB}, \{N_A, K_{AB}\}_{K_{AS}}, \{N_B, K_{AB}\}_{K_{BS}} \rangle$.

4. B envoie à A : $\langle R_{AB}, \{N_A, K_{AB}\}_{K_{AS}} \rangle$.

424

B pense...

1. La clef K_{AB} vient avec le nom N_B , donc elle doit être nouvelle...
2. ... et sans doute, j'ai parlé avec A car j'ai complètement spécifié les identités A, B et l'identificateur de session.

425

1. B envoie :

$$\langle \langle R_{AB}, A, B, \{N_M, R_{AB}, M, B\}_{K_{MS}} \rangle, \{N_B\}_{K_{BS}}, \{R_{AB}, A, B\}_{K_{BS}} \rangle$$

2. M envoie à S :

$$\langle \langle R_{AB}, M, B, \{N_M, R_{AB}, M, B\}_{K_{MS}} \rangle, \{N_B\}_{K_{BS}}, \{R_{MB}, M, B\}_{K_{BS}} \rangle$$

où $\{R_{MB}, M, B\}_{K_{BS}}$ est un chiffrement d'une vieille connexion entre M et B .

3. S envoie à B : $\langle R_{AB}, \{N_M, K_{MB}\}_{K_{MS}}, \{N_B, K_{MB}\}_{K_{BS}} \rangle$.
4. B envoie à A (intercepté par M) : $\langle R_{AB}, \{N_M, K_{MB}\}_{K_{MS}} \rangle$.

427

Exemple (attaque)

On suppose qu'un participant malhonnête M veut jouer le rôle de A auprès de B . Il commence par envoyer à B

$$\langle R_{AB}, A, B, \{N_M, R_{AB}, M, B\}_{K_{MS}} \rangle$$

ensuite il intercepte la réplique de B .

Comment continuer l'attaque ?

426

Confidentialité et intégrité des données (à nouveau !)

- Le concepteur a protégé la *confidentialité* du nom nouveau N_B .
- Cependant ce qu'il faut est l'*intégrité des données* du message, y compris le nom et l'identité des participants.

428

Sommaire : typologie d'attaques considérées

On a mentionné 7 types d'attaques.

1. Re-jeu de messages.
2. Homme au milieu.
3. Sessions parallèles.
4. Réflexion.
5. Confusion de type.
6. Omission d'identité.
7. Mauvaise utilisation de fonctionnalités cryptographiques.

429

- Ces attaques peuvent être découvertes déjà en analysant une description à *haut niveau* du protocole.
- D'autres attaques demandent une modélisation plus poussée.
- Par exemple, le *temps de réponse* d'un participant à un message peut produire une fuite d'information sur une clef censée rester secrète. On appelle ceci une attaque latérale (*side channel attack*).

430

Exemples réels

SSL/TLS, SSH, IPSEC,... (voir cours Réseaux Sécurisés, période 2).

- Conceptuellement, ces protocoles ont une taille comparable aux protocoles que nous avons considéré.
- Dans le développement de ces protocoles, on a trouvé certaines attaques que nous avons discuté.

431

AVISPA

432

Structure de l'outil

- HLPSL (High-Level Protocol Specification Language).
- Compilateur HLPSL2IF.
- Forme Intermédiaire IF.
- Actuellement 4 méthodes d'analyse :
 1. Vérification à la volée (OFMC).
 2. Satisfaction de contraintes (CL).
 3. Réduction à SAT (SATMC).
 4. Automates d'arbres.

433

Distribution

- Mode XEMACS disponible.
- Possibilité d'utiliser une interface web.
- Installation facile (dans mon expérience).
- Pour exécuter
 - > `./avispa example.hlpsl`
- Pour avoir de l'aide
 - > `./avispa -h`

434

Structure de la spécification

1. Description des rôles.
 - Paramètres.
 - Variables Locales.
 - Transitions.
 - Assertions.
2. Description d'une session.
 - Paramètres.
 - Composition de plusieurs rôles.

435

436

3. Complexité du système analysé.
- Nombre de participants.
 - Connaissance initiale de l'adversaire.
 - Participant compromis.
 - Nombre de sessions.

437

Rôles

Paramètres

```
role alice(  
  A,B      : agent,  
  K        : symmetric_key,  
  Hash     : hash_func,  
  SND,RCV  : channel(dy) )  
played_by A def=
```

439

Un exemple

Hypothèse A, B partagent une clef K .

Objectif A, B vont créer une nouvelle clef.

Description

1. $A \rightarrow B$: $\{Na\}_K$
2. $B \rightarrow A$: $\{Nb\}_K$
3. $A \rightarrow B$: $\{Nb\}_{\{h(Na.Nb)\}}$

438

Variables locales

```
local  
State      : nat,  
Na,Nb     : text,  
K1        : message  
  
init  
State     := 0
```

440

Transitions

transition

```
1. State = 0 /\ RCV(start) =|>
   State':=2 /\ Na' := new()
               /\ SND({Na'}_K)

2. State = 2 /\ RCV({Nb'}_K) =|>
   State':=4 /\ K1' := Hash(Na.Nb')
               /\ SND({Nb'}_K1')
               /\ witness(A,B,bob_alice_nb,Nb')
```

end role

441

Le rôle B

```
role bob(
A,B    : agent,
K      : symmetric_key,
Hash   : hash_func,
SND,RCV : channel(dy))
played_by B def=

local
State  : nat,
Na,Nb  : text,
K1     : message

init
State  := 1
```

443

Notation pour les transitions

- ```
2. State = 2 /\ RCV_SA(A.B.{K'.Na.Ns'}_Ka.X') =|>
 State':=4 /\ SND_BA(A.B.X'.{Na.Ns'}_K')
```
- À gauche de  $= |>$ ,  $K'$  est une variable de filtre à laquelle on affecte une valeur.
  - À droite de  $= |>$ ,  $K'$  dénote la nouvelle valeur de la variable  $K$ .

442

transition

```
1. State = 1 /\ RCV({Na'}_K) =|>
 State':=3 /\ Nb' := new()
 /\ SND({Nb'}_K)
 /\ K1':=Hash(Na'.Nb')
 /\ secret(K1',k1,{A,B})

2. State = 3 /\ RCV({Nb}_K1) =|>
 State':=5 /\ request(B,A,bob_alice_nb,Nb)
```

end role

444

## Session

```
role session(
 A,B : agent,
 K : symmetric_key,
 Hash : hash_func)
def=
local SA,SB,RA,RB : channel(dy)

composition
 alice (A,B,K,Hash,SA,RA)
/\ bob (A,B,K,Hash,SB,RB)

end role
```

445

## ...et objectifs

```
goal
 secrecy_of k1
 authentication_on bob_alice_nb
end goal

environment()
```

447

## Initialisation...

```
role environment ()
def=
const
bob_alice_nb,
k1 : protocol_id,
kab,kai,kib : symmetric_key,
a,b : agent,
h : hash_func

intruder_knowledge = {a,b,h,kai,kib}

composition
 session(a,b,kab,h)
/\ session(a,i,kai,h)
/\ session(i,b,kib,h)
```

446

## L'assertion secret

- ```
.../\ secret(K,k,{A,B})
```
- La valeur de K doit rester secrète sauf pour A et B .
 - Si par exemple, A est un intrus, alors la valeur peut être dévoilée.
 - La constante k sert uniquement à identifier l'assertion.

448

Les assertions *witness* – *request*

```
role alice ...
/\ witness(A,B, ab, Nb)
...
role bob
...
/\ request(B,A,ab,Nb)
```

- Avec *witness* on témoigne du fait que *B* va générer une valeur *Nb* qui sera utilisée dans une communication avec *A*. *ab* sert juste à identifier.
- Avec *request* on demande que la valeur *Nb* calculée par *B* a bien été générée par *B*.
- On utilise *witness* et *request* en couple dans deux rôles différents.
- Le système vérifie que chaque *request* correspond à un *witness*.

449

Sûr ? Pas si sûr...

On ajoute une autre session dans l'initialisation :

```
...
composition
session(a,b,kab,h)
/\ session(a,i,kai,h)
/\ session(i,b,kib,h)
/\ session(b,a,kab,h)
```

451

On fait tourner AVISPA...

```
avispa example.hlpls
et on obtient :
...SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/roberto/avispa-1.0/testsuite/results/new-key.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.07s
  visitedNodes: 105 nodes
  depth: 8 plies
```

450

...et voilà !

```
SUMMARY
  UNSAFE
DETAILS
  ATTACK_FOUND
PROTOCOL
  /home/roberto/avispa-1.0/testsuite/results/new-key.if
GOAL
  authentication_on_bob_alice_nb
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.06s
  visitedNodes: 45 nodes
  depth: 3 plies
```

452

ATTACK TRACE

```
i -> (a,3): start
(a,3) -> i: {Na(1)}_kab
i -> (a,12): {Na(1)}_kab
(a,12) -> i: {Nb(2)}_kab
i -> (a,3): {Nb(2)}_kab
(a,3) -> i: {Nb(2)}_(h(Na(1).Nb(2)))
i -> (a,12): {Nb(2)}_(h(Na(1).Nb(2)))
```

453

Est-ce une attaque ?

– Il faut d’abord comprendre ce qui se passe dans la trace :

```
1 : A → I(B) {Na}_K
1' : I(B) → A {Na}_K
2' : A → I(B) {Nb}_K
2 : I(B) → A {Nb}_K
3 : A → I(B) {Nb}_{h(Na,Nb)}
3' : I(B) → A {Nb}_{h(Na,Nb)}
```

– Dans cette attaque la clef reste secrète mais l’authentification échoue.

455

```
% Reached State:
% state_dummy(i,authentication_on_bob_alice_nb,i,0,17)
% request(a.b.bob_alice_nb.Nb(2),12)
% witness(a.b.bob_alice_nb.Nb(2),i)
% secret(h(Na(1).Nb(2)).k1,set_78)
% contains(b,set_78)
% contains(a,set_78)
% state_bob(a,b,kab,h,5,Na(1),Nb(2),h(Na(1).Nb(2)),set_78,12)
% state_alice(b,a,kab,h,0,dummy_nonce,dummy_nonce,dummy_msg,12)
% state_bob(b,i,kib,h,1,dummy_nonce,dummy_nonce,dummy_msg,set_75,10)
% state_alice(a,i,kai,h,0,dummy_nonce,dummy_nonce,dummy_msg,6)
% state_alice(a,b,kab,h,4,Na(1),Nb(2),h(Na(1).Nb(2)),3)
% state_bob(b,a,kab,h,1,dummy_nonce,dummy_nonce,dummy_msg,set_69,3)
```

454

Erreurs de programmation/modélisation

Erreurs de syntaxe et de typage Le système fournit un diagnostic.

Attaques et typage Certaines attaques peuvent se baser sur une confusion de type. Pour voir apparaître ces erreurs il faut que votre typage ne soit pas trop restrictif. Le type le plus général est `message`.

Spécification non exécutable L’option `-p` permet de vérifier que toutes les transitions sont exécutables. Souvent les problèmes viennent d’une mauvaise utilisation de la notation `X'`.

456

Un autre exemple : Kerberos

```
%% PROTOCOL: Simple Kerberos Variant
%% PURPOSE: Authentication, Authorisation, Key Exchange
%%
%% Kerberos is a distributed authentication service that allows a pro
%% (a client) running on behalf of a principal (a user) to prove its
%% to a verifier (an application server, or just server).
%%
%% MODELER: From Tutorial manual
%% 1. A -> S : (A.B.{Na}_Ka)
%% 2. A <- S : (A.B.{K.Na.Ns}_Ka.{K.Na.Ns}_Kb)
%% 3. A -> B : (A.B.{K.Na.Ns}_Kb.{Na.Ns}_K)
%% 4. A <- B : (A.B.{Ns.Na}_K)
```

457

Rôle A : transitions

```
transition

1. State = 0 /\ RCV_BA(start) =|>
   State':= 2 /\ Na':= new()
   /\ SND_SA(A.B.{Na'}_Ka)

2. State = 2 /\ RCV_SA(A.B.{K'.Na.Ns'}_Ka.X') =|>
   State':=4 /\ SND_BA(A.B.X'.{Na.Ns'}_K')

3. State=4 /\ RCV_BA(A.B.{Ns.Na}_K) =|>
   State':=6
   /\ request(A,B,alice_bob_na,Na)

end role
```

459

Rôle A : paramètres et variables locales

```
role alice (A, S, B : agent,
            Ka : symmetric_key,
            SND_SA, RCV_SA, SND_BA, RCV_BA : channel (dy))
played_by A
def=

local State : nat,
    K : symmetric_key,
    Na,Ns : text,
    X : {symmetric_key.text.text}_symmetric_key

init State := 0
```

458

Rôle S : paramètres et variables locales

```
role server (A, S, B : agent,
            Ka, Kb : symmetric_key,
            SND_AS, RCV_AS : channel (dy))
played_by S
def=

local State : nat,
    Ns,Na : text,
    K : symmetric_key

init State := 1
```

460

Rôle S : transitions

```
transition
1. State = 1 /\
  RCV_AS(A.B.{Na'}_Ka) =|>
  State' := 3 /\ Ns' := new() /\ K' := new()
  /\ SND_AS(A.B.{K'.Na'.Ns'}_Ka.{K'.Na'.Ns'}_Kb)
  /\ secret(K',k,{A,B,S})
end role
```

461

Rôle B : transitions

```
transition
1. State = 5 /\ RCV_AB(A.B.{K'.Na'.Ns'}_Kb.{Na'.Ns'}_K') =|>
  State' := 7 /\ SND_AB(A.B.{Ns'.Na'}_K')
  /\ witness(B,A,alice_bob_na,Na')
end role
```

463

Rôle B : paramètres et variables locales

```
role bob (A, S, B      : agent,
          Kb           : symmetric_key,
          SND_AB, RCV_AB : channel (dy))
played_by B
def=

  local State : nat,
        Ns,Na : text,
        K      : symmetric_key

  init State := 5
```

462

Session

```
role session( A, S, B : agent,
             Ka, Kb  : symmetric_key)
def=

  local SSA, RSA, SBA, RBA, SAS, RAS, SAB, RAB : channel (dy)

  composition

    alice(A, S, B, Ka, SSA, RSA, SBA, RBA)
  /\ server(A, S, B, Ka, Kb, SAS, RAS)
  /\ bob(A,S,B,Kb,SAB,RAB)

end role
```

464

Initialisation

```
role environment() def=  
  
  const a, b, s      : agent,  
  ka, kb, ki       : symmetric_key,  
    alice_bob_na, k : protocol_id  
  intruder_knowledge = {a,b,s,ki}  
  
  composition  
session(a,s,b,ka,kb)  
/\ session(a,s,i,ka,ki)  
/\   session(i,s,b,ki,kb)  
/\   session(i,s,a,ki,ka)  
/\   session(b,s,i,kb,ki)  
end role
```

465

Résultat

```
> ./avispa MYEXAMPLES/kerb-simp.hlpsl  
  
SUMMARY  
  UNSAFE  
DETAILS  
  ATTACK_FOUND  
PROTOCOL  
  /home/roberto/avispa-1.0/testsuite/results/kerb-simp.if  
GOAL  
  authentication_on_alice_bob_na  
BACKEND  
  OFMC  
COMMENTS  
STATISTICS  
  parseTime: 0.00s  
  searchTime: 0.10s  
  visitedNodes: 39 nodes  
  depth: 3 plies
```

467

Objectif

```
goal  
  
  secrecy_of k  
  authentication_on alice_bob_na  
  
end goal  
  
environment()
```

466

```
ATTACK TRACE  
i -> (a,3): start  
(a,3) -> i: a.b.{Na(1)}_ka  
i -> (s,7): a.i.{Na(1)}_ka  
(s,7) -> i: a.i.{K(2).Na(1).Ns(2)}_ka.{K(2).Na(1).Ns(2)}_ki  
i -> (a,3): a.b.{K(2).Na(1).Ns(2)}_ka.x61  
(a,3) -> i: a.b.x61.{Na(1).Ns(2)}_K(2)  
i -> (a,3): a.b.{Ns(2).Na(1)}_K(2)
```

468

Une autre attaque

On peut utiliser les options pour chercher une attaque avec un autre algorithme :

```
./avispa MYEXAMPLES/kerb-simp.hlps1 --cl-atse -short
```

ATTACK TRACE

```
i -> (b,19): start
(b,19) -> i: b.i.{n27(Na)}_kb

i -> (a,7): start
(a,7) -> i: a.i.{n11(Na)}_ka

i -> (a,3): start
(a,3) -> i: a.b.{n1(Na)}_ka

i -> (s,8): a.i.{n1(Na)}_ka
(s,8) -> i: a.i.{n17(K).n1(Na).n17(Ns)}_ka.{n17(K).n1(Na).n17(Ns)}_ki
          & Secret(n17(K),set_99); Add a to set_99; Add i to set_99;
          & Add s to set_99;
```

469

```
i -> (a,3): a.b.{n17(K).n1(Na).n17(Ns)}_ka.X(2)
(a,3) -> i: a.b.X(2).{n1(Na).n17(Ns)}_n17(K)

i -> (a,3): a.b.{n17(Ns).n1(Na)}_n17(K)
(a,3) -> i: ()
          & Request(a,b,alice_bob_na,n1(Na));
```

470

Combien de sessions faut-il pour trouver une attaque ?

– Avec 1 session :

```
composition
session(a,s,b,ka,kb)
end role
```

SUMMARY

SAFE

...

471

– Avec 2 sessions :

```
composition
session(a,s,b,ka,kb)
/\ session(a,s,i,ka,ki)
end role
```

...

ATTACK TRACE

```
i -> (a,3): start
(a,3) -> i: a.b.{Na(1)}_ka
i -> (s,7): a.i.{Na(1)}_ka
(s,7) -> i: a.i.{K(2).Na(1).Ns(2)}_ka.{K(2).Na(1).Ns(2)}_ki
i -> (a,3): a.b.{K(2).Na(1).Ns(2)}_ka.x61
(a,3) -> i: a.b.x61.{Na(1).Ns(2)}_K(2)
i -> (a,3): a.b.{Ns(2).Na(1)}_K(2)
```

472

Interprétation

$$1 : A \rightarrow I(S) \quad A.B.\{Na\}_{Ka}$$

$$1' : I \rightarrow S \quad A.I.\{Na\}_{Ka}$$

$$2' : S \rightarrow I \quad A.I.\{K.Na.Ns\}_{Ka}.\{K.Na.Ns\}_{Ki}$$

$$2 : I(S) \rightarrow A \quad A.B.\{K.Na.Ns\}_{Ka}.X$$

$$3 : A \rightarrow I(B) \quad A.B.X.\{Na.Ns\}_K$$

$$4 : I(B) \rightarrow B \quad A.B.\{Na.Na\}_K$$

Encore une violation de l'authentification.