

Introduction à la cryptographie

Claude Petit

30 janvier 2010

Table des matières

Chapitre 1

Introduction

1.1 Objectif du cours

Ce polycopié est une introduction aux techniques de cryptographie et de stéganographie moderne. Il s'adresse à des étudiants en début de second cycle universitaire (troisième année de licence) et nous avons essayé de garder les exigences mathématiques à un niveau raisonnable. Nous avons volontairement passé sous silence les aspects historiques qui sont pourtant intéressants et renvoyons le lecteur à la bibliographie pour combler ce manque.

Le but d'un cryptosystème est d'assurer la confidentialité d'un message entre son émetteur et son destinataire. Il existe deux grands types d'algorithmes: des algorithmes à clef secrète et des algorithmes à clef publique. Nous allons présenter ces deux types d'algorithmes dans les chapitres qui suivent. Dans un algorithme à clef secrète, l'émetteur et le destinataire partagent une même clef. Si l'un des deux dévoile sa clef, la confidentialité est n'est plus assurée. Il faut en outre convenir de cette clef secrète et se l'échanger à travers un canal sécurisé. Dans un système à clef publique, un tel échange de clefs n'est pas nécessaire. L'émetteur et le destinataire possède chacun une clef différente en deux parties, une partie publique et une partie privée.

Deux articles fondamentaux sont à l'origine de la cryptographie moderne. Le premier a été écrit par Claude Shannon en 1949 "the communication theory of secrecy systems" et pose les bases mathématiques des systèmes de chiffrement en définissant la notion d'information et d'entropie. Le second, "new directions in cryptography" de Whitfield Diffie et Martin Hellman, date de 1977 et expose le principe des systèmes à clef publique.

On attend de la cryptographie qu'elle résolve les problèmes de confidentialité, mais également d'authentification, de non-désavœu et d'intégrité. Chacune de ces quatre exigences doit être satisfaite par un système cryptographique. L'authentification doit permettre au destinataire d'être certain de l'identité de l'émetteur du message. Le non-désavœu doit empêcher l'expéditeur de pouvoir nier par la suite avoir envoyé le message et l'intégrité doit permettre de s'assurer que le message n'a pas été modifié entre son expédition et sa réception.

Le cours s'inspire très fortement de 3 ouvrages: la bible de la cryptographie qu'est le livre de Bruce Schneier "cryptographie appliquée", le livre de Douglas Stinson "cryptographie, théorie et pratique" et un numéro spécial de "pour la science" paru en 2002 et intitulé "l'art du secret". Bien entendu, ce polycopié ne prétend pas remplacer ces ouvrages et ne constitue qu'une toute petite introduction à un domaine très vaste et très mouvant.

1.2 Terminologie

Commençons par exposer la terminologie qui sera utilisée tout au long de cet ouvrage.

1.2.1 Alice, Bernard, Eve et les autres

Un algorithme de chiffrement transforme un message appelé texte en clair en un message chiffré à l'aide de clefs et d'une fonction de chiffrement. Le processus qui permet de recouvrer le texte en clair à partir du texte chiffré s'appelle le déchiffrement. La fonction qui transforme le texte en clair en texte chiffré est la fonction de chiffrement; celle qui effectue l'opération inverse est la fonction de déchiffrement. Ces fonctions dépendent bien sûr de clefs. Nous noterons:

x le texte en clair (quelques fois ce sera m).
y le texte chiffré.
$\mathcal{K}$ l'ensemble des clefs.
E la fonction de chiffrement (pour enciphering).
D la fonction de déchiffrement.

Lorsque ces fonctions dépendent de clefs $k_1, k_2 \in \mathcal{K}$ nous les noterons E_{k_1} et D_{k_2}. Nous avons $y = E_{k_1}(x)$ et $x = D_{k_2}(y)$. E_{k_1} et D_{k_2} sont évidemment des fonctions réciproques l'une de l'autre puisque $D_{k_2}(E_{k_1}(x)) = x$

$$D_{k_2} \circ E_{k_1} = \mathrm{Id}$$

Lorsque $k_1 = k_2$, on parle de protocole symétrique. C'est souvent le cas dans les algorithmes à clef secrète. Dans le cas contraire, on parle de système asymétrique. Un algorithme à clef publique est asymétrique.

t représentera une datation dans les protocoles où le temps intervient et l représentera souvent une durée.

Dans tout protocole, il y aura un expéditeur et un destinataire que nous appelerons Alice et Bernard comme cela est souvent l'usage dans les traités de cryptographie. Il y aura souvent aussi un espion qui tentera d'intercepter le message et que nous appellerons Eve. Martin sera également un espion malveillant qui tentera des attaques actives contre les protocoles. Camille sera l'autorité de certification et représentera une personne en laquelle on peut avoir confiance (une sorte de notaire, en fait) et qui sera chargée de diverses transactions.

1.2.2 Cryptanalyse

La cryptanalyse est le processus qui permet de déterminer le texte en clair à partir du texte chiffré sans posséder la clef. On dit également qu'on attaque le protocole cryptographique. Il existe différents types d'attaques:

- A texte chiffré: si on ne dispose que d'un texte chiffré sans connaître le texte en clair correspondant.
- A texte en clair connu: si l'on possède au moins un texte en clair et le texte chiffré correspondant.
- A texte en clair choisi: si l'on peut en plus envoyer des textes en clair et récupérer les textes chiffrés correspondants.

La taille de la clef est un paramètre très important des algorithmes à clef secrète. L'attaque exhaustive d'un tel système est l'attaque qui teste toutes les clefs secrètes possibles jusqu'à trouver la bonne. Il faut donc que l'espace des clefs $\mathcal{K}$ soit très grand pour qu'une telle attaque ne soit pas réalisable. L'attaque des anniversaires, qui sera présentée plus loin, est aussi une attaque contre le nombre de clefs possibles.

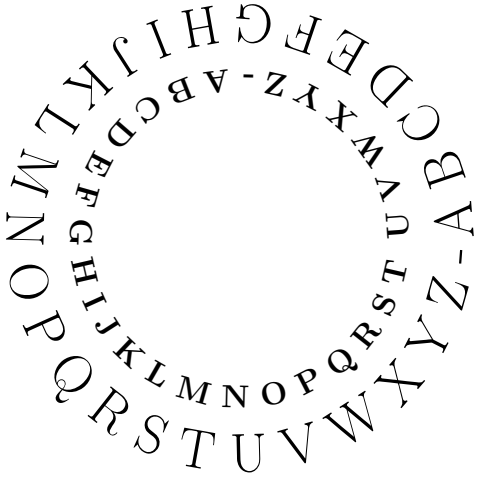
Terminons par un principe de base en cryptographie appelé principe de Kerckhoffs: la sûreté d'un algorithme cryptographique ne doit pas reposer sur le secret de l'algorithme mais sur sa robustesse et sur sa clef. Comme l'histoire l'a montré à de très nombreuses reprises, il est impossible d'espérer qu'un algorithme reste secret; c'est juste une question de temps et d'argent avant qu'il ne soit dévoilé ou volé. Les meilleurs algorithmes sont ceux qui ont été publiés, attaqués par tous les spécialistes et qui ont résisté à ces attaques. Si l'algorithme n'est pas public, personne ne trouvera ses défauts: garder un algorithme secret n'augmente pas sa sécurité, bien au contraire.

Chapitre 2

Systèmes à clefs privées

2.1 Information et entropie

2.1.1 Notion de confidentialité parfaite



Etant donné un espace probabilisé et deux évènements A et B, la Formule de Bayes nous apprend que

$$\mathbb{P}(A/B) = \frac{\mathbb{P}(B/A) \times \mathbb{P}(A)}{\mathbb{P}(B)}$$

Deux évènements sont indépendants si $\mathbb{P}(A/B) = \mathbb{P}(A)$.

Considérons l'ensemble de tous les textes en clair possibles d'une longueur donnée. On choisit une probabilité sur cet ensemble. Par exemple, on peut poser l'équiprobabilité en décidant que n'importe quel texte a la même probabilité d'être choisi. On fait de même avec toutes les clefs $k \in \mathcal{K}$ possibles ($\mathcal{K}$ est l'ensemble de toutes les clefs). Pour un texte en clair x et une clef k, $y = e_k(x)$ est le texte chiffré correspondant. Alors on peut également définir une probabilité sur l'ensemble de tous les textes chiffrés possibles en posant $\mathbb{P}(y) = \sum_{k \in \mathcal{K}} \mathbb{P}(k) \times \mathbb{P}(x)$

La formule de Bayes donne alors la probabilité que le texte en clair soit x sachant que le texte chiffré est y:

$$\mathbb{P}(x/y) = \frac{\mathbb{P}(x) \sum_k \mathbb{P}(k)}{\sum_k \mathbb{P}(k)\mathbb{P}(x)}$$

Cette probabilité permet de définir la notion de confidentialité parfaite.

Définition 1

Un système cryptographique est à confidentialité parfaite si $\mathbb{P}(x/y) = \mathbb{P}(x)$

En d'autre terme, dans un tel système, une information sur le texte chiffré n'apporte aucune information sur le texte en clair. Les deux doivent être indépendants au sens de la probabilité choisie.

On a alors également $\mathbb{P}(y/x) = \mathbb{P}(y) > 0$ de sorte que pour tout texte chiffré, on peut trouver une clef transformant un texte clair x en y. Ceci montre que le nombre de clefs doit être supérieur au nombre de textes chiffrés. En outre, le nombre de textes chiffrés doit être supérieur aux nombre de textes en clair (sinon on ne pourrait pas tous les chiffrer), ce qui montre que, dans un système à confidentialité parfaite, l'ensemble des textes en clair, des textes chiffrés et des clefs possibles ont le même nombre d'éléments. Cette condition est nécessaire mais pas suffisante.

Théorème 1

Sous les hypothèses précédentes, un système est à confidentialité parfaite si et seulement si:
- *chaque clef k est utilisée avec probabilité $1/|\mathcal{K}|$*
- *$\forall x$ et $\forall y$, il existe une unique clef $k \in \mathcal{K}$ telle que $y = e_k(x)$*

$|\mathcal{K}|$ représente le cardinal de $\mathcal{K}$, c'est à dire le nombre total de clefs possibles(et donc aussi le nombre total de textes en clair possibles). Nous ne démontrerons pas ce théorème.

2.1.2 Le chiffrement de Vernam

Cet algorithme, inventé en 1917 par Gilbert Vernam, s'appelle également algorithme du masque jetable. La clef k consiste en une suite de bits de même longueur que le texte en clair x. Le texte chiffré y est simplement la somme modulo 2, bit à bit, de x et k.

$$y = x \oplus k$$

Par ailleurs

$$x = y \oplus k$$

car $k \oplus k = 0$ et donc $y \oplus k = x \oplus k \oplus k = x$

On voit immédiatement que ce protocole vérifie les propriétés du théorème. En fait, ce système est le seul algorithme cryptographique à confidentialité parfaite et le seul qui soit donc rigoureusement incassable, mais à condition, comme l'indique le théorème, que chaque clef ne soit utilisée qu'une unique fois. Si tel n'est pas le cas, la cryptanalyse est possible. Par ailleurs, la clef doit avoir la même longueur que le texte chiffré et ne doit jamais être utilisée deux fois, ce qui provoque de très gros problèmes de gestion de clefs. C'est pour cette raison que ce protocole n'est pas tellement utilisé (il servit durant la guerre froide aux communications utilisant le téléphone rouge entre Washington et Moscou et est peut-être encore utilisé en diplomatie et dans l'armée).

2.1.3 Entropie et information

La notion d'entropie de Shannon donne une définition rigoureuse à la notion vague d'information et permet la mesure du degré d'incertitude ou de désordre. Nous allons commencer par des définitions mathématiques pour revenir ensuite à la cryptographie.

Soit X une variable aléatoire prenant comme valeurs $x_1, ..., x_n$ avec probabilité $p_1, ..., p_n$

Définition 2

l'entropie de X est le réel $H(X) = -\sum_{i=1}^{n} p_i \log p_i$

Par exemple, une variable de Bernoulli prenant deux valeurs 0 et 1 avec probabilité 1/2 a une entropie de $\ln 2$. On peut également définir l'entropie comme le nombre minimum de questions binaires (ie. dont la réponse est oui ou non) qui permette de connaître avec certitude la valeur de la variable aléatoire.

Théorème 2

- $H(X) \geq 0$
- $H(X) = 0 \iff \exists i/\ p_i = 1$
- $H(X) \leq \log n$
- $H(X) = \log n \iff p_i = 1/n\ \forall i = 1...n$

Ces propriétés (admises mais faciles à démontrer) indiquent que l'entropie est nulle lorsqu'il n'y a aucune incertitude et est maximale lorsque la loi est uniforme (en ce cas, toutes les valeurs ont même probabilité d'apparaître).

Considérons maintenant deux variables aléatoires X et Y et la loi conjointe $p_{i,j}$ de (X, Y). L'entropie du couple se calcule de la même façon que ci-dessus en posant $H(X,Y) = -\sum_{i,j} p_{i,j} \ln p_{i,j}$. On peut également définir l'entropie conditionnelle de X par rapport à Y en posant

$$H(X/Y) = -\sum_{X=x;Y=x} \mathbb{P}(x/y) \ln \mathbb{P}(x/y)$$

Théorème 3

- $H(X,Y) \leq H(X) + H(Y)$
- $H(X,Y) = H(X) + H(Y) \iff X, Y$ *indépendantes*
- $H(X/Y) = H(X,Y) - H(Y)$
- $H(X/Y) \leq H(X)$
- $H(X/Y) = H(X) \iff X, Y$ *indépendantes*

Revenons maintenant à la cryptographie. Les deux messages "hier, un crocodile a mordu un homme" et "hier, un homme a mordu un crocodile" sont formés exactement des mêmes lettres et pourtant nous sentons bien qu'ils ne contiennent pas la même quantité d'information. L'entropie d'un langage naturel mesure ainsi la moyenne de l'information fournie par une lettre dans un message qui a un sens. Dans un langage contenant tous les mots possibles avec un alphabet de 26 lettres, l'entropie d'une lettre est $\log_2 26 = 4,7$. Mais dans tout langage, les lettres successives ne sont pas équiprobables: en français, un "q" est souvent suivi d'un "u". De même, il est plus probable de trouver dans un message la phrase "Merci Bernard" que la phrase "&@-zs!!df". Ainsi, si H_n est l'entropie d'un message de longueur n, on définit le taux d'un langage comme étant le réel

$$r = \lim_{n \to +\infty} \frac{H_n}{n}$$

L'anglais a un taux compris entre 1 et 1,5 bit par lettre tandis que le français a un taux de. On définit également la redondance d'un langage possédant un alphabet de n lettres comme étant le réel $D = \log_2 n - r$

La redondance naturelle d'un langage va être utile pour cryptanalyser les messages chiffrés. Elle réduit le nombre de textes possibles à cryptanalyser.

L'entropie d'un cryptosystème est $H = \log_2(|\mathcal{K}|)$ où $|\mathcal{K}|$ est le nombre de clefs possibles du système. La distance d'unicité d'un cryptosystème est le réel

$$U = 2^{H - nD} - 1$$

U représente le minimum probabiliste de texte chiffré tel qu'il n'existe sans doute qu'un seul texte en clair correspondant. Dans le cas d'un système à clef secrète, $U = H/D$. Une clef de 56 bits donne une distance d'unicité de 8,2 lettres.

Nous définissons encore $H(k/y)$ comme étant l'entropie conditionnelle qui mesure l'information sur la clef fournie par le texte chiffré (appelé aussi ambiguïté de la clef).

Théorème 4

$H(k/y) = K(k) + H(x) - H(y)$

Connaissant le texte chiffré, le cryptanalyste cherche à déterminer la clef. Il possède un ensemble de clefs possibles, dont une seule est la bonne. Les autres clefs s'appellent des clefs parasites.

2.2 Le DES

2.2.1 Présentation

Le DES (Data Encryption Standard ou standard de chiffrement de données, en français) a été développé à partir de 1973 par IBM à partir d'un algorithme plus ancien, LUCIFER, datant du début des années 1970. Rendu public en 1975, validé par la NSA (National Security Agency), il est finalement adopté en 1977 par la NIST (bureau des standards américains) comme standard, après de nombreux débats provoqués par la méfiance de la communauté cryptographique à l'égard de la NSA. Sa sécurité est évaluée tous les 5 ans. La dernière évaluation a été faite en 1998 et le DES a été officiellement abandonné en 2000. Il fut durant de nombreuses années le protocole cryptographique le plus utilisé au

monde (il l'est encore pour quelques années, comme nous le verrons dans les exemples de la fin du cours). Les implémentations matérielles du DES sur des puces dédiées effectuent plusieurs dizaines de millions de chiffrement par seconde et traitent plusieurs gigabits par seconde.

2.2.2 Principe

Le DES traite les données par blocs de 64 bits qu'il chiffre en blocs de même taille. L'algorithme de chiffrement et de déchiffrement est le même. La clef secrète k utilisée dans le DES est de 56 bits et toute la sécurité de l'algorithme repose sur cette clef. Le DES consiste en une répétition de 16 opérations de base appelées des rondes, pendant lesquelles les bits du texte en clair sont mélangés à l'aide de la clef secrète. Chaque ronde est définie à l'aide d' une fonction de brouillage f qui dépend de la clef secrète.

Soit $x = (x_1x_2....x_{64})$ le texte en clair. Une permutation initiale P est appliquée à x avant la première ronde, dans le seul but de pouvoir traiter facilement l'algorithme lors de son implantation sur des puces (une permutation finale inverse P^{-1} de la permutation initiale sera également effectuée à la fin de l'algorithme). P et P^{-1} sont décrites ci-dessous; les entiers indiquent les indices des bits du texte chiffré à partir de ceux du texte clair. Ainsi, le vecteur $P(x)$ s'écrit $P(x) = (x_{58}x_{50}x_{42}...x_7)$ à partir des données initiales.

P

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

P^{-1}

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Les données permutées $P(x)$ sont séparées en deux parties de 32 bits, la partie gauche G_0 et la partie droite D_0. On applique alors 16 fois de suite le processus de brouillage. Lors de la ième ronde, les 32 bits de gauche G_i et les 32 bits de droite D_i sont mélangés pour produire en sortie les bits G_{i+1} et D_{i+1} sur lesquels seront appliqués la ronde suivante. A chaque ronde, la partie droite des bits est mélangée par la fonction de brouillage f. Celle-ci va donc dépendre des bits D_i mais également d'une clef k_i qui change à chaque ronde; cette clef est déterminée à partir de la clef secrète initiale k par une opération appelé diversification de la clef ou aussi permutation compressive. La sortie de la fonction de brouillage est additionnée à la partie gauche des bits par un ou exclusif. Le résultat de cette opération devient la nouvelle partie gauche, tandis que l'ancienne partie gauche devient la nouvelle partie droite. On peut résumer tout ceci de façon plus simple:

$$\begin{cases} D_{i+1} = G_i \oplus f(D_i, k_i) \\ G_{i+1} = D_i \end{cases}$$

Examinons dans le détail les différentes opérations effectuées lors d'une ronde:

Diversification de k pour obtenir k_i: la permutation compressive.

Initialement, la clef k fait 64 bits de long, mais 8 de ces bits sont utilisés comme bits de parité (1 bit tous les 8) et seuls 56 bits forment donc réellement la clef. A chaque ronde, on décale les bits de la clef k de façon différente pour générer k_i. Les bits de la clef k sont d'abord ordonnés à l'aide de la permutation ci-dessous

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

puis le résultat est divisé en deux parties de 28 bits et chacune des deux parties est décalée à gauche d'une ou deux positions:

N° de ronde	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Décalage	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Après ce décalage, 48 bits parmi les 56 sont sélectionnés par une permutation compressive (parce qu'elle ne s'applique qu'à une partie des bits) donnée par:

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

On obtient alors une clef k_i de 48 bits de long qui va être utilisée dans le brouillage.

La fonction de brouillage:

La fonction de brouillage f consiste en la succession d'une permutation expansive et d'une série de substitution par des S-boites:

La moitié droite D_i des données, qui est formée de 32 bits, est transformée en un vecteur X_i de 48 bits en répétant et permutant certains bits. Cette permutation expansive est donnée par la table ci-dessous:

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

Ce vecteur X_i est alors ajouté à la clef k_i à l'aide d'un ou exclusif (X_i et k_i font la même taille). Le but de la permutation expansive est de diluer l'information d'un bit d'entrée dans chacun des bits de sortie. Ceci s'appelle l'effet d'avalanche du DES. Soit $Y_i = X_i \oplus k_i$ le résultat de cette opération.

Les 48 bits de Y_i vont être ensuite transformés par passage dans 8 tables appelées des S-boites (S-box pour substitution box). Chaque table prend 6 bits en entrée et restitue 4 bits en sortie. On divise donc les 48 bits de Y_i en 8 paquets de 6 bits qui sont chacun envoyés dans une S-boite différente. Voici ce qui se passe dans une S-boite:

Chaque S-boite est composée de 4 lignes et 16 colonnes qui contiennent un nombre de 4 bits (ie. un entier entre 0 et 15). Soient $b_1, ..., b_6$ les bits d'entrée de la S-boite. On concatène b_1 et b_6 pour former un nombre de 2 bits (ie. un entier entre 0 et 3) qui correspond à un numéro de ligne dans la boite. On concatène également b_2 à b_5 pour former un nombre de 4 bits (ie. un entier entre 0 et 15) qui correspond à un numéro de colonne. A l'intersection de cette ligne et de cette colonne se trouve une valeur sur 4 bits: ce sera la valeur de sortie de la S-boite. On obtient donc en sortie 4 bits pour chacune des 8 S-boites, soit au total 32 bits qui forment un vecteur U_i. Le passage dans les S-boites est la partie cruciale du DES, comme nous le verrons dans les techniques de cryptanalyse. Voici le contenu des 8 S-boites:

S_1

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

S_2

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

S_3

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

S_4

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

S_5

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

S_6

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

S_7

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

S_8

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

Le vecteur U_i est encore modifié par une permutation π pour donner $V_i = \pi(U_i)$

Toutes ses opérations donnent comme résultat $V_i = f(D_i, k_i)$. Finalement, V_i est ajouté à G_i par un ou exclusif et les parties gauches et droites sont permutées. Ainsi se termine une ronde.

Le déchiffrement est identique au chiffrement, mais en inversant l'ordre des clefs de ronde. Par construction, toutes les étapes sont involutives (ie. égales à leur propre réciproque) ce qui permet un déchiffrement facile et rapide.

2.2.3 Variantes du DES

Il existe plusieurs modes opératoires du DES: ECB electronic codebook mode, CBC cipher block chaining mode, CFB cipher feedback block et OFB output feedback block. Dans le mode CBC, le résultat du chiffrement d'un bloc est injecté dans le chiffrement du bloc suivant, ie. $y_i = e_k(y_{i-1} \oplus x_i)$. Dans les modes OFB et CFB, une séquence de clefs engendrée de façon récurrente agit sur le texte en clair à l'aide d'un ou exclusif: $z_i = e_k(z_{i-1})$ et $y_i = x_i \oplus z_i$.

Le triple DES est un surchiffrement de 3 DES successifs à l'aide de deux clefs, ce qui donne un système dont la clef secrète est de 112 bits mais qui est trois fois plus lent que le DES: on chiffre un texte avec une clef k_1, puis on effectue un déchiffrement à l'aide d'une clef k_2 et à nouveau un chiffrement avec une clef k_1. Cet algorithme est encore l'un des plus utilisé dans les transactions bancaires. Etant donné la

taille de sa clef, il est toujours sûr, mais il est trois fois plus lent que le DES .

2.2.4 Critique du DES

La taille initiale des clefs dans LUCIFER était de 128 bits et la NSA l'a ramené à 56 bits dans le DES. Par ailleurs, le contenu des S-boites est resté longtemps suspect. La communauté cryptographique pensait que la NSA avait modifié l'algorithme d'IBM afin de conserver une trappe secrète lui permettant de déchiffrer facilement les messages. La principale critique concernait les S-boites, seule partie non-linéaire de l'algorithme. On dit qu'un algorithme cryptographique est linéaire lorsque la combinaison linéaire de deux entrées donne comme message en sortie la combinaison linéaire des deux sorties. De tels algorithmes sont très faciles à déchiffrer et la linéarité est donc quelque chose à éviter pour obtenir un système sûr. De fait, la partie la plus importante du DES et la seule qui en assure la sécurité est l'opération de brouillage par les S-boites. Certaines personnes pensaient que leur contenu cachait une fonction trappe permettant un déchiffrement rapide par la NSA, mais cela n'a jamais été démontré et reste peu probable. Le comportement des S-boites a beaucoup été étudié mais aucune faiblesse n'a pu être mise à jour. On s'est par contre aperçu à postériori que les S-boites permettaient une protection optimale contre certains types d'attaque. Voici les critères qui auraient guidé la conception des S-boites:

• 6 bits en entrée et 4 en sortie (c'est la taille maximale sur une puce en 1974).
• Chaque colonne d'une S-boite est une permutation de $\{0, 1, ..., 15\}$
• Aucun bit de sortie ne doit s'approcher d'une fonction linéaire des bits d'entrées.
• Un bit modifié en entrée provoque au moins deux bits modifiés en sortie.
• $\forall x$ en entrée, les sorties $S(x)$ et $S(x \oplus 001100)$ diffèrent d'au moins 2 bits.
• $\forall x$ en entrée, $\forall \epsilon, \epsilon' \in \{0, 1\}$, $S(x) \neq S(x \oplus 11\epsilon\epsilon'00)$
• Si un bit est bloqué en entrée et que l'on regarde un bit en sortie, le nombre de valeurs donnant 0 est à peu près égal au nombre de valeurs donnant 1.

Certaines valeurs de la clef k doivent également être évitées. Elles sont connues et s'appellent des clefs faibles. Si l'on utilise ces clefs, le texte en clair peut être trouvé à partir du texte chiffré. Leur liste est connue et il suffit donc de ne pas les utiliser. Il existe aussi des clefs appelées semi-faibles et d'autres appelées potentiellement faibles que l'on n'utilise pas non plus.

Le nombre de rondes est aussi important dans la sécurité de l'algorithme. On a montré qu'après 8 rondes, le texte chiffré est tellement mélangé qu'il se comporte comme une fonction aléatoire des bits de la clef et du texte en clair. Néanmoins, le DES en comporte 16. En 1982, le DES à 4 rondes est cassé, puis en 1986 le DES à 6 rondes.

L'argument le plus valable contre le DES est la longueur de sa clef, déjà considérée comme trop faible au moment de sa conception. Plusieurs cryptographes considéraient qu'une attaque exhaustive sur la valeur de la clef était réalisable. Il en existe au total 2^{56} et au fur et à mesure que les années passaient, la puissance de calcul des ordinateurs rendaient cette attaque de plus en plus vraisemblable. Finalement, en 1998, c'est une telle attaque qui est parvenu a cassé le DES 56 bits en utilisant, durant un peu plus d'un mois, quelques milliers d'ordinateurs individuels à base de pentium via internet. L'année suivante, trois sociétés se sont regroupées pour fabriquer, à un coût raisonnable, un ordinateur dédié à la cryptanalyse du DES. Il a suffit de 56 heures à cette machine pour casser le code. On estime qu'une clef secrète inférieure à 64 bits n'est pas sûre.

2.2.5 Cryptanalyse du DES

Il existe deux autres types de cryptanalyse du DES. Au final, le DES n'a pas été cassé par ces techniques mais par une attaque exhaustive contre le nombre de clefs. Néanmoins, il est intéressant de décrire ces deux techniques.

La cryptanalyse différentielle.

En 1990, Eli Biham et Adi Shamir ont inventé une technique appelée cryptanalyse différentielle afin de casser le DES et d'autres algorithmes similaires. Pour résumer, le principe consiste à comparer les textes chiffrés lorsque l'on introduit dans l'algorithme deux textes en clair ayant une différence fixée. La différence de deux suites de bits x et y est simplement la distance entre les deux, ou encore le ou exclusif $x \oplus y$. Par exemple, on peut partir de deux textes en clair n'ayant qu'un bit d'écart. Si en déplaçant la position de ce bit certains bits du texte chiffré ne changent pas, alors on a détecté une corrélation entre entrée et sortie et l'on peut s'en servir pour deviner une partie de la clef. Ces différences se propagent à travers plusieurs rondes en créant ainsi un chemin qui peut permettre de remonter au texte en clair. On

les appelle des caractéristiques. Cette technique permet de casser un DES à 8 tours en quelques minutes avec un PC récent.

Nous appellerons S_i les différentes S-boites pour $i = 1, ..., 8$. Considérons une deux séquences de bits x et x'. Leur distance est $\Delta x = x \oplus x'$. La technique de la cryptanalyse différentielle est de comparer Δx et $\Delta s = S_i(x) \oplus S_i(x')$, c'est à dire de comprarer la distance de la chaine à la sortie d'une S-boite en fonction de la distance à l'entrée d'une S-boite. Rappelons que x et x' sont des séquences de 6 bits tandis que $S_i(x)$ et $S_i(x')$ sont des séquences de 4 bits.

Considérons tous les couples (x, x') dont la sortie a pour différence un Δs donné. Comme les Δs sont formés de 4 bits, il existe 16 valeurs différentes possibles de Δs. De la même façon, comme les x et x' sont formés de 6 bits, il existe, pour chaque Δs, 64 couples différents donnant ce Δs. On classe alors dans une table le nombre de couples ayant chacune des 16 valeurs possibles de Δs.

Δs	0000	0001	0010	0011	...	1110	1111
N	n_1	n_2	n_3	n_4	...	n_{15}	n_{16}

n_1 représente le nombre de couples (parmi les 64 testés) qui possèdent $\Delta s = 0000$, n_2 représente le nombre de couples (parmi les 64 testés) qui possèdent $\Delta s = 0001$, etc. Le tableau donne la fréquence des sorties en fonction d'une entrée donnée. Il faut se rappeler que les S-boites doivent brouiller les données, et que la sortie d'une S-boite doit être aussi indépendante que possible de l'entrée. Par conséquent, on s'attend à trouver des fréquences à peu près égales à la sortie. La table doit ressembler à une loi uniforme. Si tel n'est pas le cas, c'est que certaines corrélations ont été découvertes. En étudiant la fréquence de ces caractéristiques, on peut deviner un certain nombre de clefs probables qu'il suffit ensuite de tester. Nous décrirons cette méthode de façon plus précise en TP.

On s'est aperçu après la découverte de la cryptanalyse différentielle que les S-boites du DES possédaient la meilleure résistance possible à cette attaque. L'explication vint de Coppersmith, cryptographe à IBM qui participa à la conception du DES: la cryptanalyse différentielle était connue de la NSA avant Biham et Shamir et le DES fut donc conçu pour y résister.

La cryptanalyse linéaire.

Elle a été inventée par Mitsuru Matsui en 1993. Son principe est de chercher des combinaisons linéaires entre les bits d'entrée, les bits de sortie et les bits de la clef. Lorsque de telles combinaisons apparaissent avec une probabilité qui n'est pas égale à 1/2, alors on a détecté une corrélation qui peut-être mise à profit.

La cryptanalyse linéaire nécessite de connaître 2^{43} paires de textes en clair - textes chiffrés. Une attaque fut menée à bien en 50 jours sur 12 stations HP et la clef DES 56 bits correspondante fut découverte.

Contrairement à la cryptanalyse différentielle, le DES n'est pas optimisé contre ce type d'attaque (ou bien parce qu'elle n'était pas connue, ou bien parce que la résistance à d'autres attaques ont été privilégiées).

2.3 L'AES

2.3.1 La genèse de l'AES

La dernière validation du DES devait s'arrêter en 1998. C'est d'ailleurs durant l'été 1998 qu'une association réussit sa cryptanalyse en quelques semaines avec un ordinateur spécialisé dont le coût de fabrication restait à la portée de n'importe quelle grande entreprise. Le NIST a alors lancé un nouvel appel à candidature qui a aboutit en octobre 2000 au choix d'un nouveau standard de chiffrement à clef secrète: l'AES (Advanced Encryption Standard), appelé aussi algorithme Rijndael. Ce sont deux ingénieurs belges, Joan Daemen et Vincent Rijmen, qui l'ont élaboré suivant le cahier des charges imposé par le NIST.

Ce cahier des charges possédaient de nombreuses contraintes: trouver un algorithme plus sûr que le triple DES tout en étant aussi rapide que le DES. La clef secrète devait avoir une longueur de 128, 192 ou 256 bits et la longueur des blocs à chiffrer devait être de 128 bits. Il fallait également que l'algorithme soit facilement portable de telle sorte qu'il soit utilisable aussi bien sur des super-calculateurs que sur une carte à puce. Enfin, le protocole devait être totalement libre de droits. La communauté mondiale des cryptographes a participée à l'évaluation des candidats pour finalement choisir l'AES. Depuis octobre 2000, il remplace donc le DES et le triple DES.

2.3.2 Principe

Plusieurs opérations sont mises en œuvre dans l'AES: brouillage, décalage, transformation non linéaire et addition de clef. Comme pour le DES, un certain nombre de rondes vont répéter ces opérations de bases. On considère que le message initial est découpé en blocs de 128 bits pour former le texte en clair x. Chaque bloc est réparti dans une matrice A de 4 lignes sur 4 colonnes, chaque élément de la matrice étant représenté sur un octet. Puis on applique dix rondes combinant les 4 opérations élémentaires suivantes:

Transformation non linéaire:
La transformation non linéaire f est appliquée à chacun des 16 éléments de la matrice $A = (a_{i,j})$ pour donner une matrice $B = (b_{i,j})$ de telle sorte que $b_{i,j} = f(a_{i,j})\ \forall i, j = 1..4$

Décalage des lignes:
On effectue une permutation circulaire des 3 dernières lignes de B. La 4ième ligne est décalée de 3 cases vers la droite, la 3ième ligne de 2 cases et la 2nde ligne d'une case.

Brouillage des colonnes:
Chaque colonne est multipliée par une matrice M de taille $(4, 4)$ dont les coefficients sont égaux à 1,2 ou 3. Le produit est effectué dans le corps fini $\mathbb{F}_{256}$ (chaque élément de B étant un octet, il peut prendre 256 valeurs possibles et les éléments 1,2,3 de la matrice M sont considérés dans $\mathbb{F}_{256}$). On obtient alors une matrice C.

$$\begin{pmatrix} c_{0,1} \\ c_{1,1} \\ c_{2,1} \\ c_{3,1} \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 2 & 2 \end{pmatrix} \times \begin{pmatrix} b_{0,1} \\ b_{1,1} \\ b_{2,1} \\ b_{3,1} \end{pmatrix} = \begin{pmatrix} 2b_{0,1} + 3b_{1,1} + b_{2,1} + b_{3,1} \\ b_{0,1} + 2b_{1,1} + 3b_{2,1} + b_{3,1} \\ b_{0,1} + b_{1,1} + 2b_{2,1} + 3b_{3,1} \\ 3b_{0,1} + b_{1,1} + 2b_{2,1} + 2b_{3,1} \end{pmatrix}$$

$$D = C \oplus K$$

Ajout de la clef secrète:
Une clef, donnée sous la forme d'une matrice K_i de taille $(4, 4)$ est ajoutée à la matrice C par un ou exclusif sur chaque bit des éléments de la matrice. Cette clef est différente à ronde et est déterminée à partir de la clef secrète K. On détermine K_i à partir de K par un algoritme appelé algorithme de cadencement. La clef de 128, 192 ou 256 bits est représentée sous la forme d'une matrice de 4 lignes sur N_k colonnes où N_k est le nombre de bits de K divisé par 32. $K_0 = K$ et K_i s'obtient à partir de K_{i-1} en permutant les 4 derniers octets de la clef, puis en leur appliquant la fonction S. Après avoir ajouté une constante dépendant de i au premier octet, on ajoute à l'aide d'un ou exclusif les quatre octets obtenus et les quatre premiers de la clef K_{i-1}. Les trois autres blocs de 4 octets de K_i sont les ou exclusif entre le bloc correspondant de la sous-clef K_{i-1} et le bloc précédent de K_i.

Lorsque la clef fait 128 bits, l'AES effectue 10 rondes, lorsqu'elle fait 192 bits, il effectue 12 rondes et lorsqu'elle fait 256 bits, il en effectue 14.

2.3.3 Critique de l'AES

La structure algébrique simple de l'AES pourrait cacher une faille mathématique et est beaucoup étudiée actuellement: c'est à la fois un avantage et un inconvénient. Une attaque sur 6 rondes puis sur 9 rondes ont déjà été menées à bien alors que l'algorithme n'a pas 5 ans d'existence. Ces attaques pourront-elles être améliorées pour casser l'AES ?

Un autre finaliste du nouveau standard gagné par l'AES s'appelle SERPENT. Contrairement à l'AES, il est beaucoup plus sûr car il possède 32 rondes mais est beaucoup plus lent et beaucoup moins portable que l'AES. Le choix d'un algorithme correspond toujours à un compromis entre vitesse, simplicité et sécurité...

2.4 Autres algorithmes par blocs

2.4.1 Schéma de Feistel

Il existe une foultitude d'algorithmes de chiffrement par blocs (à clef secrète) et il n'est pas question d'en donner la liste. On pourra se reporter au livre de Schneier, qui est une véritable encyclopédie du genre. Beaucoup d'algorithmes de chiffrement par blocs utilisent un principe commun appelé schéma de Feistel. Dans son article fondateur, Shannon a exposé les principes de confusion et diffusion des

algorithmes. La confusion sert à cacher les relations entre le texte en clair et le texte chiffré pour éviter la recherche des redondances ou des corrélations par des moyens statistiques, tandis que la diffusion sert à diluer l'information des bits d'entrée dans un maximum de bits de sortie. Une méthode pour s'assurer d'un bon niveau de confusion est de faire correspondre à chaque couple texte en clair - texte chiffré une clef unique (c'est ce qui est fait dans la méthode de Vernam). De cette façon, la fonction de chiffrement ressemble à une fonction totalement aléatoire. Le problème est de créer une telle fonction, mais en utilisant le moins de mémoire et de calculs possibles. Shannon a émis l'idée d'utiliser des algorithmes de chiffrement produit, c'est à dire d'itérer un petit nombre d'opérations simples pour obtenir en sortie un texte chiffré qui semble totalement aléatoire. En d'autres termes, la fonction de chiffrement doit être une bijection pseudo-aléatoire. Dans les années 1970, Feistel a trouvé une méthode itérative simple pour contrsuire de telles fonctions:

Partant d'un message de taille $2n$, on sépare le message en deux parties gauche G et droite D de n bits chacune. On choisit une fonction quelconque qui à n bits associe n bits et effectue une opération itérative entre les parties gauches et droites à chaque tour i en posant:

$$\begin{cases} G_i = D_{i-1} \\ D_i = G_{i-1} \oplus f(D_{i-1}, k_i) \end{cases}$$

La fonction f n'est pas forcément bijective, mais elle doit dépendre de la clef secrète k_i. On s'aperçoit alors que la fonction de transformation définie ci-dessus est bijective et que si l'on veut recouvrer D_{i-1} et G_{i-1} à partir de D_i et G_i; il suffit de remarquer que

$$\begin{cases} D_{i-1} = G_i \\ G_{i-1} = D_i \oplus f(G_i, k_i) \end{cases}$$

Lorsque plus de 4 itérations sont effectuées à partir du texte en clair, le résultat semble totalement aléatoire. Le DES, IDEA ou Lucifer sont des exemples de schéma de Feistel. Par contre, l'AES n'en est pas un.

2.4.2 IDEA

IDEA est un algorithme inventé par Massey et Lai en 1990, puis modifié en 1992 dans sa version finale. Il chiffre des blocs de 64 bits avec une clef secrète de 128 bits. Chaque bloc est découpé en quatre parties de 16 bits qui sont chacunes soumises à 8 rondes successives. Durant une ronde, les blocs sont additionnés par ou exclusif et multipliés entre eux et avec des clefs de ronde. Entre deux rondes, les blocs sont également permutés. L'idée des auteurs étaient de mélanger des opérations de différents groupes algébriques. Nous ne rentrerons pas plus dans les détails.

IDEA est presque aussi rapide que le DES et semble être un algorithme sûr (pour le moment). Sa version finale a été optimisée contre la cryptanalyse différentielle. Il est utilisé dans le logiciel PGP dont nous parlerons plus loin.

IDEA utilise un alphabet de 64 signes: a,...,z,A,...,Z,0,...,9,+/ et le symbole = sert pour le remplissage (appelé padding)

2.4.3 RC2,RC3,RC4,RC5,RC6

RC2 est un algoritme dû à Ron Rivest (le R de RSA) dont la taille de la clef est variable. Il chiffre des blocs de 64 bits et est plus rapide que le DES. Une variante RC3 a été rapidement cryptanalysée. RC5 utilise des blocs de longueur variables et combine des additions, des ou exclusifs et des décalages de bits (appelés rotations).

2.5 Algorithmes de chiffrement en continu

Générateur pseudo-aléatoire, suite aléatoires, registres à décalage, fonction génératrice, résistance aux corrélations, utilisation de la transformée de Fourier, non-linéarité d'une fonction.

Chapitre 3

Sytèmes à clefs publiques

3.1 Rappels d'arithmétique

3.1.1 Divisibilité et nombres premiers

Divisibilité

Sauf mention contraire, tous les nombres considérés ici sont des nombres entiers. Soient $a, b \in \mathbb{N}$

Définition 3

On dit que a divise b et l'on note a/b s'il existe d tel que $b = ad$
On dit alors que b est un multiple de a.

Ex: 3 divise 12 car $12 = 3 \times 4$

La divisibilité est une relation d'équivalence (réflexive, symétrique et transitive) sur $\mathbb{N}$ et est linéaire.

Définition 4

Le plus grand diviseur commun de deux entiers a et b s'appelle le pgcd de a et b
$d = pgcd(a, b) \iff d/a$, d/b et si c/a et c/b alors c/d

Le pgcd est unique.

Lorsque $\text{pgcd}(a, b) = 1$, on dit que a et b sont premiers entre eux (ils n'ont aucun facteur commun).

Ex: $\text{pgcd}(24, 18) = 6$ car $24 = 2 \times 2 \times 2 \times 3$ et $18 = 2 \times 3 \times 3$

Ex: $\text{pgcd}(15, 28) = 1$ car $15 = 3 \times 5$ et $28 = 2 \times 2 \times 7$

Théorème 5 (Identité de Bezout)

Soit $d = pgcd(a, b)$ alors il existe des entiers relatifs $u, v \in \mathbb{Z}$ tels que $d = au + bv$

En particulier, a, b premiers entre eux s'il existe u, v tel que $au + bv = 1$

Ex: $\text{pgcd}(12, 8) = 4$ alors $u = 1$ et $v = -1$ conviennent car $12u + 8v = 14$

Ex: $\text{pgcd}(5, 3) = 1$ alors $u = -1$ et $v = 2$ conviennent.

Algorithme d'Euclide du calcul du pgcd

Nous l'avons déjà vu en programmation. Cet algorithme a plus de 2500 ans. Il figure dans les éléments d'Euclide. Selon les historiens, il n'a pas été inventé par Euclide et serait de 200 plus vieux environ. C'est probablement le plus vieil algorithme du monde.

Connaissant a et b, on cherche à calculer $d = \text{pgcd}(a, b)$
D'après la définition de la division euclidienne, il existe b et q tels que $a = bq + r$ et $r < b$.
Ainsi, si b ne divise pas a, $r \neq 0$ et l'on peut alors poser $r_0 = a, r_1 = b$ et définir une suite récurrente $(r_n)_{n\in\mathbb{N}}$ par:

$r_0 = r_1 q_1 + r_2$ avec $r_2 < r_1$
$r_1 = r_2 q_2 + r_3$ avec $r_3 < r_2$
...
$r_{n-2} = r_{n-1} q_{n-1} + r_n$ avec $r_n < r_{n-1}$
$r_{n-1} = r_n q_n$

La suite $(r_n)_{n\in\mathbb{N}}$ est une suite à valeurs entières strictement décroissante (on peut même montrer que $r_{n+2} < r_n/2$), elle doit donc être nulle à partir d'un certain rang. Le dernier reste non nul est le pgcd recherché. En effet, par construction de l'algorithme, c'est un diviseur de a et b et c'est le plus grand possible.

Ex: Calcul de pgcd(1547,560):
$1547 = 2 \times 560 + 427$, $560 = 1 \times 427 + 133$, $427 = 3 \times 133 + 28$, $133 = 4 \times 28 + 21$, $28 = 1 \times 21 + 7$, et enfin $21 = 3 \times 7$. Le dernier reste non nul est donc $\text{pgcd}(1547, 560) = 7$.

L'algorithme d'Euclide se résume comme suit:

```
entrer a et b
faire
      (a,b) ← (b,a mod b)
      si b=1 sortir
refaire
pgcd = a
```

Algorithme d'Euclide étendu

Il permet en plus le calcul des coefficients u et v de l'Identité de Bezout, le principe restant le même. Voici l'algorithme en langage C:

```
void euclide (int a,int b,int u,int v,int d)
{
  int u1=1,v1=0,d1=a,u2=0,v2=1,d2=b,u3,v3,d3,q;
  while(d2!=0)
  {
     q=d1/d2;
     u3=u1-u2*q;v3=v1-v2*q;d3=d1-d2*q;
     u1=u2;v1=v2;d1=d2;
     u2=u3;v2=v3;d2=d3;
   }
   u=u1;v=v1;d=d1
}
```

Nombres premiers

Définition 5

Un nombre est premier s'il est supérieur à 1 et si ses seuls diviseurs sont 1 et lui-même.

Ex: 2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97, etc.

Théorème 6 (Théorème fondamental de l'arithmétique)

Tout nombre entier se décompose de façon unique en un produit de facteurs premiers.

Pour la suite, nous noterons $p_1, p_2, ..., p_n, ...$ les nombres premiers classés par ordre croissant. Ainsi, $p_1 = 2, p_2 = 3, p_3 = 5, ...$ Si x est un entier, nous noterons la décomposition en question

$$x = \prod_{i=1}^{s} p_i^{n_i} = p_1^{n_1} p_2^{n_2} ... p_s^{n_s}$$

Ex: $12 = 2^2 \times 3^1$ - $4200 = 2^3 \times 3^1 \times 5^2 \times 7^1$
Ex: $175 = 2^0 \times 3^0 \times 5^2 \times 7^1$

Nous introduisons maintenant la fonction indicatrice d'Euler, qui nous sera utile par la suite: Si $n \geq 2$ est un entier naturel, $\Phi(n)$ est égal au nombre d'entiers strictement inférieurs à n et premiers avec n. On pose par ailleurs $\Phi(1) = 1$. Autrement dit:

DÉFINITION 6

$\Phi(n) = Card\{k/\ 1 \leq k < n \text{ et } pgcd(k,n) = 1\}$
$\Phi(1) = 1$

k	1	2	3	4	5	6	7	8	9
$\Phi(k)$	1	1	2	2	4	2	6	4	6

De façon générale, $\Phi(n)$ se calcule à partir de la décomposition en facteurs premiers en utilisant la propriété suivante:

$$n = \prod_{i=1}^{s} p_i^{n_i} \ \Rightarrow\ \Phi(n) = \prod_{i=1}^{s}(p_i^{n_i} - p_i^{n_i-1}) = n \times \prod_{i=1}^{s}(1 - \frac{1}{p_i})$$

Ex: $\Phi(8) = \Phi(2^3) = 2^3 - 2^2 = 4$
Ex: $\Phi(9) = \Phi(3^2) = 3^2 - 3^1 = 6$
Ex: $\Phi(56) = \Phi(2^3 \times 7^1) = (2^3 - 2^2) \times (7^1 - 7^0) = 24$

• Si p est un nombre premier, $\Phi(p) = p - 1$
• Φ est une fonction multiplicative. C'est à dire que:

$$\boxed{\text{pgcd}(m,n) = 1 \Rightarrow \Phi(m \times n) = \Phi(m) \times \Phi(n)}$$

Ex: $\Phi(77) = \Phi(7)\Phi(11) = 6 \times 10 = 60$
Ex: $\Phi(120) = \Phi(2^3 \times 3 \times 5) = \Phi(2^3) \times \Phi(3) \times \Phi(5) = 4 \times 2 \times 4 = 32$

3.1.2 Congruences et arithmétique modulo un entier n.

Le symbole modulo.

DÉFINITION 7

Etant donné deux entiers $a, b \in \mathbb{Z}$ et étant donné $n \in \mathbb{N}$,
on dit que a est congru à b modulo n et l'on note $a \equiv b \mod n$ ssi n divise $b - a$

Ainsi, $\boxed{a \equiv b \mod n \iff n/(b-a) \iff \exists\ k \in \mathbb{N}/\ a = b + k \times n}$

En particulier, $a \equiv 0 \mod n$ ssi a est un multiple de n
b est appelé résidu de a modulo n

Ex: $19 \equiv 7 \mod 12$ car $19 = 7 + 1 \times 12$
Ex: $36 \equiv 0 \mod 12$ car $36 = 3 \times 12$
Ex: $4 \equiv 14 \mod 5$ car $14 = 4 + 2 \times 5$

Si q et r sont respectivement le quotient et le reste dans la division euclidienne de a par n, c'est à dire si $a = nq + r$, alors:

$$\boxed{r \equiv a \mod n}$$

Finalement, travailler modulo n revient à ne travailler qu'avec les restes dans la division euclidienne par n.

Remarques.

$\equiv$ est une relation d'équivalence sur les entiers et est aussi linéaire.
En outre, cette relation respecte le produit, ie:

$a \equiv b \mod n$ et $a' \equiv b' \mod n \Rightarrow aa' \equiv bb' \mod n$

L'arithmétique modulaire revient à travailler sur un cercle contenant n points régulièrement répartis: Lorsque l'on fait un tour complet, on repart de 0. La lecture de l'heure sur un cadran de montre revient à effectuer des calculs modulo 12. Vous utilisez déjà ce type d'arithmétique lorsque vous comptez en binaire ou en hexadécimal. Le principe est le même, à ceci près qu'en arithmétique modulaire, il n'y a pas de retenue.

Structure de $\mathbb{Z}/n\mathbb{Z}$

$\mathbb{Z}/n\mathbb{Z}$ est l'ensemble de tous les entiers modulo n. On a $\mathbb{Z}/n\mathbb{Z} = \{0, 1, ..., n-1\}$. On munit cet ensemble de l'addition et de la multiplication modulo n. Alors $(\mathbb{Z}/n\mathbb{Z}, +, \times)$ possède une structure d'anneau

commutatif.

Ceci veut dire que $(\mathbb{Z}/n\mathbb{Z}, +)$ est un groupe commutatif et que la multiplication est distributive par rapport à l'addition.

$\times$ fonctionne comme dans l'ensemble des réels, à ceci près que certains entiers modulo n n'ont pas d'inverse pour la multiplication. Pour le reste, les règles de calcul découlent de celles de $\mathbb{R}$ et de la linéarité de $\equiv$.

En particulier, lors d'un calcul, la valeur finale obtenue sera la même que l'on applique la réduction modulo n à chaque étape intermédiaire ou uniquement au résultat final. En cryptographie, on fait un usage intensif de l'arithmétique modulaire, ce qui permet de restreindre la taille de tous les résultats intermédiaires.

Quelques exemples de calculs:

$(9 + 11) \mod 7 = 20 \mod 7 = 6 \mod 7$
ou bien $(9 + 11) \mod 7 = (2 + 4) \mod 7 = 6 \mod 7$

$(9 \times 11) \mod 7 = 99 \mod 7 = (14 \times 7 + 1) \mod 7 = 1 \mod 7$
ou bien $(9 \times 11) \mod 7 = (2 \times 4) \mod 7 = 1 \mod 7$

Elements inversibles de $\mathbb{Z}/n\mathbb{Z}$

Lorsque n est un nombre premier, nous admettrons que $(\mathbb{Z}/n\mathbb{Z}, +, \times)$ est un corps commutatif, c'est à dire que dans ce cas, tous les éléments non nuls sont inversibles.

Définition 8

Un élément a est inversible modulo n s'il existe b tel que $ab \equiv 1 \mod n$

Ex: Dans $\mathbb{Z}/5\mathbb{Z}$, 3 admet 2 comme inverse car $3 \times 2 = 6 \equiv 1 \mod 5$. Dans $\mathbb{Z}/4\mathbb{Z}$, 2 n'a pas d'inverse.

Théorème 7

a est inversible modulo n ssi $pgcd(a, n) = 1$

En particulier, si n est un nombre premier, tout élément admet un inverse.

Théorème 8 (Petit théorème de Fermat)

Si p est un nombre premier et a non multiple de p,
$a^{p-1} = 1 \mod p$

C'est à dire que l'élévation à la puissance p dans $\mathbb{Z}/p\mathbb{Z}$ est la fonction identité.

Ex: $2^4 \equiv 1 \mod 5$

Ce théorème se généralise de la façon suivante:

Théorème 9 (Théorème de Fermat-Euler)

Si $pgcd(a, n) = 1$, on a $a^{\Phi(n)} \equiv 1 \mod n$

On peut alors exprimer l'inverse de a modulo n car $a \times a^{\Phi(n)-1} \equiv 1 \mod n$

Une autre méthode consiste à utiliser l'algorithme d'Euclide étendu et est plus rapide en général. En effet, si $\text{pgcd}(a, n) = 1$, l'algorithme d'Euclide vous donne u et v tels que $au + nv = 1$
Modulo n, cette équation devient:

$au \equiv 1 - nv \Rightarrow au \equiv 1 \mod n$

Et donc u est inverse de a modulo n.

Exponentiation modulaire et logarithmes discrets.

On peut calculer très efficacement les puissances d'un entier modulo n.

x	1	2	3	4	5	6
3^x	3	9	27	81	243	729
$3^x \mod 7$	3	2	6	4	5	1

$$3^{1234567890} \mod 7826348737 = 7590405247$$

Ainsi, le logarithme discret (en base 3) de 7590405247 est 1234567890.

La fonction à sens unique est la fonction d'exponentiation modulaire $f(x) = a^x \mod p$ Sa réciproque est la fonction $f^{-1}(y) = x$

3.1.3 Ecriture de nombres dans différentes bases

Les mots sont des nombres ! Pour convertir des données alphabétiques en ombre, on utilise la table des codes ascii qui permet de transformer chaque lettre, signe ou chiffre en un nombre compris entre 0 et 127 (ou 255 si l'on utilise la table des codes ascii étendus). On peut donc considérer qu'un mot est un nombre exprimé en base 128, chaque chiffre de ce nombre pouvant prendre 128 valeurs différentes. Si les données sont binaires, il suffit de les convertir en base 10.

Un nombre n exprimé en base N s'écrit sous la forme $d_k d_{k-1} ... d_0$ si et seulement si $n = \sum_{i=0}^{k} d_i N^i$

Ex: En décimal, $N = 10$, les digits sont $0, 1, ..., 9$ et $n = 1823 \iff n = 3 + 2 \times 10 + 8 \times 10^2 + 3 \times 10^3$
Ex: En binaire, $N = 2$, les digits sont $0, 1$ et $n = 1011 \iff n = 1 + 1 \times 2 + 0 \times 2^2 + 1 \times 2^3$
Ex: En base 128, $N = 128$, les digits sont les codes ascii et
$n = BONJOUR \iff n = 82 + 85 \times 128 + 79 \times 128^2 + ... + 66 \times 128^6$

Le nombre de chiffres d'un entier n exprimé en base N est $k = \lfloor \frac{\ln n}{\ln N} \rfloor + 1$ et k est alors l'unique entier vérifiant $N^{k-1} \leq n < N^k$. Le ième digit en base N d'un nombre décimal n est par ailleurs donné par

$$d_i = \lfloor \frac{n}{N^{i-1}} \rfloor \mod N$$

Le nombre de bits occupés en mémoire par un entier n n'est autre que le nombre de digits en base 2, ie $\lfloor \log_2 n \rfloor + 1$

3.2 Fonction à sens unique

Le concept de cryptographie à clef publique a été inventé par Diffie, Hellman et Merkle en 1976. Dans un tel système, chaque utilisateur possède une clef en deux parties: une partie publique et une partie privée. La clef publique e sert habituellement à chiffrer le message et la clef privée d sert à le déchiffrer. Lorsque Alice veut envoyer un message à Bernard, elle récupère sa clef publique dans un annuaire, chiffre son message à l'aide de cette clef et envoie le tout à Bernard. Celui-ci étant le seul à posséder sa clef privée, il est aussi le seul à pouvoir déchiffrer le message.

La fonction de chiffrement (qui dépend donc de la partie publique de la clef) doit être une fonction à sens unique avec trappe, c'est à dire une bijection facile à calculer dont la réciproque (qui est la fonction de déchiffrement) doit être très difficile à calculée pour toute autre personne que le destinataire (qui possède une trappe cachée permettant un calcul facile). C'est donc une bijection dont on peut calculer facilement la réciproque dans un sens, mais pas dans l'autre.

On peut visualiser ce processus en imaginant que le message qu'Alice envoie à Bernard est placé dans sa boite aux lettres. Tout le monde peut envoyer un message (la fente de la boîte aux lettres dans laquelle on dépose le courrier est la partie publique de la clef) mais seul Bernard est en mesure de les lire (car la clef de la boite aux lettres est la partie privée). Une autre façon de voir ce concept est le suivant: Alice place un message dans une boite qu'elle ferme à clef avec un cadenas. Elle envoie la boite à Bernard qui ajoute un second cadenas et renvoie ensuite le tout à Alice. Alice enlève alors son cadenas et renvoie la boite (qu'elle ne peut plus ouvrir) à Bernard. Celui-ci n'a plus qu'à enlever son cadenas pour lire le message: il n'y a eu aucun échange de clef !

3.3 RSA

3.3.1 Principe

Le système RSA est le protocole de chiffrement à clef publique le plus célèbre et le plus utilisé au monde (il est présent dans près de 400 millions de logiciels). Il a été inventé au MIT en 1977 par Ron Rivest, Adi Shamir et Leonard Adleman et permet de mettre en oeuvre avec une méthode simple et efficace le concept de Diffie et Hellman. Dans le système RSA, les clefs publiques et privées sont calculées à l'aide

de très grands nombres premiers et la fonction à sens unique repose sur la difficulté à factoriser les grands nombres premiers. Dans RSA, les clefs se calculent de la façon suivante:

1°. Chaque utilisateur choisit deux grands nombres premiers p et q.
2°. Il calcule $n = p \times q$ ainsi que le nombre $\phi(n) = (p-1) \times (q-1)$
3°. Il choisit un nombre e inversible modulo $\phi(n)$, c'est à dire tel que pgcd(e,$\phi(n)$)=1. N'importe quel nombre premier convient donc popur e
4°. Il calcule l'inverse d de e modulo $\phi(n)$, c'est à dire $d = e^{-1} \mod \phi(n)$
5°. Il détruit p et q qui ne serviront plus.

La clef publique de l'utilisateur est alors formée par le couple (n, e) et sa clef privée par le couple (n, d). Il peut mettre dans un annuaire la partie publique de la clef mais doit conserver secret le nombre d qui est la partie privée de sa clef.

La première opération consiste à transformer le message (qui peut être un courriel, une suite de mots ou une suite de bits) en un nombre entier. Cela se fait simplement en considérant qu'un message alphabétique est en fait un nombre en base 128 (chaque chiffre peut prendre les 128 valeurs différentes du code ascii) et un message binaire peut s'écrire simplement en base 10 (cf paragraphe précédent). L'entier x obtenu doit être plus petit que n afin de n'avoir qu'une seule représentation modulo n. Si tel n'est pas le cas, il faut découper le message en plusieurs blocs de longueur k de telle façon que $128^{\,k} < n < 128^{\,k+1}$

Pour chiffrer le message x, l'utilisateur calcule $\boxed{y = x^e \mod n}$ et pour le déchiffrer, il calcule $\boxed{x = y^d \mod n}$

La fonction de chiffrement est donc la fonction d'élévation à la puissance e modulo n et la fonction de déchiffrement est la fonction d'élévation à la puissance d modulo n. Puisque e et d sont inverses modulo $\phi(n)$, le théorème de Fermat-Euler assure que ces deux fonctions sont bien réciproques l'une de l'autre.

$d = e^{-1} \mod \phi(n) \iff ed = 1 \mod \phi(n)$

$ed = 1 + k \times \phi(n)$

$$\begin{aligned} m^{ed} &\equiv m^{1+k\times\phi(n)} \\ &\equiv m \times m^{k\times\phi(n)} \\ &\equiv m \times \left(m^k\right)^{\phi(n)} \\ &\equiv m \end{aligned}$$

un exemple simple

$p = 11047$

$q = 19501$

$n = p \times q = 215427547$

$\phi(n) = (p-1) \times (q-1) = 215397000$

$e = 65537$

$d = 65537^{-1} \mod 215397000$

$d = 160194473$

$x = 78 + 79 \times 128 + 66 \times 128^2$

$x = 1091534$

$y = x^e \mod n$

$y = 1091534^{6}5537 \mod 215427547$

$y = 109466891$

$y^d \mod n = 1091534$

3.3.2 Cryptanalyse et factorisation de nombres premiers

Factorisation d'entiers

La sécurité de l'algorithme repose sur la difficulté à factoriser de grands entiers. La première attaque contre ce protocole est donc de tenter de factoriser n. Si l'on parvient à calculer p et q, alors on peut calculer $\phi(n)$ et connaissant ce nombre ainsi que e, on peut facilement en déduire d. D'où l'importance de surveiller la capacité des ordinateurs à factoriser les nombres. Actuellement (en 2005) le record de factorisation d'entiers est de de 663 bits (il date du 28 mai 2005) et montre que $p \times q = n$. L'équipe qui a effectuée cette factorisation avait également factorisé en 2003 une valeur de n de 576 bits. Le calul des facteurs du nombre de 663 bits (200 chiffres décimaux) a nécessité plusieurs mois de calcul. Une clef de longueur inférieure pourrait donc être cassée par un gouvernement. Il existe beaucoup de méthodes de factorisations de nombres premiers. Toutes sont, à l'heure actuelle, de complexité exponentielle. La plus efficace s'appelle le crible sur corps numérique et est capable de factoriser un entier de n bits avec une complexité

$$\exp\left((\alpha + o(1))\sqrt[3]{n}(\log n)^{2/3}\right)$$

où α est une constante comprise entre 0 et 2. Nous ne rentrerons pas dans le détail des méthodes de factorisation, mais il faut connaître des ordres de grandeurs de clefs publiques qui soient sûres. Les cartes bleues de l'ancienne génération possédaient des clefs RSA de 320 bits, ce que l'on sait factoriser facilement aujourd'hui. Les cartes actuelles utilisent des clefs de 768 ou 1024 bits (231 chiffres décimaux).

La puissance de calcul des ordinateurs peut s'évaluer en Mips (million d'opérations par seconde). Un Mips/s représente environ 3×10^{13} instructions. 50 Mips est la puissance de calcul d'un pentium 100 et 50000 Mips est la puissance de calcul de l'Intel Paragon (calculateur vectoriel formé de 1800 processeurs). Une grande entreprise possède une puissance d'environ 10^7 Mips/an. Un gouvernement environ 10^9 Mips/an. Le tableau ci-dessous donne la puissance de calcul (approximative) nécessaire à la factorisation d'une clef RSA:

Nb bits	Mips/an
512	30000
768	2×10^8
1024	3×10^{11}
2048	3×10^{20}

Le record à battre actuellement est de ... La société RSA offre pour cela.

Le premier article concernant RSA a été publié en aout 1977 dans la revue pour la science. Dans le début de l'article, les auteurs donnait un nombre n

$$p = 289347938749283749823749843 31$$

$$q = 109038493824092387983749872399$$

$n = 31550063432327638712243736279 5$
$9492603597202806590874380069$
$n = 1143816257578888676692357799 76$
$1466120102182967212423625625618429 3$
$5706935245733897830597123563958705 0$
$5898907514759929002687954354 1$

$p = 3490529510847650949147849619 90$
$3898133417764638493387843990820577$

$q = 3276913299326670954996198819 08$
$3446141317764296799294253979828853 3$

Différentes attaques contre RSA

Il existe beaucoup d'autres attaques possibles contre le RSA, nous ne présenterons rapidement que certaines d'entres elles. Aucune ne remet en cause le système RSA, mais elles mettent en lumière la nécessité de faire attention au choix des paramètres p, q, d ou e ainsi qu'à la mise en oeuvre de l'algorithme.

• Attaque à texte chiffré choisi:

Eve écoute la ligne de communication d'Alice. Elle a récupéré le texte chiffré c et voudrait déterminer le texte en clair m. Elle choisit un nombre aléatoire r tel que r soit inférieur à n. Elle récupère la clef publique d'Alice et calcule:
$x = r^e \mod n$
$y = xc \mod n$
$t = r^{-1} \mod n$.
Puisque $x = r^e \mod n$ alors $r = x^d \mod n$ donc $t = x^{-d} \mod n$. Maintenant Eve fait signer à Alice la valeur y avec sa clef privée, ce qui déchiffre y car alors Alice renvoie à Eve $y^d \mod n$. Il reste à Eve à calculer :
$x^{-d}y^d \mod n = x^{-d}x^d c^d \mod n = m \mod n$.
Autrement dit, Eve a découvert le texte en clair !

• Attaque par petit exposant privé ou public:

L'attaque par petit exposant privé concerne les valeurs de d trop petites. Pour obtenir un gain de place et accélérer les calculs d'exponentiation, on peut vouloir choisir une valeur de d très petite. Malheureusement, cela rend le système vulnérable. En effet, is $q < p < 2q$ et $d < \frac{1}{3}n^{1/4}$, alors on peut retrouver facilement d. La technique consiste à utiliser des approximations par fractions continues, en remarquant que

$$|\frac{e}{\phi(n)} - \frac{k}{d}| = \frac{1}{d\phi(n)}$$

k/d est une approximation de $e/\phi(n)$ et l'on peut connaître une valeur approchée de $\phi(n)$ à partir de n. Il est alors possible de trouver une valeur approchée de k/d et d'en déduire d. On conviendra donc d'utiliser des valeurs de d assez grandes (plus d'un quart de la taille de n).

Il existe aussi une attaque pour un exposant public e trop petit. Par exemple si $e = 3$, il peut arriver que m^3 soit inférieur à n, auquel cas le déchiffrement revient à extraire une racine cubique, ce qui est très facile à effectuer: ll est recommandé de n'utiliser que des valeurs au moins égales à 65537. C'est d'ailleurs la valeur par défaut choisie en règle générale.

• Attaque par module commun:

Pour simplifier le protocole, on pourrait imaginer d'utiliser la même valeur de n pour tout le monde. Ce serait une erreur: Si c_1 et c_2 sont deux messages chiffrés à partir de m_1 et m_2 en utilisant la même valeur de n, alors on a $c_1 = m^{e_1} \mod n$ et $c_2 = m^{e_2} \mod n$. D'après le théorème chinois, on peut alors trouver deux entiers r et s tels que $re_1 + se_2 = 1 \mod n$. Alors $(c_1^{-1})^{-r}c_2^s \mod n = m \mod n$ et l'on a recouvrer le message en clair n. Ainsi, il ne faut jamais partager n.

• Attaque par bits connus:

En 1998, D. Boneh a démontré que si l'on connaissait le dernier quart des bits de d alors on pouvait reconstituer facilement la valeur complète de d. L'attaque a été étendue à d'autres situations qui montrent que si l'on connait certains bits de la clef secrète, le système n'est plus sûr.

• Attaque de Coppersmith:

En 1996, Coppersmith a démontré que si deux messages m_1 et m_2 étaient liés par une relation du type $m_2 = P(m_1)$ où P est un polynôme, alors on pouvait recouvrer ces deux messages à partir des textes chiffrés c_1 et c_2. D'autres types d'attaques utilisent le fait que la fonction de chiffrement est multiplicative (car $f(xy) = f(x)f(y)$) ce qui permet de chiffrer le produit de deux messages sans connaître le texte en clair de l'un des deux.

• Attaques physiques:

Il existe également des attaques sur les implémentations du RSA: le temps de calcul ou la consommation électrique d'une implémentation matérielle de l'algorithme (par exemple sur les cartes à puces) permet de deviner certains bits de la clef secrète et d'en déduire la totalité. Pour parer à ce type de problème, on peut masquer les temps de calcul en ajoutant une durée aléatoire à un calcul donné.

3.3.3 Implémentation

Les calculs à effectuer pour chiffrer et déchiffrer des messages par RSA sont essentiellement des exponentiations modulaires, dont on connaît des algorithmes rapides. Néanmoins, le RSA est environ 1000 fois plus lent que le DES. Il existe des processeurs dédiés à ce type de calcul, y compris sur des cartes à puces.

3.3.4 Dernier mot sur RSA

Au début des années 1970, James Ellis, Clifford Cocks et Malcolm Williamson, trois britanniques travaillant pour les services secrets anglais, avaient déjà découvert la méthode RSA. Mais celle-ci étant classifiée, elle ne pouvait être rendue publique. En 1997, vingt ans après l'article présentant RSA, on autorisa Cocks à dévoiler publiquement sa découverte. Après une rencontre avec Rivest, Shamir et Adleman, il fut décidé que les six personnes partageraient la paternité de RSA. Ellis étant mort un mois plus tôt, il ne connut jamais cette reconnaissance tardive.

3.4 Autres protocoles à clefs publiques

3.4.1 El Gamal

Au lieu de considérer l'exponentiation modulaire, on peut utiliser le calcul du logarithme discret d'un entier: étant donnés deux nombres a et b, on cherche à déterminer le nombre x tel que

$$b = a^x \mod n$$

x représente le logarithme discret à base a de b modulo n. Le calcul d'un logarithme discret est un problème dont la difficulté est comparable à la factorisation des entiers. En ce cas, la fonction de chiffrement est l'exponentiation modulo n et la fonction de déchiffrement est la recherche du logarithme discret, qui est facile lorsque l'on connait a mais difficile si on ne le connait pas (c'est la trappe de la fonction).

El-Gamal a utilisé cette technique comme système cryptographique et Diffie et Hellman l'ont utilisé dans un protocole d'échange de clefs. Neal Koblitz a découvert une variante de ce protocole en utilisant des courbes elliptiques qui sont des objets mathématiques très intéressants qui ont beaucoup d'applications dans différents domaines (décomposition en facteurs premiers, cryptographie, codes correcteurs, ...). Ces courbes elliptiques permettent d'assurer (à priori) une très bonne sécurité pour des tailles de clefs publiques plus petites que celles utilisées dans RSA (une clef de 170 bits d'un système à base de courbe elliptique est équivalent à une clef RSA de 1024 bits). A l'heure actuelle, le record d'extraction d'un logarithme discret est d'un peu plus d'une centaine de bits.

Pour mettre en oeuvre la méthode d'El-Gamal, on opère comme suit:

- On veut transmettre le texte en clair m.
- On choisit un nombre premier p et deux nombres aléatoires g et $x < p$.
- On calcule $y = g^x \mod p$.
- On choisit un nombre k premier avec $p - 1$.
- On calcule le nombre $a = g^k \mod p$ et $b = y^k m \mod p$
- Le texte chiffré est formé par le couple (a, b)
- La clef publique est (p, g, y) et la clef privée est x
- Pour déchiffrer, on calcule $m = ba^{-x} \mod p$

La valeur de k doit rester secrète et doit être changée à chaque utilisation.

3.4.2 Divers

Le protocole de Rabin repose sur la difficulté de calculer la racine carrée d'un nombre modulo n. McEliece a inventé un autre algorithme de chiffrement à clef publique utilisant les codes correcteurs d'erreurs. Il en existe bien d'autres....

Chapitre 4

Authentification et échanges de clefs

4.1 Signatures numériques

4.1.1 Problématique

En plus de la confidentialité des données, il peut-être important de vérifier l'identité de l'expéditeur (c'est le problème d'authentification et de non désavœu) et l'intégrité du message envoyé. Une signature numérique est un procédé permettant d'adjoindre à un message un petit nombre de bits qui assureront au destinataire l'identité de l'expéditeur et l'intégrité du document transmis. Cela va mettre en jeu une fonction de signature qui dépend d'une clef secrète que seul connait l'expéditeur et une fonction de vérification qui doit vérifier que le message en clair obtenu correspond à la signature. La signature doit être unique et ne doit pas être réutilisable.

Depuis mars 2000, la valeur juridique d'une signature numérique est reconnue.

4.1.2 Signature RSA

La méthode RSA permet de construire un procédé de signature: Alice veut envoyer à Bernard un message. Elle commence par le chiffrer avec sa clef secrète (qu'elle est seule à connaître), puis elle chiffre le résultat à l'aide de la clef publique de Bernard. Elle lui envoie le tout. A réception, Bernard déchiffre le message à l'aide de sa clef privée puis déchiffre le résultat à l'aide de la clef publique d'Alice. Si Alice n'était pas l'expéditeur, alors le message resterait illisible. Il est donc certain de sont authenticité et en outre la confidentialité a été assurée lors de cet échange.

Il existe des attaques spécifiques contre RSA en mode signature. L'une d'elles s'appelle la signature en aveugle: Supposons que Martin veuille faire signer un message x à Bernard. Bernard ne veut pas. Martin choisit alors un nombre r aléatoire et calcule $y = r^e \times x \mod n$ où (e, n) est la clef publique de Bernard. Il envoie y à Bernard et lui demande de signer y. Bernard calcule alors la signature $s = y^d \mod n$ et envoie le résultat à Martin. Celui-ci n'a plus qu'à calculer $s' = s/r$ qui constitue la signature de x car $s^e r^{-e} = y^{ed} r^{-e} = x \mod n$ pour retrouver. Moralité: ne jamais signer n'importe quoi !

Cette attaque est possible car la fonction de chiffrement RSA est multiplicative (ie $f(xy) = f(x)f(y)$ pour tout entier x et y); on dit aussi qu'elle est malléable. Afin d'empêcher cette attaque, on complète le message en effectuant un remplissage (on dit aussi padding) que l'on place avant le message. Les deux derniers standards PKCS 1.0 et PKCS 2.0 du RSA utilisent ce padding.

4.1.3 Signature DSA

Le DSA (Digital Signature Algorithm) est un protocole de signature à clef publique, à base de calcul de logarithme discret. Il repose sur un algorithme dérivé du système de chiffrement et de signature d'El-Gamal. C'est un standard de chiffrement américain adopté depuis 1994. Il est plus lent que le RSA, mais permet, pour une sécurité analogue, d'utiliser des signatures plus courtes (320 bits) que le RSA. Une variante appelée ECDSA utilise des courbes elliptiques pour le calcul des logarithmes discrets.

Mise en oeuvre de l'algorithme:
Il utilise une fonction de hachage que nous noterons $h(x)$ et qui provient de l'algorithme SHA. Nous présenterons les fonctions de hachage au paragraphe suivant; il suffit pour l'instant de considérer que $h(x)$ est un entier dépendant de x. Nous avons également besoin des éléments suivants:

• On choisit un nombre premier p faisant entre 512 et 1024 bits de long.

• Soit q un facteur premier de $p-1$
• Soit $g = \alpha^{(p-1)/q} \mod p$ où α est un entier $< p-1$ tel que $g > 1$
• Soit x un entier $< q$
• Soit $y = g^x \mod p$

• La clef privée est x.
• La clef publique est y.

Pour signer un message m, on opère comme suit:

• Alice choisit un entier $k < q$
• Elle calcule $r = (g^k \mod p) \mod q$ et $s = k^{-1}(h(m) + rx) \mod q$

Le couple (r, s) forme la signature du message m.

• Alice envoie (r, s) à Bernard.
• Bernard vérifie la signature en calculant:

$$\begin{cases} u_1 = h(m)s^{-1} \mod q \\ u_2 = rs^{-1} \mod q \\ v = (g^{u_1} y^{u_2} \mod p) \mod q \end{cases}$$

Si $v = r$, alors la signature est vérifiée.

4.2 Protocoles d'échanges de clefs

4.2.1 Problématique

Les algorithmes à clef secrète nécessitent l'échange d'une clef. Cette clef peut être transmise par l'intermédiaire d'un canal sécurisé, mais il se peut que la transaction doive s'effectuer sur un canal non sécurisé. Les protocoles d'échanges de clefs servent à résoudre ce problème. La distribution de clefs entre plusieurs utilisateurs dans les systèmes à clefs publiques entre aussi dans ce cadre.

Les algorithmes d'échanges de clefs nécessitent souvent l'intervention d'un serveur de clefs ou d'une autorité de certification qui joue le rôle de tiers de confiance durant l'échange.

4.2.2 Diffie et Hellman

$$x$$

$$y$$

$$b = a^x \mod p$$

$$c = a^y \mod p$$

$$b^y \mod p$$

$$c^x \mod p$$

$$b^y \mod p = (a^x)^y \mod p = a^{xy} \mod p = k$$

$$a^x \mod p = (a^y)^x \mod p = a^{yx} \mod p = k$$

$p = 17$

$a = 3$

$x = 6$

$y = 10$

$b = 3^6 \mod 17 = 15$

$c = 3^{10} \mod 17 = 8$

$k = 15^{10} \mod 17 = 4$

$k = 8^6 \mod 17 = 4$

4.2.3 Kerberos

4.3 Fonctions de hachage

4.3.1 Principe

Une fonction de hachage est un autre procédé de vérification d'intégrité d'un document. La fonction de hachage doit produire, à partir du message initial, une séquence de bits très courte (typiquement 120 ou 160 bits) qui représente une signature du texte (on dit aussi un condensé ou une empreinte numérique du texte). Le point le plus important d'une fonction de hachage est qu'une même empreinte ne doit pas pouvoir correspondre à deux messages différents. En d'autres termes, la fonction de hachage doit être injective (on dit aussi qu'elle doit éviter les collisions) et on ne doit pas pouvoir retrouver le message initial à l'aide de l'empreinte. Résumons:

A partir d'un texte en clair x, la fonction de hachage h fournit une empreinte s telle que $s = h(x)$ et $h(x) = h(x') \Rightarrow x = x'$

En général, on utilise les fonctions de hachage pour construire une empreinte qui sera signée (plutôt que de signer le message complet). Avec la méthode RSA, l'expéditeur chiffre l'empreinte avec sa clef secrète et le destinataire la déchiffre avec la clef publique. Cela va plus vite que de chiffrer tout le message (car RSA est lent). Il s'assure ainsi de l'identité de l'expéditeur ainsi que de l'intégrité du document.

4.3.2 Attaque des anniversaires

Plusieurs types d'attaques existent contre les fonctions de hachage. L' attaque qui consiste à trouver un texte différent du texte initial qui possède la même empreinte s'appelle l'attaque des anniversaires. Supposons qu'une classe comporte 23 élèves. Alors la probabilité que deux d'entre eux aient la même date d'anniversaire est supérieure à 1/2. Ceci peut sembler paradoxal car il y a 365 dates possibles et cela illustre le fait qu'il peut être envisageable de chercher un couple ayant la même propriété même si celle-ci se produit rarement.

$$1 - \frac{1}{n}$$

$$1 - \frac{i}{n}$$

$$\mathbb{P}(\bar{A}) = \prod_{i=1}^{k-1} \left(1 - \frac{i}{n}\right)$$

$$A$$

$$\forall x/\ 0 \leq x \leq 11 - x \leq e^{-x}$$

$$\mathbb{P}(\bar{A}) \leq \prod_{i=1}^{k-1} e^{-i/n} = e^{-\frac{k(k-1)}{2n}}$$

$$\mathbb{P}(A) \geq 1 - e^{-\frac{k(k-1)}{2n}}$$

$$\mathbb{P}(A) \geq \frac{1}{2} \iff k = \sqrt{2 \ln 2 \times n}$$

$$\mathbb{P}(A) \simeq \sqrt{n}$$

$n = 365$ $k = 23$ $\mathbb{P}(A) \geq 1/2$
2^n

$$2^{\frac{n}{2}}$$

Pour une empreinte de 128 bits, on doit tester 2^{64} couples de textes en moyenne pour en trouver qui ont la même empreinte. Pour une empreinte de 160 bits, il faut en tester en moyenne 2^{80}.

4.3.3 MD4, MD5

MD-4 (Message Digest) est un procédé de hachage qui a été cryptanalysé en 1996. MD-5 est une amélioration de MD-4 qui donne une empreinte de 128 bits. Certaines faiblesses de conception ont été détectées récemment. Le principe est d'opérer une suite de manipulations sur les bits qui assure toutes les propriétés requises. Le MD-5 opère sur des blocs de 512 bits.

Pour le mettre en oeuvre, on allonge le message initial pour obtenir une longueur n mulitple de 512 bits, les 64 derniers bits indiquant la longueur du message initial et les bits intermédiaires entre la fin du message et les 64 derniers bits sont formés d'un 1 suivi du nombre nécessaire de 0. Ensuite, chaque bloc de 512 bits est divisé en 16 blocs de 32 bits chacun, puis l'algorithme entre dans une boucle qui sera effectuée n fois. Cette boucle comporte 4 rondes, chacune effectuant 16 fois une opération de base dont voici le détail:

On dispose de 4 variables de 32 bits a, b, c et d initialisées à une valeur précise. On dispose également de 4 fonctions de 3 variables, non linéaires:

$$\begin{cases} f(x,y,z) = (x \wedge y) \vee (\neg x \wedge z) \\ g(x,y,z) = (x \wedge z) \vee (y \wedge \neg z) \\ h(x,y,z) = x \oplus y \oplus z \\ i(x,y,z) = y \oplus (x \vee \neg z) \end{cases}$$

Chaque opération de base calcule l'image d'une de ces fonctions pour trois des quatre variables a, b, c, d. Puis on ajoute au résultat la quatrième variable ainsi qu'une constante et un sous-bloc. On décale vers la gauche le résultat d'un nombre variables de bits, on l'ajoute encore à l'une des quatre variables et on stocke le tout à la place de a, b, c ou d. Par exemple, si x_i représente le ième sous-bloc, α_i représente la ième constante et $\ll s$ un décalage de s vers la gauche, on va calculer des expressions du type

$$\begin{cases} a = b + ((a + f(b,c,d) + x_i + \alpha_i) \ll s \\ d = a + ((d + g(a,b,c) + x_i + \alpha_i) \ll s \\ c = d + ((c + h(d,a,b) + x_i + \alpha_i) \ll s \\ a = b + ((a + i(b,c,d) + x_i + \alpha_i) \ll s \end{cases}$$

Les constantes α_i sont données par $\alpha_i = 2^{32}|\sin i|$

Au final, Les constantes modifiées au fur et à mesure des boucles sont concaténées pour donner l'empreinte.

4.3.4 SHA1

Le procédé SHA-1 (Secure Hash Algorithm) est le standard américain de fonction de hachage. Il est utilisé dans le protocole DSA de signature numérique et dans la fonction RIPEMD-160 qui a été conçue dans le cadre d'un projet européen. L'empreinte a une longueur de 160 bits.

Le principe ressemble beaucoup à celui de MD-5: Le message est allongé pour obtenir un longueur n multiple de 512 bits et 5 variables de 32 bits a, ..., e sont initialisées à des valeurs précises. La boucle principale traite un bloc de 512 bit et consiste en 4 rondes de 16 opérations de base chacune. Chaque opération de base effectue effectue un calcul d'image sur trois des 5 variables à l'aide des fonctions non linéaires suivantes:

$$\begin{cases} f_t(x,y,z) = (x \wedge y) \vee (\neg x \wedge z) & \text{si } 0 \leq t \leq 19 \\ g_t(x,y,z) = x \oplus y \oplus z & \text{si } 20 \leq t \leq 39 \\ h_t(x,y,z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z) & \text{si } 40 \leq t \leq 59 \\ i(x,y,z) = x \oplus y \oplus z & \text{si } 60 \leq t \leq 79 \end{cases}$$

t étant un paramètre entier qui varie de 0 à 79. L'algorithme utilise également des constantes k_t dépendant de t et chacun des 16 mots de 32 bits du bloc courant est transformé en 80 mots m_t de 32 bits .

En janvier 2005, une équipe a cryptanalysé le SHA-1 qui est devenu un protocole non sûr.

4.3.5 Codes d'authentification

Un CAM (Code d'Authentification de Messages) est une fonction de hachage à sens unique dépendant d'une clef secrète. Seul la personne possédant cette clef peut vérifier l'empreinte numérique.

4.4 Partage de secrets

Preuve sans apport d'information, secret morcelé, secret réparti, schéma à seuil.

Chapitre 5

Quelques exemples.

5.1 Dans les réseaux sans fil.

5.1.1 Bluetooth.

Bluetooth est une technologie de communication sans fil, radio, à courte distance, destinée à relier des appareils électroniques. IEEE 802.15 est normalisé depuis 1999, utilise la bande de fréquence de 2.4 GHz et possède un débit de 723 kbps ou 2.1 Mbps. Chaque appareil peut communiquer avec au plus 7 autres appareils. Elle permet des connexions point à point, ou bien point à multipoint. Le nom est inspiré du roi Harald « à la dent bleue » qui a réussi à unifier le Danemark, la Norvège et la Suède; le logo donne les initiales de ce roi en alphabet runique.

Les communications sont basées sur le protocole maître / esclave. Un maître est relié à sept esclaves pour formé un piconet. Le maître contrôle et synchronise les échanges. Un esclave peut avoir plusieurs maîtres. La pile de protocoles gère les échanges.

Les protocoles cryptographiques de Bluetooth se situent soit au niveau de la couche de service, soit au niveau de la couche liaison.

La clef de lien est un nombre aléatoire de 128 bits, utilisé dans la procédure d'authentification pour générer une clef de chiffrement. La durée de vie d'une clef
Le code PIN sert à authentifier l'utilisateur; sa longueur varie de 1 à 16 octets (souvent 4).
Si le mode de sécurité niveau couche de liaison est choisi, les protocoles de chiffrement sont initialisés avant l'ouverture du canal de communication. Une clef secrète est partagée par chaque paire d'appareils. La procédure de pairage génère cette clef secrète lorsque les deux appareils communiquent pour la première fois. Ce procédé se passe en trois temps:
- Création d'une clef d'initialisation. - Création d'une clef de lien. - Authentification mutuelle.
Avant de commencer le pairage, le code PIN des deux appareils doit être entré dans les deux appareils. La clef d'initialisation est créée par un algorithme appelé E22. La clef est générée à partir de l'adresse bluetooth de l'appareil, de son code PIN et d'un nombre pseudo aléatoire. Notons cette clef K_i.
La clef de lien est ensuite générée de la façon suivante: Chaque appareil (maître et esclave) génère un nombre aléatoire

5.1.2 Wifi.

5.2 La carte bancaire

En plus de l'identité I du propriétaire, la carte possède une valeur V de confidentialité (qui est un identifiant signé par la banque) vérifiant $I \times V^e = 1 \mod n$ où (e, n) forme la clef publique d'un système RSA. La banque possède le clef secrète correspondante et peut donc authentifier la carte. Une carte comportant une signature valide est considérée comme authentique.

La carte choisit un nombre x aléatoire et calcule $x_1 = x^e \mod n$
Le terminal choisit une valeur α aléatoire et l'envoie à la carte
Celle-ci calcule $x_2 = x \times V^\alpha \mod n$

La taille de cette clef était de 320 bits en 1998, ce que l'on peut factoriser en quelques jours avec un n'importe quel PC. Pour fabriquer de fausses cartes, il suffisait d'obtenir la clef publique de la banque (le nombre n) qui était présente dans tous les terminaux de paiement. La factorisation donnait alors la

clef secrète qui pouvait être utilisée pour fabriquer une signature valide avec n'importe quelle carte à puce. La sécurité du protocole RSA et celle de la carte à puce ne sont pas remis en cause. Le seul problème était la taille de la clef et le fait que pour certains achats, la carte ne contacte pas la banque. Les cartes actuelles possèdent des clefs de 792 bits.

5.3 La carte vitale

5.4 Le logiciel PGP

5.5 Standard SSL et IPsec

Chapitre 6

Annexes

6.1 Cryptographie quantique

Juste quelques mots..... Définition d'un qubit, protocole de Bennett, états intriqués d'une paire de qubits.

6.2 Aspects juridiques

Jusqu'en 1986, la cryptographie était purement et simplement interdite en France et réservée à un usage militaire. A partir de 1990, le SCSSI (Service Central de Sécurité des Systèmes Informatiques) délivre des autorisations ponctuelles. Depuis juillet 1996, la législation s'est assouplie: les procédés d'authentification et de signature sont libres ainsi que les procédés de cryptographie ayant une clef inférieure à 40 bits. Au delà, il faut une autorisation.

6.3 stéganographie

Principe du watermarking. Robustesse. Détection, décodage utilisant les codes correcteurs.

6.4 Bibliographie