Algorithmique, structures informatiques et **cryptologie**

Anaïs Barthoulot
anais.barthoulot@orange.com

Année 2022-2023

UNIVERSITÉ
CAEN
NORMANDIE

Sommaire

Qu'est ce que la cryptologie ?

- Cryptologie : cryptographie et cryptanalyse

Qu'est ce que la cryptologie ?

- Cryptologie : cryptographie et cryptanalyse
- Cryptographie : *crypto* = caché, *graphie* = écriture

Qu'est ce que la cryptologie ?

- Cryptologie : cryptographie et cryptanalyse
- Cryptographie : *crypto* = caché, *graphie* = écriture
- Cryptographie = art de ≪ cacher ≫ un message, de le rendre illisible

Qu'est ce que la cryptologie ?

- Cryptologie : cryptographie et cryptanalyse
- Cryptographie : *crypto* = caché, *graphie* = écriture
- Cryptographie = art de ≪ cacher ≫ un message, de le rendre illisible
- ATTENTION ! La cryptographie ne cache pas le fait que j'ai écris un message, juste son contenu !

Qu'est ce que la cryptologie ?

- Cryptologie : cryptographie et cryptanalyse
- Cryptographie : *crypto* = caché, *graphie* = écriture
- Cryptographie = art de « cacher » un message, de le rendre illisible
- ATTENTION ! La cryptographie ne cache pas le fait que j'ai écris un message, juste son contenu !
- Cacher le fait qu'il y a un message = stéganographie

Qu'est ce que la cryptologie ?

- Cryptologie : cryptographie et cryptanalyse
- Cryptographie : *crypto* = caché, *graphie* = écriture
- Cryptographie = art de ≪ cacher ≫ un message, de le rendre illisible
- ATTENTION ! La cryptographie ne cache pas le fait que j'ai écris un message, juste son contenu !
- Cacher le fait qu'il y a un message = stéganographie
- Cryptanalyse = attaque de la cryptographie

Où trouve-t-on de la cryptographie ?

- Sur les sites internet, c'est indiqué par le fameux http**S**

Où trouve-t-on de la cryptographie ?

- Sur les sites internet, c'est indiqué par le fameux http**S**
- Dans les cartes à puces : cartes bancaires, carte vitale

Où trouve-t-on de la cryptographie ?

- Sur les sites internet, c'est indiqué par le fameux http**S**
- Dans les cartes à puces : cartes bancaires, carte vitale
- Dans les puces RFID = Radio Frequency IDentication

Où trouve-t-on de la cryptographie ?

- Sur les sites internet, c'est indiqué par le fameux http**S**
- Dans les cartes à puces : cartes bancaires, carte vitale
- Dans les puces RFID = Radio Frequency IDentication
- Dans la télévision payante

Où trouve-t-on de la cryptographie ?

- Sur les sites internet, c'est indiqué par le fameux http**S**
- Dans les cartes à puces : cartes bancaires, carte vitale
- Dans les puces RFID = Radio Frequency IDentication
- Dans la télévision payante
- Dans les télécommunications : WIFI, ...

Vocabulaire utile

- Le **clair** : le message que l'on souhaite envoyer

Vocabulaire utile

- Le **clair** : le message que l'on souhaite envoyer

- Le **chiffré** : le message après avoir été « caché »

Vocabulaire utile

- Le **clair** : le message que l'on souhaite envoyer
- Le **chiffré** : le message après avoir été « caché »
- **Chiffrer** : l'opération qui transforme le clair en chiffré à partir d'une autre information (une clé)

Vocabulaire utile

- Le **clair** : le message que l'on souhaite envoyer
- Le **chiffré** : le message après avoir été « caché »
- **Chiffrer** : l'opération qui transforme le clair en chiffré à partir d'une autre information (une clé)
- **Déchiffrer** : l'opération qui révèle le clair à partir du chiffré, grâce à une autre information (une clé, qui peut être différente de la première)

Vocabulaire utile

- Le **clair** : le message que l'on souhaite envoyer
- Le **chiffré** : le message après avoir été « caché »
- **Chiffrer** : l'opération qui transforme le clair en chiffré à partir d'une autre information (une clé)
- **Déchiffrer** : l'opération qui révèle le clair à partir du chiffré, grâce à une autre information (une clé, qui peut être différente de la première)
- **Décrypter** : retrouver le clair à partir du chiffré sans connaître la clé

Vocabulaire utile

- Le **clair** : le message que l'on souhaite envoyer
- Le **chiffré** : le message après avoir été « caché »
- **Chiffrer** : l'opération qui transforme le clair en chiffré à partir d'une autre information (une clé)
- **Déchiffrer** : l'opération qui révèle le clair à partir du chiffré, grâce à une autre information (une clé, qui peut être différente de la première)
- **Décrypter** : retrouver le clair à partir du chiffré sans connaître la clé
- ATTENTION ! Crypter : ne veut rien dire ! Cela revient à dire que l'on peut cacher le message sans connaître de clé, ce qui n'est pas possible

Les garanties de la cryptographie

- **Confidentialité** : le message ne pourra être lu que par son destinataire légitime

Les garanties de la cryptographie

- **Confidentialité** : le message ne pourra être lu que par son destinataire légitime
- **Intégrité** : le destinataire peut s'assurer que le message n'a pas été modifié

Les garanties de la cryptographie

- **Confidentialité** : le message ne pourra être lu que par son destinataire légitime
- **Intégrité** : le destinataire peut s'assurer que le message n'a pas été modifié
- **Authenticité** : le destinataire peut s'assurer de l'origine du message (que l'expéditeur est bien celui qu'il prétend être)

Les garanties de la cryptographie

- **Confidentialité** : le message ne pourra être lu que par son destinataire légitime
- **Intégrité** : le destinataire peut s'assurer que le message n'a pas été modifié
- **Authenticité** : le destinataire peut s'assurer de l'origine du message (que l'expéditeur est bien celui qu'il prétend être)
- **Non répudiation** : l'expéditeur ne peut nier être à l'origine du message

Deux principes à retenir

- **Principe de Kerckhoffs** : pour garantir la sécurité d'un schéma de chiffrement, la méthode doit être publique (elle finira par l'être un jour de toute façon). Seule une petite partie de la méthode (appelée clé) restera secrète et sera facilement modifiable.

Deux principes à retenir

- **Principe de Kerckhoffs** : pour garantir la sécurité d'un schéma de chiffrement, la méthode doit être publique (elle finira par l'être un jour de toute façon). Seule une petite partie de la méthode (appelée clé) restera secrète et sera facilement modifiable.

- **Loi de Moore** : la puissance d'un ordinateur double tous les 10 ans (environ)

Crypto-monnaies $\neq$ cryptologie

- ATTENTION ! Appelées parfois $\ll$ Crypto $\gg$ mais à ne pas confondre avec la cryptologie !

Crypto-monnaies $\neq$ cryptologie

- ATTENTION ! Appelées parfois $\ll$ Crypto $\gg$ mais à ne pas confondre avec la cryptologie !

- Définition de l'authorité des marchés financiers (AMF) :
  une crypto-monnaie ou un crypto-actif désigne des actifs numériques virtuels qui reposent sur la technologie de la blockchain, (chaine de bloc) à travers un registre décentralisé et un protocole crypté.

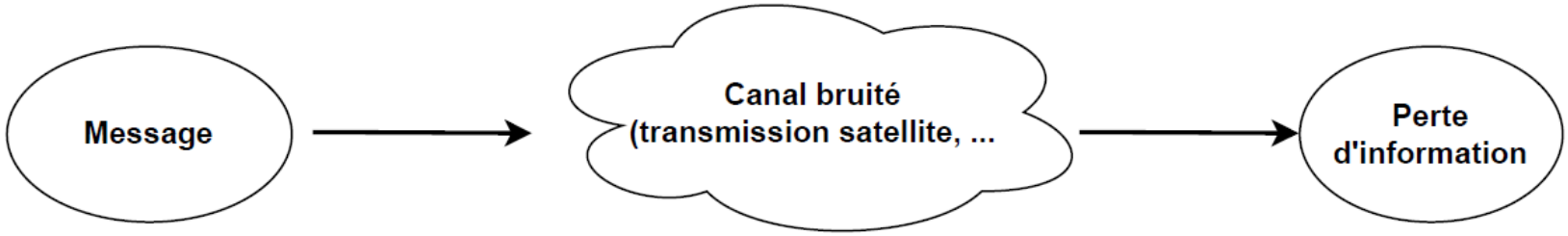
Crypto-monnaies $\neq$ cryptologie

- ATTENTION ! Appelées parfois $\ll$ Crypto $\gg$ mais à ne pas confondre avec la cryptologie !

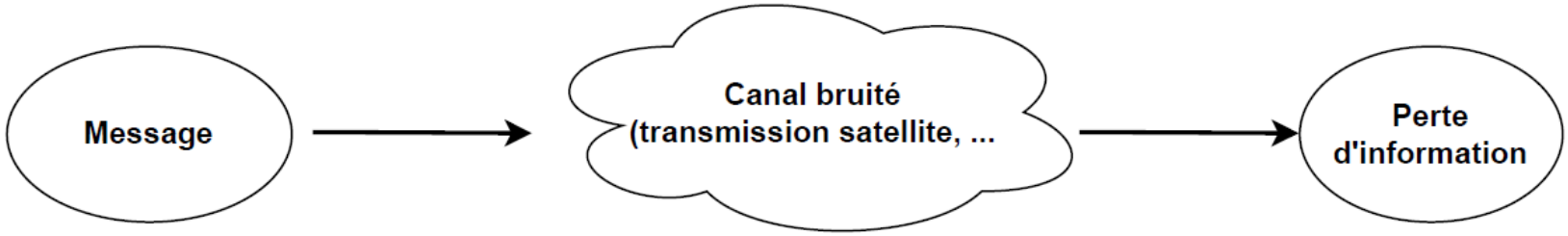
- Définition de l'authorité des marchés financiers (AMF) :
  une crypto-monnaie ou un crypto-actif désigne des actifs numériques virtuels qui reposent sur la technologie de la blockchain, (chaine de bloc) à travers un registre décentralisé et un protocole crypté.

Notez l'erreur dans la définition ;)

Codage $\neq$ cryptologie

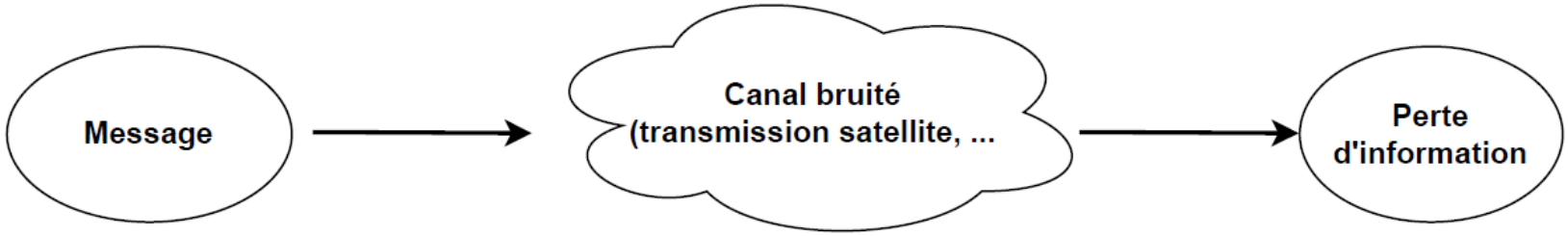


Codage $\neq$ cryptologie



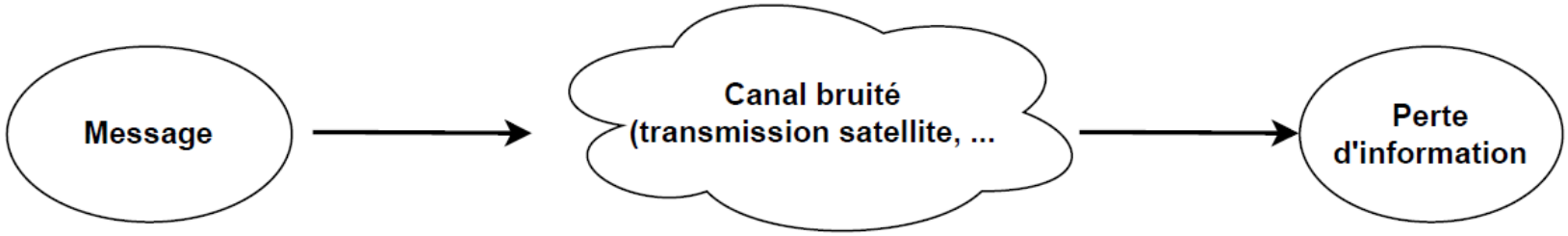
- Pour corriger ça : codes correcteurs d'erreurs

Codage $\neq$ cryptologie



- Pour corriger ça : codes correcteurs d'erreurs
- Principe : ajouter de la redondance

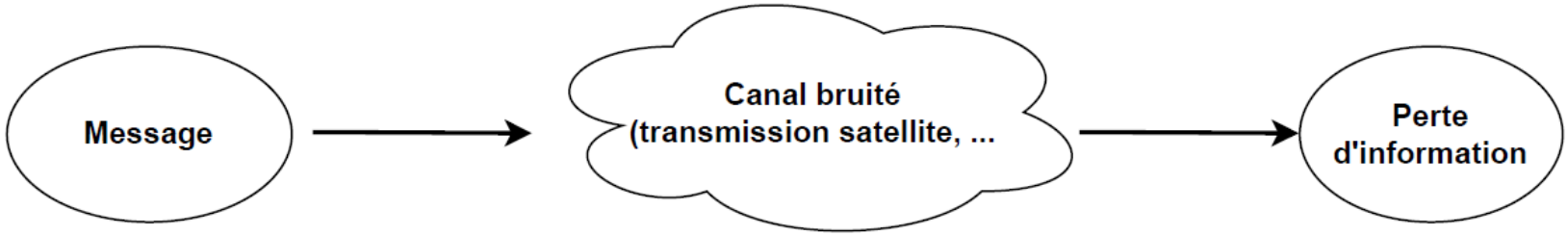
Codage $\neq$ cryptologie



- Pour corriger ça : codes correcteurs d'erreurs
- Principe : ajouter de la redondance

« O comme Oscar »

Codage $\neq$ cryptologie



- Pour corriger ça : codes correcteurs d'erreurs
- Principe : ajouter de la redondance

« O comme Oscar »

- ATTENTION ! **Coder** et **Décoder** sont des termes utilisés pour les codes correcteurs, à ne pas confondre avec **Chiffrer** et **Déchiffrer**, malgré leur ressemblance (et vice versa)

Codes correcteurs d'erreur : exemple

Information	Message
00	000
01	011
10	101
11	110

Codes correcteurs d'erreur : exemple

Information	Message
00	000
01	011
10	101
11	110

- Ajout d'un bit de parité $\longrightarrow$ indique nombre de 1 dans le message initial

Codes correcteurs d'erreur : exemple

Information	Message
00	000
01	011
10	101
11	110

- Ajout d'un bit de parité $\longrightarrow$ indique nombre de 1 dans le message initial
- Si l'on reçoit 111, on sait qu'il y a une erreur mais on ne sait pas où !

Codes correcteurs : un exemple dans la vie courante

- Vous utilisez régulièrement un objet qui est protégé par des codes correcteurs... lequel ?

Codes correcteurs : un exemple dans la vie courante

- Vous utilisez régulièrement un objet qui est protégé par des codes correcteurs... lequel ?
- Votre numéro de sécurité sociale sur votre carte vitale !

Codes correcteurs : un exemple dans la vie courante

- Vous utilisez régulièrement un objet qui est protégé par des codes correcteurs... lequel ?
- Votre numéro de sécurité sociale sur votre carte vitale !
- 13 chiffres + une clé de vérification

Codes correcteurs : un exemple dans la vie courante

- Vous utilisez régulièrement un objet qui est protégé par des codes correcteurs... lequel ?
- Votre numéro de sécurité sociale sur votre carte vitale !
- 13 chiffres + une clé de vérification
- La clé est égale à $97 - (\text{numéro} \mod 97)$

Rappel : l'opérateur **XOR**

- Addition modulo 2

Rappel : l'opérateur **XOR**

- Addition modulo 2

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

Rappel : l'opérateur **XOR**

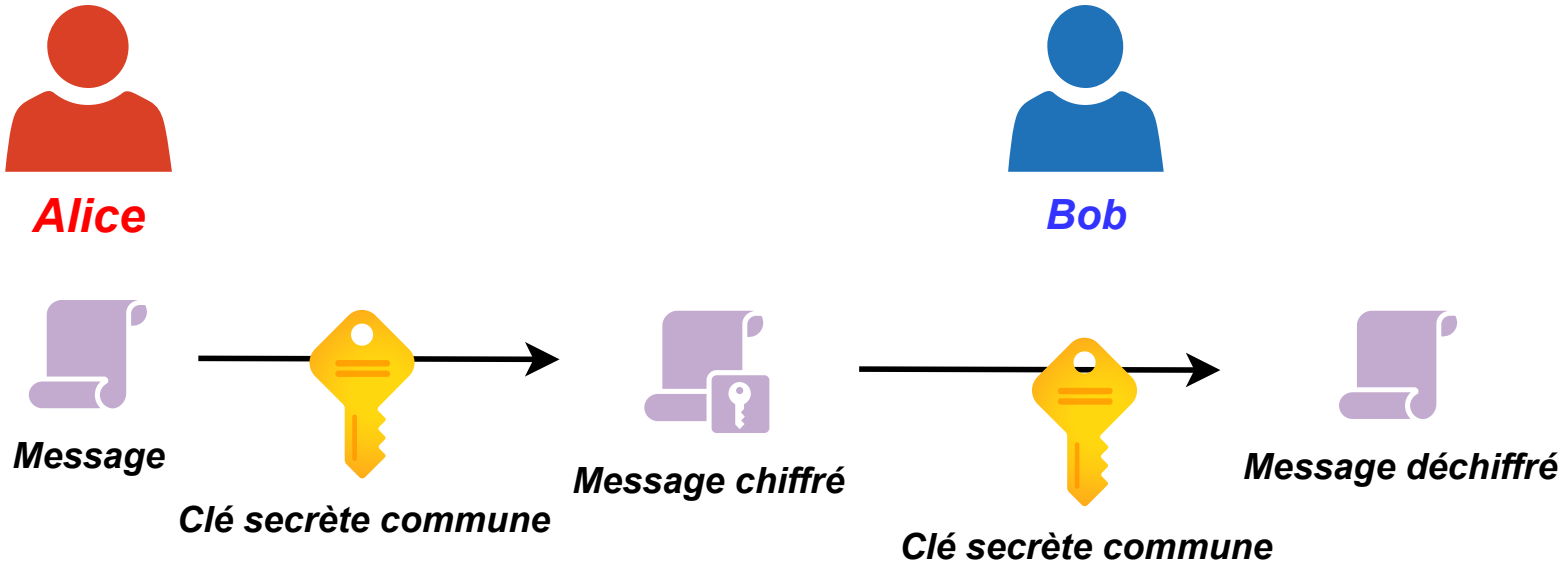
- Addition modulo 2

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

- Propriétés :
  - $A \oplus A = 0$
  - $A \oplus B = C \implies C \oplus B = A$
  - $A \oplus B \oplus B = A$

Sommaire

Cryptographie symétrique, ou à clé secrète



Chiffrements antiques : la scytale

Chiffrements antiques : la scytale

- Message est écrit le long du cylindre, une lettre par morceau de bande de ruban.

Chiffrements antiques : la scytale

- Message est écrit le long du cylindre, une lettre par morceau de bande de ruban.

- Message chiffré correspond à la lecture du ruban déroulé (lecture en colonne).

Chiffrements antiques : la scytale

- Message est écrit le long du cylindre, une lettre par morceau de bande de ruban.
- Message chiffré correspond à la lecture du ruban déroulé (lecture en colonne).
- Pour déchiffrer : connaitre nombre N de lettres par tour de ruban ou L nombre de tour de ruban ($L * N =$ nombre de lettres du message).

Chiffrements antiques : la scytale

- Message est écrit le long du cylindre, une lettre par morceau de bande de ruban.
- Message chiffré correspond à la lecture du ruban déroulé (lecture en colonne).
- Pour déchiffrer : connaitre nombre N de lettres par tour de ruban ou L nombre de tour de ruban ($L * N =$ nombre de lettres du message).
- Enrouler le message autour du cylindre et le message clair apparait. C'est un chiffrement par **permutation**

Chiffrements antiques : le chiffrement de César

a	b	c	d	e	f	g	h	i	j	k	l	$\cdots$	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	$\cdots$	21	22	23	24	25

Chiffrements antiques : le chiffrement de César

a	b	c	d	e	f	g	h	i	j	k	l	$\cdots$	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	$\cdots$	21	22	23	24	25

- Chiffrement de César : on décale chaque lettre de 3 rangs.

Chiffrements antiques : le chiffrement de César

a	b	c	d	e	f	g	h	i	j	k	l	$\cdots$	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	$\cdots$	21	22	23	24	25

- Chiffrement de César : on décale chaque lettre de 3 rangs.

Exemple :

m	e	s	s	a	g	e
12	4	18	18	0	6	4
15	7	21	21	3	9	7
p	h	v	v	d	j	h

Chiffrements antiques : le chiffrement de César

a	b	c	d	e	f	g	h	i	j	k	l	· · ·	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	· · ·	21	22	23	24	25

- Chiffrement de César : on décale chaque lettre de 3 rangs.

Exemple :

m	e	s	s	a	g	e
12	4	18	18	0	6	4
15	7	21	21	3	9	7
p	h	v	v	d	j	h

- C'est une **substitution monoalphabétique**

Chiffrements antiques : ROT 13

a	b	c	d	e	f	g	h	i	j	k	l	· · ·	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	· · ·	21	22	23	24	25

Chiffrements antiques : ROT 13

a	b	c	d	e	f	g	h	i	j	k	l	· · ·	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	· · ·	21	22	23	24	25

- Autre décalage : le chiffrement ROT 13, où les lettres sont décalées de 13 rangs.

Chiffrements antiques : ROT 13

a	b	c	d	e	f	g	h	i	j	k	l	· · ·	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	· · ·	21	22	23	24	25

- Autre décalage : le chiffrement ROT 13, où les lettres sont décalées de 13 rangs.

Exemple :

m	e	s	s	a	g	e
12	4	18	18	0	6	4
25	17	5	5	13	19	17
z	r	f	f	n	t	r

Chiffrements antiques : ROT 13

a	b	c	d	e	f	g	h	i	j	k	l	· · ·	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	· · ·	21	22	23	24	25

- Autre décalage : le chiffrement ROT 13, où les lettres sont décalées de 13 rangs.

Exemple :

m	e	s	s	a	g	e
12	4	18	18	0	6	4
25	17	5	5	13	19	17
z	r	f	f	n	t	r

ATTENTION ! $\text{Rot13}(\text{Rot13}(\text{m})) = \text{m}$!

Chiffrements antiques : ROT 13

a	b	c	d	e	f	g	h	i	j	k	l	· · ·	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	· · ·	21	22	23	24	25

- Autre décalage : le chiffrement ROT 13, où les lettres sont décalées de 13 rangs.

Exemple :

m	e	s	s	a	g	e
12	4	18	18	0	6	4
25	17	5	5	13	19	17
z	r	f	f	n	t	r

ATTENTION ! Rot13(Rot13(m)) = m !

Fonctionne avec n'importe quel décalage, qui sera la clé secrète.

Chiffrements antiques : chiffrement de Vigenère

- Clé secrète : un mot (ou une phrase)

Chiffrements antiques : chiffrement de Vigenère

- Clé secrète : un mot (ou une phrase)
- Pour chiffrer : on répète la clé sous le message, et on applique le décalage correspondant

Chiffrements antiques : chiffrement de Vigenère

- Clé secrète : un mot (ou une phrase)
- Pour chiffrer : on répète la clé sous le message, et on applique le décalage correspondant

Exemple :

m	e	s	s	a	g	e		i	m	p	o	r	t	a	n	t
c	r	y	p	t	o	l		o	g	i	e	c	r	y	p	t
o	v	q	h	t	u	p		w	s	x	s	t	k	y	c	m

Chiffrements antiques : chiffrement de Vigenère

- Clé secrète : un mot (ou une phrase)
- Pour chiffrer : on répète la clé sous le message, et on applique le décalage correspondant

  Exemple :

m	e	s	s	a	g	e		i	m	p	o	r	t	a	n	t
c	r	y	p	t	o	l		o	g	i	e	c	r	y	p	t
o	v	q	h	t	u	p		w	s	x	s	t	k	y	c	m

- C'est une **substitution polyalphabétique**

Permutation $\neq$ substitution

- **Substitution** : change les caractères du message

Permutation $\neq$ substitution

- **Substitution** : change les caractères du message

  $\longrightarrow$ cache l'information, augmente la confusion

Permutation $\neq$ substitution

- **Substitution** : change les caractères du message

  $\longrightarrow$ cache l'information, augmente la confusion

- **Permutation** : change la place des caractères

Permutation $\neq$ substitution

- **Substitution** : change les caractères du message

  $\longrightarrow$ cache l'information, augmente la confusion

- **Permutation** : change la place des caractères

  $\longrightarrow$ disperse le bruit, propage la confusion

Enigma (1926)

- Machine électromécanique
- Utilisée par les allemands pendant la seconde guerre mondiale
- Utilise des rotors pour décaler les lettres

Chiffrement par blocs ou par flot

- On considère deux grandes familles de schémas de chiffrement

Chiffrement par blocs ou par flot

- On considère deux grandes familles de schémas de chiffrement
- Ceux qui chiffrent le message lettre par lettre (ou bit à bit) : chiffrement par flot

Chiffrement par blocs ou par flot

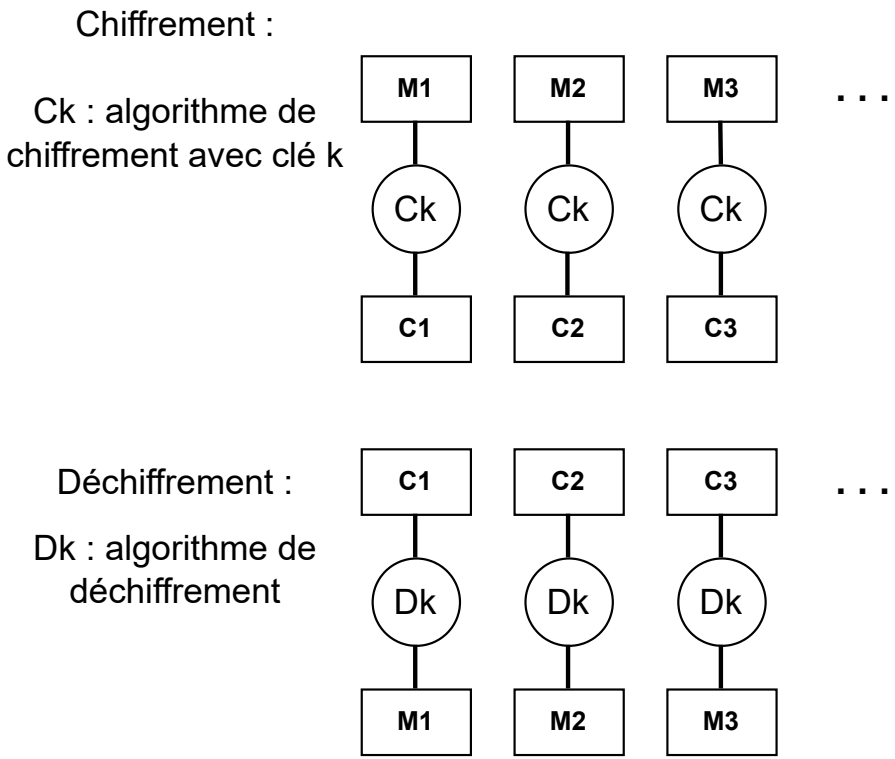
- On considère deux grandes familles de schémas de chiffrement
- Ceux qui chiffrent le message lettre par lettre (ou bit à bit) : chiffrement par flot
- Ceux qui chiffrent le message par bloc de lettres (ou bloc de bits) : chiffrement par bloc

Chiffrement par blocs : différents traitements

- Combiner les blocs (chaînage) améliore la sécurité

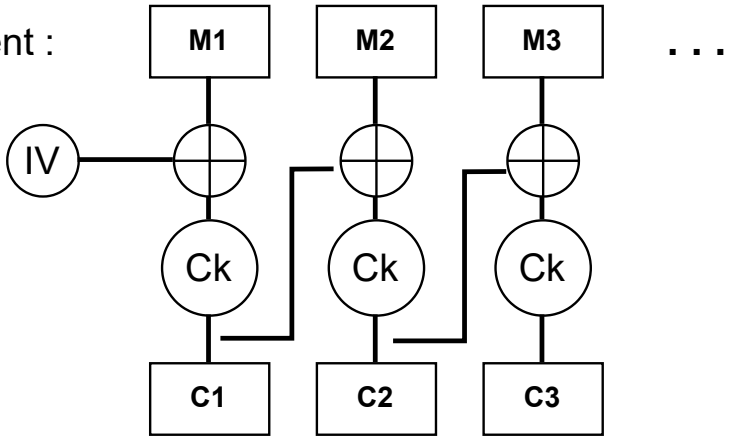
Chiffrement par blocs : différents traitements

- Combiner les blocs (chaînage) améliore la sécurité
- Mode **ECB** (Electronic Code Book)

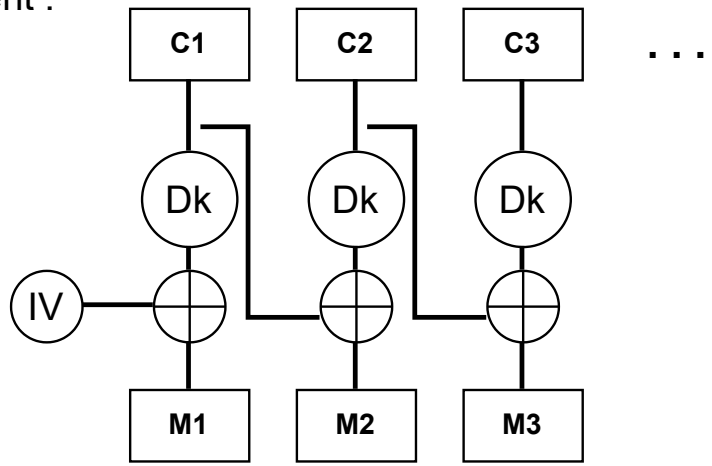


Mode **CBC** (Cipher Block Chaining)

Chiffrement :

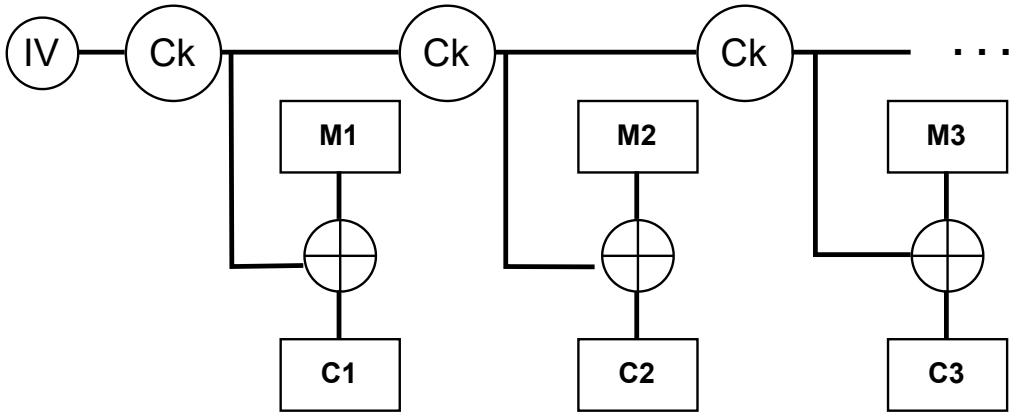


Déchiffrement :

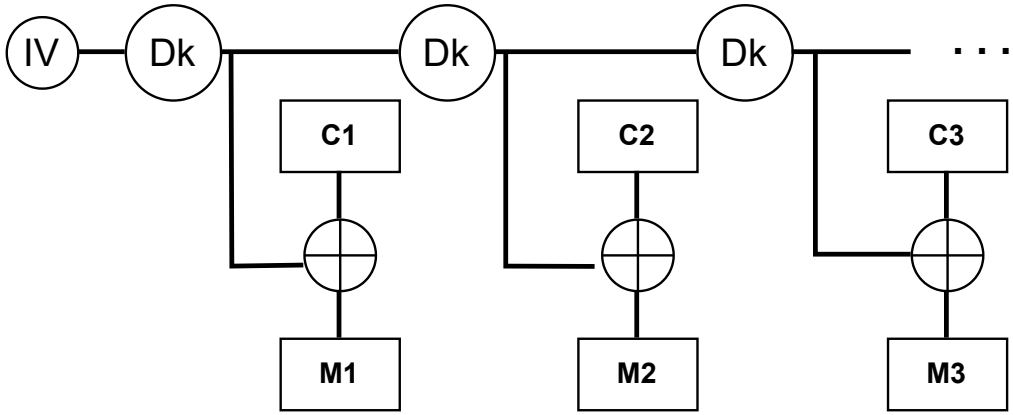


Mode **OFB** (Output Feed Back)

Chiffrement :



Déchiffrement :



Il existe d'autres modes : CFB, CTR, ...

Schéma de Feistel : utilisé dans les chiffrements par bloc

- Souvent on utilise le schéma suivant :

Schéma de Feistel : utilisé dans les chiffrements par bloc

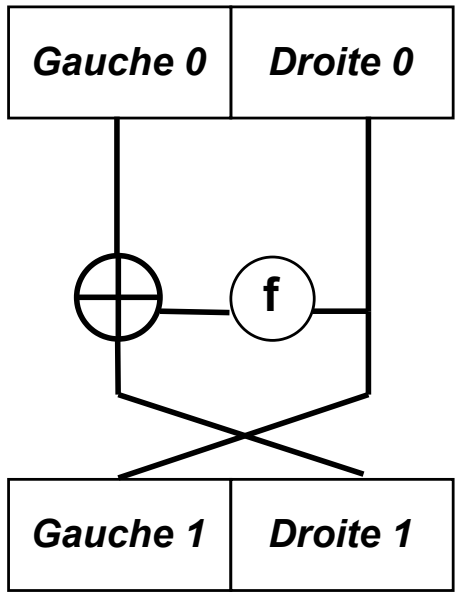
- Souvent on utilise le schéma suivant :

$$\begin{cases} Gauche_n = Droite_{n-1} \\ Droite_n = Gauche_{n-1} \oplus f(Droite_{n-1}) \end{cases}$$

Schéma de Feistel : utilisé dans les chiffrements par bloc

- Souvent on utilise le schéma suivant :

$$\begin{cases} Gauche_n = Droite_{n-1} \\ Droite_n = Gauche_{n-1} \oplus f(Droite_{n-1}) \end{cases}$$



Un schéma de chiffrement par blocs : le D.E.S.

- Data Encryption Standard (D.E.S.)

Un schéma de chiffrement par blocs : le D.E.S.

- Data Encryption Standard (D.E.S.)
- Standard de chiffrement en 1975

Un schéma de chiffrement par blocs : le D.E.S.

- Data Encryption Standard (D.E.S.)
- Standard de chiffrement en 1975
- Taille des blocs : 64 bits (8 octets)

Un schéma de chiffrement par blocs : le D.E.S.

- Data Encryption Standard (D.E.S.)
- Standard de chiffrement en 1975
- Taille des blocs : 64 bits (8 octets)
- 16 sous clés de 48 bits, dérivées d'une clé maître K de 64 bits

Un schéma de chiffrement par blocs : le D.E.S.

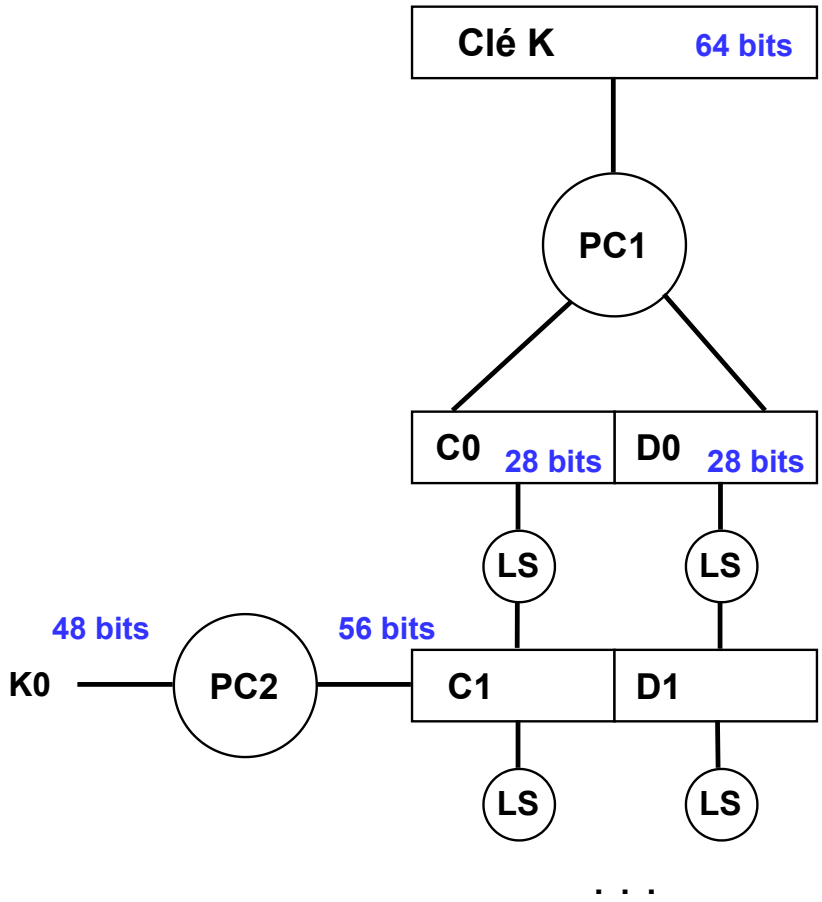
- Data Encryption Standard (D.E.S.)
- Standard de chiffrement en 1975
- Taille des blocs : 64 bits (8 octets)
- 16 sous clés de 48 bits, dérivées d'une clé maître K de 64 bits
- K contient des bits de parité $\longrightarrow$ détecte erreur lors de transmission ou stockage

D.E.S. : génération des clés

- K sous forme de matrice 8×8 avec dernier bit de chaque octet positionné tel qu'il y ait un nombre **impair** de 1 dans l'écriture binaire de l'octet

D.E.S. : génération des clés

- K sous forme de matrice 8×8 avec dernier bit de chaque octet positionné tel qu'il y ait un nombre **impair** de 1 dans l'écriture binaire de l'octet



D.E.S. : tables PC et LS

- PC1 est une table de permutation 8×7 (les bits de parité ne sont pas pris en compte)

57	49	41	33	25	17	9
1	58	50	42	34	26	18
...						

D.E.S. : tables PC et LS

- PC1 est une table de permutation 8×7 (les bits de parité ne sont pas pris en compte)

57	49	41	33	25	17	9
1	58	50	42	34	26	18
...						

- PC2, PC3, ... n'utilisent que 48 bits, table de 8×6

D.E.S. : tables PC et LS

- PC1 est une table de permutation 8×7 (les bits de parité ne sont pas pris en compte)

57	49	41	33	25	17	9
1	58	50	42	34	26	18
...						

- PC2, PC3, ... n'utilisent que 48 bits, table de 8×6

- LS = Left shift : décalage circulaire à gauche
  Exemple : 1, 2, 3, 4 $\rightarrow$ 2, 3, 4, 1

D.E.S. : tables PC et LS

- PC1 est une table de permutation 8×7 (les bits de parité ne sont pas pris en compte)

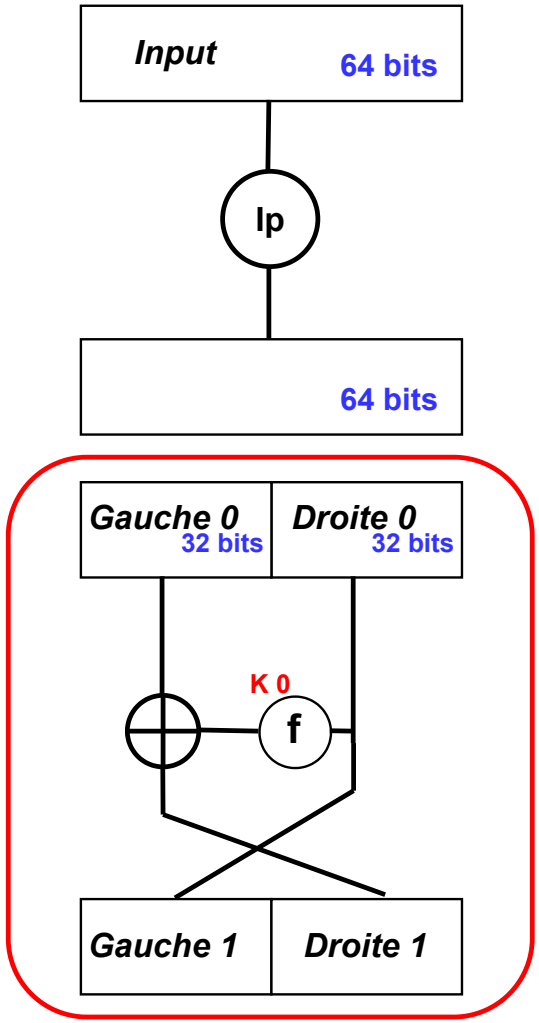
57	49	41	33	25	17	9
1	58	50	42	34	26	18
...						

- PC2, PC3, ... n'utilisent que 48 bits, table de 8×6

- LS = Left shift : décalage circulaire à gauche
  Exemple : 1, 2, 3, 4 $\rightarrow$ 2, 3, 4, 1

- Deux LS à la suite, sauf aux tours 1,2,9,16

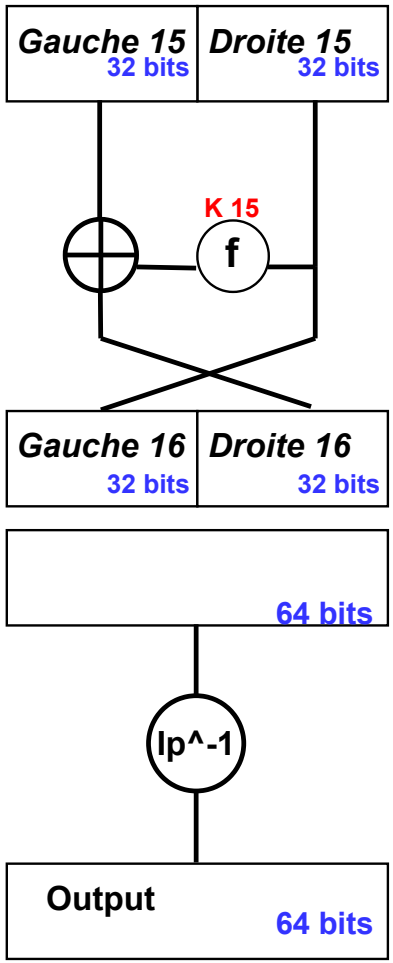
D.E.S. : fonctionnement (1)



1 tour

16 tours au total

D.E.S. : fonctionnement (2)



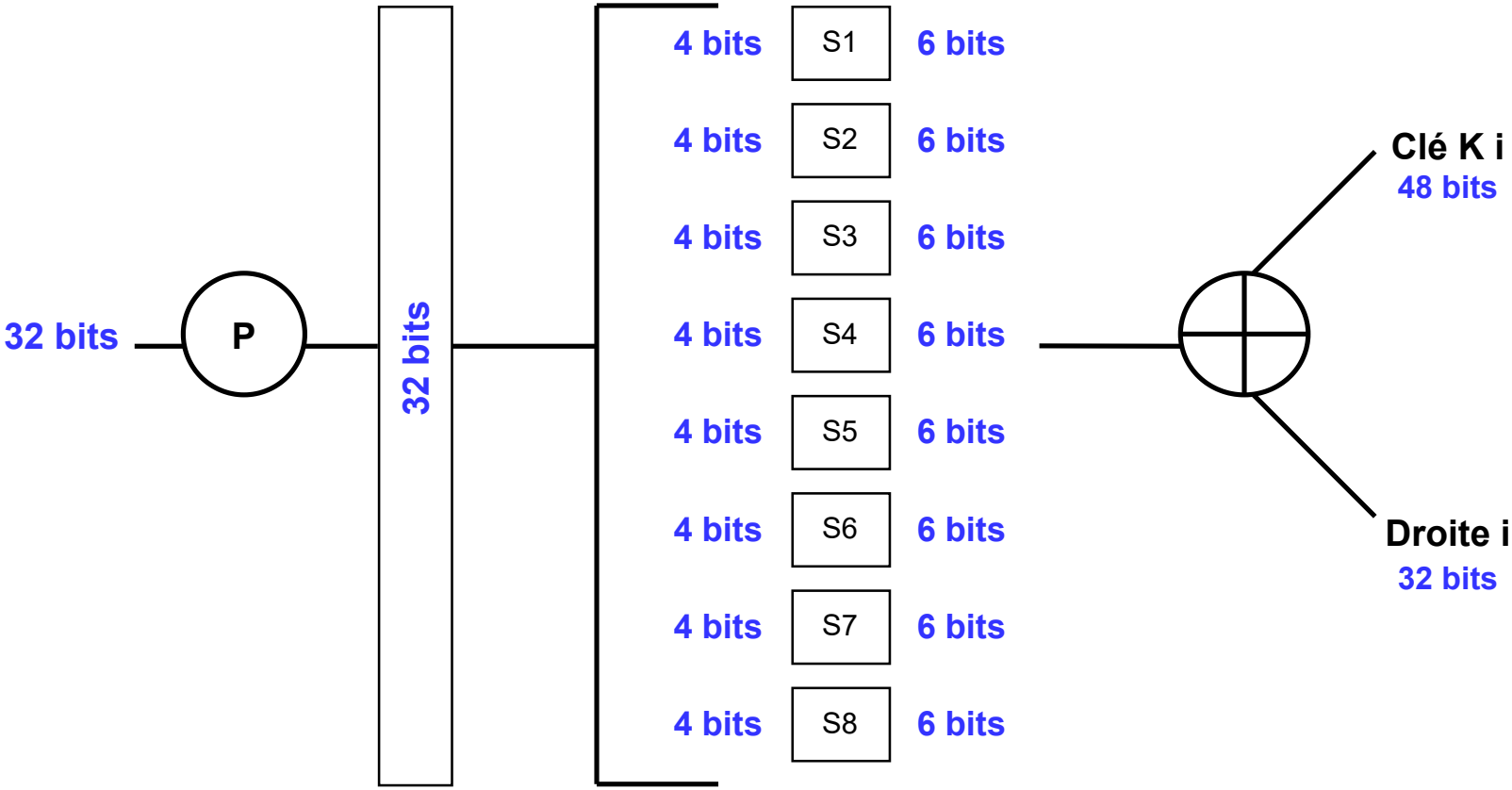
D.E.S. : Ip et Ip^{-1}

- Permutation initiale et finale, table 8×8

Initial Permutation							
58	50	42	34	26	18	10	02
60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06
64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01
59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05
63	55	47	39	31	23	15	07

Final Permutation							
40	08	48	16	56	24	64	32
39	07	47	15	55	23	63	31
38	06	46	14	54	22	62	30
37	05	45	13	53	21	61	29
36	04	44	12	52	20	60	28
35	03	43	11	51	19	59	27
34	02	42	10	50	18	58	26
33	01	41	09	49	17	57	25

D.E.S. : fonction f



D.E.S. : table d'expension

- Sous bloc de taille 32 bits

D.E.S. : table d'expension

- Sous bloc de taille 32 bits
- Sous clé de taille 48 bits

D.E.S. : table d'expension

- Sous bloc de taille 32 bits
- Sous clé de taille 48 bits
- Le XOR n'est pas possible : problème de taille !

D.E.S. : table d'expension

- Sous bloc de taille 32 bits
- Sous clé de taille 48 bits
- Le XOR n'est pas possible : problème de taille !

- Solution : la table d'expension

D.E.S. : table d'expension

- Sous bloc de taille 32 bits
- Sous clé de taille 48 bits
- Le XOR n'est pas possible : problème de taille !

- Solution : la table d'expension

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

D.E.S. : S-Box et P-Box

- 8 boîtes S (S-Box) qui prennent 6 bits en entrée, en sortent 4

D.E.S. : S-Box et P-Box

- 8 boîtes S (S-Box) qui prennent 6 bits en entrée, en sortent 4
- Table de 4×16 : chaque ligne est permutation des nombres de 0 à 15

D.E.S. : S-Box et P-Box

- 8 boîtes S (S-Box) qui prennent 6 bits en entrée, en sortent 4
- Table de 4×16 : chaque ligne est permutation des nombres de 0 à 15

- Par exemple 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

D.E.S. : S-Box et P-Box

- 8 boîtes S (S-Box) qui prennent 6 bits en entrée, en sortent 4
- Table de 4×16 : chaque ligne est permutation des nombres de 0 à 15

- Par exemple 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

- Les deux bits aux extrémités indiquent le numéro de ligne

D.E.S. : S-Box et P-Box

- 8 boîtes S (S-Box) qui prennent 6 bits en entrée, en sortent 4
- Table de 4×16 : chaque ligne est permutation des nombres de 0 à 15

- Par exemple 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

- Les deux bits aux extrémités indiquent le numéro de ligne
- Les quatre du milieu indiquent le numéro de colonne

D.E.S. : S-Box et P-Box

- 8 boîtes S (S-Box) qui prennent 6 bits en entrée, en sortent 4
- Table de 4×16 : chaque ligne est permutation des nombres de 0 à 15

- Par exemple 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

- Les deux bits aux extrémités indiquent le numéro de ligne
- Les quatre du milieu indiquent le numéro de colonne
- *Exemple* : 011001, ligne 1, colonne 12 $\rightarrow S_1(011000) = 9 = (1001)_2$

D.E.S. : S-Box et P-Box

- 8 boîtes S (S-Box) qui prennent 6 bits en entrée, en sortent 4
- Table de 4×16 : chaque ligne est permutation des nombres de 0 à 15

- Par exemple 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

- Les deux bits aux extrémités indiquent le numéro de ligne
- Les quatre du milieu indiquent le numéro de colonne
- *Exemple* : 011001, ligne 1, colonne 12 $\rightarrow S_1(011000) = 9 = (1001)_2$

- P-Box, boîtes P, sont juste des permutations

Déchiffrement

- Reprendre l'algorithme de chiffrement avec $K15$ au premier tour, puis $K14$ jusqu'à $K0$ au dernier tour.

Déchiffrement

- Reprendre l'algorithme de chiffrement avec $K15$ au premier tour, puis $K14$ jusqu'à $K0$ au dernier tour.

- ATTENTION ! D.E.S. a taille de clé trop faible pour une utilisation professionnelle

Chiffrement par flot

- But : concevoir suite chiffrante K qui semble aléatoire

Chiffrement par flot

- But : concevoir suite chiffrante K qui semble aléatoire
- Mais qui est déterministe et initialisée par une clé k

Chiffrement par flot

- But : concevoir suite chiffrante K qui semble aléatoire
- Mais qui est déterministe et initialisée par une clé k

$$M \oplus \text{suite-chiffrante}_K = C$$
$$C \oplus \text{suite-chiffrante}_K = M$$

Chiffrement par flot

- But : concevoir suite chiffrante K qui semble aléatoire
- Mais qui est déterministe et initialisée par une clé k

$$M \oplus \text{suite-chiffrante}_K = C$$
$$C \oplus \text{suite-chiffrante}_K = M$$

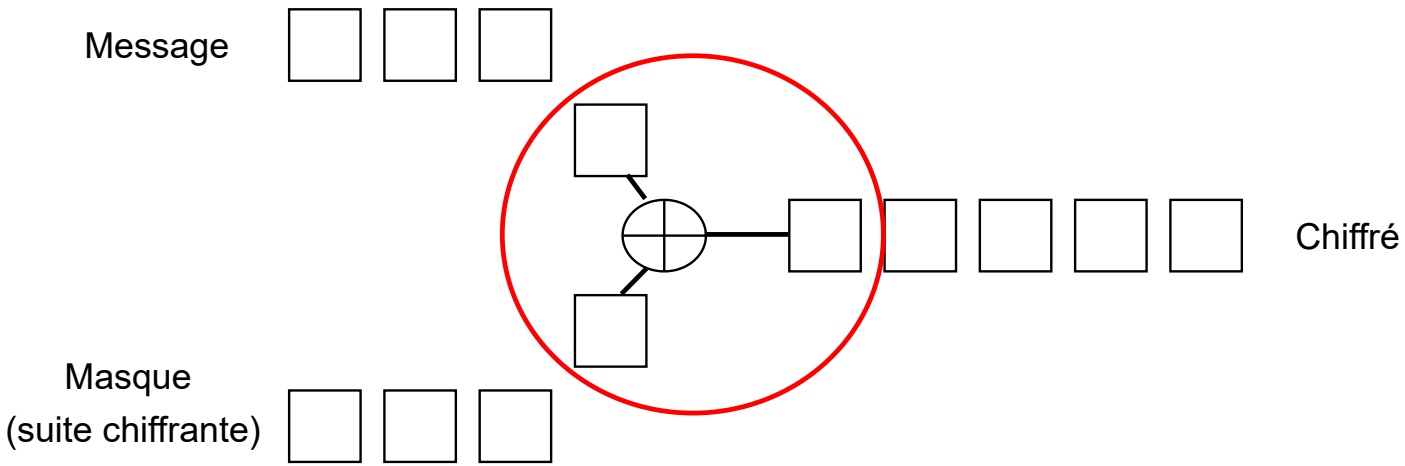
- Même initialisation $\rightarrow$ même suite chiffrante

Chiffrement par flot

- But : concevoir suite chiffrante K qui semble aléatoire
- Mais qui est déterministe et initialisée par une clé k

$$M \oplus \text{suite-chiffrante}_K = C$$
$$C \oplus \text{suite-chiffrante}_K = M$$

- Même initialisation $\rightarrow$ même suite chiffrante



ONE TIME PAD (ou chiffrement de Vernam)

M=	0	1	1	1	0	1	1	0	1	0
Suite K =	1	0	1	1	0	0	1	1	1	1
$C = M \oplus K =$	1	1	0	0	0	1	0	1	0	1
$M = C \oplus K =$	0	1	1	1	0	1	1	0	1	0

ONE TIME PAD (ou chiffrement de Vernam)

M=	0	1	1	1	0	1	1	0	1	0
Suite K =	1	0	1	1	0	0	1	1	1	1
$C = M \oplus K =$	1	1	0	0	0	1	0	1	0	1
$M = C \oplus K =$	0	1	1	1	0	1	1	0	1	0

- Chiffrement l'un des plus sûrs que l'on connait, il est incassable si utilisé dans les conditions suivantes :

ONE TIME PAD (ou chiffrement de Vernam)

M=	0	1	1	1	0	1	1	0	1	0
Suite K =	1	0	1	1	0	0	1	1	1	1
$C = M \oplus K =$	1	1	0	0	0	1	0	1	0	1
$M = C \oplus K =$	0	1	1	1	0	1	1	0	1	0

- Chiffrement l'un des plus sûrs que l'on connait, il est incassable si utilisé dans les conditions suivantes :
  - Clé K complètement indistinguable de l'aléatoire

ONE TIME PAD (ou chiffrement de Vernam)

M=	0	1	1	1	0	1	1	0	1	0
Suite K =	1	0	1	1	0	0	1	1	1	1
$C = M \oplus K =$	1	1	0	0	0	1	0	1	0	1
$M = C \oplus K =$	0	1	1	1	0	1	1	0	1	0

- Chiffrement l'un des plus sûrs que l'on connait, il est incassable si utilisé dans les conditions suivantes :
  - Clé K complètement indistinguable de l'aléatoire
  - Clé K de taille au moins égale au message (on ne doit pas répéter la clé)

ONE TIME PAD (ou chiffrement de Vernam)

M=	0	1	1	1	0	1	1	0	1	0
Suite K =	1	0	1	1	0	0	1	1	1	1
$C = M \oplus K =$	1	1	0	0	0	1	0	1	0	1
$M = C \oplus K =$	0	1	1	1	0	1	1	0	1	0

- Chiffrement l'un des plus sûrs que l'on connait, il est incassable si utilisé dans les conditions suivantes :
  - Clé K complètement indistinguable de l'aléatoire
  - Clé K de taille au moins égale au message (on ne doit pas répéter la clé)
  - On ne réutilisera pas la clé

ONE TIME PAD (ou chiffrement de Vernam)

M=	0	1	1	1	0	1	1	0	1	0
Suite K =	1	0	1	1	0	0	1	1	1	1
$C = M \oplus K =$	1	1	0	0	0	1	0	1	0	1
$M = C \oplus K =$	0	1	1	1	0	1	1	0	1	0

- Chiffrement l'un des plus sûrs que l'on connait, il est incassable si utilisé dans les conditions suivantes :
  - Clé K complètement indistinguable de l'aléatoire
  - Clé K de taille au moins égale au message (on ne doit pas répéter la clé)
  - On ne réutilisera pas la clé
- C'est le masque jetable (One Time Pad), ou chiffrement de Vernam si on utilise des lettres

ONE TIME PAD (ou chiffrement de Vernam)

M=	0	1	1	1	0	1	1	0	1	0
Suite K =	1	0	1	1	0	0	1	1	1	1
$C = M \oplus K =$	1	1	0	0	0	1	0	1	0	1
$M = C \oplus K =$	0	1	1	1	0	1	1	0	1	0

- Chiffrement l'un des plus sûrs que l'on connait, il est incassable si utilisé dans les conditions suivantes :
  - Clé K complètement indistinguable de l'aléatoire
  - Clé K de taille au moins égale au message (on ne doit pas répéter la clé)
  - On ne réutilisera pas la clé
- C'est le masque jetable (One Time Pad), ou chiffrement de Vernam si on utilise des lettres
- Chiffrement **parfait** en théorie de l'information

ONE TIME PAD (ou chiffrement de Vernam)

M=	0	1	1	1	0	1	1	0	1	0
Suite K =	1	0	1	1	0	0	1	1	1	1
$C = M \oplus K =$	1	1	0	0	0	1	0	1	0	1
$M = C \oplus K =$	0	1	1	1	0	1	1	0	1	0

- Chiffrement l'un des plus sûrs que l'on connait, il est incassable si utilisé dans les conditions suivantes :
  - Clé K complètement indistinguable de l'aléatoire
  - Clé K de taille au moins égale au message (on ne doit pas répéter la clé)
  - On ne réutilisera pas la clé
- C'est le masque jetable (One Time Pad), ou chiffrement de Vernam si on utilise des lettres
- Chiffrement **parfait** en théorie de l'information
- Inconvénient : il faut transmettre la clé de manière sûre

Rappels sur les probabilités

- Probabilités : c'est le rapport entre le nombre de cas favorables et le nombre de cas possibles ($0 \leq \textit{proba} \leq 1$)

Rappels sur les probabilités

- Probabilités : c'est le rapport entre le nombre de cas favorables et le nombre de cas possibles ($0 \leq proba \leq 1$)
- Deux événements S_1, S_2 sont **indépendants** si et seulement si

$$p(S_1 \cap S_2) = p(S_1) \times p(S_2)$$

Rappels sur les probabilités

- Probabilités : c'est le rapport entre le nombre de cas favorables et le nombre de cas possibles ($0 \leq proba \leq 1$)
- Deux événements S_1, S_2 sont **indépendants** si et seulement si

$$p(S_1 \cap S_2) = p(S_1) \times p(S_2)$$

- Probabilité **conditionnelle** : soient deux événements S_1 et S_2 avec $p(S_2) > 0$. La probabilité de S_1 **sachant** S_2 réalisé est

$$p(S_1|S_2) = p(S_1 \cap S_2)/p(S_2)$$

Rappels sur les probabilités

- Probabilités : c'est le rapport entre le nombre de cas favorables et le nombre de cas possibles ($0 \leq proba \leq 1$)
- Deux événements S_1, S_2 sont **indépendants** si et seulement si

$$p(S_1 \cap S_2) = p(S_1) \times p(S_2)$$

- Probabilité **conditionnelle** : soient deux événements S_1 et S_2 avec $p(S_2) > 0$. La probabilité de S_1 **sachant** S_2 réalisé est

$$p(S_1|S_2) = p(S_1 \cap S_2)/p(S_2)$$

- Si S_1 et S_2 sont indépendants

$$p(S_1|S_2) = \frac{p(S_1) \times p(S_2)}{p(S_2)} = p(S_1)$$

Rappels sur les probabilités

- Probabilités : c'est le rapport entre le nombre de cas favorables et le nombre de cas possibles ($0 \leq proba \leq 1$)
- Deux événements S_1, S_2 sont **indépendants** si et seulement si

$$p(S_1 \cap S_2) = p(S_1) \times p(S_2)$$

- Probabilité **conditionnelle** : soient deux événements S_1 et S_2 avec $p(S_2) > 0$. La probabilité de S_1 **sachant** S_2 réalisé est

$$p(S_1|S_2) = p(S_1 \cap S_2)/p(S_2)$$

- Si S_1 et S_2 sont indépendants

$$p(S_1|S_2) = \frac{p(S_1) \times p(S_2)}{p(S_2)} = p(S_1)$$

- Théorème de Bayes : $p(S_1|S_2) \times p(S_2) = p(S_2|S_1) \times p(S_1)$ (les deux quantités valent $p(S_1 \cap S_2)$)

Entropie : introduction

- Une mesure de la quantité d'information d'une source (=langue, signal, texte, $\cdots$) a été proposée par Shannon

Entropie : introduction

- Une mesure de la quantité d'information d'une source (=langue, signal, texte, $\cdots$) a été proposée par Shannon

- On dispose d'une source X (variable aléatoire) qui prend les valeurs x comme états possibles

Entropie : introduction

- Une mesure de la quantité d'information d'une source (=langue, signal, texte, $\cdots$) a été proposée par Shannon

- On dispose d'une source X (variable aléatoire) qui prend les valeurs x comme états possibles

- La formule de l'**entropie** (mesure binaire de la quantité d'information) est donnée par

$$H(X) = -\sum_{x} p(X = x) log_2(p(X = x))$$

où log_2 est le logarithme en base 2

Entropie : introduction

- Une mesure de la quantité d'information d'une source (=langue, signal, texte, $\cdots$) a été proposée par Shannon

- On dispose d'une source X (variable aléatoire) qui prend les valeurs x comme états possibles

- La formule de l'**entropie** (mesure binaire de la quantité d'information) est donnée par

$$H(X) = -\sum_{x} p(X = x) log_2(p(X = x))$$

où log_2 est le logarithme en base 2

- Par convention $p_i log_2(p_i) = 0$ lorsque $p_i = 0$

Entropie : introduction

- Une mesure de la quantité d'information d'une source (=langue, signal, texte, $\cdots$) a été proposée par Shannon

- On dispose d'une source X (variable aléatoire) qui prend les valeurs x comme états possibles

- La formule de l'**entropie** (mesure binaire de la quantité d'information) est donnée par

$$H(X) = -\sum_{x} p(X = x) log_2(p(X = x))$$

où log_2 est le logarithme en base 2

- Par convention $p_i log_2(p_i) = 0$ lorsque $p_i = 0$
- L'entropie est **maximale** si les éléments apparaissent de façon équiprobable

Entropie : introduction

- Une mesure de la quantité d'information d'une source (=langue, signal, texte, $\cdots$) a été proposée par Shannon

- On dispose d'une source X (variable aléatoire) qui prend les valeurs x comme états possibles

- La formule de l'**entropie** (mesure binaire de la quantité d'information) est donnée par

$$H(X) = -\sum_{x} p(X = x) log_2(p(X = x))$$

où log_2 est le logarithme en base 2

- Par convention $p_i log_2(p_i) = 0$ lorsque $p_i = 0$
- L'entropie est **maximale** si les éléments apparaissent de façon équiprobable
- Plus de détails dans les prochains cours...

Sécurité parfaite

- Un chiffrement est difficile à casser lorsque la connaissance du chiffré C n'apporte aucune information sur le clair M

Sécurité parfaite

- Un chiffrement est difficile à casser lorsque la connaissance du chiffré C n'apporte aucune information sur le clair M
- Pour un système de chiffrement : $H(K|C) = H(M) + H(K) - H(C)$

Sécurité parfaite

- Un chiffrement est difficile à casser lorsque la connaissance du chiffré C n'apporte aucune information sur le clair M
- Pour un système de chiffrement : $H(K|C) = H(M) + H(K) - H(C)$
- Un système de chiffrement sera dit **parfait** si

$$H(M|C) = H(M)$$

Sécurité parfaite

- Un chiffrement est difficile à casser lorsque la connaissance du chiffré C n'apporte aucune information sur le clair M
- Pour un système de chiffrement : $H(K|C) = H(M) + H(K) - H(C)$
- Un système de chiffrement sera dit **parfait** si

$$H(M|C) = H(M)$$

- Avec théorie de l'information on a identifié deux grands principes de construction :

Sécurité parfaite

- Un chiffrement est difficile à casser lorsque la connaissance du chiffré C n'apporte aucune information sur le clair M
- Pour un système de chiffrement : $H(K|C) = H(M) + H(K) - H(C)$
- Un système de chiffrement sera dit **parfait** si

$$H(M|C) = H(M)$$

- Avec théorie de l'information on a identifié deux grands principes de construction :
  - Principe de **confusion** : la relation entre la clé et le chiffré doit être la plus complexe possible

Sécurité parfaite

- Un chiffrement est difficile à casser lorsque la connaissance du chiffré C n'apporte aucune information sur le clair M
- Pour un système de chiffrement : $H(K|C) = H(M) + H(K) - H(C)$
- Un système de chiffrement sera dit **parfait** si

$$H(M|C) = H(M)$$

- Avec théorie de l'information on a identifié deux grands principes de construction :
  - Principe de **confusion** : la relation entre la clé et le chiffré doit être la plus complexe possible
  - Principe de **diffusion** : la dépendance entre les bits de sorties et les bits d'entrées doit être minimale

Sécurité parfaite

- Un chiffrement est difficile à casser lorsque la connaissance du chiffré C n'apporte aucune information sur le clair M
- Pour un système de chiffrement : $H(K|C) = H(M) + H(K) - H(C)$
- Un système de chiffrement sera dit **parfait** si

$$H(M|C) = H(M)$$

- Avec théorie de l'information on a identifié deux grands principes de construction :
  - Principe de **confusion** : la relation entre la clé et le chiffré doit être la plus complexe possible
  - Principe de **diffusion** : la dépendance entre les bits de sorties et les bits d'entrées doit être minimale

- En utilisant de la substitution et de la permutation on se conforme à ces principes

Schéma de confusion/diffusion (Shannon)

- Permutation P **inversible**

Schéma de confusion/diffusion (Shannon)

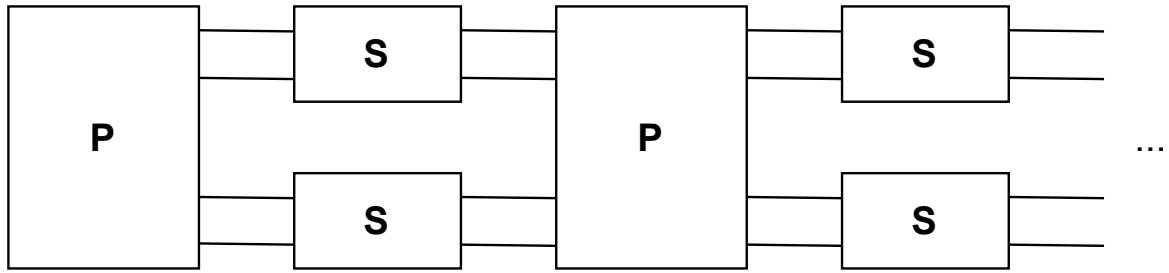
- Permutation P **inversible**
- Substitution S **inversible**

Schéma de confusion/diffusion (Shannon)

- Permutation P **inversible**
- Substitution S **inversible**
- Si P, S pas inversibles on ne peut pas déchiffrer

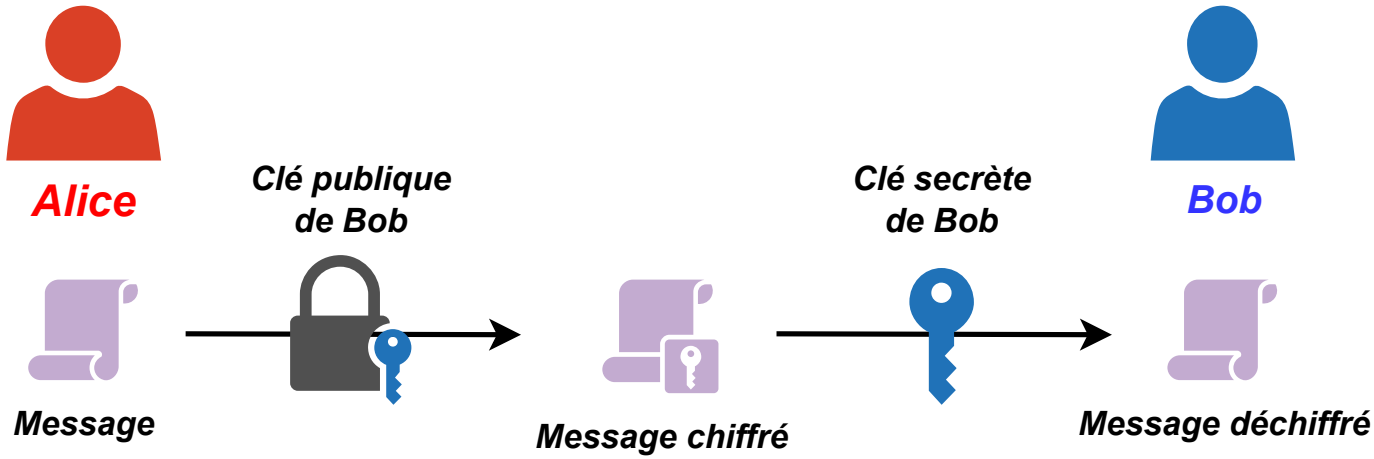
Schéma de confusion/diffusion (Shannon)

- Permutation P **inversible**
- Substitution S **inversible**
- Si P, S pas inversibles on ne peut pas déchiffrer



Sommaire

Cryptographie asymétrique, ou à clé publique



La cryptographie à clé publique

- Une partie de la clé est maintenant connue de tous : la clé publique pk

La cryptographie à clé publique

- Une partie de la clé est maintenant connue de tous : la clé publique pk
- L'autre partie reste secrète : la clé secrète sk

La cryptographie à clé publique

- Une partie de la clé est maintenant connue de tous : la clé publique pk
- L'autre partie reste secrète : la clé secrète sk
- Avantage : il n'y a plus besoin de partager un secret commun

La cryptographie à clé publique

- Une partie de la clé est maintenant connue de tous : la clé publique pk
- L'autre partie reste secrète : la clé secrète sk
- Avantage : il n'y a plus besoin de partager un secret commun
- Important : il est difficile de retrouver sk à partir de pk

La cryptographie à clé publique

- Une partie de la clé est maintenant connue de tous : la clé publique pk
- L'autre partie reste secrète : la clé secrète sk
- Avantage : il n'y a plus besoin de partager un secret commun
- Important : il est difficile de retrouver sk à partir de pk
- Un schéma connu : R.S.A.

Division euclidienne

Division euclidienne

Théorème : pour tout couple d'entiers (a, b) appartenant à $\mathbb{Z} \times \mathbb{Z}^*$, il existe un unique couple d'entiers (q, r) tel que

$$a = b \cdot q + r \quad \text{et } 0 \leq r < |b| :$$

L'entier r (resp. q) est appelé le **reste** (resp. le **quotient**) de la division euclidienne de a par b.

Division euclidienne

Division euclidienne

Théorème : pour tout couple d'entiers (a, b) appartenant à $\mathbb{Z} \times \mathbb{Z}^*$, il existe un unique couple d'entiers (q, r) tel que

$$a = b \cdot q + r \quad \text{et } 0 \leq r < |b| :$$

L'entier r (resp. q) est appelé le **reste** (resp. le **quotient**) de la division euclidienne de a par b.

- *Preuve* : admise.

Division euclidienne

Division euclidienne

Théorème : pour tout couple d'entiers (a, b) appartenant à $\mathbb{Z} \times \mathbb{Z}^*$, il existe un unique couple d'entiers (q, r) tel que

$$a = b \cdot q + r \quad \text{et } 0 \leq r < |b| :$$

L'entier r (resp. q) est appelé le **reste** (resp. le **quotient**) de la division euclidienne de a par b.

- *Preuve* : admise.
- *Exemple* :

$$26 = 5 \cdot 5 + 1$$
$$31 = 3 \cdot 10 + 1$$
$$-52 = 6 \cdot (-7) - 3$$

PGCD

PGCD

Définition : soient $a_1, \cdots, a_k$ des entiers relatifs. Le PGCD (plus grand commun diviseur) de $a_1, \cdots, a_k$ est le plus grand entier positif d divisant tous les a_i, $i = 1, \cdots, k$.

PGCD

PGCD

Définition : soient $a_1, \cdots, a_k$ des entiers relatifs. Le PGCD (plus grand commun diviseur) de $a_1, \cdots, a_k$ est le plus grand entier positif d divisant tous les a_i, $i = 1, \cdots, k$.

- *Exemple* :

$$pgcd(8, 2) = 2$$
$$pgcd(3, 5) = 1$$
$$pgcd(3, 6, 9) = 3$$

Propriétés du PGCD

- $pgcd(a_1, \cdots, a_k) = pgcd(a_1, d)$ avec $d = pgcd(a_2, \cdots, a_k)$

Propriétés du PGCD

- $pgcd(a_1, \cdots, a_k) = pgcd(a_1, d)$ avec $d = pgcd(a_2, \cdots, a_k)$
- $pgcd(a, b) = pgcd(|a|, |b|)$

Propriétés du PGCD

- $pgcd(a_1, \cdots, a_k) = pgcd(a_1, d)$ avec $d = pgcd(a_2, \cdots, a_k)$
- $pgcd(a, b) = pgcd(|a|, |b|)$
- Le pgcd est **commutatif** : $pgcd(a, b) = pgcd(b, a)$

Propriétés du PGCD

- $pgcd(a_1, \cdots, a_k) = pgcd(a_1, d)$ avec $d = pgcd(a_2, \cdots, a_k)$
- $pgcd(a, b) = pgcd(|a|, |b|)$
- Le pgcd est **commutatif** : $pgcd(a, b) = pgcd(b, a)$
- $pgcd(a, 0) = |a|$

Propriétés du PGCD

- $pgcd(a_1, \cdots, a_k) = pgcd(a_1, d)$ avec $d = pgcd(a_2, \cdots, a_k)$
- $pgcd(a, b) = pgcd(|a|, |b|)$
- Le pgcd est **commutatif** : $pgcd(a, b) = pgcd(b, a)$
- $pgcd(a, 0) = |a|$
- Si $a = bq + r$ est la division euclidienne de a par b alors $pgcd(a, b) = pgcd(b, r)$

Propriétés du PGCD

- $pgcd(a_1, \cdots, a_k) = pgcd(a_1, d)$ avec $d = pgcd(a_2, \cdots, a_k)$
- $pgcd(a, b) = pgcd(|a|, |b|)$
- Le pgcd est **commutatif** : $pgcd(a, b) = pgcd(b, a)$
- $pgcd(a, 0) = |a|$
- Si $a = bq + r$ est la division euclidienne de a par b alors $pgcd(a, b) = pgcd(b, r)$

Comment calculer le pgcd ?

Algorithme d'Euclide

Algorithm Algorithme d'Euclide

Input : deux entiers a et b

Output : le pgcd de a et b

1: **while** $b \neq 0$ **do**
2: Faire la division euclidenne de a par b : $a = b \cdot q + r$
3: $a \leftarrow b$
4: $b \leftarrow r$
5: **end while**
6: **return** $|a|$

Complexité : $O((l(a) + l(b))^2)$ avec $l(x)$ le nombre de bits de x.

On verra plus tard la notation O.

Algorithme d'Euclide : exemples

- On applique l'algorithme d'Euclide avec en entrée 68 et 3

Algorithme d'Euclide : exemples

- On applique l'algorithme d'Euclide avec en entrée 68 et 3

$$68 = 3 \cdot 22 + 2$$
$$3 = 2 \cdot 1 + 1$$
$$2 = 1 \cdot 2 + 0$$

Algorithme d'Euclide : exemples

- On applique l'algorithme d'Euclide avec en entrée 68 et 3

$$\begin{aligned} 68 &= 3 \cdot 22 + 2 \\ 3 &= 2 \cdot 1 + 1 \\ 2 &= 1 \cdot 2 + 0 \end{aligned}$$

Sortie : $pgcd(68, 3) = 1$

Algorithme d'Euclide : exemples

- On applique l'algorithme d'Euclide avec en entrée 68 et 3

$$\begin{aligned} 68 &= 3 \cdot 22 + 2 \\ 3 &= 2 \cdot 1 + 1 \\ 2 &= 1 \cdot 2 + 0 \end{aligned}$$

Sortie : $pgcd(68, 3) = 1$

- On applique l'algorithme d'Euclide avec en entrée 112 et 6

Algorithme d'Euclide : exemples

- On applique l'algorithme d'Euclide avec en entrée 68 et 3

$$\begin{aligned} 68 &= 3 \cdot 22 + 2 \\ 3 &= 2 \cdot 1 + 1 \\ 2 &= 1 \cdot 2 + 0 \end{aligned}$$

Sortie : $pgcd(68, 3) = 1$

- On applique l'algorithme d'Euclide avec en entrée 112 et 6

$$\begin{aligned} 112 &= 6 \cdot 18 + 4 \\ 6 &= 4 \cdot 1 + 2 \\ 4 &= 2 \cdot 2 + 0 \end{aligned}$$

Algorithme d'Euclide : exemples

- On applique l'algorithme d'Euclide avec en entrée 68 et 3

$$\begin{aligned} 68 &= 3 \cdot 22 + 2 \\ 3 &= 2 \cdot 1 + 1 \\ 2 &= 1 \cdot 2 + 0 \end{aligned}$$

Sortie : $pgcd(68, 3) = 1$

- On applique l'algorithme d'Euclide avec en entrée 112 et 6

$$\begin{aligned} 112 &= 6 \cdot 18 + 4 \\ 6 &= 4 \cdot 1 + 2 \\ 4 &= 2 \cdot 2 + 0 \end{aligned}$$

Sortie : $pgcd(112, 6) = 2$

Coefficients de Bézout

Définition et proposition

Pour tout couple d'entiers a et b, il existe un couple d'entiers u et v, appelés **coefficients de Bezout**, et vérifiant

$$a \cdot u + b \cdot v = pgcd(a, b)$$

Coefficients de Bézout

Définition et proposition

Pour tout couple d'entiers a et b, il existe un couple d'entiers u et v, appelés **coefficients de Bezout**, et vérifiant

$$a \cdot u + b \cdot v = pgcd(a, b)$$

Exemples :

$$1 = 3 \cdot 23 - 1 \cdot 68$$
$$2 = 6 \cdot 19 - 1 \cdot 112$$

ATTENTION ! Les coefficients ne sont pas uniques !

Coefficients de Bézout

Définition et proposition

Pour tout couple d'entiers a et b, il existe un couple d'entiers u et v, appelés **coefficients de Bezout**, et vérifiant

$$a \cdot u + b \cdot v = pgcd(a, b)$$

Exemples :

$$1 = 3 \cdot 23 - 1 \cdot 68$$
$$2 = 6 \cdot 19 - 1 \cdot 112$$

ATTENTION ! Les coefficients ne sont pas uniques !

Comment calculer les coefficients de Bézout ?

Algorithme d'Euclide étendu

Algorithm Algorithme d'Euclide étendu

Input : deux entiers a et b

Output : le triplet u, v, d avec $a \cdot u + b \cdot v = d = pgcd(a, b)$

1: $(A, B) \leftarrow (a, b)$
2: $(u_0, u_1) \leftarrow (1, 0)$
3: $(v_0, v_1) \leftarrow (0, 1)$
4: **while** $B \neq 0$ **do**
5: Faire la division euclidenne de A par B : $A = B \cdot q + r$
6: $(A, B) \leftarrow (B, r)$
7: $(u_0, u_1) \leftarrow (u_1, u_0 - q \cdot u_1)$
8: $(v_0, v_1) \leftarrow (v_1, v_0 - q \cdot v_1)$
9: **end while**
10: **return** A, u_0, v_0

A chaque début ou fin de boucle, on a $a \cdot u_0 + b \cdot v_0 = A$, $a \cdot u_1 + b \cdot v_1 = B$. A la fin de la dernière boucle, on a $A = pgcd(a, b), B = 0$

Nombres premiers et premiers entre eux

Définition et théorème

Deux entiers a et b sont **premiers entre eux** si $pgcd(a, b) = 1$. Deux entiers a et b sont premiers entre eux si et seulement si il existe deux entiers u et v tels que $au + bv = 1$.

Nombres premiers et premiers entre eux

Définition et théorème

Deux entiers a et b sont **premiers entre eux** si $pgcd(a, b) = 1$. Deux entiers a et b sont premiers entre eux si et seulement si il existe deux entiers u et v tels que $au + bv = 1$.

- *Preuve* : admise.

Nombres premiers et premiers entre eux

Définition et théorème

Deux entiers a et b sont **premiers entre eux** si $pgcd(a, b) = 1$. Deux entiers a et b sont premiers entre eux si et seulement si il existe deux entiers u et v tels que $au + bv = 1$.

- *Preuve* : admise.

Définition

Un entier n est dit premier si et seulement si

1. $n \geq 2$
2. n n'est divisible que par $1, -1, n$ et $-n$

Nombres premiers et premiers entre eux

Définition et théorème

Deux entiers a et b sont **premiers entre eux** si $pgcd(a, b) = 1$. Deux entiers a et b sont premiers entre eux si et seulement si il existe deux entiers u et v tels que $au + bv = 1$.

- *Preuve* : admise.

Définition

Un entier n est dit premier si et seulement si

1. $n \geq 2$
2. n n'est divisible que par $1, -1, n$ et $-n$

- *Exemple* : 2,3,5,7,11,13,17,19 $\cdots$

Nombres premiers et premiers entre eux

Définition et théorème

Deux entiers a et b sont **premiers entre eux** si $pgcd(a, b) = 1$. Deux entiers a et b sont premiers entre eux si et seulement si il existe deux entiers u et v tels que $au + bv = 1$.

- *Preuve* : admise.

Définition

Un entier n est dit premier si et seulement si

1. $n \geq 2$
2. n n'est divisible que par $1, -1, n$ et $-n$

- *Exemple* : 2,3,5,7,11,13,17,19 $\cdots$

- ATTENTION ! Les nombres pairs (sauf 2) ne sont pas premiers !

Quelques propositions

Théorème

Il existe une infinité de nombres premiers.

Quelques propositions

Théorème

Il existe une infinité de nombres premiers.

- *Remarque* : les nombres premiers sont de plus en plus rares lorsque leur taille augmente.

Quelques propositions

Théorème

Il existe une infinité de nombres premiers.

- *Remarque* : les nombres premiers sont de plus en plus rares lorsque leur taille augmente.

Proposition

Lemme d'Euclide : si p est un nombre premier qui divise le produit $a \cdot b$, alors p divise a ou b.

Quelques propositions

Théorème

Il existe une infinité de nombres premiers.

- *Remarque* : les nombres premiers sont de plus en plus rares lorsque leur taille augmente.

Proposition

Lemme d'Euclide : si p est un nombre premier qui divise le produit $a \cdot b$, alors p divise a ou b.

Théorème

Lemme de Gauss : soient a, b, c trois entiers tels que $a|bc$. Si a est premier avec b, alors a divise c.

Génération de nombres premiers et test de primalité

- Pour construire un nombre premier de taille fixée : on tire au hasard un nombre et on teste sa primalité

Génération de nombres premiers et test de primalité

- Pour construire un nombre premier de taille fixée : on tire au hasard un nombre et on teste sa primalité

- Il existe pour ça de nombreux tests : le premier se base sur le théorème de Fermat

Génération de nombres premiers et test de primalité

- Pour construire un nombre premier de taille fixée : on tire au hasard un nombre et on teste sa primalité

- Il existe pour ça de nombreux tests : le premier se base sur le théorème de Fermat

Théorème de Fermat

Soit p un entier premier et $a \in \mathbb{Z}$ tel que $pgcd(a, p) = 1$. Alors

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

Génération de nombres premiers et test de primalité

- Pour construire un nombre premier de taille fixée : on tire au hasard un nombre et on teste sa primalité

- Il existe pour ça de nombreux tests : le premier se base sur le théorème de Fermat

Théorème de Fermat

Soit p un entier premier et $a \in \mathbb{Z}$ tel que $pgcd(a, p) = 1$. Alors

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

- Le test de Fermat donne une condition nécessaire mais pas suffisante ! N premier $\implies a^{N-1} \mod N = 1$, $a^{N-1} \mod N = 1 \not\implies N$ premier, $a^{N-1} \mod N \neq 1 \implies N$ non premier

Test de Fermat

Algorithm Test de primalité de Fermat

Input : un entier N, un entier k

Output : « N est probablement premier » ou « N n'est pas premier »

1: **for** $i = 0$ à k **do**
2: Choisir $a \leq N$ aléatoirement
3: Calculer $s = a^{N-1} \mod N$
4: **if** $s \neq 1 \mod N$ **then**
5: **return** « N n'est pas premier »
6: **end if**
7: **end for**
8: **return** « N est probablement premier »

Test de Fermat

Algorithm Test de primalité de Fermat

Input : un entier N, un entier k

Output : « N est probablement premier » ou « N n'est pas premier »

1: **for** $i = 0$ à k **do**
2: Choisir $a \leq N$ aléatoirement
3: Calculer $s = a^{N-1} \mod N$
4: **if** $s \neq 1 \mod N$ **then**
5: **return** « N n'est pas premier »
6: **end if**
7: **end for**
8: **return** « N est probablement premier »

Exemple : pour $N = 341$, $a = 2$ et on a $2^{340} = 1 \mod 341$.
L'algorithme répond « probablement premier », mais $341 = 13 \times 11$!

Test de Fermat

Algorithm Test de primalité de Fermat

Input : un entier N, un entier k

Output : « N est probablement premier » ou « N n'est pas premier »

1: **for** $i = 0$ à k **do**
2: Choisir $a \leq N$ aléatoirement
3: Calculer $s = a^{N-1} \mod N$
4: **if** $s \neq 1 \mod N$ **then**
5: **return** « N n'est pas premier »
6: **end if**
7: **end for**
8: **return** « N est probablement premier »

Exemple : pour $N = 341$, $a = 2$ et on a $2^{340} = 1 \mod 341$.
L'algorithme répond « probablement premier », mais $341 = 13 \times 11$!

- Les **nombres de Carmichael** sont des entiers composés qui passent toujours le test de primalité de Fermat !

Petit théorème de Fermat (version forte)

Petit théorème de Fermat (version forte)

Soit n un entier impair. On écrit $n-1 = 2^k q$ avec q impair. Quelque soit b tel que $pgcd(n,b) = 1$, si n est premier alors on a une des deux conditions suivantes

1. $b^q \equiv 1 \mod n$
2. $b^{2^i q} \equiv -1 \mod n$ pour $0 \leq i \leq k-1$

Petit théorème de Fermat (version forte)

Petit théorème de Fermat (version forte)

Soit n un entier impair. On écrit $n-1 = 2^k q$ avec q impair. Quelque soit b tel que $pgcd(n,b) = 1$, si n est premier alors on a une des deux conditions suivantes

1. $b^q \equiv 1 \mod n$
2. $b^{2^i q} \equiv -1 \mod n$ pour $0 \leq i \leq k-1$

- De ce théorème on construit un algorithme (probabiliste) de primalité plus ≪ fort ≫ que le précédent

Petit théorème de Fermat (version forte)

Petit théorème de Fermat (version forte)

Soit n un entier impair. On écrit $n-1 = 2^k q$ avec q impair. Quelque soit b tel que $pgcd(n,b) = 1$, si n est premier alors on a une des deux conditions suivantes

1. $b^q \equiv 1 \mod n$
2. $b^{2^i q} \equiv -1 \mod n$ pour $0 \leq i \leq k-1$

- De ce théorème on construit un algorithme (probabiliste) de primalité plus « fort » que le précédent
- C'est un algorithme de Monté-Carlo biaisé vers le faux (non-premier)

Petit théorème de Fermat (version forte)

Petit théorème de Fermat (version forte)

Soit n un entier impair. On écrit $n-1 = 2^k q$ avec q impair. Quelque soit b tel que $pgcd(n,b) = 1$, si n est premier alors on a une des deux conditions suivantes

1. $b^q \equiv 1 \mod n$
2. $b^{2^i q} \equiv -1 \mod n$ pour $0 \leq i \leq k-1$

- De ce théorème on construit un algorithme (probabiliste) de primalité plus « fort » que le précédent
- C'est un algorithme de Monté-Carlo biaisé vers le faux (non-premier)
- La probabilité d'erreur dans l'autre cas est $p_1 \leq \frac{1}{4}$

Petit théorème de Fermat (version forte)

Petit théorème de Fermat (version forte)

Soit n un entier impair. On écrit $n-1 = 2^k q$ avec q impair. Quelque soit b tel que $pgcd(n,b) = 1$, si n est premier alors on a une des deux conditions suivantes

1. $b^q \equiv 1 \mod n$
2. $b^{2^i q} \equiv -1 \mod n$ pour $0 \leq i \leq k-1$

- De ce théorème on construit un algorithme (probabiliste) de primalité plus « fort » que le précédent
- C'est un algorithme de Monté-Carlo biaisé vers le faux (non-premier)
- La probabilité d'erreur dans l'autre cas est $p_1 \leq \frac{1}{4}$
- En effectuant au plus k appels au test, la probabilité d'erreur est $p_k \leq 4^{-k}$

Test Miller Rabin

Algorithm Test de primalité de Miller Rabin

Input : N un entier impair

Output : Retourne « non premier » ou « probablement premier »

1: Calculer k et q tels que $N-1=2^k q$ avec q impair
2: Choisir b premier avec N aléatoirement dans $]1, \cdots, N-1[$
3: **if** $b^q - 1 \equiv 0 \mod N$ ou $b^q + 1 \equiv 0 \mod N$ ou $b^{2q} + 1 \equiv 0 \mod N$ ou $\cdots$ ou $b^{2^{k-1}q} + 1 \equiv 0 \mod N$ **then**
4: Retourner « probablement premier »
5: **else**
6: Retourner « non premier »
7: **end if**

Décomposition en facteurs premiers

Théorème

Théorème fondamental de l'arithmétique : tout nombre entier positif et ≥ 2 peut s'écrire sous la forme d'un produit fini de nombres premiers, la décomposition étant unique, à l'ordre près des facteurs premiers.

Décomposition en facteurs premiers

Théorème

Théorème fondamental de l'arithmétique : tout nombre entier positif et ≥ 2 peut s'écrire sous la forme d'un produit fini de nombres premiers, la décomposition étant unique, à l'ordre près des facteurs premiers.

- *Remarque* : factoriser un grand entier est considéré aujourd'hui comme un problème difficile. Ce problème est à la base de nombreux protocoles cryptographiques dont RSA.

Indicatrice d'Euler

Définition

Soit n un entier. Le nombre de nombres premiers avec n compris entre 1 et $n-1$ est noté $\phi(n)$. La fonction ϕ s'appelle l'indicatrice d'Euler.

Indicatrice d'Euler

Définition

Soit n un entier. Le nombre de nombres premiers avec n compris entre 1 et $n-1$ est noté $\phi(n)$. La fonction ϕ s'appelle l'indicatrice d'Euler.

- Plus formellement, $\phi(n) = \#\{m \in \{1, \cdots, n-1\} | pgcd(m, n) = 1\}$

Indicatrice d'Euler

Définition

Soit n un entier. Le nombre de nombres premiers avec n compris entre 1 et $n-1$ est noté $\phi(n)$. La fonction ϕ s'appelle l'indicatrice d'Euler.

- Plus formellement, $\phi(n) = \#\{m \in \{1, \cdots, n-1\} | pgcd(m, n) = 1\}$
- Si p est premier, alors $\phi(p) = p - 1$

Indicatrice d'Euler

Définition

Soit n un entier. Le nombre de nombres premiers avec n compris entre 1 et $n-1$ est noté $\phi(n)$. La fonction ϕ s'appelle l'indicatrice d'Euler.

- Plus formellement, $\phi(n) = \#\{m \in \{1, \cdots, n-1\} \,|pgcd(m,n) = 1\}$
- Si p est premier, alors $\phi(p) = p - 1$
- Si p est premier et α un entier positif, alors

$$\phi(p^\alpha) = (p-1)p^{\alpha-1} = p^\alpha - p^{\alpha-1}$$

Indicatrice d'Euler

Définition

Soit n un entier. Le nombre de nombres premiers avec n compris entre 1 et $n-1$ est noté $\phi(n)$. La fonction ϕ s'appelle l'indicatrice d'Euler.

- Plus formellement, $\phi(n) = \#\{m \in \{1, \cdots, n-1\} \,|pgcd(m,n) = 1\}$
- Si p est premier, alors $\phi(p) = p - 1$
- Si p est premier et α un entier positif, alors

$$\phi(p^\alpha) = (p-1)p^{\alpha-1} = p^\alpha - p^{\alpha-1}$$

- Si la décomposition en facteurs premiers de n est donnée par $n = p_1^{k_1} \cdot p_2^{k_2} \cdots p_l^{k_l}$ alors

$$\phi(n) = \phi(p_1^{k_1}) \cdots \phi(p_l^{k_l}) = (p_1 - 1)p_1^{k_1-1} \cdots (p_l - 1)p_l^{k_l-1}$$

Indicatrice d'Euler : exemples

- $$\phi(2) = 1$$

Indicatrice d'Euler : exemples

- $$\phi(2) = 1$$
- $$\phi(5) = 4$$

Indicatrice d'Euler : exemples

- $$\phi(2) = 1$$
- $$\phi(5) = 4$$
- $$\phi(8) = \phi(2^3) = (2-1) \cdot 2^{3-1} = 4$$

Indicatrice d'Euler : exemples

- $$\phi(2) = 1$$
- $$\phi(5) = 4$$
- $$\phi(8) = \phi(2^3) = (2-1) \cdot 2^{3-1} = 4$$
- $$\phi(15) = \phi(3)\phi(5) = 2 \cdot 4 = 8$$

Indicatrice d'Euler : exemples

- $$\phi(2) = 1$$
- $$\phi(5) = 4$$
- $$\phi(8) = \phi(2^3) = (2-1) \cdot 2^{3-1} = 4$$
- $$\phi(15) = \phi(3)\phi(5) = 2 \cdot 4 = 8$$
- $$\phi(144) = \phi(16 \cdot 9) = \phi(4^2 \cdot 3^2) = 48$$

Ensemble $\mathbb{Z}/n\mathbb{Z}$

Définition

Soient a et n deux entiers. La **classe résiduelle** de a modulo n, notée $\overline{a}$, est l'ensemble $\overline{a} = a + n\mathbb{Z} = \{\cdots, a-2n, a-n, a, a+n, a+2n, \cdots\}$

Ensemble $\mathbb{Z}/n\mathbb{Z}$

Définition

Soient a et n deux entiers. La **classe résiduelle** de a modulo n, notée $\overline{a}$, est l'ensemble $\overline{a} = a + n\mathbb{Z} = \{\cdots, a-2n, a-n, a, a+n, a+2n, \cdots\}$

Exemple : $a = 3, n = 7$, $\overline{3} = \{-18, -11, -4, 3, 10, 17, 24, \cdots\}$

Ensemble $\mathbb{Z}/n\mathbb{Z}$

Définition

Soient a et n deux entiers. La **classe résiduelle** de a modulo n, notée $\overline{a}$, est l'ensemble $\overline{a} = a + n\mathbb{Z} = \{\cdots, a-2n, a-n, a, a+n, a+2n, \cdots\}$

Exemple : $a = 3, n = 7, \ \overline{3} = \{-18, -11, -4, 3, 10, 17, 24, \cdots\}$

Définition

Soit n un entier. $\mathbb{Z}/n\mathbb{Z}$ est défini comme étant l'ensemble $\left\{\overline{0}, \overline{1}, \cdots, \overline{n-1}\right\}$

où $\overline{a}$ est la classe résiduelle de a modulo n.

Calculs dans $\mathbb{Z}/n\mathbb{Z}$

- **Addition** : $\overline{a} + \overline{b} = \overline{a + b}$

Calculs dans $\mathbb{Z}/n\mathbb{Z}$

- **Addition** : $\overline{a} + \overline{b} = \overline{a + b}$

  Exemple : $n = 13,\ a = 5,\ b = 9,$
  $\overline{a} + \overline{b} = \overline{a + b} = \overline{5} + \overline{9} = \overline{5 + 9} = \overline{14} = \overline{1}$

Calculs dans $\mathbb{Z}/n\mathbb{Z}$

- **Addition** : $\overline{a} + \overline{b} = \overline{a + b}$

  Exemple : $n = 13,\ a = 5,\ b = 9,$
  $\overline{a} + \overline{b} = \overline{a + b} = \overline{5} + \overline{9} = \overline{5 + 9} = \overline{14} = \overline{1}$

- **Multiplication** : $\overline{a} \cdot \overline{b} = \overline{a \cdot b}$

Calculs dans $\mathbb{Z}/n\mathbb{Z}$

- **Addition** : $\bar{a} + \bar{b} = \overline{a + b}$

  Exemple : $n = 13,\ a = 5,\ b = 9,$
  $\bar{a} + \bar{b} = \overline{a + b} = \bar{5} + \bar{9} = \overline{5 + 9} = \overline{14} = \bar{1}$

- **Multiplication** : $\bar{a} \cdot \bar{b} = \overline{a \cdot b}$

  Exemple : $n = 13,\ a = 5,\ b = 9,$
  $\bar{a} \cdot \bar{b} = \overline{a \cdot b} = \bar{5} \cdot \bar{9} = \overline{5 \cdot 9} = \overline{25} = \bar{12}$

Inverse modulaire

Définition

Un élément $\bar{x}$ de $\mathbb{Z}/n\mathbb{Z}$ est dit inversible s'il existe un élément $\bar{y}$ de $\mathbb{Z}/n\mathbb{Z}$ tel que $\bar{x} \cdot \bar{y} = \bar{1}$.

Inverse modulaire

Définition

Un élément $\bar{x}$ de $\mathbb{Z}/n\mathbb{Z}$ est dit inversible s'il existe un élément $\bar{y}$ de $\mathbb{Z}/n\mathbb{Z}$ tel que $\bar{x} \cdot \bar{y} = \bar{1}$.

- *Exemple* : dans $\mathbb{Z}/5\mathbb{Z}$, $1, 2, 3, 4$ sont inversibles : $1 \cdot 1 = 1$, $2 \cdot 3 = 1$, $4 \cdot 4 = 1$

Inverse modulaire

Définition

Un élément $\bar{x}$ de $\mathbb{Z}/n\mathbb{Z}$ est dit inversible s'il existe un élément $\bar{y}$ de $\mathbb{Z}/n\mathbb{Z}$ tel que $\bar{x} \cdot \bar{y} = \bar{1}$.

- *Exemple* : dans $\mathbb{Z}/5\mathbb{Z}$, $1, 2, 3, 4$ sont inversibles : $1 \cdot 1 = 1$, $2 \cdot 3 = 1$, $4 \cdot 4 = 1$
- L'ensemble des éléments inversibles est noté $(\mathbb{Z}/n\mathbb{Z})^\times$

Inverse modulaire

Définition

Un élément $\bar{x}$ de $\mathbb{Z}/n\mathbb{Z}$ est dit inversible s'il existe un élément $\bar{y}$ de $\mathbb{Z}/n\mathbb{Z}$ tel que $\bar{x} \cdot \bar{y} = \bar{1}$.

- *Exemple* : dans $\mathbb{Z}/5\mathbb{Z}$, $1, 2, 3, 4$ sont inversibles : $1 \cdot 1 = 1$, $2 \cdot 3 = 1$, $4 \cdot 4 = 1$
- L'ensemble des éléments inversibles est noté $(\mathbb{Z}/n\mathbb{Z})^\times$

Proposition

1. Un élément $\bar{x}$ de $\mathbb{Z}/n\mathbb{Z}$ est **inversible** si et seulement si $pgcd(x, n) = 1$.
2. Le cardinal de $(\mathbb{Z}/n\mathbb{Z})^\times$ est $\phi(n)$.

Cas particulier

Corollaire

p est un nombre premier si et seulement si tous les éléments de $\mathbb{Z}/p\mathbb{Z}$ sauf $\overline{0}$ sont inversibles.

Cas particulier

Corollaire

p est un nombre premier si et seulement si tous les éléments de $\mathbb{Z}/p\mathbb{Z}$ sauf $\overline{0}$ sont inversibles.

- *Exemples* :
  - $n = 15$, $pgcd(15, 10) = 5$ donc $\overline{10}$ n'est pas inversible modulo 15
  - $p = 17$ est un nombre premier donc $|(\mathbb{Z}/p\mathbb{Z})^{\times}| = \phi(17) = 16$

Cas particulier

Corollaire

p est un nombre premier si et seulement si tous les éléments de $\mathbb{Z}/p\mathbb{Z}$ sauf $\overline{0}$ sont inversibles.

- *Exemples* :
  - $n = 15$, $pgcd(15, 10) = 5$ donc $\overline{10}$ n'est pas inversible modulo 15
  - $p = 17$ est un nombre premier donc $|(\mathbb{Z}/p\mathbb{Z})^{\times}| = \phi(17) = 16$

- *Remarque* : on calcule des inverses modulaires grâce à l'algorithme d'Euclide étendu

Théorèmes d'Euler et de Fermat

Théorème d'Euler

Soit n un entier et x tel que $pgcd(x, n) = 1$. Alors

$$x^{\phi(n)} = 1 \mod n$$

Théorèmes d'Euler et de Fermat

Théorème d'Euler

Soit n un entier et x tel que $pgcd(x, n) = 1$. Alors

$$x^{\phi(n)} = 1 \mod n$$

Petit théorème de Fermat

Soit p un nombre premier. Pour tout x non multiple de p, on a

$$x^{p-1} = 1 \mod p$$

Exponentiation modulaire : $a^k \mod n$

- Naïvement on calcule $b = \underbrace{a \times a \times \cdots \times a}_{k}$ fois puis on calcule $b \mod n$ si $b \geq n$

Exponentiation modulaire : $a^k \mod n$

- Naïvement on calcule $b = \underbrace{a \times a \times \cdots \times a}_{k}$ fois puis on calcule $b \mod n$ si $b \geq n$
- Mais dès lors qu'on utilise des grands nombres, ce calcul prend trop de temps !

Exponentiation modulaire : $a^k \mod n$

- Naïvement on calcule $b = \underbrace{a \times a \times \cdots \times a}_{k \text{ fois}}$ puis on calcule $b \mod n$ si $b \geq n$
- Mais dès lors qu'on utilise des grands nombres, ce calcul prend trop de temps !
- Pour rendre le calcul plus efficace, on va utiliser l'algorithme suivant :

Algorithm Square and multiply algorithm

Input : les entiers a, k et n avec $k = \sum_{i=0}^{p} k_i 2^i$ (écriture binaire de k)
Output : $a^k \mod n$

1: $h \leftarrow 1$
2: **for** $i = p$ to 0 **do**
3: $\quad h \leftarrow h \times h \mod n$
4: $\quad$ **if** $k_i = 1$ **then**
5: $\quad\quad h \leftarrow h \times a \mod n$
6: $\quad$ **end if**
7: **end for**

Square and multiply : un exemple pour comprendre (1)

- $N = 13$, $a = 7$, $k = 17$. Calculer $a^k \mod N$ avec l'algorithme square and multiply

Square and multiply : un exemple pour comprendre (1)

- $N = 13$, $a = 7$, $k = 17$. Calculer $a^k \mod N$ avec l'algorithme square and multiply
- D'abord écrire k sous forme binaire

Square and multiply : un exemple pour comprendre (1)

- $N = 13$, $a = 7$, $k = 17$. Calculer $a^k \mod N$ avec l'algorithme square and multiply
- D'abord écrire k sous forme binaire
- $k = 17 = (10001)_2$ et $p = 4$

Square and multiply : un exemple pour comprendre (1)

- $N = 13$, $a = 7$, $k = 17$. Calculer $a^k \mod N$ avec l'algorithme square and multiply
- D'abord écrire k sous forme binaire
- $k = 17 = (10001)_2$ et $p = 4$
- $h = 1$

Square and multiply : un exemple pour comprendre (1)

- $N = 13$, $a = 7$, $k = 17$. Calculer $a^k \mod N$ avec l'algorithme square and multiply
- D'abord écrire k sous forme binaire
- $k = 17 = (10001)_2$ et $p = 4$
- $h = 1$
- Pour $i = 4$:

Square and multiply : un exemple pour comprendre (1)

- $N = 13$, $a = 7$, $k = 17$. Calculer $a^k \mod N$ avec l'algorithme square and multiply
- D'abord écrire k sous forme binaire
- $k = 17 = (10001)_2$ et $p = 4$
- $h = 1$
- Pour $i = 4$:
  - $h = 1 \times 1 \mod 13 = 1$

Square and multiply : un exemple pour comprendre (1)

- $N = 13$, $a = 7$, $k = 17$. Calculer $a^k \mod N$ avec l'algorithme square and multiply
- D'abord écrire k sous forme binaire
- $k = 17 = (10001)_2$ et $p = 4$
- $h = 1$
- Pour $i = 4$:
  - $h = 1 \times 1 \mod 13 = 1$
  - $k_4 = 1$ donc $h = 1 \times 7 \mod 13$

Square and multiply : un exemple pour comprendre (1)

- $N = 13$, $a = 7$, $k = 17$. Calculer $a^k \mod N$ avec l'algorithme square and multiply
- D'abord écrire k sous forme binaire
- $k = 17 = (10001)_2$ et $p = 4$
- $h = 1$
- Pour $i = 4$:
  - $h = 1 \times 1 \mod 13 = 1$
  - $k_4 = 1$ donc $h = 1 \times 7 \mod 13$
- Pour $i = 3$:

Square and multiply : un exemple pour comprendre (1)

- $N = 13$, $a = 7$, $k = 17$. Calculer $a^k \mod N$ avec l'algorithme square and multiply
- D'abord écrire k sous forme binaire
- $k = 17 = (10001)_2$ et $p = 4$
- $h = 1$
- Pour $i = 4$:
  - $h = 1 \times 1 \mod 13 = 1$
  - $k_4 = 1$ donc $h = 1 \times 7 \mod 13$
- Pour $i = 3$:
  - $h = 7 \times 7 \mod 13 = 10$

Square and multiply : un exemple pour comprendre (1)

- $N = 13$, $a = 7$, $k = 17$. Calculer $a^k \mod N$ avec l'algorithme square and multiply
- D'abord écrire k sous forme binaire
- $k = 17 = (10001)_2$ et $p = 4$
- $h = 1$
- Pour $i = 4$:
  - $h = 1 \times 1 \mod 13 = 1$
  - $k_4 = 1$ donc $h = 1 \times 7 \mod 13$
- Pour $i = 3$:
  - $h = 7 \times 7 \mod 13 = 10$
  - $k_3 \neq 1$

Square and multiply : un exemple pour comprendre (2)

- Pour $i = 2$:

Square and multiply : un exemple pour comprendre (2)

- Pour $i = 2$:
  - $h = 10 \times 10 \mod 13 = 9$

Square and multiply : un exemple pour comprendre (2)

- Pour $i = 2$:
  - $h = 10 \times 10 \mod 13 = 9$
  - $k_2 \neq 1$

Square and multiply : un exemple pour comprendre (2)

- Pour $i = 2$:
  - $h = 10 \times 10 \mod 13 = 9$
  - $k_2 \neq 1$
- Pour $i = 1$:

Square and multiply : un exemple pour comprendre (2)

- Pour $i = 2$:
  - $h = 10 \times 10 \mod 13 = 9$
  - $k_2 \neq 1$
- Pour $i = 1$:
  - $h = 9 \times 9 \mod 13 = 3$

Square and multiply : un exemple pour comprendre (2)

- Pour $i = 2$:
  - $h = 10 \times 10 \mod 13 = 9$
  - $k_2 \neq 1$
- Pour $i = 1$:
  - $h = 9 \times 9 \mod 13 = 3$
  - $k_1 \neq 1$

Square and multiply : un exemple pour comprendre (2)

- Pour $i = 2$:
  - $h = 10 \times 10 \mod 13 = 9$
  - $k_2 \neq 1$
- Pour $i = 1$:
  - $h = 9 \times 9 \mod 13 = 3$
  - $k_1 \neq 1$
- Pour $i = 0$:

Square and multiply : un exemple pour comprendre (2)

- Pour $i = 2$:
  - $h = 10 \times 10 \mod 13 = 9$
  - $k_2 \neq 1$
- Pour $i = 1$:
  - $h = 9 \times 9 \mod 13 = 3$
  - $k_1 \neq 1$
- Pour $i = 0$:
  - $h = 3 \times 3 \mod 13 = 9$

Square and multiply : un exemple pour comprendre (2)

- Pour $i = 2$:
  - $h = 10 \times 10 \mod 13 = 9$
  - $k_2 \neq 1$
- Pour $i = 1$:
  - $h = 9 \times 9 \mod 13 = 3$
  - $k_1 \neq 1$
- Pour $i = 0$:
  - $h = 3 \times 3 \mod 13 = 9$
  - $k_0 = 1$ donc $h = 9 \times 7 \mod 13 = 11$

Square and multiply : un exemple pour comprendre (2)

- Pour $i = 2$:
  - $h = 10 \times 10 \mod 13 = 9$
  - $k_2 \neq 1$
- Pour $i = 1$:
  - $h = 9 \times 9 \mod 13 = 3$
  - $k_1 \neq 1$
- Pour $i = 0$:
  - $h = 3 \times 3 \mod 13 = 9$
  - $k_0 = 1$ donc $h = 9 \times 7 \mod 13 = 11$
- Le résultat est : 11

Structure de groupe

Définition

Soit G un ensemble et $*$ une application de $G \times G$ dans G (aussi appelée loi). Le couple $(G, *)$ est un groupe si et seulement si :

1. La loi $*$ est **associative**, i.e. : $\forall (x, y, z) \in G^3$, $(x * y) * z = x * (y * z)$
2. Il existe $e \in G$, appelé **élément neutre**, tel que $\forall x \in G$, $x * e = e * x = x$
3. Tout élément x de G admet un inverse, i.e. : $\forall x \in G$, $\exists y \in G$, $x * y = y * x = e$. L'inverse de x est noté x^{-1}.

Structure de groupe

Définition

Soit G un ensemble et $*$ une application de $G \times G$ dans G (aussi appelée loi). Le couple $(G, *)$ est un groupe si et seulement si :

1. La loi $*$ est **associative**, i.e. : $\forall (x, y, z) \in G^3$, $(x * y) * z = x * (y * z)$
2. Il existe $e \in G$, appelé **élément neutre**, tel que $\forall x \in G$, $x * e = e * x = x$
3. Tout élément x de G admet un inverse, i.e. : $\forall x \in G$, $\exists y \in G$, $x * y = y * x = e$. L'inverse de x est noté x^{-1}.

- *Notation* : $g^r = g * g * \cdots * g$ (r fois)

Structure de groupe

Définition

Soit G un ensemble et $*$ une application de $G \times G$ dans G (aussi appelée loi). Le couple $(G, *)$ est un groupe si et seulement si :

1. La loi $*$ est **associative**, i.e. : $\forall (x, y, z) \in G^3$, $(x * y) * z = x * (y * z)$
2. Il existe $e \in G$, appelé **élément neutre**, tel que $\forall x \in G$, $x * e = e * x = x$
3. Tout élément x de G admet un inverse, i.e. : $\forall x \in G$, $\exists y \in G$, $x * y = y * x = e$. L'inverse de x est noté x^{-1}.

- *Notation* : $g^r = g * g * \cdots * g$ (r fois)
- **Ordre** : l'ordre d'un élément g, noté $o(g)$ est le plus petit entier k tel que $g^k = e$

Structure de groupe

Définition

Soit G un ensemble et $*$ une application de $G \times G$ dans G (aussi appelée loi). Le couple $(G, *)$ est un groupe si et seulement si :

1. La loi $*$ est **associative**, i.e. : $\forall (x, y, z) \in G^3$, $(x * y) * z = x * (y * z)$
2. Il existe $e \in G$, appelé **élément neutre**, tel que $\forall x \in G$, $x * e = e * x = x$
3. Tout élément x de G admet un inverse, i.e. : $\forall x \in G$, $\exists y \in G$, $x * y = y * x = e$. L'inverse de x est noté x^{-1}.

- *Notation* : $g^r = g * g * \cdots * g$ (r fois)
- **Ordre** : l'ordre d'un élément g, noté $o(g)$ est le plus petit entier k tel que $g^k = e$

- **Générateur** : $x \in G$ est un générateur de G si tous les éléments de G peuvent s'écrire x^k for $k \in \mathbb{Z}$. On note alors $G =< x >$. Pour $(G = \mathbb{Z}/p\mathbb{Z}, +)$ avec p premier, tous les éléments $\neq 0$ sont générateurs.

Un schéma connu : R.S.A. (Rivest, Shamir, Adleman.)

- Génération des clés

Un schéma connu : R.S.A. (Rivest, Shamir, Adleman.)

- Génération des clés
  - p et q deux grands nombres premiers (1024 bits)

Un schéma connu : R.S.A. (Rivest, Shamir, Adleman.)

- Génération des clés
  - p et q deux grands nombres premiers (1024 bits)
  - $N = pq$ (2048 bits)

Un schéma connu : R.S.A. (Rivest, Shamir, Adleman.)

- Génération des clés
  - p et q deux grands nombres premiers (1024 bits)
  - $N = pq$ (2048 bits)
  - e, d sont deux entiers premiers à $\phi(N) = (p-1)(q-1)$ tels que $ed \equiv 1 \pmod{\phi}(N)$

Un schéma connu : R.S.A. (Rivest, Shamir, Adleman.)

- Génération des clés
  - p et q deux grands nombres premiers (1024 bits)
  - $N = pq$ (2048 bits)
  - e, d sont deux entiers premiers à $\phi(N) = (p-1)(q-1)$ tels que $ed \equiv 1 \pmod{\phi}(N)$
  - Clé publique $\text{pk} = (N, e)$

Un schéma connu : R.S.A. (Rivest, Shamir, Adleman.)

- Génération des clés
  - p et q deux grands nombres premiers (1024 bits)
  - $N = pq$ (2048 bits)
  - e, d sont deux entiers premiers à $\phi(N) = (p-1)(q-1)$ tels que $ed \equiv 1 \pmod{\phi)(N}$
  - Clé publique $\text{pk} = (N, e)$
  - Clé secrète $\text{sk} = (d, p, q)$

- Chiffrer : message $m \in \mathbb{Z}/N\mathbb{Z}$

$$c \equiv m^e \pmod{N}$$

Un schéma connu : R.S.A. (Rivest, Shamir, Adleman.)

- Génération des clés
  - p et q deux grands nombres premiers (1024 bits)
  - $N = pq$ (2048 bits)
  - e, d sont deux entiers premiers à $\phi(N) = (p-1)(q-1)$ tels que $ed \equiv 1 \pmod{\phi)(N}$
  - Clé publique $\text{pk} = (N, e)$
  - Clé secrète $\text{sk} = (d, p, q)$

- Chiffrer : message $m \in \mathbb{Z}/N\mathbb{Z}$

$$c \equiv m^e \pmod{N}$$

- Déchiffrer :

$$m \equiv c^d \pmod{N}$$

Fonctionnement déchiffrement R.S.A.

$$
\begin{aligned}
c^d \pmod N &\equiv m^{ed} \pmod N \\
&\equiv m^{k\phi(N)+1} \pmod N \\
&\equiv m^1 \cdot (m^{\phi(N)})^k \pmod N \\
&\equiv m
\end{aligned}
$$

R.S.A. : un exemple

- Génération des clés :

R.S.A. : un exemple

- Génération des clés :
  - On choisit $p = 7$ et $q = 11$, donc $N = 77$

R.S.A. : un exemple

- Génération des clés :
  - On choisit $p = 7$ et $q = 11$, donc $N = 77$
  - On a $\phi(77) = 60$ et on choisit $e = 13$ qui est premier à $\phi(77)$

R.S.A. : un exemple

- Génération des clés :
  - On choisit $p = 7$ et $q = 11$, donc $N = 77$
  - On a $\phi(77) = 60$ et on choisit $e = 13$ qui est premier à $\phi(77)$
  - Avec Euclide étendu on calcule $13^{-1} \equiv 37 \pmod{77}$

R.S.A. : un exemple

- Génération des clés :
  - On choisit $p = 7$ et $q = 11$, donc $N = 77$
  - On a $\phi(77) = 60$ et on choisit $e = 13$ qui est premier à $\phi(77)$
  - Avec Euclide étendu on calcule $13^{-1} \equiv 37 \pmod{77}$
  - On a alors pk $= (77, 13)$ et sk $= (7, 11, 37)$

R.S.A. : un exemple

- Génération des clés :
  - On choisit $p = 7$ et $q = 11$, donc $N = 77$
  - On a $\phi(77) = 60$ et on choisit $e = 13$ qui est premier à $\phi(77)$
  - Avec Euclide étendu on calcule $13^{-1} \equiv 37 \pmod{77}$
  - On a alors $\text{pk} = (77, 13)$ et $\text{sk} = (7, 11, 37)$

- Chiffrement : pour chiffrer message $m = 5$, on calcule $5^{13} \pmod{77} = 26$

R.S.A. : un exemple

- Génération des clés :
  - On choisit $p = 7$ et $q = 11$, donc $N = 77$
  - On a $\phi(77) = 60$ et on choisit $e = 13$ qui est premier à $\phi(77)$
  - Avec Euclide étendu on calcule $13^{-1} \equiv 37 \pmod{77}$
  - On a alors $\text{pk} = (77, 13)$ et $\text{sk} = (7, 11, 37)$

- Chiffrement : pour chiffrer message $m = 5$, on calcule $5^{13} \pmod{77} = 26$

- Déchiffrement : pour déchiffrer, on calcule $26^{37} \pmod{77} = 5$

La confiance : une notion importante en cryptologie !

- On peut vouloir utiliser des bibliothèques cryptographiques en boite noire (on ne connait que le résultat des algorithmes, pas leur fonctionnement)

La confiance : une notion importante en cryptologie !

- On peut vouloir utiliser des bibliothèques cryptographiques en boite noire (on ne connait que le résultat des algorithmes, pas leur fonctionnement)

- Une clé peut alors être générée avec les apparences (notamment statistiques) d'une clé aléatoire mais qui en pratique contient une porte dérobée (backdoor)

La confiance : une notion importante en cryptologie !

- On peut vouloir utiliser des bibliothèques cryptographiques en boite noire (on ne connait que le résultat des algorithmes, pas leur fonctionnement)

- Une clé peut alors être générée avec les apparences (notamment statistiques) d'une clé aléatoire mais qui en pratique contient une porte dérobée (backdoor)

- Cette porte dérobée peut ensuite être utilisée pour déchiffrer tous les messages !

La confiance : une notion importante en cryptologie !

- On peut vouloir utiliser des bibliothèques cryptographiques en boite noire (on ne connait que le résultat des algorithmes, pas leur fonctionnement)

- Une clé peut alors être générée avec les apparences (notamment statistiques) d'une clé aléatoire mais qui en pratique contient une porte dérobée (backdoor)

- Cette porte dérobée peut ensuite être utilisée pour déchiffrer tous les messages !

- Pour plus d'informations : `https://www.numerama.com/politique/27869-la-nsa-aurait-paye-10-millions-de-dollars-pour-corr html`
  `https://www.researchgate.net/publication/225139661_The_Dark_Side_of_Black-Box_Cryptography_or_Should_We_Trust_Capstone`

Le logarithme discret

Problème du logarithme discret dans $\mathbb{Z}/p\mathbb{Z}$

Soit p un nombre premier, g un élément de $(\mathbb{Z}/p\mathbb{Z})^\times$, r un entier et $x \equiv g^r \mod p$. Le problème du logarithme discret consiste à retrouver r connaissant p, g et x.

Le logarithme discret

Problème du logarithme discret dans $\mathbb{Z}/p\mathbb{Z}$

Soit p un nombre premier, g un élément de $(\mathbb{Z}/p\mathbb{Z})^\times$, r un entier et $x \equiv g^r \mod p$. Le problème du logarithme discret consiste à retrouver r connaissant p, g et x.

Problème du logarithme discret dans un groupe

Soit $(G, *)$ un groupe (commutatif) fini, g un élément de G, d'ordre q grand, r un entier $\leq q$ et $x = g^r$. Le problème du logarithme discret consiste à retrouver r connaissant G, g et x.

Un autre schéma : El Gamal

- Génération des clés

Un autre schéma : El Gamal

- Génération des clés
  - Choisir un nombre premier p et un générateur g de $\mathbb{Z}/p\mathbb{Z}$

Un autre schéma : El Gamal

- Génération des clés
  - Choisir un nombre premier p et un générateur g de $\mathbb{Z}/p\mathbb{Z}$
  - Générer aléatoirement un entier $r \in [1, \cdots, p-1]$

Un autre schéma : El Gamal

- Génération des clés
  - Choisir un nombre premier p et un générateur g de $\mathbb{Z}/p\mathbb{Z}$
  - Générer aléatoirement un entier $r \in [1, \cdots, p-1]$
  - Calculer $B \equiv g^r \mod p$

Un autre schéma : El Gamal

- Génération des clés
  - Choisir un nombre premier p et un générateur g de $\mathbb{Z}/p\mathbb{Z}$
  - Générer aléatoirement un entier $r \in [1, \cdots, p-1]$
  - Calculer $B \equiv g^r \mod p$
  - $\mathsf{pk} = (g, p, B)$ et $\mathsf{sk} = r$

Un autre schéma : El Gamal

- Génération des clés
  - Choisir un nombre premier p et un générateur g de $\mathbb{Z}/p\mathbb{Z}$
  - Générer aléatoirement un entier $r \in [1, \cdots, p-1]$
  - Calculer $B \equiv g^r \mod p$
  - $\mathsf{pk} = (g, p, B)$ et $\mathsf{sk} = r$
- Chiffrement : le message à chiffrer m est tel que $1 \leq m < p$

Un autre schéma : El Gamal

- Génération des clés
  - Choisir un nombre premier p et un générateur g de $\mathbb{Z}/p\mathbb{Z}$
  - Générer aléatoirement un entier $r \in [1, \cdots, p-1]$
  - Calculer $B \equiv g^r \mod p$
  - $\mathsf{pk} = (g, p, B)$ et $\mathsf{sk} = r$
- Chiffrement : le message à chiffrer m est tel que $1 \leq m < p$
  - Générer un entier $a \in [1, \cdots, p-1]$

Un autre schéma : El Gamal

- Génération des clés
  - Choisir un nombre premier p et un générateur g de $\mathbb{Z}/p\mathbb{Z}$
  - Générer aléatoirement un entier $r \in [1, \cdots, p-1]$
  - Calculer $B \equiv g^r \mod p$
  - $\mathsf{pk} = (g, p, B)$ et $\mathsf{sk} = r$
- Chiffrement : le message à chiffrer m est tel que $1 \leq m < p$
  - Générer un entier $a \in [1, \cdots, p-1]$
  - Calculer $c_1 \equiv g^a \mod p$ et $c_2 \equiv m \cdot B^a \mod p$

Un autre schéma : El Gamal

- Génération des clés
  - Choisir un nombre premier p et un générateur g de $\mathbb{Z}/p\mathbb{Z}$
  - Générer aléatoirement un entier $r \in [1, \cdots, p-1]$
  - Calculer $B \equiv g^r \mod p$
  - $\text{pk} = (g, p, B)$ et $\text{sk} = r$
- Chiffrement : le message à chiffrer m est tel que $1 \leq m < p$
  - Générer un entier $a \in [1, \cdots, p-1]$
  - Calculer $c_1 \equiv g^a \mod p$ et $c_2 \equiv m \cdot B^a \mod p$
  - Message chiffré $= (c_1, c_2)$

Un autre schéma : El Gamal

- Génération des clés
  - Choisir un nombre premier p et un générateur g de $\mathbb{Z}/p\mathbb{Z}$
  - Générer aléatoirement un entier $r \in [1, \cdots, p-1]$
  - Calculer $B \equiv g^r \mod p$
  - pk $= (g, p, B)$ et sk $= r$
- Chiffrement : le message à chiffrer m est tel que $1 \leq m < p$
  - Générer un entier $a \in [1, \cdots, p-1]$
  - Calculer $c_1 \equiv g^a \mod p$ et $c_2 \equiv m \cdot B^a \mod p$
  - Message chiffré $= (c_1, c_2)$
- Déchiffrement :

Un autre schéma : El Gamal

- Génération des clés
  - Choisir un nombre premier p et un générateur g de $\mathbb{Z}/p\mathbb{Z}$
  - Générer aléatoirement un entier $r \in [1, \cdots, p-1]$
  - Calculer $B \equiv g^r \mod p$
  - pk $= (g, p, B)$ et sk $= r$
- Chiffrement : le message à chiffrer m est tel que $1 \leq m < p$
  - Générer un entier $a \in [1, \cdots, p-1]$
  - Calculer $c_1 \equiv g^a \mod p$ et $c_2 \equiv m \cdot B^a \mod p$
  - Message chiffré $= (c_1, c_2)$
- Déchiffrement :
  - Calculer $d_1 \equiv c_1^{-1} \mod p$ et $m' \equiv c_2 \cdot d_1^r \mod p$

Un autre schéma : El Gamal

- Génération des clés
  - Choisir un nombre premier p et un générateur g de $\mathbb{Z}/p\mathbb{Z}$
  - Générer aléatoirement un entier $r \in [1, \cdots, p-1]$
  - Calculer $B \equiv g^r \mod p$
  - pk $= (g, p, B)$ et sk $= r$
- Chiffrement : le message à chiffrer m est tel que $1 \leq m < p$
  - Générer un entier $a \in [1, \cdots, p-1]$
  - Calculer $c_1 \equiv g^a \mod p$ et $c_2 \equiv m \cdot B^a \mod p$
  - Message chiffré $= (c_1, c_2)$
- Déchiffrement :
  - Calculer $d_1 \equiv c_1^{-1} \mod p$ et $m' \equiv c_2 \cdot d_1^r \mod p$
- Chiffrement réussi si $m' = m$

Un autre schéma : El Gamal

- Génération des clés
  - Choisir un nombre premier p et un générateur g de $\mathbb{Z}/p\mathbb{Z}$
  - Générer aléatoirement un entier $r \in [1, \cdots, p-1]$
  - Calculer $B \equiv g^r \mod p$
  - pk $= (g, p, B)$ et sk $= r$
- Chiffrement : le message à chiffrer m est tel que $1 \leq m < p$
  - Générer un entier $a \in [1, \cdots, p-1]$
  - Calculer $c_1 \equiv g^a \mod p$ et $c_2 \equiv m \cdot B^a \mod p$
  - Message chiffré $= (c_1, c_2)$
- Déchiffrement :
  - Calculer $d_1 \equiv c_1^{-1} \mod p$ et $m' \equiv c_2 \cdot d_1^r \mod p$
- Chiffrement réussi si $m' = m$
- $c_2 \cdot d_1^r = (m \cdot B^a) \cdot ((g^a)^{-1})^r = m \cdot g^{ra} \cdot (g^{-a})^r = m \cdot g^{ra-ar} = m$

El Gamal : sécurité et exemple

- *Exemple* :

El Gamal : sécurité et exemple

- *Exemple* :
  - Génération de la clé

El Gamal : sécurité et exemple

- *Exemple* :
  - Génération de la clé
    - $p = 661$

El Gamal : sécurité et exemple

- *Exemple* :
  - Génération de la clé
    - $p = 661$
    - On choisit $g = 23$ un générateur

El Gamal : sécurité et exemple

- *Exemple* :
  - Génération de la clé
    - ★ $p = 661$
    - ★ On choisit $g = 23$ un générateur
    - ★ On choisit un entier aléatoirement dans $[1, \cdots, 661] = 7$

El Gamal : sécurité et exemple

- *Exemple* :
  - Génération de la clé
    - ★ $p = 661$
    - ★ On choisit $g = 23$ un générateur
    - ★ On choisit un entier aléatoirement dans $[1, \cdots, 661] = 7$
    - ★ $B = 23^7 \mod 661 = 566$

El Gamal : sécurité et exemple

- *Exemple* :
  - Génération de la clé
    - ★ $p = 661$
    - ★ On choisit $g = 23$ un générateur
    - ★ On choisit un entier aléatoirement dans $[1, \cdots, 661] = 7$
    - ★ $B = 23^7 \mod 661 = 566$
    - ★ $\text{pk} = (661, 23, 566)$, $\text{sk} = 7$

El Gamal : sécurité et exemple

- *Exemple* :
  - Génération de la clé
    - $p = 661$
    - On choisit $g = 23$ un générateur
    - On choisit un entier aléatoirement dans $[1, \cdots, 661] = 7$
    - $B = 23^7 \mod 661 = 566$
    - $\text{pk} = (661, 23, 566)$, $\text{sk} = 7$
  - Pour chiffrer $m = 192$, on choisit $a = 13$ et le chiffré est $c_1 = 105, c_2 = 237$

El Gamal : sécurité et exemple

- *Exemple* :
  - Génération de la clé
    - $p = 661$
    - On choisit $g = 23$ un générateur
    - On choisit un entier aléatoirement dans $[1, \cdots, 661] = 7$
    - $B = 23^7 \mod 661 = 566$
    - $\text{pk} = (661, 23, 566)$, $\text{sk} = 7$
  - Pour chiffrer $m = 192$, on choisit $a = 13$ et le chiffré est $c_1 = 105, c_2 = 237$
  - Pour déchiffrer : $c_2 \cdot (c_1^{-1})^7 = 192$

El Gamal : sécurité et exemple

- *Exemple* :
  - Génération de la clé
    - $p = 661$
    - On choisit $g = 23$ un générateur
    - On choisit un entier aléatoirement dans $[1, \cdots, 661] = 7$
    - $B = 23^7 \mod 661 = 566$
    - $\text{pk} = (661, 23, 566)$, $\text{sk} = 7$
  - Pour chiffrer $m = 192$, on choisit $a = 13$ et le chiffré est $c_1 = 105, c_2 = 237$
  - Pour déchiffrer : $c_2 \cdot (c_1^{-1})^7 = 192$

- Sécurité : elle repose sur le problème du logarithme discret

Échange de clés avec le logarithme discret

- On a vu que le partage de clé est un problème important de la cryptographie (notamment dans la cryptographie à clé secrète)

Échange de clés avec le logarithme discret

- On a vu que le partage de clé est un problème important de la cryptographie (notamment dans la cryptographie à clé secrète)
- Il existe un protocole basé sur le logarithme discret qui permet d'obtenir une clé commune : protocole Diffie Hellman

Échange de clés avec le logarithme discret

- On a vu que le partage de clé est un problème important de la cryptographie (notamment dans la cryptographie à clé secrète)

- Il existe un protocole basé sur le logarithme discret qui permet d'obtenir une clé commune : protocole Diffie Hellman
  - $(G, *)$ est un groupe et $g \in G$ d'ordre un grand premier q

Échange de clés avec le logarithme discret

- On a vu que le partage de clé est un problème important de la cryptographie (notamment dans la cryptographie à clé secrète)

- Il existe un protocole basé sur le logarithme discret qui permet d'obtenir une clé commune : protocole Diffie Hellman
  - $(G, *)$ est un groupe et $g \in G$ d'ordre un grand premier q
  - Alice tire au hasard $a \in [1, \cdots, q-1]$ et pose $y_A = g^a$

Échange de clés avec le logarithme discret

- On a vu que le partage de clé est un problème important de la cryptographie (notamment dans la cryptographie à clé secrète)

- Il existe un protocole basé sur le logarithme discret qui permet d'obtenir une clé commune : protocole Diffie Hellman
  - $(G, *)$ est un groupe et $g \in G$ d'ordre un grand premier q
  - Alice tire au hasard $a \in [1, \cdots, q-1]$ et pose $y_A = g^a$
  - Bob tire au hasard $b \in [1, \cdots, q-1]$ et pose $y_B = g^b$

Échange de clés avec le logarithme discret

- On a vu que le partage de clé est un problème important de la cryptographie (notamment dans la cryptographie à clé secrète)

- Il existe un protocole basé sur le logarithme discret qui permet d'obtenir une clé commune : protocole Diffie Hellman
  - $(G, *)$ est un groupe et $g \in G$ d'ordre un grand premier q
  - Alice tire au hasard $a \in [1, \cdots, q-1]$ et pose $y_A = g^a$
  - Bob tire au hasard $b \in [1, \cdots, q-1]$ et pose $y_B = g^b$
  - Alice et Bob envoient respectivement y_A, y_B à l'autre

Échange de clés avec le logarithme discret

- On a vu que le partage de clé est un problème important de la cryptographie (notamment dans la cryptographie à clé secrète)

- Il existe un protocole basé sur le logarithme discret qui permet d'obtenir une clé commune : protocole Diffie Hellman
  - $(G, *)$ est un groupe et $g \in G$ d'ordre un grand premier q
  - Alice tire au hasard $a \in [1, \cdots, q-1]$ et pose $y_A = g^a$
  - Bob tire au hasard $b \in [1, \cdots, q-1]$ et pose $y_B = g^b$
  - Alice et Bob envoient respectivement y_A, y_B à l'autre
  - Alice calcule $y_B^a = g^{ba}$ et Bob calcule $y_A^b = g^{ab}$

Échange de clés avec le logarithme discret

- On a vu que le partage de clé est un problème important de la cryptographie (notamment dans la cryptographie à clé secrète)

- Il existe un protocole basé sur le logarithme discret qui permet d'obtenir une clé commune : protocole Diffie Hellman
  - $(G, *)$ est un groupe et $g \in G$ d'ordre un grand premier q
  - Alice tire au hasard $a \in [1, \cdots, q-1]$ et pose $y_A = g^a$
  - Bob tire au hasard $b \in [1, \cdots, q-1]$ et pose $y_B = g^b$
  - Alice et Bob envoient respectivement y_A, y_B à l'autre
  - Alice calcule $y_B^a = g^{ba}$ et Bob calcule $y_A^b = g^{ab}$
  - Leur clé commune est g^{ab}

Sommaire

La cryptanalyse

- La cryptanalyse consiste à attaquer les schémas cryptographiques

La cryptanalyse

- La cryptanalyse consiste à attaquer les schémas cryptographiques
- Il existe différents types d'attaque

La cryptanalyse

- La cryptanalyse consiste à attaquer les schémas cryptographiques
- Il existe différents types d'attaque
  - A chiffré seul : l'attaquant ne connait qu'un message chiffré

La cryptanalyse

- La cryptanalyse consiste à attaquer les schémas cryptographiques
- Il existe différents types d'attaque
  - A chiffré seul : l'attaquant ne connait qu'un message chiffré
  - A clair/chiffré connu : l'attaquant connait un couple (clair, chiffré)

La cryptanalyse

- La cryptanalyse consiste à attaquer les schémas cryptographiques
- Il existe différents types d'attaque
  - A chiffré seul : l'attaquant ne connait qu'un message chiffré
  - A clair/chiffré connu : l'attaquant connait un couple (clair, chiffré)
  - A clair choisi : l'attaquant choisit un clair et obtient le chiffré associé

La cryptanalyse

- La cryptanalyse consiste à attaquer les schémas cryptographiques

- Il existe différents types d'attaque
  - A chiffré seul : l'attaquant ne connait qu'un message chiffré
  - A clair/chiffré connu : l'attaquant connait un couple (clair, chiffré)
  - A clair choisi : l'attaquant choisit un clair et obtient le chiffré associé
  - A clair choisi adaptif : l'attaquant choisit différents clairs de manière adaptive et obtient les chiffrés associés

La cryptanalyse

- La cryptanalyse consiste à attaquer les schémas cryptographiques
- Il existe différents types d'attaque
  - A chiffré seul : l'attaquant ne connait qu'un message chiffré
  - A clair/chiffré connu : l'attaquant connait un couple (clair, chiffré)
  - A clair choisi : l'attaquant choisit un clair et obtient le chiffré associé
  - A clair choisi adaptif : l'attaquant choisit différents clairs de manière adaptive et obtient les chiffrés associés
- Les attaques précédentes sont présentées du plus difficile au plus facile

Complexité d'une attaque

- La complexité d'une attaque se mesure en calculant la complexité de l'algorithme :

Complexité d'une attaque

- La complexité d'une attaque se mesure en calculant la complexité de l'algorithme :
  - temps de calcul = nombre d'opérations élémentaires (addition, comparaison...) qu'il faut pour la réaliser

Complexité d'une attaque

- La complexité d'une attaque se mesure en calculant la complexité de l'algorithme :
  - temps de calcul = nombre d'opérations élémentaires (addition, comparaison...) qu'il faut pour la réaliser
  - taille des variables utilisées

Complexité d'une attaque

- La complexité d'une attaque se mesure en calculant la complexité de l'algorithme :
  - temps de calcul = nombre d'opérations élémentaires (addition, comparaison...) qu'il faut pour la réaliser
  - taille des variables utilisées

- On utilise (souvent) la notation de Landau O ($\ll$ Grand O $\gg$) pour exprimer la complexité d'un algorithme : soient f, g deux fonctions. On dit que f est dominée par g en $+\infty$ si

$$\exists N, k \in \mathbb{N}, \forall x > N, |f(x)| \leq k|g(x)|$$

Complexité d'une attaque

- La complexité d'une attaque se mesure en calculant la complexité de l'algorithme :
  - temps de calcul = nombre d'opérations élémentaires (addition, comparaison...) qu'il faut pour la réaliser
  - taille des variables utilisées

- On utilise (souvent) la notation de Landau O ($\ll$ Grand O $\gg$) pour exprimer la complexité d'un algorithme : soient f, g deux fonctions. On dit que f est dominée par g en $+\infty$ si

$$\exists N, k \in \mathbb{N}, \forall x > N, |f(x)| \leq k|g(x)|$$

- Si pour trouver la clé d'un schéma symétrique, on teste toutes les clés possible de même taille, on parle alors d'attaque **exhaustive** ou encore par d'attaque **force brute**

Complexité d'une attaque

- La complexité d'une attaque se mesure en calculant la complexité de l'algorithme :
  - temps de calcul = nombre d'opérations élémentaires (addition, comparaison...) qu'il faut pour la réaliser
  - taille des variables utilisées

- On utilise (souvent) la notation de Landau O (« Grand O ») pour exprimer la complexité d'un algorithme : soient f, g deux fonctions. On dit que f est dominée par g en $+\infty$ si

$$\exists N, k \in \mathbb{N}, \forall x > N, |f(x)| \leq k|g(x)|$$

- Si pour trouver la clé d'un schéma symétrique, on teste toutes les clés possible de même taille, on parle alors d'attaque **exhaustive** ou encore par d'attaque **force brute**

- Actuellement on estime qu'il est calculatoirement impossible de faire plus de 2^{80} opérations élémentaires

Quelques classes de complexité

Type	Notation
Constant	$O(1)$
Logarithmique	$O(log(n)$
Poly logarithmique	$O(log(n)^2)$
Linéaire	$O(n)$
Polynomiale	$O(n^e)$
Sous exponentielle	$O(n^{log(n)})$
Exponentielle	$O(c^n)$

Quelques classes de complexité

Type	Notation
Constant	$O(1)$
Logarithmique	$O(log(n)$
Poly logarithmique	$O(log(n)^2)$
Linéaire	$O(n)$
Polynomiale	$O(n^e)$
Sous exponentielle	$O(n^{log(n)})$
Exponentielle	$O(c^n)$

- Les classes de « constante » à « polynomiale » correspondent aux problèmes dit « faciles », et les autres aux problèmes dit « difficiles »

Coût des opérations arithmétiques simples

- $a + b$ / $a - b$: $O(ln(max(a, b)) = O(ln(a) + ln(b)) = O(ln(N))$ si $0 \leq a, b \leq N$ coût **linéaire**

Coût des opérations arithmétiques simples

- $a + b$ / $a - b$: $O(ln(max(a, b)) = O(ln(a) + ln(b)) = O(ln(N))$ si $0 \leq a, b \leq N$ coût **linéaire**

- $a \times b$: $O(\underbrace{ln(a) + ln(a) + \cdots + ln(a)}) = O(ln(a) \times ln(b)) = O((ln(N))^2)$ si $0 \leq a, b < N$, coût **quadratique**

Coût des opérations arithmétiques simples

- $a + b$ / $a - b$: $O(ln(max(a, b)) = O(ln(a) + ln(b)) = O(ln(N))$ si $0 \leq a, b \leq N$ coût **linéaire**
- $a \times b$: $O(\underbrace{ln(a) + ln(a) + \cdots + ln(a)}) = O(ln(a) \times ln(b)) = O((ln(N))^2)$ si $0 \leq a, b < N$, coût **quadratique**
- a/b : $O(\frac{ln(b)+\cdots+ln(b)}{ln(a)-ln(b)}) = O((ln(a) - ln(b))ln(b)) = O((ln(N))^2)$ si $0 \leq a, b < N$, coût **quadratique**

Attaque fréquentielle

- Attaque sur les systèmes alphabétiques : étude de la fréquence d'apparition des lettres dans le chiffré

Attaque fréquentielle

- Attaque sur les systèmes alphabétiques : étude de la fréquence d'apparition des lettres dans le chiffré

- En particulier si c'est un système de **substitution monoalphabétique**, les caractéristiques du chiffré sont identiques à celles du texte clair

Attaque fréquentielle

- Attaque sur les systèmes alphabétiques : étude de la fréquence d'apparition des lettres dans le chiffré
- En particulier si c'est un système de **substitution monoalphabétique**, les caractéristiques du chiffré sont identiques à celles du texte clair
- La lettre la plus fréquente dans le chiffré doit correspondre à la lettre la plus fréquente dans le clair

Attaque fréquentielle

- Attaque sur les systèmes alphabétiques : étude de la fréquence d'apparition des lettres dans le chiffré
- En particulier si c'est un système de **substitution monoalphabétique**, les caractéristiques du chiffré sont identiques à celles du texte clair
- La lettre la plus fréquente dans le chiffré doit correspondre à la lettre la plus fréquente dans le clair
- En français, le « E » apparait le plus souvent

Attaque fréquentielle : visuellement

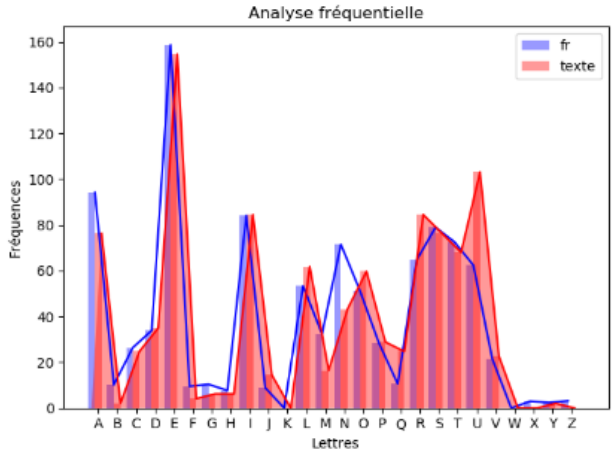


Figure 1: Fréquence d'apparition de chaque lettre. Pour un texte raisonnablement long, en français, l'histogramme est remarquablement constant

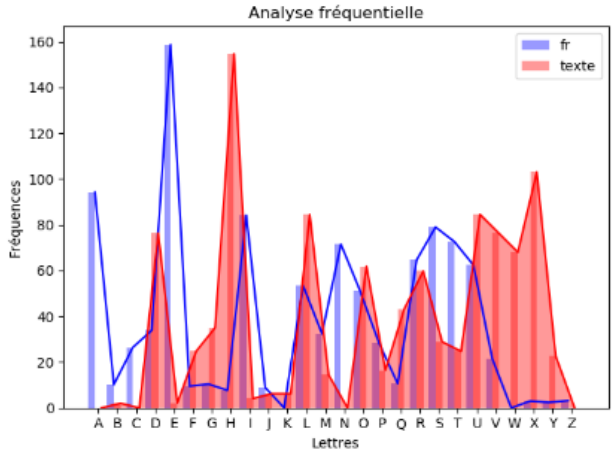


Figure 2: Dans le cas du chiffre de César, on déduit le décalage, ici 3.

Source : `https://ensip.gitlab.io/programmation_e2/PDF/C_P2A_50_crypto_anafreq.pdf`

Test de Kasiski

- Pour les chiffrements alphabétiques symétriques

Test de Kasiski

- Pour les chiffrements alphabétiques symétriques
- Si deux parties de texte identiques sont chiffrées avec la même partie de clé, on obtient des parties de chiffrés identiques

Test de Kasiski

- Pour les chiffrements alphabétiques symétriques
- Si deux parties de texte identiques sont chiffrées avec la même partie de clé, on obtient des parties de chiffrés identiques
- Inversement, si deux parties de chiffré sont identiques, on a de fortes chances que la distance entre les deux parties identiques constitue un multiple de la longueur de la clé

Test de Kasiski

- Pour les chiffrements alphabétiques symétriques
- Si deux parties de texte identiques sont chiffrées avec la même partie de clé, on obtient des parties de chiffrés identiques
- Inversement, si deux parties de chiffré sont identiques, on a de fortes chances que la distance entre les deux parties identiques constitue un multiple de la longueur de la clé
- Ce test est utile pour le schéma de Vigenère : on repère deux couples de parties identiques dans le chiffré, et la longueur de la clé est le pgcd de l'écart entre les éléments de chaque couple

Indice de coïncidence

- Idée : regarder la proportion de paires identiques parmi l'ensemble des paires de lettres du message M

Indice de coïncidence

- Idée : regarder la proportion de paires identiques parmi l'ensemble des paires de lettres du message M

- Pour un message de taille n, le nombre de paires possibles est $\binom{n}{2} = \frac{n(n-1)}{2}$

Indice de coïncidence

- Idée : regarder la proportion de paires identiques parmi l'ensemble des paires de lettres du message M

- Pour un message de taille n, le nombre de paires possibles est $\binom{n}{2} = \frac{n(n-1)}{2}$

- Maintenant on regarde les lettres identiques de M :
  - il y a n_1 lettres « A » : $\binom{n_1}{2}$
  - ...
  - il y a n_{26} lettres « Z » : $\binom{n_{26}}{2}$

Indice de coïncidence

- Idée : regarder la proportion de paires identiques parmi l'ensemble des paires de lettres du message M

- Pour un message de taille n, le nombre de paires possibles est $\binom{n}{2} = \frac{n(n-1)}{2}$

- Maintenant on regarde les lettres identiques de M :
  - il y a n_1 lettres « A » : $\binom{n_1}{2}$
  - ...
  - il y a n_{26} lettres « Z » : $\binom{n_{26}}{2}$

- L'indice de coïncidence vaut

$$I_C(M) = \sum_{i=1}^{26} \frac{\binom{n_i}{2}}{\binom{n}{2}} = \frac{1}{n(n-1)} \sum_{i=1}^{26} n_i(n_i - 1)$$

Indice de coïncidence et cryptanalyse

- Pour un texte aléatoire (les lettres du message sont réparties de façon équiprobables) : $I_C(M) \approx 1/26$

Indice de coïncidence et cryptanalyse

- Pour un texte aléatoire (les lettres du message sont réparties de façon équiprobables) : $I_C(M) \approx 1/26$
- Pour un texte en français : $I_C(M) \approx 0,08$

Indice de coïncidence et cryptanalyse

- Pour un texte aléatoire (les lettres du message sont réparties de façon équiprobables) : $I_C(M) \approx 1/26$
- Pour un texte en français : $I_C(M) \approx 0,08$
- Si le texte est chiffré par une substitution monoalphabétique, l'indice de coïncidence reste inchangé

Indice de coïncidence et cryptanalyse

- Pour un texte aléatoire (les lettres du message sont réparties de façon équiprobables) : $I_C(M) \approx 1/26$
- Pour un texte en français : $I_C(M) \approx 0,08$
- Si le texte est chiffré par une substitution monoalphabétique, l'indice de coïncidence reste inchangé
- Cela permet de distinguer un texte aléatoire d'un texte normal, à partir d'un chiffré

Indice de coïncidence et cryptanalyse du schéma de Vigenère

- Avec l'indice de coïncidence, on peut retrouver la longueur de la clé (comme avec le test de Kasiski)

Indice de coïncidence et cryptanalyse du schéma de Vigenère

- Avec l'indice de coïncidence, on peut retrouver la longueur de la clé (comme avec le test de Kasiski)

- L'idée est de tester toutes les tailles possibles : clé de taille $k = 1$ (la clé est une lettre), clé de taille $k = 2$, ... jusqu'à la longueur du chiffré

Indice de coïncidence et cryptanalyse du schéma de Vigenère

- Avec l'indice de coïncidence, on peut retrouver la longueur de la clé (comme avec le test de Kasiski)

- L'idée est de tester toutes les tailles possibles : clé de taille $k = 1$ (la clé est une lettre), clé de taille $k = 2$, ... jusqu'à la longueur du chiffré

- Pour chaque k, on découpe le chiffré en k sous messages : $k = 1$ le sous message est le chiffré, $k = 2$ le premier sous message est le chiffré aux positions $1, 3, 5, \cdots$, et le deuxième est le chiffré aux positions $2, 4, 6, \cdots$

Indice de coïncidence et cryptanalyse du schéma de Vigenère

- Avec l'indice de coïncidence, on peut retrouver la longueur de la clé (comme avec le test de Kasiski)

- L'idée est de tester toutes les tailles possibles : clé de taille $k = 1$ (la clé est une lettre), clé de taille $k = 2$, ... jusqu'à la longueur du chiffré

- Pour chaque k, on découpe le chiffré en k sous messages : $k = 1$ le sous message est le chiffré, $k = 2$ le premier sous message est le chiffré aux positions $1, 3, 5, \cdots$, et le deuxième est le chiffré aux positions $2, 4, 6, \cdots$

- Pour chaque k, on calcule l'I_C des k sous messages

Indice de coïncidence et cryptanalyse du schéma de Vigenère

- Avec l'indice de coïncidence, on peut retrouver la longueur de la clé (comme avec le test de Kasiski)
- L'idée est de tester toutes les tailles possibles : clé de taille $k = 1$ (la clé est une lettre), clé de taille $k = 2$, ... jusqu'à la longueur du chiffré
- Pour chaque k, on découpe le chiffré en k sous messages : $k = 1$ le sous message est le chiffré, $k = 2$ le premier sous message est le chiffré aux positions $1, 3, 5, \cdots$, et le deuxième est le chiffré aux positions $2, 4, 6, \cdots$
- Pour chaque k, on calcule l'I_C des k sous messages
- Le sous message ayant un I_C le plus proche de 0.08 nous donne la longueur de la clé : c'est la valeur k associée à ce sous message !

Sommaire

Vote électronique

- Dans un système de vote électronique, comme dans un système de vote classique, il y a plusieurs acteurs :

Vote électronique

- Dans un système de vote électronique, comme dans un système de vote classique, il y a plusieurs acteurs :
  - Autorité électorale : organise les élections, vérifie la bonne tenue du vote, ...

Vote électronique

- Dans un système de vote électronique, comme dans un système de vote classique, il y a plusieurs acteurs :
  - Autorité électorale : organise les élections, vérifie la bonne tenue du vote, ...
  - Assesseurs : responsable de la collecte des votes, vérifient l'éligibilité d'un votant, et comptent les votes

Vote électronique

- Dans un système de vote électronique, comme dans un système de vote classique, il y a plusieurs acteurs :
  - Autorité électorale : organise les élections, vérifie la bonne tenue du vote, ...
  - Assesseurs : responsable de la collecte des votes, vérifient l'éligibilité d'un votant, et comptent les votes
  - Électeurs : sont des personnes enregistrées sur une liste électorale et qui peuvent participer au vote

Vote électronique

- Dans un système de vote électronique, comme dans un système de vote classique, il y a plusieurs acteurs :
  - Autorité électorale : organise les élections, vérifie la bonne tenue du vote, ...
  - Assesseurs : responsable de la collecte des votes, vérifient l'éligibilité d'un votant, et comptent les votes
  - Électeurs : sont des personnes enregistrées sur une liste électorale et qui peuvent participer au vote

- Attention les acteurs précédents peuvent agir de manière frauduleuse : un électeur peut voter deux fois par exemple, ...

Les propriétés requises pour le vote électronique

- **Universellement vérifiable** : chaque acteur peut vérifier la présence et la validité de n'importe quel vote

Les propriétés requises pour le vote électronique

- **Universellement vérifiable** : chaque acteur peut vérifier la présence et la validité de n'importe quel vote
- **Éligibilité** : seulement les votes corrects venant d'électeurs légitimes sont pris en compte

Les propriétés requises pour le vote électronique

- **Universellement vérifiable** : chaque acteur peut vérifier la présence et la validité de n'importe quel vote
- **Éligibilité** : seulement les votes corrects venant d'électeurs légitimes sont pris en compte
- **Secret du vote** : personne ne peut lier un vote et la personne à son origine

Les propriétés requises pour le vote électronique

- **Universellement vérifiable** : chaque acteur peut vérifier la présence et la validité de n'importe quel vote
- **Éligibilité** : seulement les votes corrects venant d'électeurs légitimes sont pris en compte
- **Secret du vote** : personne ne peut lier un vote et la personne à son origine
- **Pas de résultat partiel** : les assesseurs ne doivent pas publier de résultats avant la fin de l'élection

Les propriétés requises pour le vote électronique

- **Universellement vérifiable** : chaque acteur peut vérifier la présence et la validité de n'importe quel vote
- **Éligibilité** : seulement les votes corrects venant d'électeurs légitimes sont pris en compte
- **Secret du vote** : personne ne peut lier un vote et la personne à son origine
- **Pas de résultat partiel** : les assesseurs ne doivent pas publier de résultats avant la fin de l'élection
- **Sans reçu** : un électeur ne peut pas fournir la preuve de son vote ; cela empêche la coercition et l'achat de vote

Les propriétés requises pour le vote électronique

- **Universellement vérifiable** : chaque acteur peut vérifier la présence et la validité de n'importe quel vote
- **Éligibilité** : seulement les votes corrects venant d'électeurs légitimes sont pris en compte
- **Secret du vote** : personne ne peut lier un vote et la personne à son origine
- **Pas de résultat partiel** : les assesseurs ne doivent pas publier de résultats avant la fin de l'élection
- **Sans reçu** : un électeur ne peut pas fournir la preuve de son vote ; cela empêche la coercition et l'achat de vote
- **Pas de vote double** : un électeur ne peut pas voter deux fois ou plus

Les propriétés requises pour le vote électronique

- **Universellement vérifiable** : chaque acteur peut vérifier la présence et la validité de n'importe quel vote
- **Éligibilité** : seulement les votes corrects venant d'électeurs légitimes sont pris en compte
- **Secret du vote** : personne ne peut lier un vote et la personne à son origine
- **Pas de résultat partiel** : les assesseurs ne doivent pas publier de résultats avant la fin de l'élection
- **Sans reçu** : un électeur ne peut pas fournir la preuve de son vote ; cela empêche la coercition et l'achat de vote
- **Pas de vote double** : un électeur ne peut pas voter deux fois ou plus
- **Simplicité** : l'électeur doit avoir seulement une interaction avec le système de vote

Un outil : le chiffrement homomorphe

- Reprenons le schéma R.S.A. : soient a, b deux clairs et c_a, c_b leur chiffré respectif. Calculez $c_a \times c_b$. Que remarquez vous ?

Un outil : le chiffrement homomorphe

- Reprenons le schéma R.S.A. : soient a, b deux clairs et c_a, c_b leur chiffré respectif. Calculez $c_a \times c_b$. Que remarquez vous ?

$$c_a \times c_b \equiv a^e \cdot b^e \mod N \equiv (ab)^e \mod N = c_{ab}$$

Un outil : le chiffrement homomorphe

- Reprenons le schéma R.S.A. : soient a, b deux clairs et c_a, c_b leur chiffré respectif. Calculez $c_a \times c_b$. Que remarquez vous ?

$$c_a \times c_b \equiv a^e \cdot b^e \mod N \equiv (ab)^e \mod N = c_{ab}$$

- On a une propriété multiplicative : on dit que R.S.A. est un schéma multiplicativement **homomorphe** !

Un outil : le chiffrement homomorphe

- Reprenons le schéma R.S.A. : soient a, b deux clairs et c_a, c_b leur chiffré respectif. Calculez $c_a \times c_b$. Que remarquez vous ?

$$c_a \times c_b \equiv a^e \cdot b^e \mod N \equiv (ab)^e \mod N = c_{ab}$$

- On a une propriété multiplicative : on dit que R.S.A. est un schéma multiplicativement **homomorphe** !

Schéma homomorphe (définition simple)

Un schéma de chiffrement Enc est dit **homomorphe** si il existe deux opérations $*, \times$ telles que $\forall m_1, m_2$, $\text{Enc}(m_1) * \text{Enc}(m_2) = \text{Enc}(m_1 \times m_2)$

Un outil : le chiffrement homomorphe

- Reprenons le schéma R.S.A. : soient a, b deux clairs et c_a, c_b leur chiffré respectif. Calculez $c_a \times c_b$. Que remarquez vous ?

$$c_a \times c_b \equiv a^e \cdot b^e \mod N \equiv (ab)^e \mod N = c_{ab}$$

- On a une propriété multiplicative : on dit que R.S.A. est un schéma multiplicativement **homomorphe** !

Schéma homomorphe (définition simple)

Un schéma de chiffrement Enc est dit **homomorphe** si il existe deux opérations $*, \times$ telles que $\forall m_1, m_2$, $\text{Enc}(m_1) * \text{Enc}(m_2) = \text{Enc}(m_1 \times m_2)$

- Pour R.S.A. $* = \times$: multiplication

Un outil : le chiffrement homomorphe

- Reprenons le schéma R.S.A. : soient a, b deux clairs et c_a, c_b leur chiffré respectif. Calculez $c_a \times c_b$. Que remarquez vous ?

$$c_a \times c_b \equiv a^e \cdot b^e \mod N \equiv (ab)^e \mod N = c_{ab}$$

- On a une propriété multiplicative : on dit que R.S.A. est un schéma multiplicativement **homomorphe** !

Schéma homomorphe (définition simple)

Un schéma de chiffrement Enc est dit **homomorphe** si il existe deux opérations $*, \times$ telles que $\forall m_1, m_2$, $\text{Enc}(m_1) * \text{Enc}(m_2) = \text{Enc}(m_1 \times m_2)$

- Pour R.S.A. $* = \times$: multiplication

- La définition se généralise facilement pour $k \geq 2$ messages

Chiffrement homomorphe et vote électronique : le cas du référendum

- Prenons le cas d'un référendum : pour voter « oui » on vote 1, et pour voter « non » on vote 0

Chiffrement homomorphe et vote électronique : le cas du référendum

- Prenons le cas d'un référendum : pour voter « oui » on vote 1, et pour voter « non » on vote 0

- On utilise un schéma homomorphe (à clé publique) avec $\times$ qui est l'opération d'addition

Chiffrement homomorphe et vote électronique : le cas du référendum

- Prenons le cas d'un référendum : pour voter « oui » on vote 1, et pour voter « non » on vote 0

- On utilise un schéma homomorphe (à clé publique) avec $\times$ qui est l'opération d'addition

- Chaque votant i chiffre son vote v_i (0 ou 1) avec la clé publique et transmet $\mathrm{Enc}_i(pk, v_i) = c_i$ aux assesseurs

Chiffrement homomorphe et vote électronique : le cas du référendum

- Prenons le cas d'un référendum : pour voter « oui » on vote 1, et pour voter « non » on vote 0
- On utilise un schéma homomorphe (à clé publique) avec $\times$ qui est l'opération d'addition
- Chaque votant i chiffre son vote v_i (0 ou 1) avec la clé publique et transmet $\mathrm{Enc}_i(pk, v_i) = c_i$ aux assesseurs
- A la fin du vote, les assesseurs calculent

$$\mathrm{Dec}(c_1 * c_2 * \cdots * c_N) = \mathrm{Dec}(\mathrm{Enc}(pk, (v_1 + v_2 + \cdots + v_N))) = k$$

Chiffrement homomorphe et vote électronique : le cas du référendum

- Prenons le cas d'un référendum : pour voter « oui » on vote 1, et pour voter « non » on vote 0
- On utilise un schéma homomorphe (à clé publique) avec $\times$ qui est l'opération d'addition
- Chaque votant i chiffre son vote v_i (0 ou 1) avec la clé publique et transmet $\mathrm{Enc}_i(pk, v_i) = c_i$ aux assesseurs
- A la fin du vote, les assesseurs calculent

$$\mathrm{Dec}(c_1 * c_2 * \cdots * c_N) = \mathrm{Dec}(\mathrm{Enc}(pk, (v_1 + v_2 + \cdots + v_N))) = k$$

- Si $k \geq N/2$ alors le « oui » gagne, sinon c'est le « non »

Inconvénients de cette méthode

- Avec le chiffrement homomorphe, le vote peut satisfaire certaines propriétés voulues : éligibilité, pas de double vote, simplicité, ...

Inconvénients de cette méthode

- Avec le chiffrement homomorphe, le vote peut satisfaire certaines propriétés voulues : éligibilité, pas de double vote, simplicité, ...

- Le problème c'est qu'un votant mal intentionné peut tricher facilement (plus ou moins) : en votant un grand nombre positif il favorise le ≪ oui ≫, en votant un grand nombre négatif il favorise le ≪ non ≫

Inconvénients de cette méthode

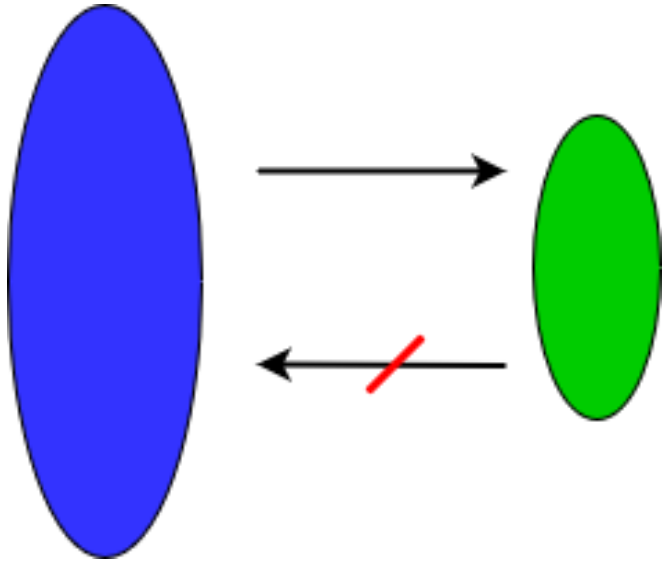
- Avec le chiffrement homomorphe, le vote peut satisfaire certaines propriétés voulues : éligibilité, pas de double vote, simplicité, ...
- Le problème c'est qu'un votant mal intentionné peut tricher facilement (plus ou moins) : en votant un grand nombre positif il favorise le « oui », en votant un grand nombre négatif il favorise le « non »
- De plus, le chiffrement homomorphe est très lourd en terme de calcul : il est donc inefficace en réalité

Fonctions de hachage

- C'est une fonction f à **sens unique**

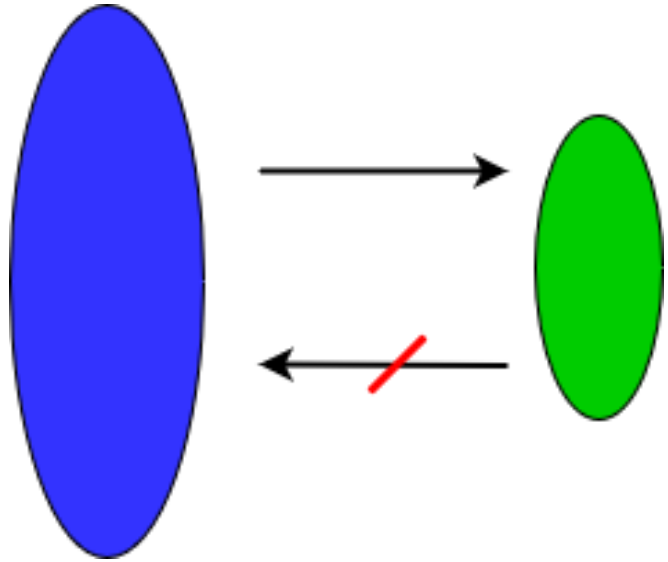
Fonctions de hachage

- C'est une fonction f à **sens unique**



Fonctions de hachage

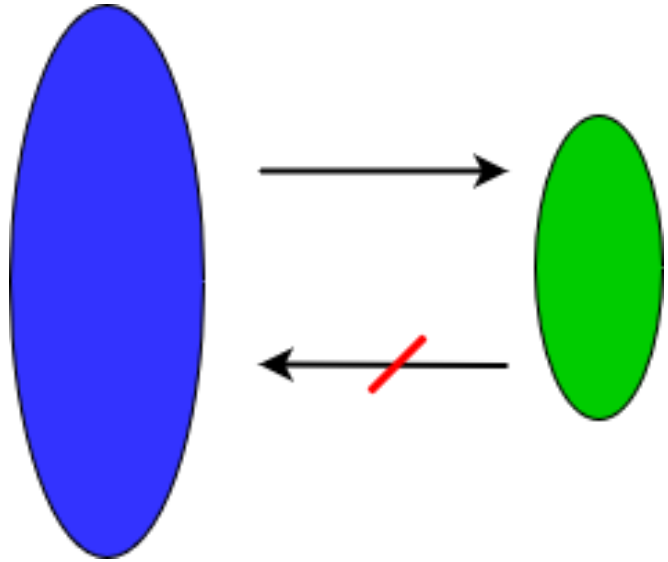
- C'est une fonction f à **sens unique**



- Fonction à sens unique = dure à inverser

Fonctions de hachage

- C'est une fonction f à **sens unique**

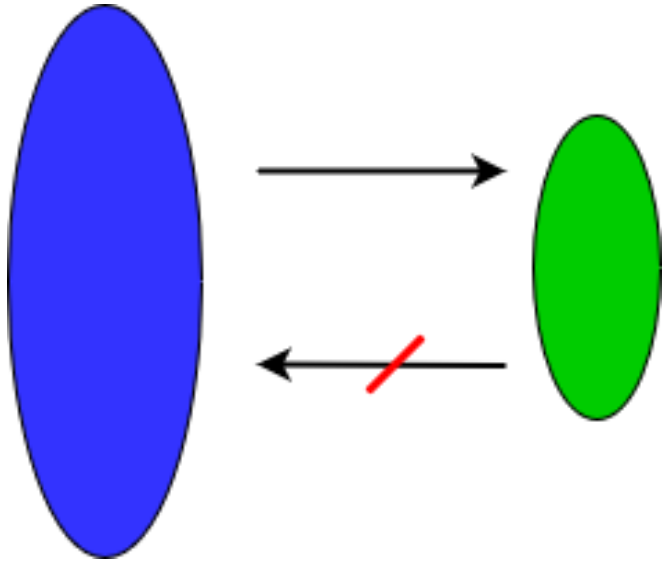


- Fonction à sens unique = dure à inverser

- Résistante aux collisions : difficile de trouver $x \neq x^{'}$ tels que $f(x) = f(x^{'})$

Fonctions de hachage

- C'est une fonction f à **sens unique**



- Fonction à sens unique = dure à inverser
- Résistante aux collisions : difficile de trouver $x \neq x^{'}$ tels que $f(x) = f(x^{'})$
- Sont utilisées pour la compression de données

Fonction de hachage et intégrité

- Les fonctions de hachage calculent une ≪ empreinte ≫ du message passé en entrée

Fonction de hachage et intégrité

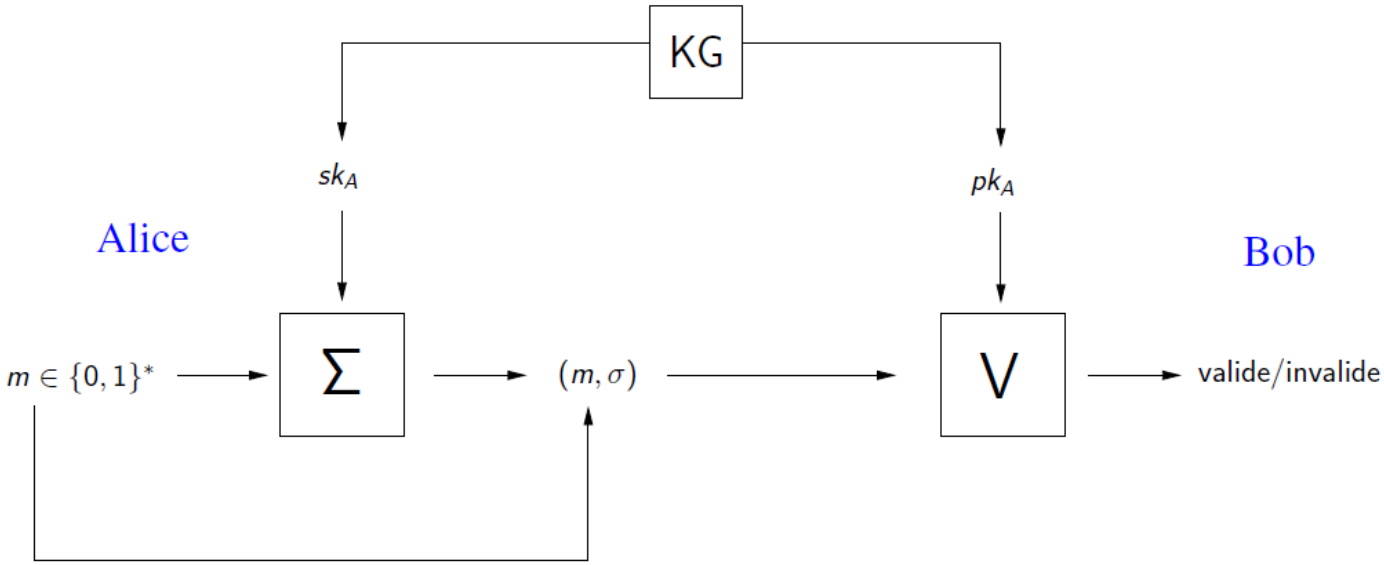
- Les fonctions de hachage calculent une « empreinte » du message passé en entrée

- Si l'on reçoit un message chiffré, et une empreinte du clair on peut vérifier une fois qu'on a déchiffré que le message envoyé n'a pas été modifié (à moins de modifier aussi l'empreinte $\rightarrow$ peut être compliqué)

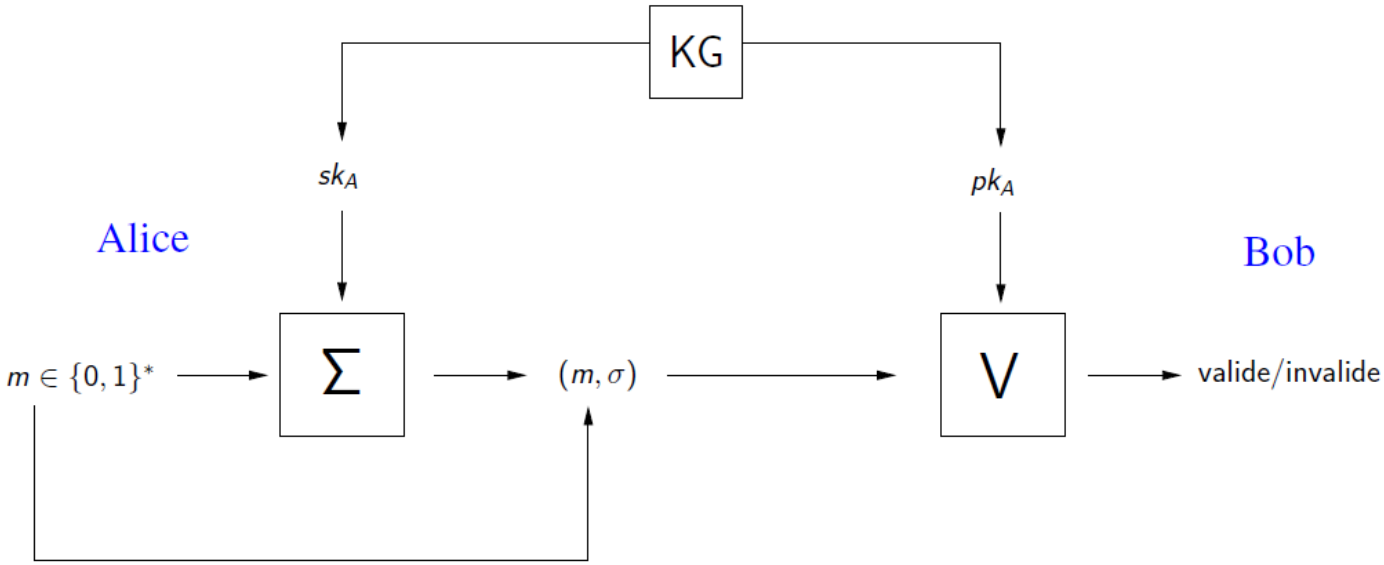
Fonction de hachage et intégrité

- Les fonctions de hachage calculent une « empreinte » du message passé en entrée
- Si l'on reçoit un message chiffré, et une empreinte du clair on peut vérifier une fois qu'on a déchiffré que le message envoyé n'a pas été modifié (à moins de modifier aussi l'empreinte $\rightarrow$ peut être compliqué)
- Lorsqu'on souhaite télécharger un fichier sur internet, on peut s'assurer qu'il n'a pas été modifié (et donc qu'il n'est pas potentiellement malveillant) en vérifiant son empreinte (si elle est fournie)

Signature



Signature



- La signature dépend du message !

Signature : propriétés

- **Authenticité** : on doit pouvoir retrouver de manière certaine l'identité du signataire

Signature : propriétés

- **Authenticité** : on doit pouvoir retrouver de manière certaine l'identité du signataire

- **Intégrité** : une fois le document signé, on ne peut plus le modifier

Signature : propriétés

- **Authenticité** : on doit pouvoir retrouver de manière certaine l'identité du signataire
- **Intégrité** : une fois le document signé, on ne peut plus le modifier
- **Non-répudiation** : la personne ayant signé le document ne peut le nier

Signature : propriétés

- **Authenticité** : on doit pouvoir retrouver de manière certaine l'identité du signataire
- **Intégrité** : une fois le document signé, on ne peut plus le modifier
- **Non-répudiation** : la personne ayant signé le document ne peut le nier
- **Résistance à la forge** (non réutilisable) : la signature fait partie du document signé, elle ne peut être réutilisée pour un autre message

Signature : propriétés

- **Authenticité** : on doit pouvoir retrouver de manière certaine l'identité du signataire
- **Intégrité** : une fois le document signé, on ne peut plus le modifier
- **Non-répudiation** : la personne ayant signé le document ne peut le nier
- **Résistance à la forge** (non réutilisable) : la signature fait partie du document signé, elle ne peut être réutilisée pour un autre message
- **Vérification universelle** : tout le monde peut vérifier la validité d'une signature

Sécurité des schémas de signatures

- Buts du forgeur F :

Sécurité des schémas de signatures

- Buts du forgeur F :
  - cassage total : F retrouve la clé secrète du signataire

Sécurité des schémas de signatures

- Buts du forgeur F :
  - cassage total : F retrouve la clé secrète du signataire
  - forge universelle : F peut signer n'importe quel message

Sécurité des schémas de signatures

- Buts du forgeur F :
  - cassage total : F retrouve la clé secrète du signataire
  - forge universelle : F peut signer n'importe quel message
  - forge selective : F peut signer un message de son choix

Sécurité des schémas de signatures

- Buts du forgeur F :
  - cassage total : F retrouve la clé secrète du signataire
  - forge universelle : F peut signer n'importe quel message
  - forge selective : F peut signer un message de son choix
  - forge existentielle : F peut générer un couple message/signature valide

Sécurité des schémas de signatures

- Buts du forgeur F :
  - cassage total : F retrouve la clé secrète du signataire
  - forge universelle : F peut signer n'importe quel message
  - forge selective : F peut signer un message de son choix
  - forge existentielle : F peut générer un couple message/signature valide

- à l'aide d' :

Sécurité des schémas de signatures

- Buts du forgeur F :
  - cassage total : F retrouve la clé secrète du signataire
  - forge universelle : F peut signer n'importe quel message
  - forge selective : F peut signer un message de son choix
  - forge existentielle : F peut générer un couple message/signature valide

- à l'aide d' :
  - une attaque sans message : F ne connaît que la clé publique

Sécurité des schémas de signatures

- Buts du forgeur F :
  - cassage total : F retrouve la clé secrète du signataire
  - forge universelle : F peut signer n'importe quel message
  - forge selective : F peut signer un message de son choix
  - forge existentielle : F peut générer un couple message/signature valide

- à l'aide d' :
  - une attaque sans message : F ne connaît que la clé publique
  - une attaque à messages connus : F a accès à une liste de couples message/signature de ce signataire

Sécurité des schémas de signatures

- Buts du forgeur F :
  - cassage total : F retrouve la clé secrète du signataire
  - forge universelle : F peut signer n'importe quel message
  - forge selective : F peut signer un message de son choix
  - forge existentielle : F peut générer un couple message/signature valide

- à l'aide d' :
  - une attaque sans message : F ne connaît que la clé publique
  - une attaque à messages connus : F a accès à une liste de couples message/signature de ce signataire
  - une attaque à messages choisis : F obtient des signatures de messages de son choix

Signature R.S.A.

- La paire de clés :

Signature R.S.A.

- La paire de clés :
  - p et q sont deux grands premiers (2048 bits)

Signature R.S.A.

- La paire de clés :
  - p et q sont deux grands premiers (2048 bits)
  - $N = pq$

Signature R.S.A.

- La paire de clés :
  - p et q sont deux grands premiers (2048 bits)
  - $N = pq$
  - e et d sont deux entiers premiers à $\phi(N) = (p-1)(q-1)$ tels que $ed = 1 \mod \phi(N)$

Signature R.S.A.

- La paire de clés :
  - p et q sont deux grands premiers (2048 bits)
  - $N = pq$
  - e et d sont deux entiers premiers à $\phi(N) = (p-1)(q-1)$ tels que $ed = 1 \mod \phi(N)$
  - (N, e) est la clé publique pk

Signature R.S.A.

- La paire de clés :
  - p et q sont deux grands premiers (2048 bits)
  - $N = pq$
  - e et d sont deux entiers premiers à $\phi(N) = (p-1)(q-1)$ tels que $ed = 1 \mod \phi(N)$
  - (N, e) est la clé publique pk
  - (d, p, q) est la clé secrète sk

Signature R.S.A.

- La paire de clés :
  - p et q sont deux grands premiers (2048 bits)
  - $N = pq$
  - e et d sont deux entiers premiers à $\phi(N) = (p-1)(q-1)$ tels que $ed = 1 \mod \phi(N)$
  - (N, e) est la clé publique pk
  - (d, p, q) est la clé secrète sk
  - $h : \{0,1\}^* \leftarrow \mathbb{Z}/N\mathbb{Z}$ une fonction de hachage (connue)

Signature R.S.A.

- La paire de clés :
  - p et q sont deux grands premiers (2048 bits)
  - $N = pq$
  - e et d sont deux entiers premiers à $\phi(N) = (p-1)(q-1)$ tels que $ed = 1 \mod \phi(N)$
  - (N, e) est la clé publique pk
  - (d, p, q) est la clé secrète sk
  - $h : \{0,1\}^* \leftarrow \mathbb{Z}/N\mathbb{Z}$ une fonction de hachage (connue)

- Pour signer un message $m \in \mathbb{Z}/N\mathbb{Z}$: $\sigma = h(m)^d \mod N$

Signature R.S.A.

- La paire de clés :
  - p et q sont deux grands premiers (2048 bits)
  - $N = pq$
  - e et d sont deux entiers premiers à $\phi(N) = (p-1)(q-1)$ tels que $ed = 1 \mod \phi(N)$
  - (N, e) est la clé publique pk
  - (d, p, q) est la clé secrète sk
  - $h : \{0,1\}^* \leftarrow \mathbb{Z}/N\mathbb{Z}$ une fonction de hachage (connue)
- Pour signer un message $m \in \mathbb{Z}/N\mathbb{Z}$: $\sigma = h(m)^d \mod N$
- Vérification : σ valide $\iff h(m) = \sigma^e = (h(m)^d)^e = h(m)$

Variations des signatures

- Signature **aveugle** : signer un document masqué et garantir l'anonymat ; Aucune connaissance de son contenu par le signataire

Variations des signatures

- Signature **aveugle** : signer un document masqué et garantir l'anonymat ; Aucune connaissance de son contenu par le signataire

- Signature **de groupe** :
  1. seulement les membres du groupe peuvent signer
  2. Destinataire doit pouvoir vérifier la validité de la signature sans obtenir l'identité du signataire
  3. En cas de force majeur : le signataire peut être retrouvé

Variations des signatures

- Signature **aveugle** : signer un document masqué et garantir l'anonymat ; Aucune connaissance de son contenu par le signataire

- Signature **de groupe** :
  1. seulement les membres du groupe peuvent signer
  2. Destinataire doit pouvoir vérifier la validité de la signature sans obtenir l'identité du signataire
  3. En cas de force majeur : le signataire peut être retrouvé

- Signature **d'anneau** : signature de groupe sans possibilité d'identifier le signataire

Variations des signatures

- Signature **aveugle** : signer un document masqué et garantir l'anonymat ; Aucune connaissance de son contenu par le signataire

- Signature **de groupe** :
  1. seulement les membres du groupe peuvent signer
  2. Destinataire doit pouvoir vérifier la validité de la signature sans obtenir l'identité du signataire
  3. En cas de force majeur : le signataire peut être retrouvé

- Signature **d'anneau** : signature de groupe sans possibilité d'identifier le signataire

- Signature « **K parmi N** » : signature valable uniquement si au moins K membres parmi N membres d'entreprises signent le document

Variations des signatures

- Signature **aveugle** : signer un document masqué et garantir l'anonymat ; Aucune connaissance de son contenu par le signataire

- Signature **de groupe** :
  1. seulement les membres du groupe peuvent signer
  2. Destinataire doit pouvoir vérifier la validité de la signature sans obtenir l'identité du signataire
  3. En cas de force majeur : le signataire peut être retrouvé

- Signature **d'anneau** : signature de groupe sans possibilité d'identifier le signataire

- Signature « **K parmi N** » : signature valable uniquement si au moins K membres parmi N membres d'entreprises signent le document

- **Double signature** : les données sont séparées : données de paiement et données de commande.

Preuve de connaissance à divulgation nulle

Protocole zero-knowledge / Preuve de connaissance à divulgation nulle

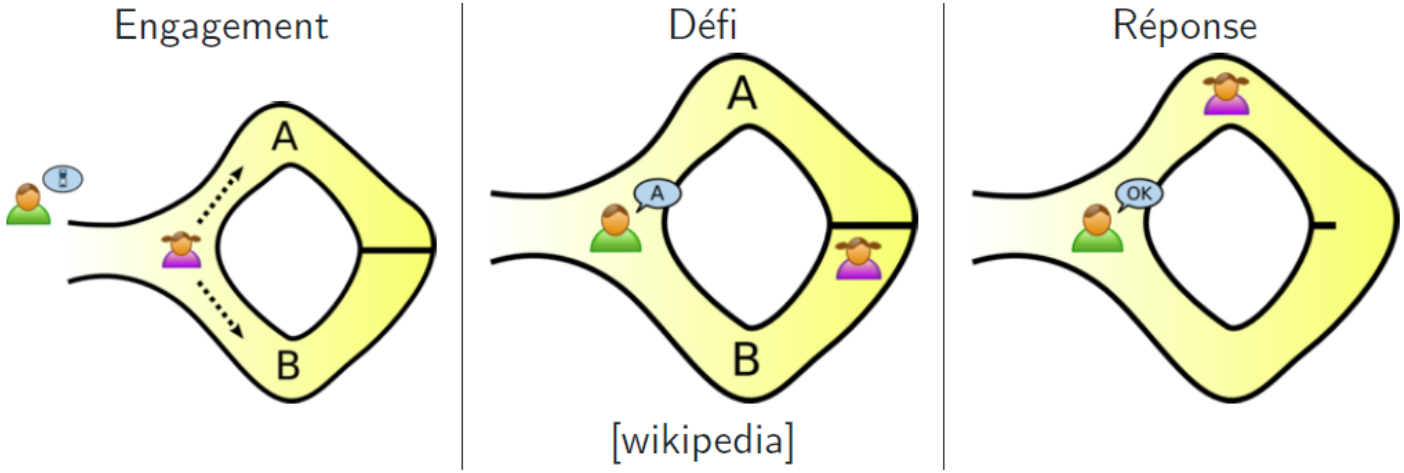
Dans un protocole zero-knowledge (ZK), un prouveur prouve mathématiquement auprès d'un vérifieur qu'il connaît un secret/une assertion sans en révéler la moindre information (idéal pour l'authentification).

Preuve de connaissance à divulgation nulle

Protocole zero-knowledge / Preuve de connaissance à divulgation nulle

Dans un protocole zero-knowledge (ZK), un prouveur prouve mathématiquement auprès d'un vérifieur qu'il connaît un secret/une assertion sans en révéler la moindre information (idéal pour l'authentification).

- *Exemple* (classique) : Alice (prouveur) veut prouver à Bob (vérifieur) qu'elle connaît la clé.



Le protocole d'authentification de Fiat-Shamir

- Génération des clés

Le protocole d'authentification de Fiat-Shamir

- Génération des clés
  - $N = pq$ avec p, q deux grands premiers

Le protocole d'authentification de Fiat-Shamir

- Génération des clés
  - $N = pq$ avec p, q deux grands premiers
  - a random dans $[0, N-1]$, $A = a^2 \mod N$ (a est la racine carrée de A)

Le protocole d'authentification de Fiat-Shamir

- Génération des clés
  - $N = pq$ avec p, q deux grands premiers
  - a random dans $[0, N-1]$, $A = a^2 \mod N$ (a est la racine carrée de A)
  - *Clé publique* : (A, N) *Clé privée* : (a, p, q)

Le protocole d'authentification de Fiat-Shamir

- Génération des clés
  - $N = pq$ avec p, q deux grands premiers
  - a random dans $[0, N-1]$, $A = a^2 \mod N$ (a est la racine carrée de A)
  - *Clé publique* : (A, N) *Clé privée* : (a, p, q)
- Le but du prouver P : prouver qu'il connait a sans le révéler à vérifieur V

Le protocole d'authentification de Fiat-Shamir

- Génération des clés
  - $N = pq$ avec p, q deux grands premiers
  - a random dans $[0, N-1]$, $A = a^2 \mod N$ (a est la racine carrée de A)
  - *Clé publique* : (A, N) *Clé privée* : (a, p, q)
- Le but du prouver P : prouver qu'il connait a sans le révéler à vérifieur V

- Engagement : $k \in [0, N-1]$ random, P envoie $K = k^2 \mod N$ à V

Le protocole d'authentification de Fiat-Shamir

- Génération des clés
  - $N = pq$ avec p, q deux grands premiers
  - a random dans $[0, N-1]$, $A = a^2 \mod N$ (a est la racine carrée de A)
  - *Clé publique* : (A, N) *Clé privée* : (a, p, q)
- Le but du prouver P : prouver qu'il connait a sans le révéler à vérifieur V

- Engagement : $k \in [0, N-1]$ random, P envoie $K = k^2 \mod N$ à V
- Défi : V choisi $r \in \{0, 1\}$ et l'envoie à P

Le protocole d'authentification de Fiat-Shamir

- Génération des clés
  - $N = pq$ avec p, q deux grands premiers
  - a random dans $[0, N-1]$, $A = a^2 \mod N$ (a est la racine carrée de A)
  - *Clé publique* : (A, N) *Clé privée* : (a, p, q)
- Le but du prouver P : prouver qu'il connait a sans le révéler à vérifieur V

- Engagement : $k \in [0, N-1]$ random, P envoie $K = k^2 \mod N$ à V
- Défi : V choisi $r \in \{0, 1\}$ et l'envoie à P
- Réponse : $y = ka^r \mod N$

Le protocole d'authentification de Fiat-Shamir

- Génération des clés
  - $N = pq$ avec p, q deux grands premiers
  - a random dans $[0, N-1]$, $A = a^2 \mod N$ (a est la racine carrée de A)
  - *Clé publique* : (A, N) *Clé privée* : (a, p, q)
- Le but du prouver P : prouver qu'il connait a sans le révéler à vérifieur V

- Engagement : $k \in [0, N-1]$ random, P envoie $K = k^2 \mod N$ à V
- Défi : V choisi $r \in \{0, 1\}$ et l'envoie à P
- Réponse : $y = ka^r \mod N$
- Vérification : $y^2 = KA^r \mod N$

Le protocole d'authentification de Fiat-Shamir

- Génération des clés
  - $N = pq$ avec p, q deux grands premiers
  - a random dans $[0, N-1]$, $A = a^2 \mod N$ (a est la racine carrée de A)
  - *Clé publique* : (A, N) *Clé privée* : (a, p, q)
- Le but du prouver P : prouver qu'il connait a sans le révéler à vérifieur V

- Engagement : $k \in [0, N-1]$ random, P envoie $K = k^2 \mod N$ à V
- Défi : V choisi $r \in \{0, 1\}$ et l'envoie à P
- Réponse : $y = ka^r \mod N$
- Vérification : $y^2 = KA^r \mod N$
- En effet, $y^2 = (ka^r)^2 = k^2 a^{2r} = KA^r$ si P connait a

Private Information Retrieval

- On a une base de données

Private Information Retrieval

- On a une base de données
- On souhaite obtenir l'élement x de la base de données

Private Information Retrieval

- On a une base de données
- On souhaite obtenir l'élement x de la base de données
- Mais sans révéler à la BDD quel élément on souhaite

Private Information Retrieval

- On a une base de données
- On souhaite obtenir l'élement x de la base de données
- Mais sans révéler à la BDD quel élément on souhaite
- Possible grâce au Private Information Retrieval (PIR)

Private Information Retrieval

- On a une base de données
- On souhaite obtenir l'élement x de la base de données
- Mais sans révéler à la BDD quel élément on souhaite
- Possible grâce au Private Information Retrieval (PIR)
- Problème : on doit interroger tous les bits de la BDD

Private Set Intersection

- Soient S_1, S_2 deux ensembles

Private Set Intersection

- Soient S_1, S_2 deux ensembles
- But : connaître $S_1 \cap S_2$ en connaissant seulement un des deux et sans révéler l'autre

Private Set Intersection

- Soient S_1, S_2 deux ensembles
- But : connaître $S_1 \cap S_2$ en connaissant seulement un des deux et sans révéler l'autre
- *Exemple* : le FBI possède une liste L_1 de potentiels terroristes, et une compagnie aérienne possède une liste L_2 des passagers d'un vol. La compagnie souhaite savoir si un de ses passagers est sur la liste du FBI, et le FBI ne souhaite pas dévoiler sa liste. Comment calculer $L_1 \cap L_2$?

Private Set Intersection

- Soient S_1, S_2 deux ensembles
- But : connaître $S_1 \cap S_2$ en connaissant seulement un des deux et sans révéler l'autre
- *Exemple* : le FBI possède une liste L_1 de potentiels terroristes, et une compagnie aérienne possède une liste L_2 des passagers d'un vol. La compagnie souhaite savoir si un de ses passagers est sur la liste du FBI, et le FBI ne souhaite pas dévoiler sa liste. Comment calculer $L_1 \cap L_2$?
- Possible grâce au Private Set Intersection (PSI)

PSI : un exemple

- Alice possède l'ensemble de valeurs $S_1 = \{2, 3, 5\}$ et Bob l'ensemble de valeurs $S_2 = \{3, 5, 7\}$

PSI : un exemple

- Alice possède l'ensemble de valeurs $S_1 = \{2, 3, 5\}$ et Bob l'ensemble de valeurs $S_2 = \{3, 5, 7\}$
- Alice veut connaître $S_1 \cap S_2$ sans connaître S_2

PSI : un exemple

- Alice possède l'ensemble de valeurs $S_1 = \{2, 3, 5\}$ et Bob l'ensemble de valeurs $S_2 = \{3, 5, 7\}$
- Alice veut connaître $S_1 \cap S_2$ sans connaître S_2
- Alice calcule le polynôme $P[X] = (X - 2)(X - 3)(X - 5) = X^3 - 10X^2 + 31X - 30 = m_3X^3 + m_2X^2 + m_1X + m_0$

PSI : un exemple

- Alice possède l'ensemble de valeurs $S_1 = \{2, 3, 5\}$ et Bob l'ensemble de valeurs $S_2 = \{3, 5, 7\}$
- Alice veut connaître $S_1 \cap S_2$ sans connaître S_2
- Alice calcule le polynôme $P[X] = (X - 2)(X - 3)(X - 5) = X^3 - 10X^2 + 31X - 30 = m_3X^3 + m_2X^2 + m_1X + m_0$
- Alice possède un algorithme de chiffrement $\mathcal{E}$ **homomorphe** pour l'addition

PSI : un exemple

- Alice possède l'ensemble de valeurs $S_1 = \{2, 3, 5\}$ et Bob l'ensemble de valeurs $S_2 = \{3, 5, 7\}$
- Alice veut connaître $S_1 \cap S_2$ sans connaître S_2
- Alice calcule le polynôme $P[X] = (X - 2)(X - 3)(X - 5) = X^3 - 10X^2 + 31X - 30 = m_3X^3 + m_2X^2 + m_1X + m_0$
- Alice possède un algorithme de chiffrement $\mathcal{E}$ **homomorphe** pour l'addition
- Alice chiffre m_0, m_1, m_2, m_3 et envoie les chiffrés à Bob

PSI : un exemple

- Alice possède l'ensemble de valeurs $S_1 = \{2, 3, 5\}$ et Bob l'ensemble de valeurs $S_2 = \{3, 5, 7\}$
- Alice veut connaître $S_1 \cap S_2$ sans connaître S_2
- Alice calcule le polynôme $P[X] = (X-2)(X-3)(X-5) = X^3 - 10X^2 + 31X - 30 = m_3X^3 + m_2X^2 + m_1X + m_0$
- Alice possède un algorithme de chiffrement $\mathcal{E}$ **homomorphe** pour l'addition
- Alice chiffre m_0, m_1, m_2, m_3 et envoie les chiffrés à Bob
- Bob calcule alors $\mathcal{E}(\mathrm{pk}_A, P(3)), \mathcal{E}(\mathrm{pk}_A, P(5)), \mathcal{E}(\mathrm{pk}_A, P(7))$

PSI : un exemple

- Alice possède l'ensemble de valeurs $S_1 = \{2, 3, 5\}$ et Bob l'ensemble de valeurs $S_2 = \{3, 5, 7\}$
- Alice veut connaître $S_1 \cap S_2$ sans connaître S_2
- Alice calcule le polynôme $P[X] = (X-2)(X-3)(X-5) = X^3 - 10X^2 + 31X - 30 = m_3X^3 + m_2X^2 + m_1X + m_0$
- Alice possède un algorithme de chiffrement $\mathcal{E}$ **homomorphe** pour l'addition
- Alice chiffre m_0, m_1, m_2, m_3 et envoie les chiffrés à Bob
- Bob calcule alors $\mathcal{E}(\mathrm{pk}_A, P(3)), \mathcal{E}(\mathrm{pk}_A, P(5)), \mathcal{E}(\mathrm{pk}_A, P(7))$
- Avec sa clé secrète, Alice déchiffre tous les résultats

Pourquoi ça fonctionne ?

- Si élément a de S_2 appartient à S_1 alors $P(a) = 0$ car a racine de $P[Z]$.

Pourquoi ça fonctionne ?

- Si élément a de S_2 appartient à S_1 alors $P(a) = 0$ car a racine de $P[Z]$.
- Vérifions :

Pourquoi ça fonctionne ?

- Si élément a de S_2 appartient à S_1 alors $P(a) = 0$ car a racine de $P[Z]$.
- Vérifions :
  - $P(3) = 3^3 - 10 \cdot 3^2 + 31 \cdot 3 - 30 = 0$

Pourquoi ça fonctionne ?

- Si élément a de S_2 appartient à S_1 alors $P(a) = 0$ car a racine de $P[Z]$.
- Vérifions :
  - $P(3) = 3^3 - 10 \cdot 3^2 + 31 \cdot 3 - 30 = 0$
  - $P(5) = 5^3 - 10 \cdot 5^2 + 31 \cdot 5 - 30 = 0$

Pourquoi ça fonctionne ?

- Si élément a de S_2 appartient à S_1 alors $P(a) = 0$ car a racine de $P[Z]$.
- Vérifions :
  - $P(3) = 3^3 - 10 \cdot 3^2 + 31 \cdot 3 - 30 = 0$
  - $P(5) = 5^3 - 10 \cdot 5^2 + 31 \cdot 5 - 30 = 0$
  - $P(7) = 7^3 - 10 \cdot 7^2 + 31 \cdot 7 - 30 = 40$

Pourquoi ça fonctionne ?

- Si élément a de S_2 appartient à S_1 alors $P(a) = 0$ car a racine de $P[Z]$.
- Vérifions :
  - $P(3) = 3^3 - 10 \cdot 3^2 + 31 \cdot 3 - 30 = 0$
  - $P(5) = 5^3 - 10 \cdot 5^2 + 31 \cdot 5 - 30 = 0$
  - $P(7) = 7^3 - 10 \cdot 7^2 + 31 \cdot 7 - 30 = 40$
- Donc Alice en déduit que $S_1 \cdot S_2 = \{3, 5\}$...

Pourquoi ça fonctionne ?

- Si élément a de S_2 appartient à S_1 alors $P(a) = 0$ car a racine de $P[Z]$.
- Vérifions :
  - $P(3) = 3^3 - 10 \cdot 3^2 + 31 \cdot 3 - 30 = 0$
  - $P(5) = 5^3 - 10 \cdot 5^2 + 31 \cdot 5 - 30 = 0$
  - $P(7) = 7^3 - 10 \cdot 7^2 + 31 \cdot 7 - 30 = 40$
- Donc Alice en déduit que $S_1 \cdot S_2 = \{3, 5\}$...
- ... sans connaître S_2 !

Pourquoi ça fonctionne ?

- Si élément a de S_2 appartient à S_1 alors $P(a) = 0$ car a racine de $P[Z]$.
- Vérifions :
  - $P(3) = 3^3 - 10 \cdot 3^2 + 31 \cdot 3 - 30 = 0$
  - $P(5) = 5^3 - 10 \cdot 5^2 + 31 \cdot 5 - 30 = 0$
  - $P(7) = 7^3 - 10 \cdot 7^2 + 31 \cdot 7 - 30 = 40$
- Donc Alice en déduit que $S_1 \cdot S_2 = \{3, 5\}$...
- ... sans connaître S_2 !

- Pourquoi utiliser un chiffrement additivement homomorphe ?

$$\underbrace{\mathcal{E}(\mathrm{pk}_A, m_1) * \cdots * \mathcal{E}(\mathrm{pk}_A, m_1)}_{a \text{ times}} = \mathcal{E}(\mathrm{pk}_A, m_1 \cdot a)$$

Certificats et P.K.I.

- Un certificat contient :

Certificats et P.K.I.

- Un certificat contient :
  - une clé publique

Certificats et P.K.I.

- Un certificat contient :
  - une clé publique
  - un nom associé

Certificats et P.K.I.

- Un certificat contient :
  - une clé publique
  - un nom associé
  - une période de validité

Certificats et P.K.I.

- Un certificat contient :
  - une clé publique
  - un nom associé
  - une période de validité
  - l'adresse (URL) du centre de révocation

Certificats et P.K.I.

- Un certificat contient :
  - une clé publique
  - un nom associé
  - une période de validité
  - l'adresse (URL) du centre de révocation
  - la signature de ce certificat par l'autorité de certification

Certificats et P.K.I.

- Un certificat contient :
  - une clé publique
  - un nom associé
  - une période de validité
  - l'adresse (URL) du centre de révocation
  - la signature de ce certificat par l'autorité de certification

- Une infrastructure de gestion de clés publiques (PKI en anglais) est un ensemble de procédures visant à mettre en place des communications sécurisées à l'aide de certificats.

Certificats et P.K.I.

- Un certificat contient :
  - une clé publique
  - un nom associé
  - une période de validité
  - l'adresse (URL) du centre de révocation
  - la signature de ce certificat par l'autorité de certification

- Une infrastructure de gestion de clés publiques (PKI en anglais) est un ensemble de procédures visant à mettre en place des communications sécurisées à l'aide de certificats.

- Les diapositives suivantes sont issues du cours « Sécurité des Usages TIC » de P-F. Bonnefoi, Université de Limoges

Public Key Infrastructure (P.K.I.)

- Buts de la P.K.I. :

Public Key Infrastructure (P.K.I.)

- Buts de la P.K.I. :
  - Gérer les problèmes posés par le maintien de lien entre des clés publiques et des identités au travers de différentes applications. Sans PKI, il faudrait définir de nombreuses solutions de sécurité et espérer une certaine interopérabilité ainsi qu'un même niveau de protection entre elles.

Public Key Infrastructure (P.K.I.)

- Buts de la P.K.I. :
  - Gérer les problèmes posés par le maintien de lien entre des clés publiques et des identités au travers de différentes applications. Sans PKI, il faudrait définir de nombreuses solutions de sécurité et espérer une certaine interopérabilité ainsi qu'un même niveau de protection entre elles.
  - Gérer le partage de la confiance entre usagers, en utilisant un tiers pour confirmer la propriété d'un « credential », c-à-d un document conférant une identité ou une qualité, appelé certificat

Public Key Infrastructure (P.K.I.)

- Buts de la P.K.I. :
  - Gérer les problèmes posés par le maintien de lien entre des clés publiques et des identités au travers de différentes applications. Sans PKI, il faudrait définir de nombreuses solutions de sécurité et espérer une certaine interopérabilité ainsi qu'un même niveau de protection entre elles.
  - Gérer le partage de la confiance entre usagers, en utilisant un tiers pour confirmer la propriété d'un « credential », c-à-d un document conférant une identité ou une qualité, appelé certificat
  - Être reconnu « comme de confiance » par les différents usagers : un usager n'a plus à connaître directement un autre usager avec lequel il veut établir une relation de confiance :

Public Key Infrastructure (P.K.I.)

- Buts de la P.K.I. :
  - Gérer les problèmes posés par le maintien de lien entre des clés publiques et des identités au travers de différentes applications. Sans PKI, il faudrait définir de nombreuses solutions de sécurité et espérer une certaine interopérabilité ainsi qu'un même niveau de protection entre elles.
  - Gérer le partage de la confiance entre usagers, en utilisant un tiers pour confirmer la propriété d'un « credential », c-à-d un document conférant une identité ou une qualité, appelé certificat
  - Être reconnu « comme de confiance » par les différents usagers : un usager n'a plus à connaître directement un autre usager avec lequel il veut établir une relation de confiance :
    - ★ il connaît un tiers de confiance partagé avec cet autre usager ;

Public Key Infrastructure (P.K.I.)

- Buts de la P.K.I. :
  - Gérer les problèmes posés par le maintien de lien entre des clés publiques et des identités au travers de différentes applications. Sans PKI, il faudrait définir de nombreuses solutions de sécurité et espérer une certaine interopérabilité ainsi qu'un même niveau de protection entre elles.
  - Gérer le partage de la confiance entre usagers, en utilisant un tiers pour confirmer la propriété d'un « credential », c-à-d un document conférant une identité ou une qualité, appelé certificat
  - Être reconnu « comme de confiance » par les différents usagers : un usager n'a plus à connaître directement un autre usager avec lequel il veut établir une relation de confiance :
    - ★ il connaît un tiers de confiance partagé avec cet autre usager ;
    - ★ il établit un lien de confiance avec cet autre usager au travers du tiers.

Public Key Infrastructure (P.K.I.)

- Buts de la P.K.I. :
  - Gérer les problèmes posés par le maintien de lien entre des clés publiques et des identités au travers de différentes applications. Sans PKI, il faudrait définir de nombreuses solutions de sécurité et espérer une certaine interopérabilité ainsi qu'un même niveau de protection entre elles.
  - Gérer le partage de la confiance entre usagers, en utilisant un tiers pour confirmer la propriété d'un « credential », c-à-d un document conférant une identité ou une qualité, appelé certificat
  - Être reconnu « comme de confiance » par les différents usagers : un usager n'a plus à connaître directement un autre usager avec lequel il veut établir une relation de confiance :
    - ★ il connaît un tiers de confiance partagé avec cet autre usager ;
    - ★ il établit un lien de confiance avec cet autre usager au travers du tiers.
  - C'est le modèle du « tiers de confiance ».

Composants et fonctionnement d'une P.K.I.

- Composants d'une P.K.I :

Composants et fonctionnement d'une P.K.I.

- Composants d'une P.K.I :
  - des certificats électroniques ;

Composants et fonctionnement d'une P.K.I.

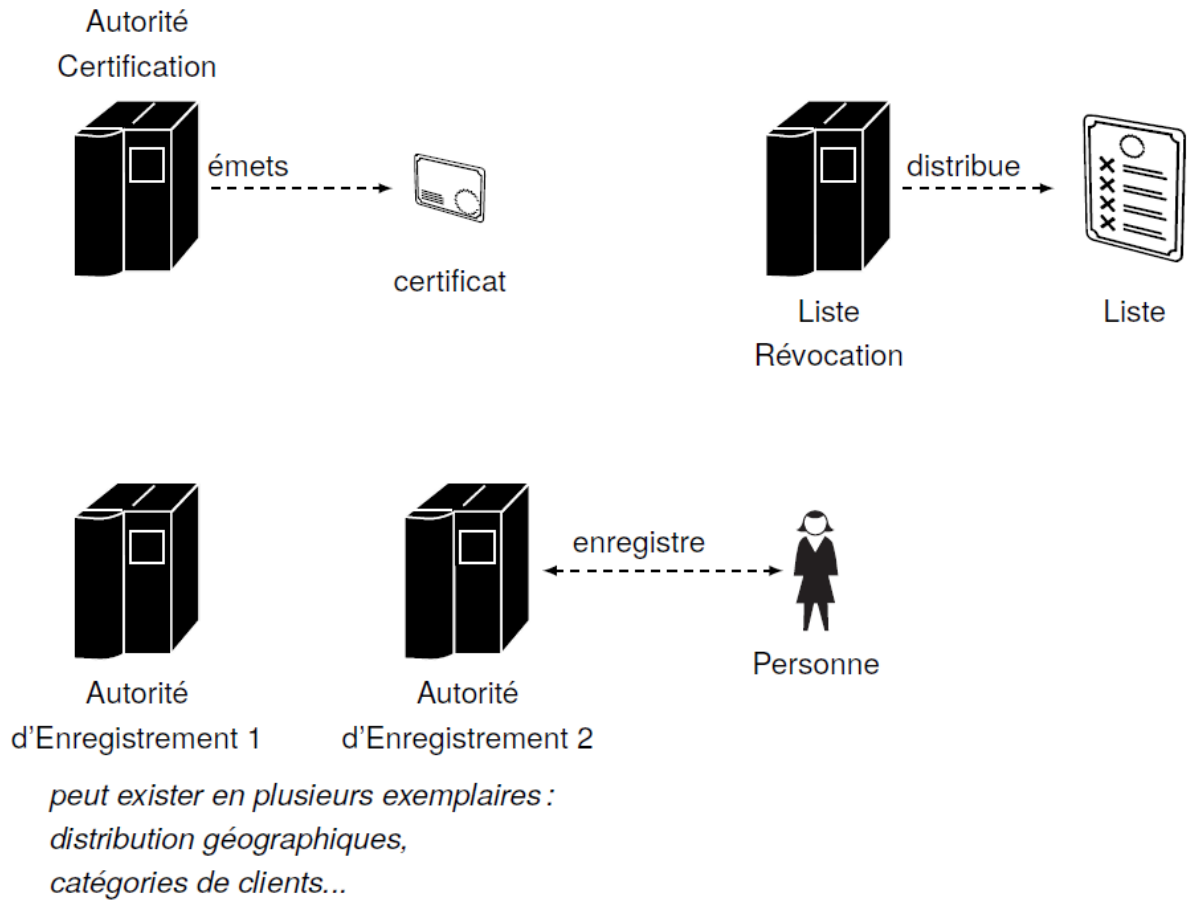
- Composants d'une P.K.I :
  - des certificats électroniques ;
  - des autorités d'enregistrement, ≪ Registration Authority ≫, et de certifications ≪ Certification Authority ≫

Composants et fonctionnement d'une P.K.I.

- Composants d'une P.K.I :
  - des certificats électroniques ;
  - des autorités d'enregistrement, ≪ Registration Authority ≫, et de certifications ≪ Certification Authority ≫
  - un procédé standardisé de vérification

Composants et fonctionnement d'une P.K.I.

- Composants d'une P.K.I :
  - des certificats électroniques ;
  - des autorités d'enregistrement, « Registration Authority », et de certifications « Certification Authority »
  - un procédé standardisé de vérification



Sommaire

Taille de clés et sécurité

Date	Symétrique	Factorisation Module	Logarithme discret Clef	Logarithme discret Groupe	Courbe elliptique GF(p)	Courbe elliptique GF(2^n)	Hash
2014 - 2020	100	2048	200	2048	200	200	200
2021 - 2030	128	2048	200	2048	256	256	256
> 2030	128	3072	200	3072	256	256	256

Recommandations de l'ANSSI (2014)

Taille de clés et sécurité

Date	Symétrique	Factorisation Module	Logarithme discret Clef	Logarithme discret Groupe	Courbe elliptique GF(p)	Courbe elliptique GF(2^n)	Hash
2014 - 2020	100	2048	200	2048	200	200	200
2021 - 2030	128	2048	200	2048	256	256	256
> 2030	128	3072	200	3072	256	256	256

Recommandations de l'ANSSI (2014)

- Les tailles de clés sont exprimées en bits, pour une sécurité minimale

Taille de clés et sécurité

Date	Symétrique	Factorisation Module	Logarithme discret Clef	Logarithme discret Groupe	Courbe elliptique GF(p)	Courbe elliptique GF(2^n)	Hash
2014 - 2020	100	2048	200	2048	200	200	200
2021 - 2030	128	2048	200	2048	256	256	256
> 2030	128	3072	200	3072	256	256	256

Recommandations de l'ANSSI (2014)

- Les tailles de clés sont exprimées en bits, pour une sécurité minimale
- Mais rappelez vous la loi de Moore : la puissance des ordinateurs double tous les 10 ans (environ) il faut donc augmenter aussi les tailles de clés pour garder la sécurité !

Taille de clés et sécurité

Date	Symétrique	Factorisation Module	Logarithme discret Clef	Logarithme discret Groupe	Courbe elliptique GF(p)	Courbe elliptique GF(2^n)	Hash
2014 - 2020	100	2048	200	2048	200	200	200
2021 - 2030	128	2048	200	2048	256	256	256
> 2030	128	3072	200	3072	256	256	256

Recommandations de l'ANSSI (2014)

- Les tailles de clés sont exprimées en bits, pour une sécurité minimale
- Mais rappelez vous la loi de Moore : la puissance des ordinateurs double tous les 10 ans (environ) il faut donc augmenter aussi les tailles de clés pour garder la sécurité !

- Cryptographie sur les courbes elliptiques a besoin de taille de clé plus petite ...

Taille de clés et sécurité

Date	Symétrique	Factorisation Module	Logarithme discret Clef	Logarithme discret Groupe	Courbe elliptique GF(p)	Courbe elliptique $GF(2^n)$	Hash
2014 - 2020	100	2048	200	2048	200	200	200
2021 - 2030	128	2048	200	2048	256	256	256
> 2030	128	3072	200	3072	256	256	256

Recommandations de l'ANSSI (2014)

- Les tailles de clés sont exprimées en bits, pour une sécurité minimale
- Mais rappelez vous la loi de Moore : la puissance des ordinateurs double tous les 10 ans (environ) il faut donc augmenter aussi les tailles de clés pour garder la sécurité !

- Cryptographie sur les courbes elliptiques a besoin de taille de clé plus petite ...
- ... mais les calculs sont plus compliqués qu'avec l'arithmétique modulaire

Courbes elliptiques

Courbe elliptique

Une courbe elliptique est l'ensemble des points vérifiant une équation du type

$$E : y^2 = x^3 + ax + b$$

auquel on ajoute un point **infini** ∞

Courbes elliptiques

Courbe elliptique

Une courbe elliptique est l'ensemble des points vérifiant une équation du type

$$E : y^2 = x^3 + ax + b$$

auquel on ajoute un point **infini** ∞

- Une telle équation est une « forme courte de Weierstrass »

Courbes elliptiques

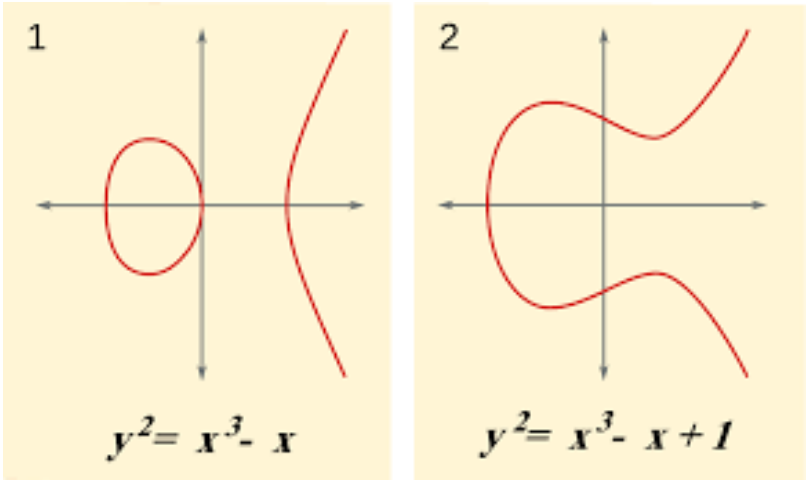
Courbe elliptique

Une courbe elliptique est l'ensemble des points vérifiant une équation du type

$$E : y^2 = x^3 + ax + b$$

auquel on ajoute un point **infini** ∞

- Une telle équation est une « forme courte de Weierstrass »
- *Exemples* :



Opérations sur les courbes elliptiques : addition

- $E : y^2 = x^3 + ax + b$

Opérations sur les courbes elliptiques : addition

- $E : y^2 = x^3 + ax + b$

- Calcul de $P_3 = P_1 + P_2$ avec $P_1 = (x_1, y_1) \in E$, $P_2 = (x_2, y_2) \in E$, $x_1 \neq x_2$

Opérations sur les courbes elliptiques : addition

- $E : y^2 = x^3 + ax + b$
- Calcul de $P_3 = P_1 + P_2$ avec $P_1 = (x_1, y_1) \in E$, $P_2 = (x_2, y_2) \in E$, $x_1 \neq x_2$
- Posons $a = \frac{y_2 - y_1}{x_2 - x_1}$, alors $x_3 = a^2 - x_1 - x_2$ et $y_3 = a(x_1 - x_3) - y_1$

Opérations sur les courbes elliptiques : addition

- $E : y^2 = x^3 + ax + b$
- Calcul de $P_3 = P_1 + P_2$ avec $P_1 = (x_1, y_1) \in E$, $P_2 = (x_2, y_2) \in E$, $x_1 \neq x_2$
- Posons $a = \frac{y_2 - y_1}{x_2 - x_1}$, alors $x_3 = a^2 - x_1 - x_2$ et $y_3 = a(x_1 - x_3) - y_1$
- Calcul de $2P_1 = P_2 = (x_2, y_2)$ avec $P_1 = (x_1, y_1) \in E$

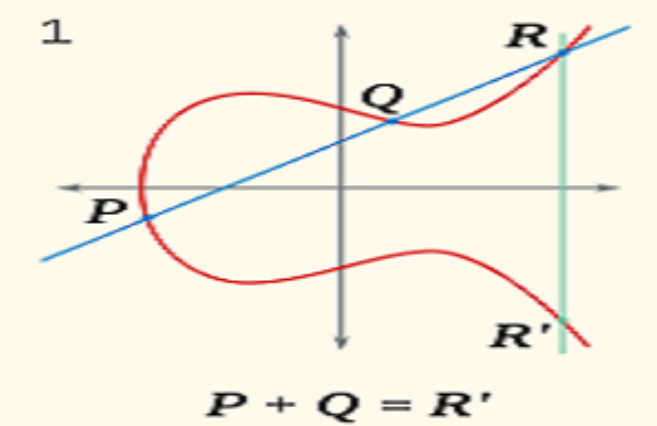
Opérations sur les courbes elliptiques : addition

- $E : y^2 = x^3 + ax + b$
- Calcul de $P_3 = P_1 + P_2$ avec $P_1 = (x_1, y_1) \in E$, $P_2 = (x_2, y_2) \in E$, $x_1 \neq x_2$
- Posons $a = \frac{y_2 - y_1}{x_2 - x_1}$, alors $x_3 = a^2 - x_1 - x_2$ et $y_3 = a(x_1 - x_3) - y_1$
- Calcul de $2P_1 = P_2 = (x_2, y_2)$ avec $P_1 = (x_1, y_1) \in E$
- Posons $a = \frac{3x_1^2 + a}{2y_1}$ alors $x_2 = a^2 - 2x_1$ et $y_2 = a(x_1 - x_2) - y_1$

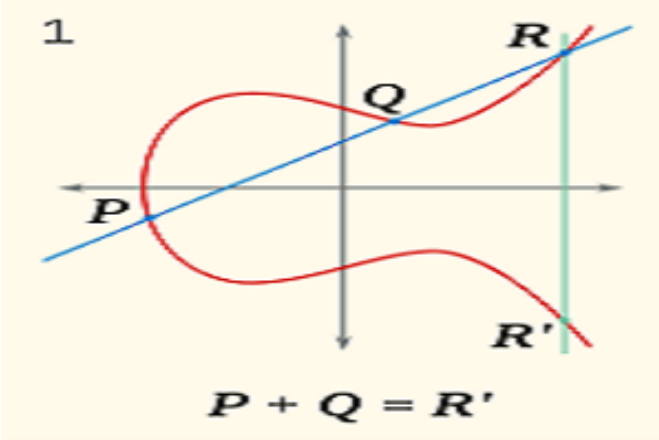
Opérations sur les courbes elliptiques : addition

- $E : y^2 = x^3 + ax + b$
- Calcul de $P_3 = P_1 + P_2$ avec $P_1 = (x_1, y_1) \in E$, $P_2 = (x_2, y_2) \in E$, $x_1 \neq x_2$
- Posons $a = \frac{y_2 - y_1}{x_2 - x_1}$, alors $x_3 = a^2 - x_1 - x_2$ et $y_3 = a(x_1 - x_3) - y_1$
- Calcul de $2P_1 = P_2 = (x_2, y_2)$ avec $P_1 = (x_1, y_1) \in E$
- Posons $a = \frac{3x_1^2 + a}{2y_1}$ alors $x_2 = a^2 - 2x_1$ et $y_2 = a(x_1 - x_2) - y_1$
- Les points d'une courbe elliptique avec l'addition forment un groupe (commutatif) dont l'élément neutre est le point à l'infini.

Addition sur courbes elliptiques : exemples

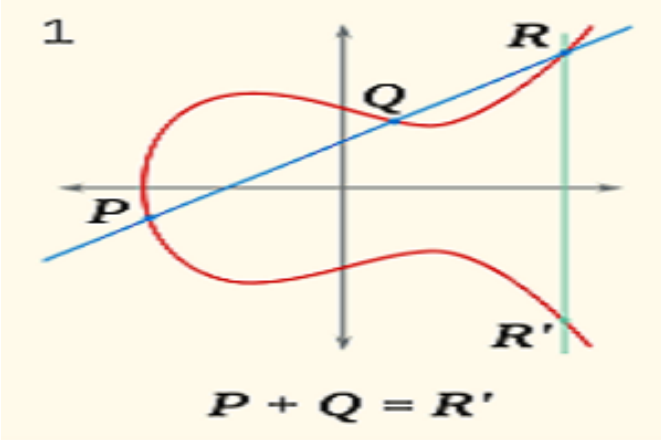


Addition sur courbes elliptiques : exemples



- Notez que ici $R^{'} = -R$

Addition sur courbes elliptiques : exemples



- Notez que ici $R^{'} = -R$
- C'est une méthode graphique pour calculer $P + Q$

El Gamal sur courbes elliptiques

- Génération des clés

El Gamal sur courbes elliptiques

- Génération des clés
  - Choisir une courbe elliptique E et un point P sur cette courbe

El Gamal sur courbes elliptiques

- Génération des clés
  - Choisir une courbe elliptique E et un point P sur cette courbe
  - Générer aléatoirement un entier r

El Gamal sur courbes elliptiques

- Génération des clés
  - Choisir une courbe elliptique E et un point P sur cette courbe
  - Générer aléatoirement un entier r
  - Calculer $B = r \cdot P = (P + P + \cdots + P \text{ r fois})$

El Gamal sur courbes elliptiques

- Génération des clés
  - Choisir une courbe elliptique E et un point P sur cette courbe
  - Générer aléatoirement un entier r
  - Calculer $B = r \cdot P = (P + P + \cdots + P \text{ r fois})$
  - pk $= (E, P, B)$ et sk $= r$

El Gamal sur courbes elliptiques

- Génération des clés
  - Choisir une courbe elliptique E et un point P sur cette courbe
  - Générer aléatoirement un entier r
  - Calculer $B = r \cdot P = (P + P + \cdots + P \text{ r fois})$
  - pk $= (E, P, B)$ et sk $= r$
- Chiffrement : le message à chiffrer m est un point de la courbe

El Gamal sur courbes elliptiques

- Génération des clés
  - Choisir une courbe elliptique E et un point P sur cette courbe
  - Générer aléatoirement un entier r
  - Calculer $B = r \cdot P = (P + P + \cdots + P \text{ r fois})$
  - pk $= (E, P, B)$ et sk $= r$
- Chiffrement : le message à chiffrer m est un point de la courbe
  - Générer un entier a

El Gamal sur courbes elliptiques

- Génération des clés
  - Choisir une courbe elliptique E et un point P sur cette courbe
  - Générer aléatoirement un entier r
  - Calculer $B = r \cdot P = (P + P + \cdots + P \text{ r fois})$
  - pk $= (E, P, B)$ et sk $= r$
- Chiffrement : le message à chiffrer m est un point de la courbe
  - Générer un entier a
  - Calculer $c_1 = a \cdot P$ et $c_2 = m + a \cdot B$

El Gamal sur courbes elliptiques

- Génération des clés
  - Choisir une courbe elliptique E et un point P sur cette courbe
  - Générer aléatoirement un entier r
  - Calculer $B = r \cdot P = (P + P + \cdots + P \text{ r fois})$
  - pk $= (E, P, B)$ et sk $= r$
- Chiffrement : le message à chiffrer m est un point de la courbe
  - Générer un entier a
  - Calculer $c_1 = a \cdot P$ et $c_2 = m + a \cdot B$
  - Message chiffré $= (c_1, c_2)$

El Gamal sur courbes elliptiques

- Génération des clés
  - Choisir une courbe elliptique E et un point P sur cette courbe
  - Générer aléatoirement un entier r
  - Calculer $B = r \cdot P = (P + P + \cdots + P \text{ r fois})$
  - pk $= (E, P, B)$ et sk $= r$
- Chiffrement : le message à chiffrer m est un point de la courbe
  - Générer un entier a
  - Calculer $c_1 = a \cdot P$ et $c_2 = m + a \cdot B$
  - Message chiffré $= (c_1, c_2)$
- Déchiffrement : calculer $d_1 = -c_1$ et $m' = c_2 + r \cdot d_1$

El Gamal sur courbes elliptiques

- Génération des clés
  - Choisir une courbe elliptique E et un point P sur cette courbe
  - Générer aléatoirement un entier r
  - Calculer $B = r \cdot P = (P + P + \cdots + P \text{ r fois})$
  - pk $= (E, P, B)$ et sk $= r$
- Chiffrement : le message à chiffrer m est un point de la courbe
  - Générer un entier a
  - Calculer $c_1 = a \cdot P$ et $c_2 = m + a \cdot B$
  - Message chiffré $= (c_1, c_2)$
- Déchiffrement : calculer $d_1 = -c_1$ et $m' = c_2 + r \cdot d_1$
- Chiffrement réussi si $m' = m$

Cryptographie sur courbes elliptiques : les couplages

Couplage

Soient $\mathbb{G}, \mathbb{G}_T$ deux groupes multiplicatif d'ordre premier p, et g un générateur de $\mathbb{G}_1$. Soit $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ une application. On dit e est un **couplage** si

1. e est **bilinéaire** : $\forall a, b \in \mathbb{Z}_p$, $g^a, g^b \in \mathbb{G}$, on a $e(g^a, g^b) = e(g, g)^{ab} = e(g, g^{ab}) = e(g^{ab}, g)$
2. e est non dégénerée, i.e. $< e(g, g) >= \mathbb{G}_T$

Cryptographie sur courbes elliptiques : les couplages

Couplage

Soient $\mathbb{G}, \mathbb{G}_T$ deux groupes multiplicatif d'ordre premier p, et g un générateur de $\mathbb{G}_1$. Soit $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ une application. On dit e est un **couplage** si

1. e est **bilinéaire** : $\forall a, b \in \mathbb{Z}_p$, $g^a, g^b \in \mathbb{G}$, on a $e(g^a, g^b) = e(g, g)^{ab} = e(g, g^{ab}) = e(g^{ab}, g)$
2. e est non dégénerée, i.e. $< e(g, g) >= \mathbb{G}_T$

- Permet de faire des schémas basés sur l'identité par exemple (IBE)

IBE de Boneh-Franklin

- $\mathbb{G} =< g >$ d'ordre p premier et e un couplage

IBE de Boneh-Franklin

- $\mathbb{G} =< g >$ d'ordre p premier et e un couplage
- s random dans $\mathbb{Z}_p$, on pose $\text{msk} = s, \text{mpk} = h = g^s$

IBE de Boneh-Franklin

- $\mathbb{G} =< g >$ d'ordre p premier et e un couplage
- s random dans $\mathbb{Z}_p$, on pose $\text{msk} = s, \text{mpk} = h = g^s$
- Pour créer clé pour identité id

IBE de Boneh-Franklin

- $\mathbb{G} =< g >$ d'ordre p premier et e un couplage
- s random dans $\mathbb{Z}_p$, on pose $\text{msk} = s, \text{mpk} = h = g^s$
- Pour créer clé pour identité id
  - Choisit $H : \{0,1\}^* \to \mathbb{G}$ fonction de hachage

IBE de Boneh-Franklin

- $\mathbb{G} =< g >$ d'ordre p premier et e un couplage
- s random dans $\mathbb{Z}_p$, on pose $\text{msk} = s, \text{mpk} = h = g^s$
- Pour créer clé pour identité id
  - Choisit $H : \{0,1\}^* \to \mathbb{G}$ fonction de hachage
  - La clé est $\text{sk}_{id} = H(id)^s$

IBE de Boneh-Franklin

- $\mathbb{G} =< g >$ d'ordre p premier et e un couplage
- s random dans $\mathbb{Z}_p$, on pose $\text{msk} = s, \text{mpk} = h = g^s$
- Pour créer clé pour identité id
  - Choisit $H : \{0,1\}^* \to \mathbb{G}$ fonction de hachage
  - La clé est $\text{sk}_{id} = H(id)^s$
- Pour chiffrer $M \in \mathbb{G}_T$ pour id :

IBE de Boneh-Franklin

- $\mathbb{G} =< g >$ d'ordre p premier et e un couplage
- s random dans $\mathbb{Z}_p$, on pose $\text{msk} = s, \text{mpk} = h = g^s$
- Pour créer clé pour identité id
  - Choisit $H : \{0,1\}^* \to \mathbb{G}$ fonction de hachage
  - La clé est $\text{sk}_{id} = H(id)^s$
- Pour chiffrer $M \in \mathbb{G}_T$ pour id :
  - r random dans $\mathbb{Z}_p$

IBE de Boneh-Franklin

- $\mathbb{G} =< g >$ d'ordre p premier et e un couplage
- s random dans $\mathbb{Z}_p$, on pose $\text{msk} = s, \text{mpk} = h = g^s$
- Pour créer clé pour identité id
  - Choisit $H : \{0,1\}^* \to \mathbb{G}$ fonction de hachage
  - La clé est $\text{sk}_{id} = H(id)^s$
- Pour chiffrer $M \in \mathbb{G}_T$ pour id :
  - r random dans $\mathbb{Z}_p$
  - $\text{ct} = (c_1, c_2) = (g^r, M \cdot e(g^s, H(id)^r))$

IBE de Boneh-Franklin

- $\mathbb{G} =< g >$ d'ordre p premier et e un couplage
- s random dans $\mathbb{Z}_p$, on pose $\text{msk} = s, \text{mpk} = h = g^s$
- Pour créer clé pour identité id
  - Choisit $H : \{0,1\}^* \to \mathbb{G}$ fonction de hachage
  - La clé est $\text{sk}_{id} = H(id)^s$
- Pour chiffrer $M \in \mathbb{G}_T$ pour id :
  - r random dans $\mathbb{Z}_p$
  - $\text{ct} = (c_1, c_2) = (g^r, M \cdot e(g^s, H(id)^r))$
- Pour déchiffrer :
  $c_2/e(c_1, \text{sk}_{id}) = M \cdot e(g^s, H(id)^r)/e(g^s, H(id)^r) = M$

Ordinateur quantique

Ordinateur quantique (définition de futura-sciences)

Un ordinateur quantique est l'équivalent des ordinateurs classiques mais qui effectuerait ses calculs en utilisant directement les lois de la physique quantique et, à la base, celle dite de superposition des états quantiques. Alors qu'un ordinateur classique manipule des bits d'information, qui sont soit des 0 soit des 1, un ordinateur quantique utilise des qubits. Ceux-ci sont des généralisations des bits classiques, qui sont en quelque sorte une superposition simultanée de ces deux états.

Ordinateur quantique

Ordinateur quantique (définition de futura-sciences)

Un ordinateur quantique est l'équivalent des ordinateurs classiques mais qui effectuerait ses calculs en utilisant directement les lois de la physique quantique et, à la base, celle dite de superposition des états quantiques. Alors qu'un ordinateur classique manipule des bits d'information, qui sont soit des 0 soit des 1, un ordinateur quantique utilise des qubits. Ceux-ci sont des généralisations des bits classiques, qui sont en quelque sorte une superposition simultanée de ces deux états.

- En 2021, IBM a construit un processeur quantique de 127 qubits

Ordinateur quantique

Ordinateur quantique (définition de futura-sciences)

Un ordinateur quantique est l'équivalent des ordinateurs classiques mais qui effectuerait ses calculs en utilisant directement les lois de la physique quantique et, à la base, celle dite de superposition des états quantiques. Alors qu'un ordinateur classique manipule des bits d'information, qui sont soit des 0 soit des 1, un ordinateur quantique utilise des qubits. Ceux-ci sont des généralisations des bits classiques, qui sont en quelque sorte une superposition simultanée de ces deux états.

- En 2021, IBM a construit un processeur quantique de 127 qubits
- Mais encore insuffisant ...

Ordinateur quantique

Ordinateur quantique (définition de futura-sciences)

Un ordinateur quantique est l'équivalent des ordinateurs classiques mais qui effectuerait ses calculs en utilisant directement les lois de la physique quantique et, à la base, celle dite de superposition des états quantiques. Alors qu'un ordinateur classique manipule des bits d'information, qui sont soit des 0 soit des 1, un ordinateur quantique utilise des qubits. Ceux-ci sont des généralisations des bits classiques, qui sont en quelque sorte une superposition simultanée de ces deux états.

- En 2021, IBM a construit un processeur quantique de 127 qubits
- Mais encore insuffisant ...
- ... même si IBM prévoit d'obtenir un processeur à plus de 1000 qubits en 2023

Algorithme de Grover

- L'algorithme de Grover, proposé en 1996 permet de trouver la solution à un problème dans un ensemble à n éléments en $O(\sqrt{N})$. Concrétement, si la recherche se faisait avec une compléxité en $O(2^n)$ avec cet algorithme elle se fait en $O(2^{n/2})$

Algorithme de Grover

- L'algorithme de Grover, proposé en 1996 permet de trouver la solution à un problème dans un ensemble à n éléments en $O(\sqrt{N})$. Concrétement, si la recherche se faisait avec une compléxité en $O(2^n)$ avec cet algorithme elle se fait en $O(2^{n/2})$

- Si un ordinateur quantique assez puissant est créé, avec cet algorithme il est possible de casser les schémas de cryptographie actuels !

Algorithme de Grover

- L'algorithme de Grover, proposé en 1996 permet de trouver la solution à un problème dans un ensemble à n éléments en $O(\sqrt{N})$. Concrétement, si la recherche se faisait avec une compléxité en $O(2^n)$ avec cet algorithme elle se fait en $O(2^{n/2})$

- Si un ordinateur quantique assez puissant est créé, avec cet algorithme il est possible de casser les schémas de cryptographie actuels !

- Il existe néanmoins certains domaines de la cryptographie dans lesquels se trouvent des problèmes supposés résistants à un ordinateur quantique : les codes correcteurs d'erreurs, les réseaux euclidiens ou encore les polynômes multivariés

Cryptographie à base de codes correcteurs

- On a vu ce qu'était les codes correcteurs en introduction

Cryptographie à base de codes correcteurs

- On a vu ce qu'était les codes correcteurs en introduction

- La sécurité de la cryptographie sur les codes repose sur le fait que le décodage d'un code (linéaire) est « difficile » (plus exactement NP difficile) en général

Cryptographie à base de codes correcteurs

- On a vu ce qu'était les codes correcteurs en introduction
- La sécurité de la cryptographie sur les codes repose sur le fait que le décodage d'un code (linéaire) est « difficile » (plus exactement NP difficile) en général
- Un code linéaire peut être décrit par une matrice (dite génératrice) : ça sera la clé publique du schéma de chiffrement

Cryptographie à base de codes correcteurs

- On a vu ce qu'était les codes correcteurs en introduction
- La sécurité de la cryptographie sur les codes repose sur le fait que le décodage d'un code (linéaire) est « difficile » (plus exactement NP difficile) en général
- Un code linéaire peut être décrit par une matrice (dite génératrice) : ça sera la clé publique du schéma de chiffrement
- Pour chiffrer un message sous forme de vecteur, on le multiplie par la matrice et on ajoute un aléa qui servira d'« erreur » (i.e. un autre vecteur)

Cryptographie à base de codes correcteurs

- On a vu ce qu'était les codes correcteurs en introduction
- La sécurité de la cryptographie sur les codes repose sur le fait que le décodage d'un code (linéaire) est « difficile » (plus exactement NP difficile) en général
- Un code linéaire peut être décrit par une matrice (dite génératrice) : ça sera la clé publique du schéma de chiffrement
- Pour chiffrer un message sous forme de vecteur, on le multiplie par la matrice et on ajoute un aléa qui servira d'« erreur » (i.e. un autre vecteur)
- Le déchiffrement se fait via le décodage du code

Codes linéaires

- $s \in \mathbb{N}^*$, p premier, $q = p^s$, $\mathbb{F}_q$ est un corps fini, $n \in \mathbb{N}^*$ et $\mathbb{F}_q^n$ est le produit cartésien $\mathbb{F}_q \times \mathbb{F}_q \times \cdots \mathbb{F}_q$ (très souvent $q = 2$)

Codes linéaires

- $s \in \mathbb{N}^*$, p premier, $q = p^s$, $\mathbb{F}_q$ est un corps fini, $n \in \mathbb{N}^*$ et $\mathbb{F}_q^n$ est le produit cartésien $\mathbb{F}_q \times \mathbb{F}_q \times \cdots \mathbb{F}_q$ (très souvent $q = 2$)

- Un code de longueur n est dit **linéaire** s'il est un sous-espace vectoriel sur $\mathbb{F}_q$ de $\mathbb{F}_q^n$.

Codes linéaires

- $s \in \mathbb{N}^*$, p premier, $q = p^s$, $\mathbb{F}_q$ est un corps fini, $n \in \mathbb{N}^*$ et $\mathbb{F}_q^n$ est le produit cartésien $\mathbb{F}_q \times \mathbb{F}_q \times \cdots \mathbb{F}_q$ (très souvent $q = 2$)

- Un code de longueur n est dit **linéaire** s'il est un sous-espace vectoriel sur $\mathbb{F}_q$ de $\mathbb{F}_q^n$.

- Autrement dit, un mot d'un code linéaire est obtenu après transformation linéaire du mot initial

Codes linéaires

- $s \in \mathbb{N}^*$, p premier, $q = p^s$, $\mathbb{F}_q$ est un corps fini, $n \in \mathbb{N}^*$ et $\mathbb{F}_q^n$ est le produit cartésien $\mathbb{F}_q \times \mathbb{F}_q \times \cdots \mathbb{F}_q$ (très souvent $q = 2$)

- Un code de longueur n est dit **linéaire** s'il est un sous-espace vectoriel sur $\mathbb{F}_q$ de $\mathbb{F}_q^n$.

- Autrement dit, un mot d'un code linéaire est obtenu après transformation linéaire du mot initial

- *Exemple :* le code binaire de longueur 7 formé des mots suivants $(a, b, c, d, b + c + d, a + c + d, a + b + d)$ avec $a, b, c, d \in \mathbb{F}_2$ est linéaire. Pourquoi ?

Codes linéaires

- $s \in \mathbb{N}^*$, p premier, $q = p^s$, $\mathbb{F}_q$ est un corps fini, $n \in \mathbb{N}^*$ et $\mathbb{F}_q^n$ est le produit cartésien $\mathbb{F}_q \times \mathbb{F}_q \times \cdots \mathbb{F}_q$ (très souvent $q = 2$)

- Un code de longueur n est dit **linéaire** s'il est un sous-espace vectoriel sur $\mathbb{F}_q$ de $\mathbb{F}_q^n$.

- Autrement dit, un mot d'un code linéaire est obtenu après transformation linéaire du mot initial

- *Exemple :* le code binaire de longueur 7 formé des mots suivants $(a, b, c, d, b + c + d, a + c + d, a + b + d)$ avec $a, b, c, d \in \mathbb{F}_2$ est linéaire. Pourquoi ?

- Addition de deux mots du code est un mot du code, et multiplication d'un mot du code par un scalaire est un mot du code

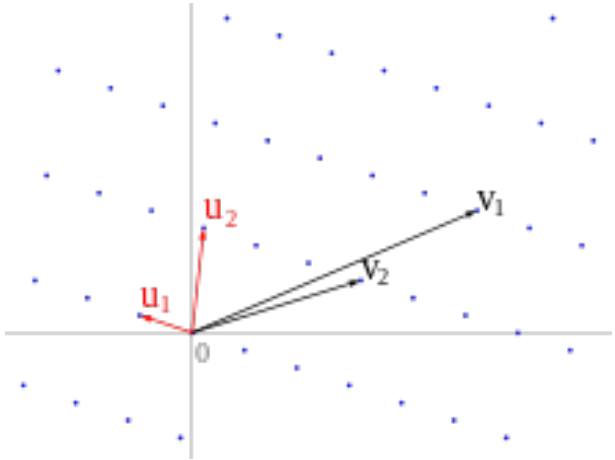
Réseaux euclidiens

- Formellement un réseau euclidien est un sous-groupe discret d'un espace (vectoriel) euclidien, de rang fini n.

Réseaux euclidiens

- Formellement un réseau euclidien est un sous-groupe discret d'un espace (vectoriel) euclidien, de rang fini n.

- De manière plus visuelle, un réseau est un ≪ pavage ≫ de l'espace :

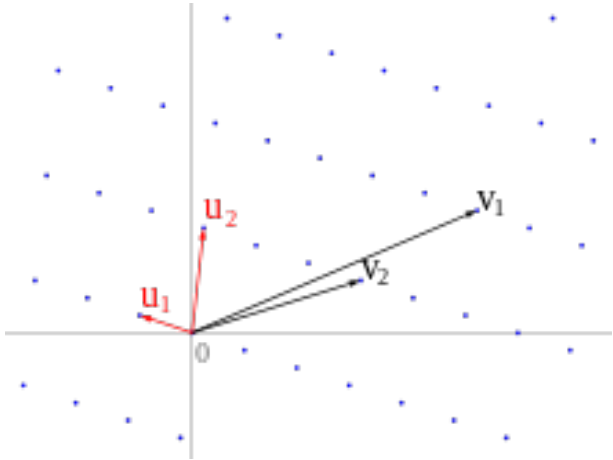


Source : wikipédia

Réseaux euclidiens

- Formellement un réseau euclidien est un sous-groupe discret d'un espace (vectoriel) euclidien, de rang fini n.

- De manière plus visuelle, un réseau est un ≪ pavage ≫ de l'espace :



Source : wikipédia

- Ils possèdent différentes bases que l'on peut qualifier de ≪ bonnes ≫ ou ≪ mauvaises ≫ : dans la figure précédente (u_1, u_2) est une bonne base car le parallélépipède qu'elle forme se rapproche du carré alors que (v_1, v_2) est une mauvaise base

Cryptographie sur les réseaux euclidiens

- Idée est d'utiliser une mauvaise base pour chiffrer et une bonne base pour déchiffrer

Cryptographie sur les réseaux euclidiens

- Idée est d'utiliser une mauvaise base pour chiffrer et une bonne base pour déchiffrer

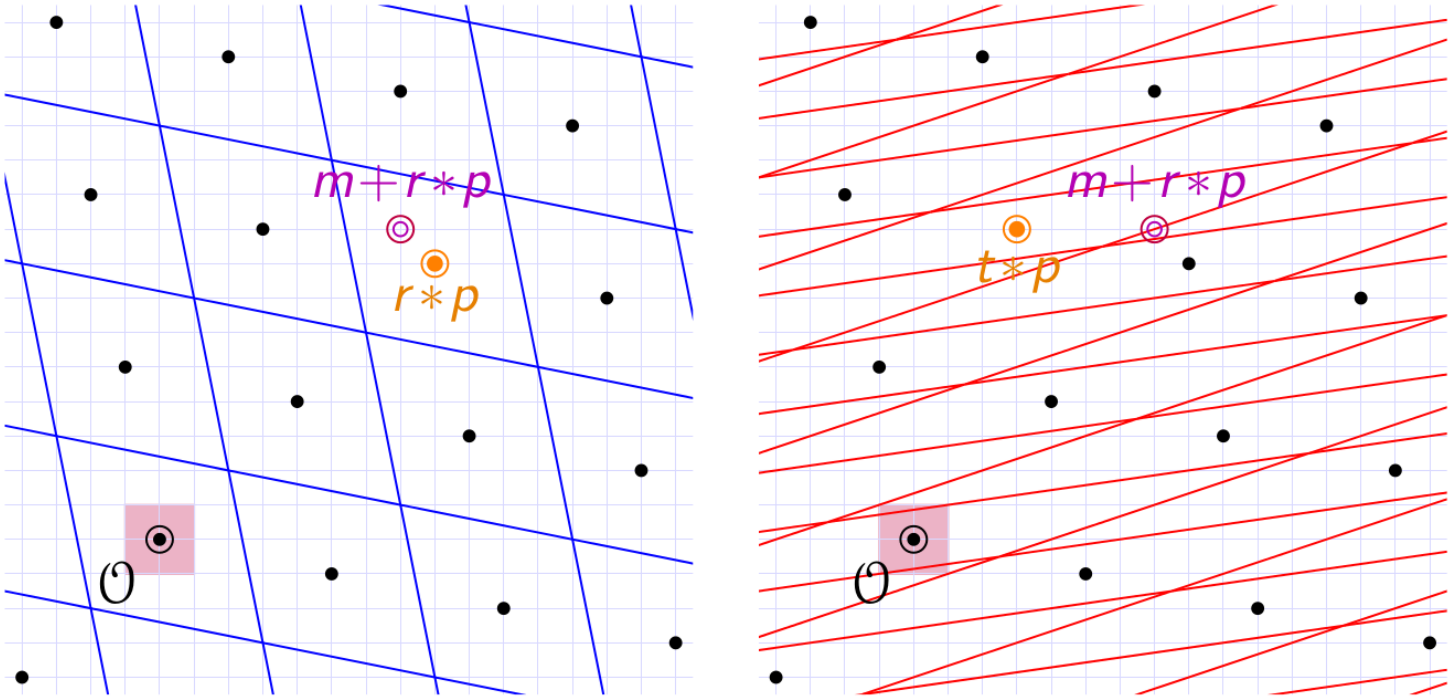


Figure – Source : CNRS

Cryptographie multivariée

- Un polynôme P est une combinaison de puissance d'une indéterminée, notée X : $P[X] = \sum_{i=0}^{l} p_i X^i = p_0 + p_1 X + p_2 X^2 + \cdots + p_l X^l$

Cryptographie multivariée

- Un polynôme P est une combinaison de puissance d'une indéterminée, notée X : $P[X] = \sum_{i=0}^{l} p_i X^i = p_0 + p_1 X + p_2 X^2 + \cdots + p_l X^l$

- Un polynôme **multivarié** est défini avec plusieurs indéterminées, notées $X_1, \cdots, X_k$

Cryptographie multivariée

- Un polynôme P est une combinaison de puissance d'une indéterminée, notée X : $P[X] = \sum_{i=0}^{l} p_i X^i = p_0 + p_1 X + p_2 X^2 + \cdots + p_l X^l$
- Un polynôme **multivarié** est défini avec plusieurs indéterminées, notées $X_1, \cdots, X_k$
- *Exemple* : $P[X, Y] = X^2 + Y^2 - 1$

Cryptographie multivariée

- Un polynôme P est une combinaison de puissance d'une indéterminée, notée X : $P[X] = \sum_{i=0}^{l} p_i X^i = p_0 + p_1 X + p_2 X^2 + \cdots + p_l X^l$
- Un polynôme **multivarié** est défini avec plusieurs indéterminées, notées $X_1, \cdots, X_k$
- *Exemple* : $P[X, Y] = X^2 + Y^2 - 1$
- La cryptographie multivariée date de 1988 (même si le schéma initial a été cassé) : FHE est un algorithme de chiffrement de cryptographie multivariée

Cryptographie multivariée

- Un polynôme P est une combinaison de puissance d'une indéterminée, notée X : $P[X] = \sum_{i=0}^{l} p_i X^i = p_0 + p_1 X + p_2 X^2 + \cdots + p_l X^l$
- Un polynôme **multivarié** est défini avec plusieurs indéterminées, notées $X_1, \cdots, X_k$
- *Exemple* : $P[X, Y] = X^2 + Y^2 - 1$
- La cryptographie multivariée date de 1988 (même si le schéma initial a été cassé) : FHE est un algorithme de chiffrement de cryptographie multivariée
- Se base sur le fait que la résolution de systèmes d'équations polynomiales est un problème « difficile » (NP difficile pour être exacte), en général

Sommaire

LFSR (1)

- Rappel : pour faire du chiffrement ONE TIME PAD (chiffrement à flot) on a besoin d'une suite chiffrante qui semble aléatoire

LFSR (1)

- Rappel : pour faire du chiffrement ONE TIME PAD (chiffrement à flot) on a besoin d'une suite chiffrante qui semble aléatoire
- Pour en créer une, on peut utiliser les LFSR (Linear Feedback Shift Register) : registre à décalage à rétroaction linéaire.

LFSR (1)

- Rappel : pour faire du chiffrement ONE TIME PAD (chiffrement à flot) on a besoin d'une suite chiffrante qui semble aléatoire
- Pour en créer une, on peut utiliser les LFSR (Linear Feedback Shift Register) : registre à décalage à rétroaction linéaire.

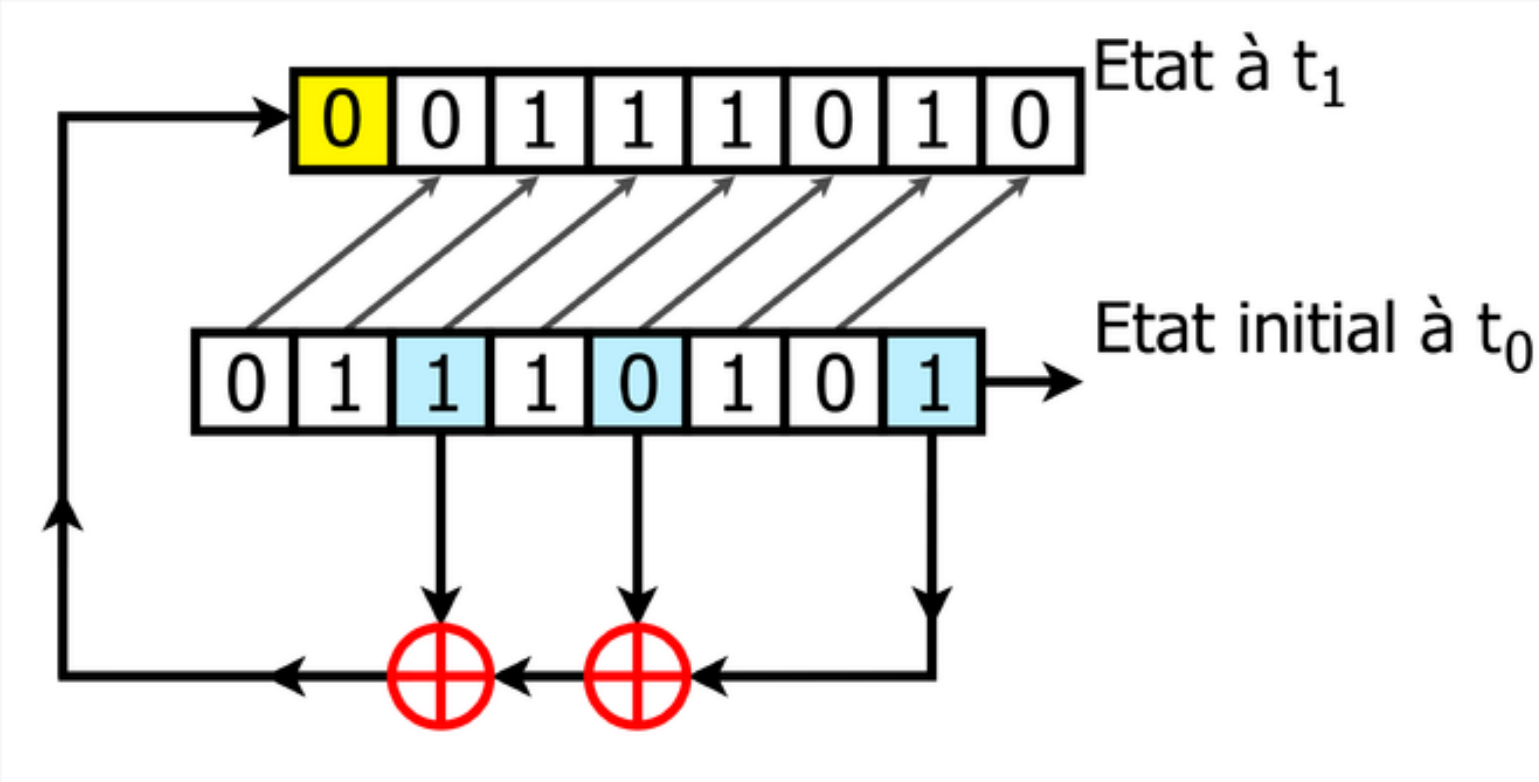


Figure – Source : wikipédia

LFSR (2)

- Un LFSR est composé de L registres (i.e. L cases) et est défini avec un polynôme (le polynôme de rétroaction) :
  $C(X) = 1 = a_1X + a_2X^2 + \cdots + a_LX^L$ (avec $a_i = 0$ ou 1).

LFSR (2)

- Un LFSR est composé de L registres (i.e. L cases) et est défini avec un polynôme (le polynôme de rétroaction) :
  $C(X) = 1 = a_1X + a_2X^2 + \cdots + a_LX^L$ (avec $a_i = 0$ ou 1).

- Les bits de la suite chiffrante sont obtenus de la manière suivante :

LFSR (2)

- Un LFSR est composé de L registres (i.e. L cases) et est défini avec un polynôme (le polynôme de rétroaction) :
  $C(X) = 1 = a_1X + a_2X^2 + \cdots + a_LX^L$ (avec $a_i = 0$ ou 1).

- Les bits de la suite chiffrante sont obtenus de la manière suivante :
  - Décalage vers la droite de une case

LFSR (2)

- Un LFSR est composé de L registres (i.e. L cases) et est défini avec un polynôme (le polynôme de rétroaction) :
  $C(X) = 1 = a_1X + a_2X^2 + \cdots + a_LX^L$ (avec $a_i = 0$ ou 1).

- Les bits de la suite chiffrante sont obtenus de la manière suivante :
  - Décalage vers la droite de une case
  - Il y a un bit de sortie : s_i

LFSR (2)

- Un LFSR est composé de L registres (i.e. L cases) et est défini avec un polynôme (le polynôme de rétroaction) :
  $C(X) = 1 = a_1X + a_2X^2 + \cdots + a_LX^L$ (avec $a_i = 0$ ou 1).

- Les bits de la suite chiffrante sont obtenus de la manière suivante :
  - Décalage vers la droite de une case
  - Il y a un bit de sortie : s_i
  - On calcule un bit de rétroaction par : $s_j = a_1s_1 \oplus \cdots \oplus a_Ls_{j-L}$.

LFSR (2)

- Un LFSR est composé de L registres (i.e. L cases) et est défini avec un polynôme (le polynôme de rétroaction) :
  $C(X) = 1 = a_1X + a_2X^2 + \cdots + a_LX^L$ (avec $a_i = 0$ ou 1).

- Les bits de la suite chiffrante sont obtenus de la manière suivante :
  - Décalage vers la droite de une case
  - Il y a un bit de sortie : s_i
  - On calcule un bit de rétroaction par : $s_j = a_1s_1 \oplus \cdots \oplus a_Ls_{j-L}$.

- ATTENTION ! On ne peut pas utiliser directement les LFSR pour du chiffrement à flot, car si on connaît $2L$ bits de clair d'affilé on peut retrouver tout message clair !

LFSR : exemple

- On prend $L = 3$ et $C(X) = 1 + X + X^3$ avec initialisation $(s_0, s_1, s_2) = (1, 1, 0)$. Calculons la suite chiffrante.

LFSR : exemple

- On prend $L = 3$ et $C(X) = 1 + X + X^3$ avec initialisation $(s_0, s_1, s_2) = (1, 1, 0)$. Calculons la suite chiffrante.

$t = 0$	1	1	0	sortie $\rightarrow 0$	rétroaction $s_0 \oplus s_2 = 1 + 0 = 1$
$t = 1$	1	1	1	sortie $\rightarrow 1$	rétroaction $s_0 \oplus s_2 = 1 + 1 = 0$
$t = 2$	0	1	1	sortie $\rightarrow 1$	rétroaction $s_0 \oplus s_2 = 0 + 1 = 1$
$t = 3$	1	0	1	sortie $\rightarrow 1$	rétroaction $s_0 \oplus s_2 = 1 + 1 = 0$
$t = 4$	0	1	0	sortie $\rightarrow 0$	rétroaction $s_0 \oplus s_2 = 0 + 0 = 0$
$t = 5$	0	0	1	sortie $\rightarrow 1$	rétroaction $s_0 \oplus s_2 = 0 + 1 = 1$
$t = 6$	1	0	0	sortie $\rightarrow 0$	rétroaction $s_0 \oplus s_2 = 1 + 0 = 1$
$t = 7$	1	1	0	sortie $\rightarrow 0$	rétroaction $s_0 \oplus s_2 = 1 + 0 = 1$
$t = 8$	1	1	1	sortie $\rightarrow 1$	rétroaction $s_0 \oplus s_2 = 1 + 1 = 0$
$t = 9$	0	1	1	sortie $\rightarrow 1$	rétroaction $s_0 \oplus s_2 = 0 + 1 = 1$

LFSR : exemple

- On prend $L = 3$ et $C(X) = 1 + X + X^3$ avec initialisation $(s_0, s_1, s_2) = (1, 1, 0)$. Calculons la suite chiffrante.

$t = 0$	1	1	0	sortie $\rightarrow 0$	rétroaction $s_0 \oplus s_2 = 1 + 0 = 1$
$t = 1$	1	1	1	sortie $\rightarrow 1$	rétroaction $s_0 \oplus s_2 = 1 + 1 = 0$
$t = 2$	0	1	1	sortie $\rightarrow 1$	rétroaction $s_0 \oplus s_2 = 0 + 1 = 1$
$t = 3$	1	0	1	sortie $\rightarrow 1$	rétroaction $s_0 \oplus s_2 = 1 + 1 = 0$
$t = 4$	0	1	0	sortie $\rightarrow 0$	rétroaction $s_0 \oplus s_2 = 0 + 0 = 0$
$t = 5$	0	0	1	sortie $\rightarrow 1$	rétroaction $s_0 \oplus s_2 = 0 + 1 = 1$
$t = 6$	1	0	0	sortie $\rightarrow 0$	rétroaction $s_0 \oplus s_2 = 1 + 0 = 1$
$t = 7$	1	1	0	sortie $\rightarrow 0$	rétroaction $s_0 \oplus s_2 = 1 + 0 = 1$
$t = 8$	1	1	1	sortie $\rightarrow 1$	rétroaction $s_0 \oplus s_2 = 1 + 1 = 0$
$t = 9$	0	1	1	sortie $\rightarrow 1$	rétroaction $s_0 \oplus s_2 = 0 + 1 = 1$

- On remarque que le nombre d'états est fini : il y a une cyclicité si on exclut le cas nul. Au plus, la période vaut $2^L - 1$.

A.E.S

Les diapositives suivantes sont issues du cours ≪ Développement de logiciels cryptographiques ≫ de Christophe Clavier

- Advanced Encryption Standard (AES)

A.E.S

Les diapositives suivantes sont issues du cours ≪ Développement de logiciels cryptographiques ≫ de Christophe Clavier

- Advanced Encryption Standard (AES)
- Standard de chiffrement du NIST en 2001

A.E.S

Les diapositives suivantes sont issues du cours ≪ Développement de logiciels cryptographiques ≫ de Christophe Clavier

- Advanced Encryption Standard (AES)
- Standard de chiffrement du NIST en 2001
- Inventé par Joan Daemen et Vincent Rijmen

A.E.S

Les diapositives suivantes sont issues du cours « Développement de logiciels cryptographiques » de Christophe Clavier

- Advanced Encryption Standard (AES)
- Standard de chiffrement du NIST en 2001
- Inventé par Joan Daemen et Vincent Rijmen
- N'est pas basé sur un schéma de Feistel mais sur un réseau de substitution/permutation

A.E.S

Les diapositives suivantes sont issues du cours « Développement de logiciels cryptographiques » de Christophe Clavier

- Advanced Encryption Standard (AES)
- Standard de chiffrement du NIST en 2001
- Inventé par Joan Daemen et Vincent Rijmen
- N'est pas basé sur un schéma de Feistel mais sur un réseau de substitution/permutation
- Trois modes sont disponibles, pour un niveau de sécurité différent :
  1. AES 128, avec 10 tours : message d'entrée de 128 bits, clé de 128 bits
  2. AES 192, avec 12 tours : message d'entrée de 128 bits, clé de 192 bits
  3. AES 256, avec 14 tours : message d'entrée de 128 bits, clé de 256 bits

A.E.S

Les diapositives suivantes sont issues du cours « Développement de logiciels cryptographiques » de Christophe Clavier

- Advanced Encryption Standard (AES)
- Standard de chiffrement du NIST en 2001
- Inventé par Joan Daemen et Vincent Rijmen
- N'est pas basé sur un schéma de Feistel mais sur un réseau de substitution/permutation
- Trois modes sont disponibles, pour un niveau de sécurité différent :
  1. AES 128, avec 10 tours : message d'entrée de 128 bits, clé de 128 bits
  2. AES 192, avec 12 tours : message d'entrée de 128 bits, clé de 192 bits
  3. AES 256, avec 14 tours : message d'entrée de 128 bits, clé de 256 bits
- Prend en entrée des éléments de $GF(2^8)$: chaque octet est un élément de $GF(2^8)$

Addition dans le corps de Galois $GF(2^8)$

- Chaque octet est représenté par un polynôme (de degré au plus 7), à coefficients dans $GF(2) = \{0, 1\}$

Addition dans le corps de Galois $GF(2^8)$

- Chaque octet est représenté par un polynôme (de degré au plus 7), à coefficients dans $GF(2) = \{0, 1\}$
- L'addition dans $GF(2^8)$ est simplement l'addition de polynômes

Addition dans le corps de Galois $GF(2^8)$

- Chaque octet est représenté par un polynôme (de degré au plus 7), à coefficients dans $GF(2) = \{0, 1\}$
- L'addition dans $GF(2^8)$ est simplement l'addition de polynômes
- *Exemple :*
  - $a(x) = x^6 + x^3 + x^2 + 1$, sous forme binaire $a = 01001101$
  - $b(x) = x^7 + x^6 + x^5 + 1$, sous forme binaire $b = 11100001$

Addition dans le corps de Galois $GF(2^8)$

- Chaque octet est représenté par un polynôme (de degré au plus 7), à coefficients dans $GF(2) = \{0, 1\}$
- L'addition dans $GF(2^8)$ est simplement l'addition de polynômes
- *Exemple :*
  - $a(x) = x^6 + x^3 + x^2 + 1$, sous forme binaire $a = 01001101$
  - $b(x) = x^7 + x^6 + x^5 + 1$, sous forme binaire $b = 11100001$
  - $c(x) = a(x) + b(x) = x^7 + x^5 + x^3 + x^2$, sous forme binaire $c = 10101100$

Addition dans le corps de Galois $GF(2^8)$

- Chaque octet est représenté par un polynôme (de degré au plus 7), à coefficients dans $GF(2) = \{0, 1\}$
- L'addition dans $GF(2^8)$ est simplement l'addition de polynômes
- *Exemple :*
  - $a(x) = x^6 + x^3 + x^2 + 1$, sous forme binaire $a = 01001101$
  - $b(x) = x^7 + x^6 + x^5 + 1$, sous forme binaire $b = 11100001$
  - $c(x) = a(x) + b(x) = x^7 + x^5 + x^3 + x^2$, sous forme binaire $c = 10101100$

$$c = a \oplus b$$

Multiplication dans $GF(2^8)$

- La multiplication de deux polynômes $a(x), b(x)$ de degrés au plus 7 peut donner un polynôme de degré supérieur à 7

Multiplication dans $GF(2^8)$

- La multiplication de deux polynômes $a(x), b(x)$ de degrés au plus 7 peut donner un polynôme de degré supérieur à 7
- Il faut donc réduire le résultat modulo un polynôme **irréductible** $m(x)$ de degré 8 ($m(x) = x^8 + x^4 + x^3 + x + 1$ pour l'AES)

Multiplication dans $GF(2^8)$

- La multiplication de deux polynômes $a(x), b(x)$ de degrés au plus 7 peut donner un polynôme de degré supérieur à 7
- Il faut donc réduire le résultat modulo un polynôme **irréductible** $m(x)$ de degré 8 ($m(x) = x^8 + x^4 + x^3 + x + 1$ pour l'AES)
- *Exemple :*
  - $a(x) = x^6 + x^3 + x^2 + 1$, sous forme binaire $a = 01001101$
  - $b(x) = x^7 + x^6 + x^5 + 1$, sous forme binaire $b = 11100001$

Multiplication dans $GF(2^8)$

- La multiplication de deux polynômes $a(x), b(x)$ de degrés au plus 7 peut donner un polynôme de degré supérieur à 7
- Il faut donc réduire le résultat modulo un polynôme **irréductible** $m(x)$ de degré 8 ($m(x) = x^8 + x^4 + x^3 + x + 1$ pour l'AES)
- *Exemple :*
  - $a(x) = x^6 + x^3 + x^2 + 1$, sous forme binaire $a = 01001101$
  - $b(x) = x^7 + x^6 + x^5 + 1$, sous forme binaire $b = 11100001$
  - $a(x) \cdot b(x) = x^{13} + x^{12} + x^{11} + x^{10} + x^5 + x^3 + x^2 + 1$

Multiplication dans $GF(2^8)$

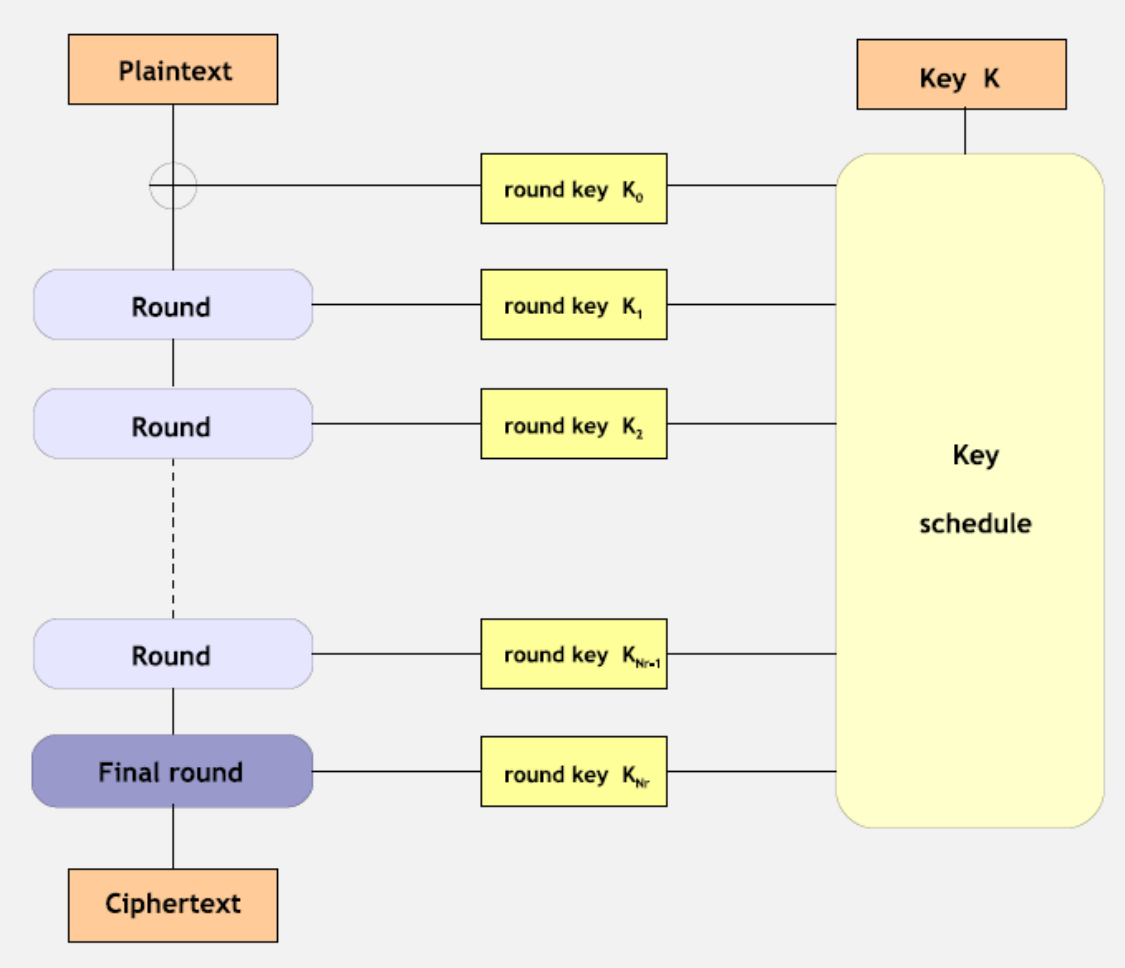
- La multiplication de deux polynômes $a(x), b(x)$ de degrés au plus 7 peut donner un polynôme de degré supérieur à 7
- Il faut donc réduire le résultat modulo un polynôme **irréductible** $m(x)$ de degré 8 ($m(x) = x^8 + x^4 + x^3 + x + 1$ pour l'AES)
- *Exemple :*
  - $a(x) = x^6 + x^3 + x^2 + 1$, sous forme binaire $a = 01001101$
  - $b(x) = x^7 + x^6 + x^5 + 1$, sous forme binaire $b = 11100001$
  - $a(x) \cdot b(x) = x^{13} + x^{12} + x^{11} + x^{10} + x^5 + x^3 + x^2 + 1$
  - $c(x) = a(x) \cdot b(x) \mod m(x) = x^3 + x^5 + x^4 + x^3 + x^2 + x + 1$

Multiplication dans $GF(2^8)$

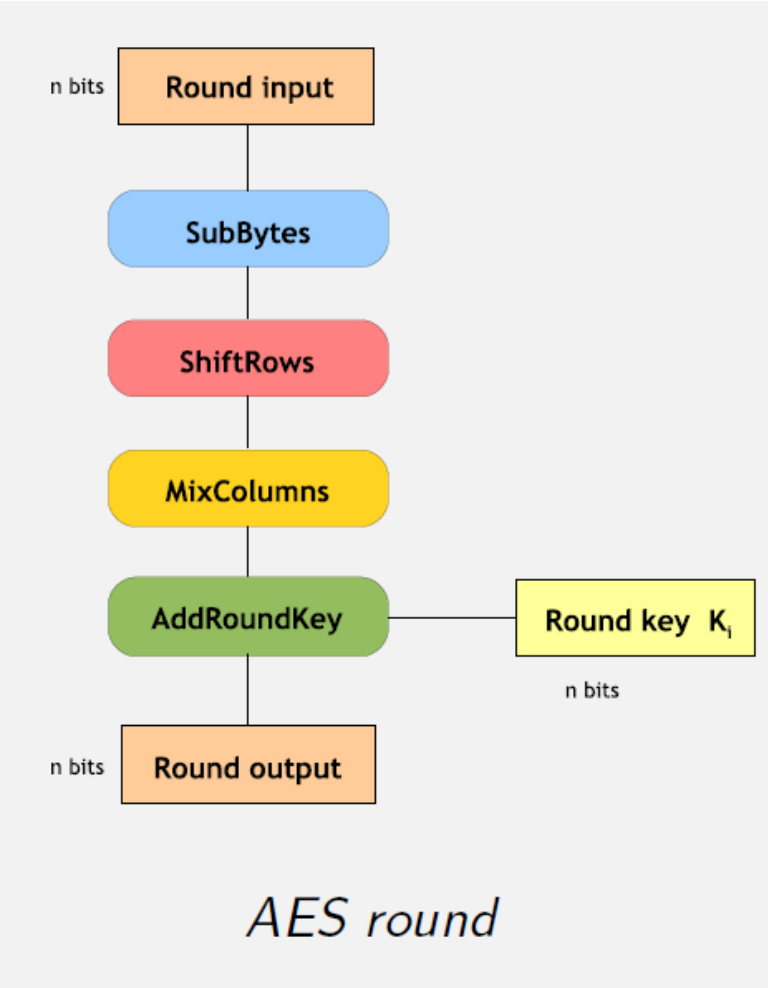
- La multiplication de deux polynômes $a(x), b(x)$ de degrés au plus 7 peut donner un polynôme de degré supérieur à 7
- Il faut donc réduire le résultat modulo un polynôme **irréductible** $m(x)$ de degré 8 ($m(x) = x^8 + x^4 + x^3 + x + 1$ pour l'AES)
- *Exemple :*
  - $a(x) = x^6 + x^3 + x^2 + 1$, sous forme binaire $a = 01001101$
  - $b(x) = x^7 + x^6 + x^5 + 1$, sous forme binaire $b = 11100001$
  - $a(x) \cdot b(x) = x^{13} + x^{12} + x^{11} + x^{10} + x^5 + x^3 + x^2 + 1$
  - $c(x) = a(x) \cdot b(x) \mod m(x) = x^3 + x^5 + x^4 + x^3 + x^2 + x + 1$

- La réduction modulaire se fait par division euclidienne de polynômes obtenu par le polynôme irréductible $m(x)$ (comme pour les entiers)

L'algorithme AES en entier

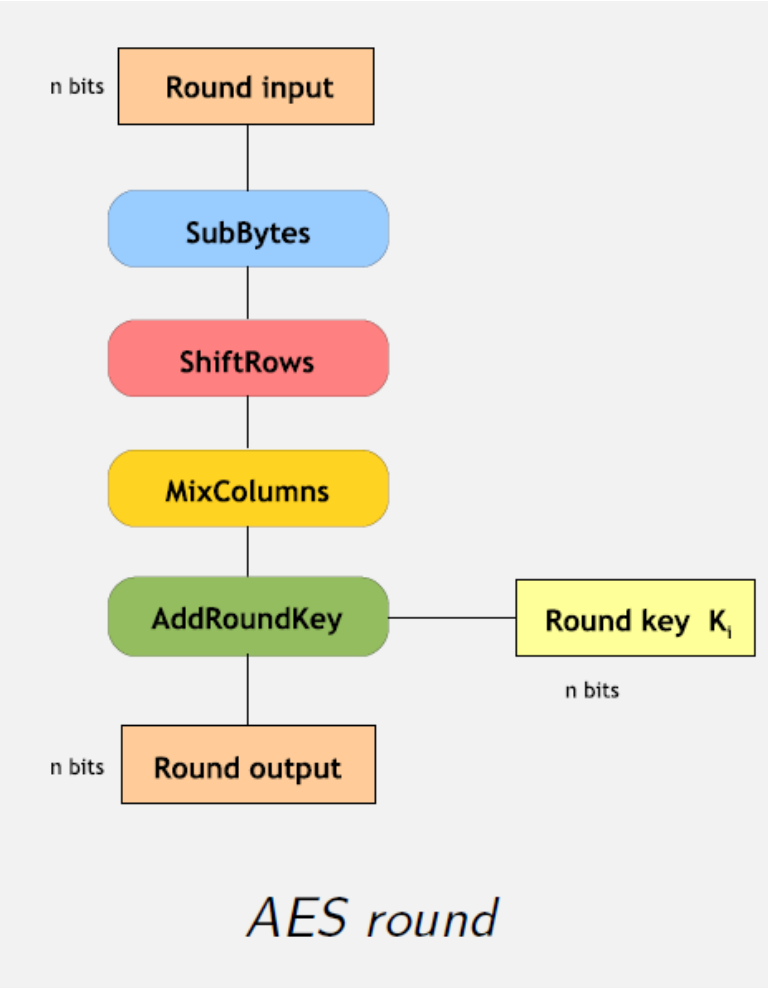


Un tour d'AES



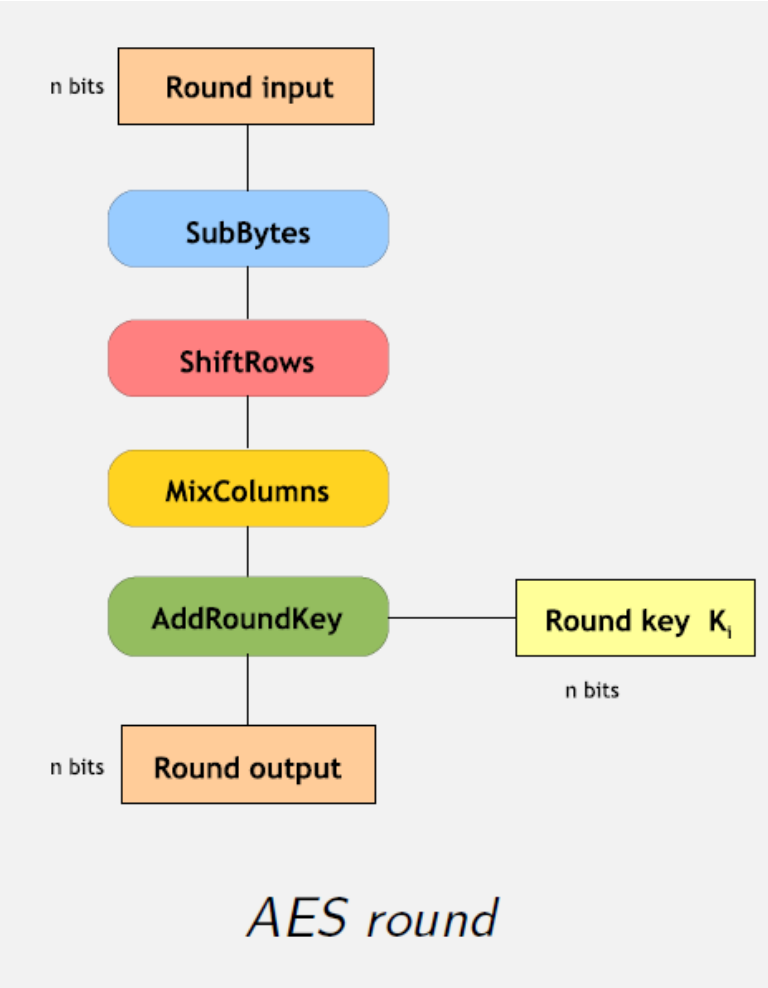
- **SubBytes** : substitution non linéaire sur les octets (S-Box)

Un tour d'AES



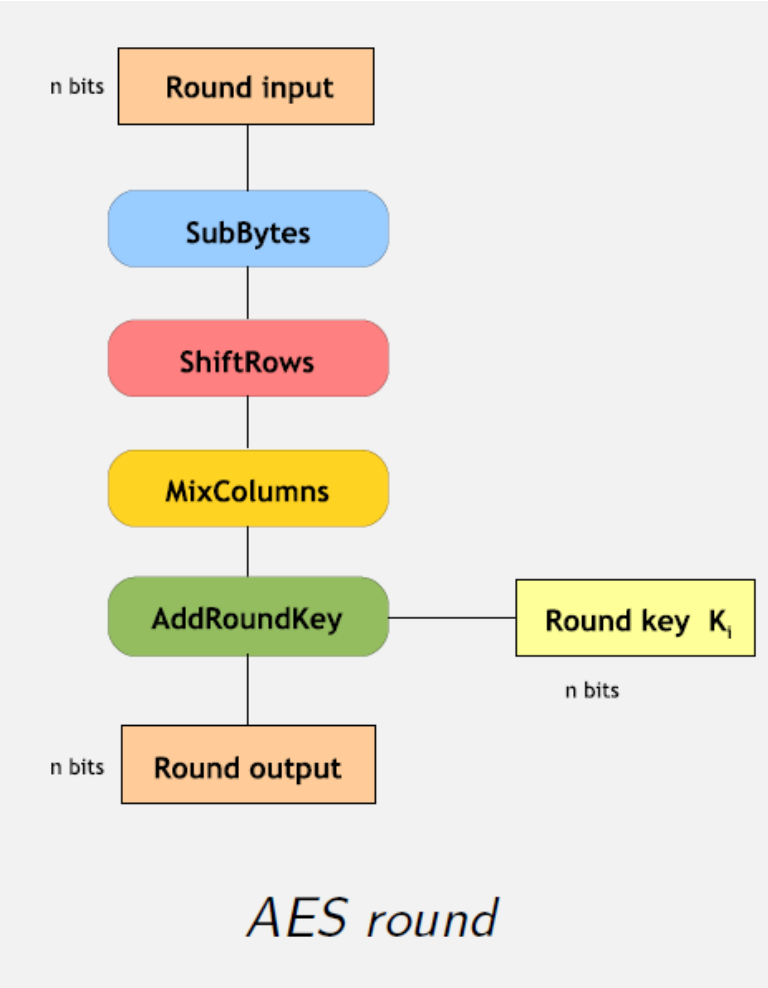
- **SubBytes** : substitution non linéaire sur les octets (S-Box)
- **ShiftRows** : permutation des octets par ligne

Un tour d'AES



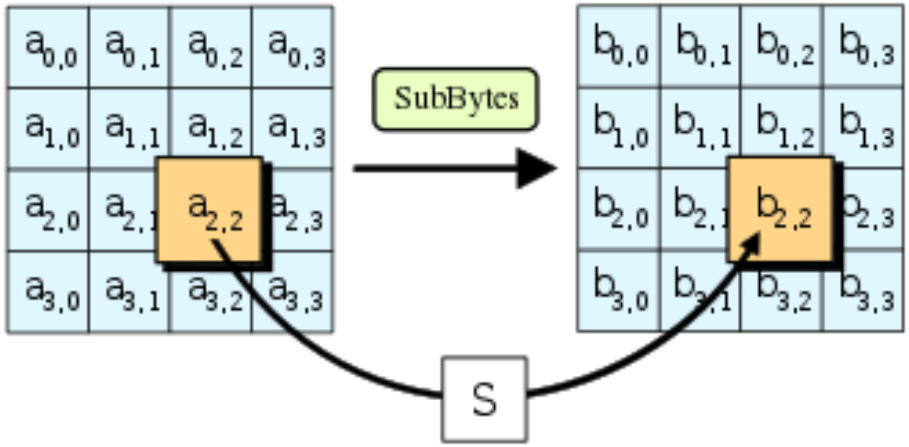
- **SubBytes** : substitution non linéaire sur les octets (S-Box)
- **ShiftRows** : permutation des octets par ligne
- **MixColumns** : les colonnes sont multipliées par une matrice circulante

Un tour d'AES

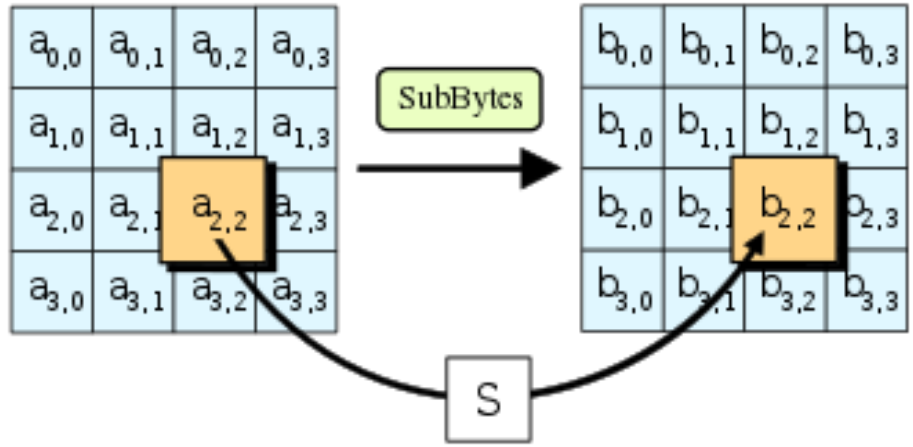


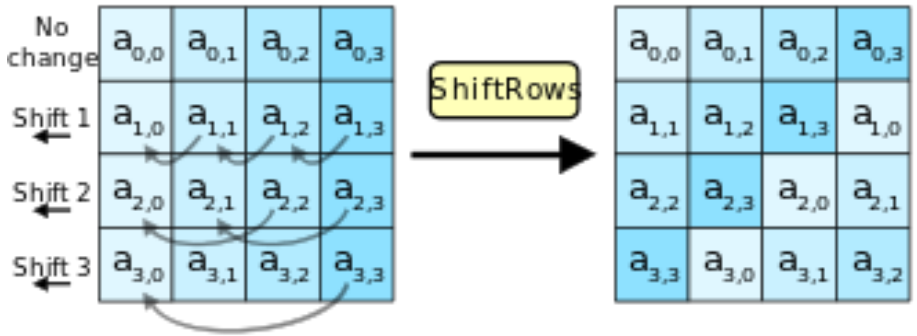
- **SubBytes** : substitution non linéaire sur les octets (S-Box)
- **ShiftRows** : permutation des octets par ligne
- **MixColumns** : les colonnes sont multipliées par une matrice circulante
- **AddRoundKey** : addition des octets dans $GF(2^8)$ (XOR avec K_i)

SubBytes et ShiftRows

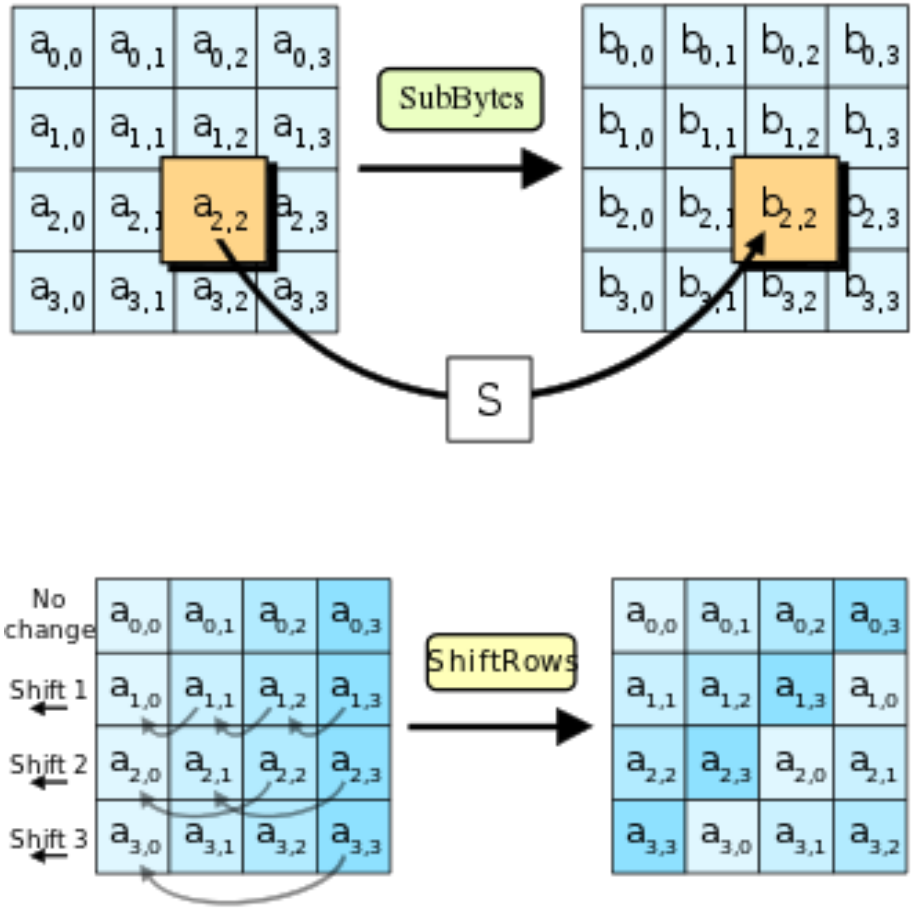


SubBytes et ShiftRows



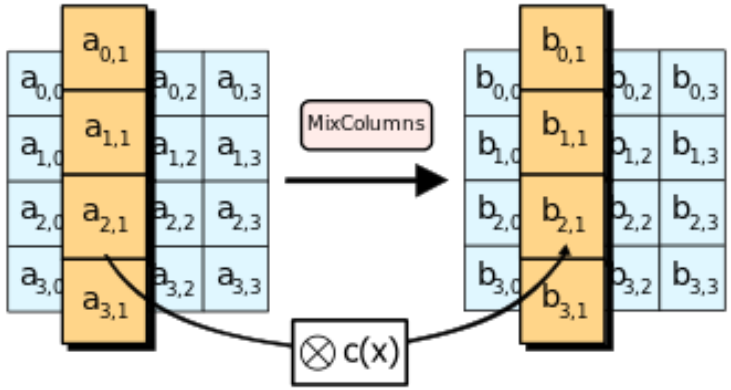


SubBytes et ShiftRows

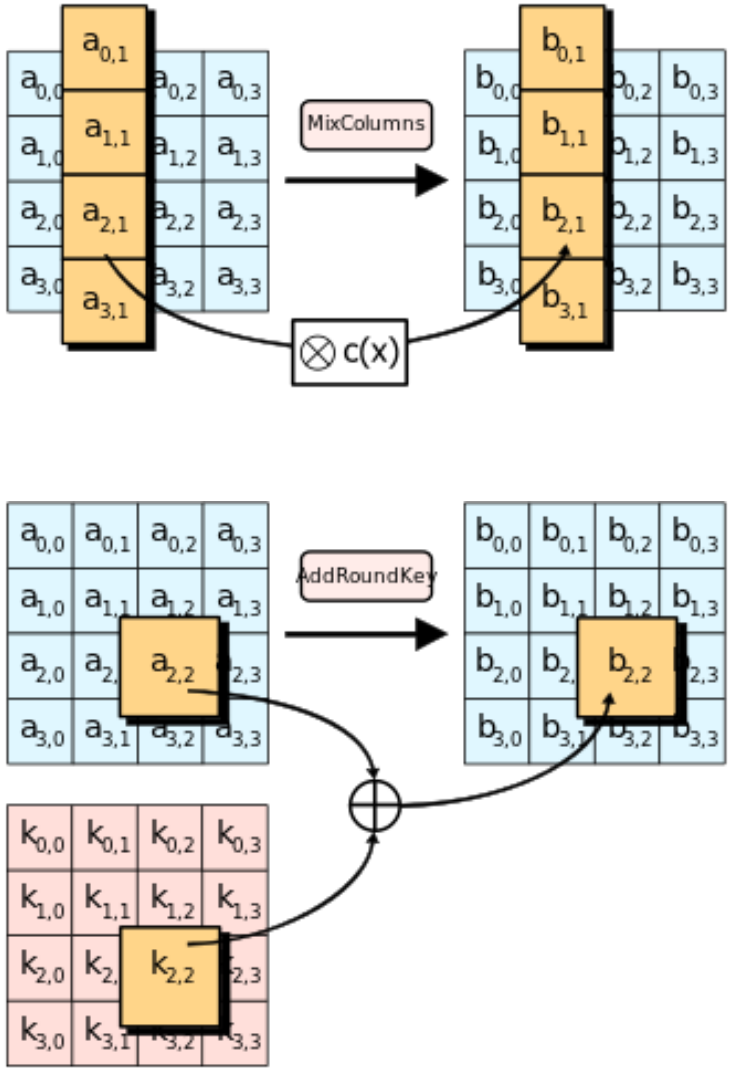


Images from boowiki.info

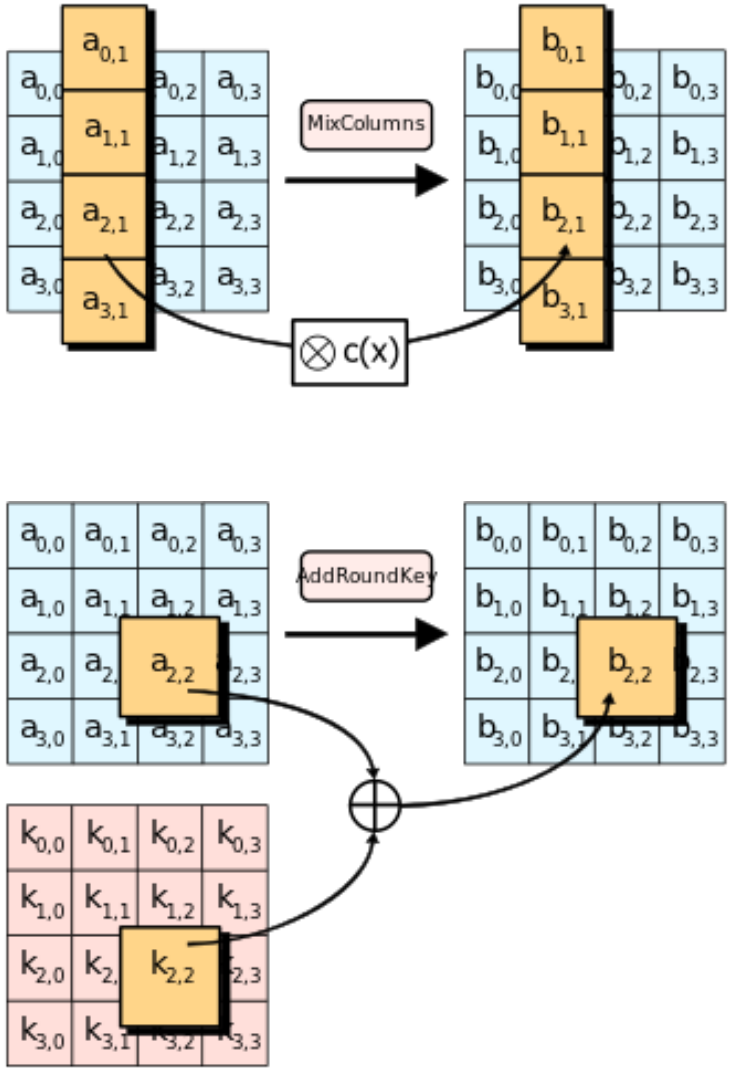
MixColumns et AddRoundKey



MixColumns et AddRoundKey



MixColumns et AddRoundKey



Images from boowiki.info

Particularité du dernier tour