Cryptographie 2013

Pascal Boyer (LAGA)

Université Paris 13

janvier-février 2013

Vous trouverez sur

http://www.math.univ-paris13.fr/ boyer/enseignement/images.html

- un polycopié de cours ainsi que
- des liens vers les trois tds.

Cryptographie 2013

Pascal Boyer (LAGA)

Université Paris 13

janvier-février 2013

Vous trouverez sur

http://www.math.univ-paris13.fr/ boyer/enseignement/images.html

- un polycopié de cours ainsi que
- des liens vers les trois tds.

1. Généralités
   - Historique
   - Présentation axiomatique
2. Exemples simples
   - Codes de permutation
   - Codes de substitution
   - Cryptanalyse
   - Codes polyalphabétiques
   - Code de VIC
3. Codes modernes
   - Chiffrement en chaînes
   - Codes à confidentialité parfaite
   - Registres à décalages
   - Applications aux GSM et WIFI
   - DES et AES
4. Codes à clefs publiques
   - RSA
   - El Gamal
   - Menezes-Vanstone
5. Fonctions de Hachage
   - Problématique
   - Description de SHA-1

6. Protocoles
   - Signature et datation
   - Certificats
   - MAC
   - Échange de clefs
   - Mots de passe
   - Preuve sans transfert de connaissance
   - Transfert inconscient
   - Partage de secret
   - Carte bleue
   - SSL et TLS
   - PGP

7. Rappels mathématiques
   - Théorie de la complexité
   - Division euclidienne
   - Congruences
   - Corps finis
   - Petit théorème de Fermat and c
   - Sur les nombres premiers
   - Méthode de factorisation
   - Polynômes
   - Courbes elliptiques

Historique

La **cryptographie** ou science du secret est un art très ancien qui se développe parallèlement à la **stéganographie**.

Scytale

Sparte vers -450 AJC, (principe des *codes de permutation*)

Code de Jules César

vers -50 AJC, (principe des *codes de substitution*, $n = n + 3$) cryptanalysé par les arabes (9e), amélioré (ajout de blancs, mauvaise orthographe, et qui a coûté la vie à Marie Stuart (fin 16e)

Historique

La **cryptographie** ou science du secret est un art très ancien qui se développe parallèlement à la **stéganographie**.

Scytale

Sparte vers -450 AJC, (principe des *codes de permutation*)

Code de Jules César

vers -50 AJC, (principe des *codes de substitution*, $n = n + 3$) cryptanalysé par les arabes (9e), amélioré (ajout de blancs, mauvaise orthographe, et qui a coûté la vie à Marie Stuart (fin 16e)

Historique

La **cryptographie** ou science du secret est un art très ancien qui se développe parallèlement à la **stéganographie**.

Scytale

Sparte vers -450 AJC, (principe des *codes de permutation*)

Code de Jules César

vers -50 AJC, (principe des *codes de substitution*, $n = n + 3$) cryptanalysé par les arabes (9e), amélioré (ajout de blancs, mauvaise orthographe, et qui a coûté la vie à Marie Stuart (fin 16e)

Historique

Code de Vigenère 1586

Premier chiffre polyalphabétique, invulnérable à l'analyse statistique avec un nombre immense de clef, il resta négligé pendant deux siècles lui préférant des chiffres de substitution homophonique: l'exemple le plus remarquable est **le grand chiffre** de Louis XIV (17e) déchiffré seulement à la fin du 19e.

Codes à répertoires

Très anciens, utilisés intensivement jusqu'au début du 20-ième siècle.

Enigma: début XXe

Utilisée par l'armée allemande durant la seconde guerre mondiale, décryptée par les polonais grâce a une répétition récurrence, puis par Alan Turing via la recherche **de mots probables**.

Historique

Code de Vigenère 1586

Premier chiffre polyalphabétique, invulnérable à l'analyse statistique avec un nombre immense de clef, il resta négligé pendant deux siècles lui préférant des chiffres de substitution homophonique: l'exemple le plus remarquable est **le grand chiffre** de Louis XIV (17e) déchiffré seulement à la fin du 19e.

Codes à répertoires

Très anciens, utilisés intensivement jusqu'au début du 20-ième siècle.

Enigma: début XXe

Utilisée par l'armée allemande durant la seconde guerre mondiale, décryptée par les polonais grâce a une répétition récurrence, puis par Alan Turing via la recherche **de mots probables**.

Historique

Code de Vigenère 1586

Premier chiffre polyalphabétique, invulnérable à l'analyse statistique avec un nombre immense de clef, il resta négligé pendant deux siècles lui préférant des chiffres de substitution homophonique: l'exemple le plus remarquable est **le grand chiffre** de Louis XIV (17e) déchiffré seulement à la fin du 19e.

Codes à répertoires

Très anciens, utilisés intensivement jusqu'au début du 20-ième siècle.

Enigma: début XXe

Utilisée par l'armée allemande durant la seconde guerre mondiale, décryptée par les polonais grâce a une répétition récurrence, puis par Alan Turing via la recherche **de mots probables**.

Principe de Kerckhoffs

Jusqu'au milieu du XXe siècle, la sécurité d'un chiffre reposait sur le secret de son fonctionnement. Le problème est que dès que ce secret est éventé, cf. par exemple le chiffre de VIC, il faut changer entièrement le cryptosystème ce qui est complexe et couteux.

Principe de Kerckhoffs

La sécurité d'un cryptosystème **ne repose pas sur le secret du cryptosystème** mais seulement sur la **clef du cryptosystème** qui est un paramètre facile à transmettre secrètement et à changer, de taille réduite (actuellement de 64 à 2048 bits).

Loin d'affaiblir la sécurité du chiffre, la diffusion du fonctionnement d'un cryptosystème analysé par le plus grand nombre, permet, en l'éprouvant, de valider sa sécurité ou à défaut, renseigne sur l'urgence d'en changer.

Principe de Kerckhoffs

Jusqu'au milieu du XXe siècle, la sécurité d'un chiffre reposait sur le secret de son fonctionnement. Le problème est que dès que ce secret est éventé, cf. par exemple le chiffre de VIC, il faut changer entièrement le cryptosystème ce qui est complexe et couteux.

Principe de Kerckhoffs

La sécurité d'un cryptosystème **ne repose pas sur le secret du cryptosystème** mais seulement sur la **clef du cryptosystème** qui est un paramètre facile à transmettre secrètement et à changer, de taille réduite (actuellement de 64 à 2048 bits).

Loin d'affaiblir la sécurité du chiffre, la diffusion du fonctionnement d'un cryptosystème analysé par le plus grand nombre, permet, en l'éprouvant, de valider sa sécurité ou à défaut, renseigne sur l'urgence d'en changer.

Principe de Kerckhoffs

Jusqu'au milieu du XXe siècle, la sécurité d'un chiffre reposait sur le secret de son fonctionnement. Le problème est que dès que ce secret est éventé, cf. par exemple le chiffre de VIC, il faut changer entièrement le cryptosystème ce qui est complexe et couteux.

Principe de Kerckhoffs

La sécurité d'un cryptosystème **ne repose pas sur le secret du cryptosystème** mais seulement sur la **clef du cryptosystème** qui est un paramètre facile à transmettre secrètement et à changer, de taille réduite (actuellement de 64 à 2048 bits).

Loin d'affaiblir la sécurité du chiffre, la diffusion du fonctionnement d'un cryptosystème analysé par le plus grand nombre, permet, en l'éprouvant, de valider sa sécurité ou à défaut, renseigne sur l'urgence d'en changer.

Codes modernes

On peut distinguer deux grandes familles de codes classiques:

Codes à clefs secrètes

dits aussi **symétriques** qui mêlent **codes de permutation** et **codes de substitution**. Les exemples les plus célèbres sont **DES et AES**.

codes à clefs publiques

dits aussi **asymétriques** qui reposent sur la notion mathématique de **fonctions à sens unique**. Citons dans cette catégorie les codes **RSA et El Gamal**.

Suivant le **principe de Kerckhoffs**, ces cryptosystèmes sont connus de tous, leur sécurité reposant sur l'existence de clés au coeur du chiffre.

Codes modernes

On peut distinguer deux grandes familles de codes classiques:

Codes à clefs secrètes

dits aussi **symétriques** qui mêlent **codes de permutation** et **codes de substitution**. Les exemples les plus célèbres sont **DES et AES**.

codes à clefs publiques

dits aussi **asymétriques** qui reposent sur la notion mathématique de **fonctions à sens unique**. Citons dans cette catégorie les codes **RSA et El Gamal**.

Suivant le **principe de Kerckhoffs**, ces cryptosystèmes sont connus de tous, leur sécurité reposant sur l'existence de clés au coeur du chiffre.

Codes modernes

On peut distinguer deux grandes familles de codes classiques:

Codes à clefs secrètes

dits aussi **symétriques** qui mêlent **codes de permutation** et **codes de substitution**. Les exemples les plus célèbres sont **DES et AES**.

codes à clefs publiques

dits aussi **asymétriques** qui reposent sur la notion mathématique de **fonctions à sens unique**. Citons dans cette catégorie les codes **RSA et El Gamal**.

Suivant le **principe de Kerckhoffs**, ces cryptosystèmes sont connus de tous, leur sécurité reposant sur l'existence de clés au coeur du chiffre.

Procédure de chiffrement/déchiffrement

Un expéditeur **Alice** veut envoyer un message à un destinataire **Bob** en évitant les oreilles indiscrète d'**Ève**, et les attaques malveillantes de **Martin**.

Pour cela Alice utilise un **cryptosystème** (un **chiffre**) qu'elle partage avec Bob. En général le cryptosystème dépend d'un paramètre, la **clef**, qui peut être publique ou secrète.

Décryptage

Pour décrypter Bob dispose d'une **clef de décryptage**, nécessairement secrète, qui peut être égale ou différente de la clef de cryptage et d'un algorithme de décryptage.

Procédure de chiffrement/déchiffrement

Un expéditeur **Alice** veut envoyer un message à un destinataire **Bob** en évitant les oreilles indiscrète d'**Ève**, et les attaques malveillantes de **Martin**.

Pour cela Alice utilise un **cryptosystème** (un **chiffre**) qu'elle partage avec Bob. En général le cryptosystème dépend d'un paramètre, la **clef**, qui peut être publique ou secrète.

Décryptage

Pour décrypter Bob dispose d'une **clef de décryptage**, nécessairement secrète, qui peut être égale ou différente de la clef de cryptage et d'un algorithme de décryptage.

Procédure de chiffrement/déchiffrement

Un expéditeur **Alice** veut envoyer un message à un destinataire **Bob** en évitant les oreilles indiscrète d'**Ève**, et les attaques malveillantes de **Martin**.

Pour cela Alice utilise un **cryptosystème** (un **chiffre**) qu'elle partage avec Bob. En général le cryptosystème dépend d'un paramètre, la **clef**, qui peut être publique ou secrète.

Décryptage

Pour décrypter Bob dispose d'une **clef de décryptage**, nécessairement secrète, qui peut être égale ou différente de la clef de cryptage et d'un algorithme de décryptage.

Définition d'un code

Définition

Un code est la donnée de

- deux ensembles finis de blocs, $\mathcal{P}$ les **mots en clair** et $\mathcal{C}$ les **mots codés**
- d'un ensemble fini $\mathcal{K}$ l'**espace des clefs**
- Pour tout $k \in \mathcal{K}$ on a une règle de chiffrement $e_k \in \mathcal{E}$ qui transforme les mots en clair en mots codés et une régle de déchiffrement $d_k \in \mathcal{D}$ qui transforme les mots codés en mots en clair telles que si $x \in \mathcal{P}$ on a $d_k(e_k(x)) = x$.

Exemple

Le code de César dans lequel $\mathcal{P} = \mathcal{E}$ est l'ensemble des mots écrits à l'aide de l'alphabet ordinaire. $\mathcal{K}$ est un nombre entre 0 et 26. La règle de chiffrement e_k consiste à remplacer la lettre de rang n dans l'alphabet par celle celle de rang $n + k$ modulo 26

Définition d'un code

Définition

Un code est la donnée de

- deux ensembles finis de blocs, $\mathcal{P}$ les **mots en clair** et $\mathcal{C}$ les **mots codés**
- d'un ensemble fini $\mathcal{K}$ l'**espace des clefs**
- Pour tout $k \in \mathcal{K}$ on a une règle de chiffrement $e_k \in \mathcal{E}$ qui transforme les mots en clair en mots codés et une régle de déchiffrement $d_k \in \mathcal{D}$ qui transforme les mots codés en mots en clair telles que si $x \in \mathcal{P}$ on a $d_k(e_k(x)) = x$.

Exemple

Le code de César dans lequel $\mathcal{P} = \mathcal{E}$ est l'ensemble des mots écrits à l'aide de l'alphabet ordinaire. $\mathcal{K}$ est un nombre entre 0 et 26. La règle de chiffrement e_k consiste à remplacer la lettre de rang n dans l'alphabet par celle celle de rang $n + k$ modulo 26

Définition d'un code

Définition

Un code est la donnée de

- deux ensembles finis de blocs, $\mathcal{P}$ les **mots en clair** et $\mathcal{C}$ les **mots codés**
- d'un ensemble fini $\mathcal{K}$ l'**espace des clefs**
- Pour tout $k \in \mathcal{K}$ on a une règle de chiffrement $e_k \in \mathcal{E}$ qui transforme les mots en clair en mots codés et une régle de déchiffrement $d_k \in \mathcal{D}$ qui transforme les mots codés en mots en clair telles que si $x \in \mathcal{P}$ on a $d_k(e_k(x)) = x$.

Exemple

Le code de César dans lequel $\mathcal{P} = \mathcal{E}$ est l'ensemble des mots écrits à l'aide de l'alphabet ordinaire. $\mathcal{K}$ est un nombre entre 0 et 26. La règle de chiffrement e_k consiste à remplacer la lettre de rang n dans l'alphabet par celle celle de rang $n + k$ modulo 26

Définition d'un code

Définition

Un code est la donnée de

- deux ensembles finis de blocs, $\mathcal{P}$ les **mots en clair** et $\mathcal{C}$ les **mots codés**
- d'un ensemble fini $\mathcal{K}$ l'**espace des clefs**
- Pour tout $k \in \mathcal{K}$ on a une règle de chiffrement $e_k \in \mathcal{E}$ qui transforme les mots en clair en mots codés et une régle de déchiffrement $d_k \in \mathcal{D}$ qui transforme les mots codés en mots en clair telles que si $x \in \mathcal{P}$ on a $d_k(e_k(x)) = x$.

Exemple

Le code de César dans lequel $\mathcal{P} = \mathcal{E}$ est l'ensemble des mots écrits à l'aide de l'alphabet ordinaire. $\mathcal{K}$ est un nombre entre 0 et 26. La règle de chiffrement e_k consiste à remplacer la lettre de rang n dans l'alphabet par celle celle de rang $n + k$ modulo 26

Clés

code à clef secrète ou code symétrique

Si la clef de chiffrement est la même que la clef de déchiffrement on a affaire à un **code symétrique**.

code à clef publique ou code asymétrique

Si les clefs de chiffrement et déchiffrement sont différentes, on dit que **le code est asymétrique**.

Clés

code à clef secrète ou code symétrique

Si la clef de chiffrement est la même que la clef de déchiffrement on a affaire à un **code symétrique**.

code à clef publique ou code asymétrique

Si les clefs de chiffrement et déchiffrement sont différentes, on dit que **le code est asymétrique**.

Constitution d'un code moderne

- Un **algorithme de chiffrement** $f = f_{K_C}$, dépendant d'un paramètre K_C, la clé de chiffrement. L'algorithme est fixé et public, seule la clé change, **principe de Kerkhoffs**).
- La **valeur de la clé de chiffrement**, K_C qui est secrète ou non suivant que l'on a affaire à un code à clef secrète ou à un code à clef publique.
- Un **algorithme de déchiffrement** $g = g_{K_D} = f^{-1}$ (supposé connu de tous) dépendant d'une **clé de déchiffrement**, K_D, différente ou non de K_C.
- La valeur de la clé de déchiffrement, K_D, qui est toujours secrète.

Constitution d'un code moderne

- Un **algorithme de chiffrement** $f = f_{K_C}$, dépendant d'un paramètre K_C, la clé de chiffrement. L'algorithme est fixé et public, seule la clé change, **principe de Kerkhoffs**).
- La **valeur de la clé de chiffrement**, K_C qui est secrète ou non suivant que l'on a affaire à un code à clef secrète ou à un code à clef publique.
- Un **algorithme de déchiffrement** $g = g_{K_D} = f^{-1}$ (supposé connu de tous) dépendant d'une **clé de déchiffrement**, K_D, différente ou non de K_C.
- La valeur de la clé de déchiffrement, K_D, qui est toujours secrète.

Constitution d'un code moderne

- Un **algorithme de chiffrement** $f = f_{K_C}$, dépendant d'un paramètre K_C, la clé de chiffrement. L'algorithme est fixé et public, seule la clé change, **principe de Kerkhoffs**).
- La **valeur de la clé de chiffrement**, K_C qui est secrète ou non suivant que l'on a affaire à un code à clef secrète ou à un code à clef publique.
- Un **algorithme de déchiffrement** $g = g_{K_D} = f^{-1}$ (supposé connu de tous) dépendant d'une **clé de déchiffrement**, K_D, différente ou non de K_C.
- La valeur de la clé de déchiffrement, K_D, qui est toujours secrète.

Constitution d'un code moderne

- Un **algorithme de chiffrement** $f = f_{K_C}$, dépendant d'un paramètre K_C, la clé de chiffrement. L'algorithme est fixé et public, seule la clé change, **principe de Kerkhoffs**).
- La **valeur de la clé de chiffrement**, K_C qui est secrète ou non suivant que l'on a affaire à un code à clef secrète ou à un code à clef publique.
- Un **algorithme de déchiffrement** $g = g_{K_D} = f^{-1}$ (supposé connu de tous) dépendant d'une **clé de déchiffrement**, K_D, différente ou non de K_C.
- La valeur de la clé de déchiffrement, K_D, qui est toujours secrète.

Cryptanalyse

Il est très important toutes les fois que l'on utilise un cryptosystème d'évaluer son degré de sécurité et sa résistance aux attaques.

Cryptanalyse

La cryptanalyse est l'ensemble des techniques qui permettent à un attaquant de décrypter un message ou de retrouver la clef d'un code secret

Les différents types d'attaques

Attaque à texte chiffré connu

L'opposant ne connait que le message chiffré y.

Attaque à texte clair connu

L'opposant dispose d'un texte clair x et du message chiffré correspondant y

Attaque à texte clair choisi

L'opposant a accès à une machine chiffrante. Il peut choisir un texte clair et obtenir le texte chiffré correspondant y.

Attaque à texte chiffré choisi

L'opposant a accès à une machine déchiffrante. Il peut choisir un texte chiffré, y et obtenir le texte clair correspondant x.

Les différents types d'attaques

Attaque à texte chiffré connu

L'opposant ne connait que le message chiffré y.

Attaque à texte clair connu

L'opposant dispose d'un texte clair x et du message chiffré correspondant y

Attaque à texte clair choisi

L'opposant a accès à une machine chiffrante. Il peut choisir un texte clair et obtenir le texte chiffré correspondant y.

Attaque à texte chiffré choisi

L'opposant a accès à une machine déchiffrante. Il peut choisir un texte chiffré, y et obtenir le texte clair correspondant x.

Les différents types d'attaques

Attaque à texte chiffré connu

L'opposant ne connait que le message chiffré y.

Attaque à texte clair connu

L'opposant dispose d'un texte clair x et du message chiffré correspondant y

Attaque à texte clair choisi

L'opposant a accès à une machine chiffrante. Il peut choisir un texte clair et obtenir le texte chiffré correspondant y.

Attaque à texte chiffré choisi

L'opposant a accès à une machine déchiffrante. Il peut choisir un texte chiffré, y et obtenir le texte clair correspondant x.

Les différents types d'attaques

Attaque à texte chiffré connu

L'opposant ne connait que le message chiffré y.

Attaque à texte clair connu

L'opposant dispose d'un texte clair x et du message chiffré correspondant y

Attaque à texte clair choisi

L'opposant a accès à une machine chiffrante. Il peut choisir un texte clair et obtenir le texte chiffré correspondant y.

Attaque à texte chiffré choisi

L'opposant a accès à une machine déchiffrante. Il peut choisir un texte chiffré, y et obtenir le texte clair correspondant x.

Différentes notions de sécurité d'un cryptosystème

Sécurité inconditionnelle

Elle ne préjuge pas de la puissance de calcul du cryptanalyste qui peut être illimitée.

Sécurité prouvée

qui réduit la sécurité du cryptosystème à un problème bien connu réputé difficile, par exemple on pourrait prouver un théorème disant qu'un système cryptographique donné est sûr si un entier donné n ne peut être factorisé.

Sécurité calculatoire

qui repose sur l'impossibilité de faire en un temps raisonnable les calculs nécessaires pour décrypter un message.

Différentes notions de sécurité d'un cryptosystème

Sécurité inconditionnelle

Elle ne préjuge pas de la puissance de calcul du cryptanalyste qui peut être illimitée.

Sécurité prouvée

qui réduit la sécurité du cryptosystème à un problème bien connu réputé difficile, par exemple on pourrait prouver un théorème disant qu'un système cryptographique donné est sûr si un entier donné n ne peut être factorisé.

Sécurité calculatoire

qui repose sur l'impossibilité de faire en un temps raisonnable les calculs nécessaires pour décrypter un message.

Différentes notions de sécurité d'un cryptosystème

Sécurité inconditionnelle

Elle ne préjuge pas de la puissance de calcul du cryptanalyste qui peut être illimitée.

Sécurité prouvée

qui réduit la sécurité du cryptosystème à un problème bien connu réputé difficile, par exemple on pourrait prouver un théorème disant qu'un système cryptographique donné est sûr si un entier donné n ne peut être factorisé.

Sécurité calculatoire

qui repose sur l'impossibilité de faire en un temps raisonnable les calculs nécessaires pour décrypter un message.

Objectifs des codes actuels

Confidentialité des données

Les messages ne peuvent être déchiffrés que par le destinataire.

Intégrité des données

Les messages ne peuvent être modifiés par un tiers non autorisé.

Authentification

L'identité des différents participants peut être vérifiée.

Non-répudiation

L'expéditeur ne peut nier avoir émis le message et le destinaire ne peut nier l'avoir reçu.

Objectifs des codes actuels

Confidentialité des données

Les messages ne peuvent être déchiffrés que par le destinataire.

Intégrité des données

Les messages ne peuvent être modifiés par un tiers non autorisé.

Authentification

L'identité des différents participants peut être vérifiée.

Non-répudiation

L'expéditeur ne peut nier avoir émis le message et le destinaire ne peut nier l'avoir reçu.

Objectifs des codes actuels

Confidentialité des données

Les messages ne peuvent être déchiffrés que par le destinataire.

Intégrité des données

Les messages ne peuvent être modifiés par un tiers non autorisé.

Authentification

L'identité des différents participants peut être vérifiée.

Non-répudiation

L'expéditeur ne peut nier avoir émis le message et le destinaire ne peut nier l'avoir reçu.

Objectifs des codes actuels

Confidentialité des données

Les messages ne peuvent être déchiffrés que par le destinataire.

Intégrité des données

Les messages ne peuvent être modifiés par un tiers non autorisé.

Authentification

L'identité des différents participants peut être vérifiée.

Non-répudiation

L'expéditeur ne peut nier avoir émis le message et le destinaire ne peut nier l'avoir reçu.

Qualités des codes modernes

Il doit **coder et décoder rapidement** (en temps réel pour certaines applications).

Il doit résister aux attaques connues et si possible avoir une **sécurité prouvée**.

Confidentialité parfaite

Qualité des codes pour lesquels un couple (message clair, message chiffré) ne donne aucune information sur la clef.

Pas de codes réunissant toutes ces qualités simultanément, donc **compromis adaptés à chaque situation**.

Qualités des codes modernes

Il doit **coder et décoder rapidement** (en temps réel pour certaines applications).

Il doit résister aux attaques connues et si possible avoir une **sécurité prouvée**.

Confidentialité parfaite

Qualité des codes pour lesquels un couple (message clair, message chiffré) ne donne aucune information sur la clef.

Pas de codes réunissant toutes ces qualités simultanément, donc **compromis adaptés à chaque situation**.

Qualités des codes modernes

Il doit **coder et décoder rapidement** (en temps réel pour certaines applications).

Il doit résister aux attaques connues et si possible avoir une **sécurité prouvée**.

Confidentialité parfaite

Qualité des codes pour lesquels un couple (message clair, message chiffré) ne donne aucune information sur la clef.

Pas de codes réunissant toutes ces qualités simultanément, donc **compromis adaptés à chaque situation**.

Plan

Codes de permutation ou de transposition

On partage le texte en blocs, on garde **le même alphabet** mais on **change la place des lettres** à l'intérieur d'un bloc (on les permute).

méthode de la grille (principe de la scytale)

Message en clair:

RENDEZ VOUS DEMAIN MIDI VILLETANEUSE

Codes de permutation ou de transposition

On partage le texte en blocs, on garde **le même alphabet** mais on **change la place des lettres** à l'intérieur d'un bloc (on les permute).

méthode de la grille (principe de la scytale)

Message en clair:

RENDEZ VOUS DEMAIN MIDI VILLETANEUSE

Codes de permutation ou de transposition

On partage le texte en blocs, on garde **le même alphabet** mais on **change la place des lettres** à l'intérieur d'un bloc (on les permute).

méthode de la grille (principe de la scytale)

Message en clair:

RENDEZ VOUS DEMAIN MIDI VILLETANEUSE

Codes de permutation ou de transposition

On écrit le message dans une **grille de largeur fixée à l'avance**, ici 6, cette largeur est la **clef secrète** partagée entre Alice et Bob.

R	E	N	D	E	Z
□	V	O	U	S	□
D	E	M	A	I	N
□	M	I	D	I	□
V	I	L	L	E	T
A	N	E	U	S	E

Codes de permutation ou de transposition

On écrit le message dans une **grille de largeur fixée à l'avance**, ici 6, cette largeur est la **clef secrète** partagée entre Alice et Bob.

R	E	N	D	E	Z
□	V	O	U	S	□
D	E	M	A	I	N
□	M	I	D	I	□
V	I	L	L	E	T
A	N	E	U	S	E

Codes de permutation ou de transposition

On lit le texte en colonne; message crypté:

R□D□VAEVEMINNOMILEDUADLUESIIESZ□N□TEC

Ajout d'une clef

Pour pouvoir changer facilement de code en gardant le **même algorithme de codage** on rajoute une **clé secrète** constituée par exemple par l'ordre de lecture des colonnes. La largeur de la grille peut alors être rendue publique.

Codes de permutation ou de transposition

On lit le texte en colonne; message crypté:

R□D□VAEVEMINNOMILEDUADLUESIIESZ□N□TEC

Ajout d'une clef

Pour pouvoir changer facilement de code en gardant le **même algorithme de codage** on rajoute une **clé secrète** constituée par exemple par l'ordre de lecture des colonnes. La largeur de la grille peut alors être rendue publique.

Codes de permutation ou de transposition

On lit le texte en colonne; message crypté:

R□D□VAEVEMINNOMILEDUADLUESIIESZ□N□TEC

Ajout d'une clef

Pour pouvoir changer facilement de code en gardant le **même algorithme de codage** on rajoute une **clé secrète** constituée par exemple par l'ordre de lecture des colonnes. La largeur de la grille peut alors être rendue publique.

Codes de permutation ou de transposition

Exemple

on choisit la clé: **CAPTER**

On numérote les colonne en fonction du rang des lettres du mot CAPTER dans l'alphabet c'est à dire

$$2, 1, 4, 6, 3, 5$$

et on lit les colonnes dans l'ordre indiqué.

EVEMINR□D□DADUADLUZ□N□TENOMILEESIIES

On a 6! codes différents.

Codes de permutation ou de transposition

Exemple

on choisit la clé: **CAPTER**

On numérote les colonne en fonction du rang des lettres du mot CAPTER dans l'alphabet c'est à dire

$$\mathbf{2, 1, 4, 6, 3, 5}$$

et on lit les colonnes dans l'ordre indiqué.

EVEMINR□D□DADUADLUZ□N□TENOMILEESIIES

On a 6! codes différents.

Codes de permutation ou de transposition

Exemple

on choisit la clé: **CAPTER**

On numérote les colonne en fonction du rang des lettres du mot CAPTER dans l'alphabet c'est à dire

$$2, 1, 4, 6, 3, 5$$

et on lit les colonnes dans l'ordre indiqué.

EVEMINR□D□DADUADLUZ□N□TENOMILEESIIES

On a 6! codes différents.

Codes de permutation ou de transposition

Pour décoder

EVEMINR□D□DADUADLUZ□N□TENOMILEESIIES

on range en colonne sur la grille en suivant l'ordre des colonnes donnés par le mot de code

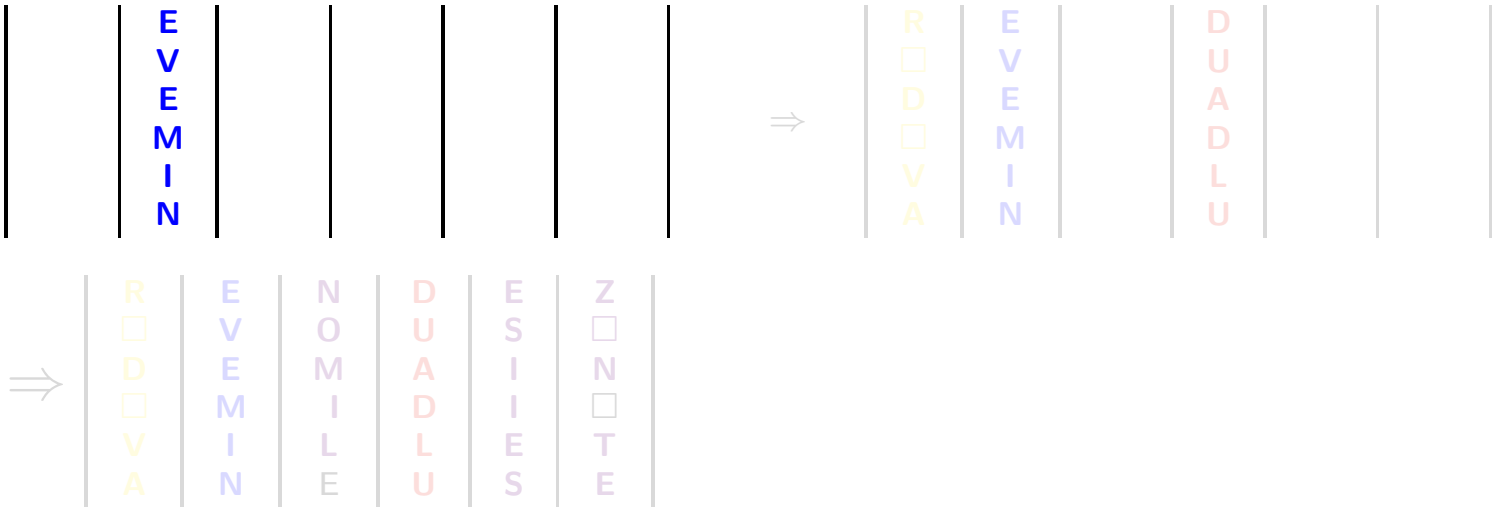
On a affaire à un **code à clef secrète** ou **code symétrique** car la clef de décodage est la même que la clef de codage.

Codes de permutation ou de transposition

Pour décoder

EVEMINR□D□DADUADLUZ□N□TENOMILEESIIES

on range en colonne sur la grille en suivant l'ordre des colonnes donnés par le mot de code



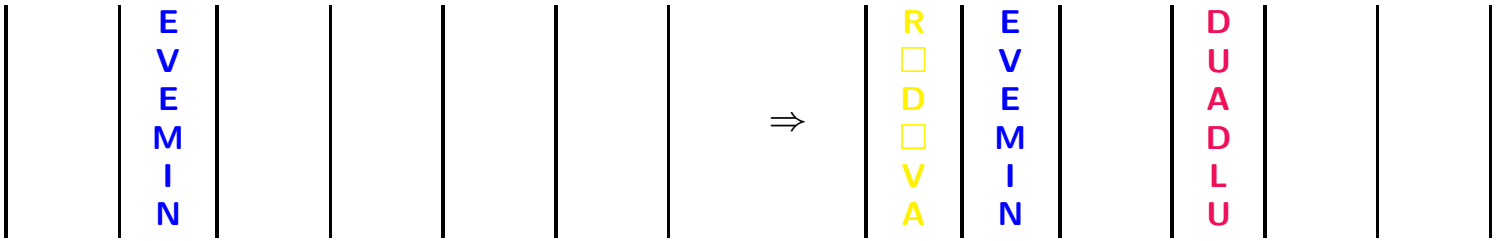
On a affaire à un **code à clef secrète** ou **code symétrique** car la clef de décodage est la même que la clef de codage.

Codes de permutation ou de transposition

Pour décoder

EVEMINR□D□DADUADLUZ□N□TENOMILEESIIES

on range en colonne sur la grille en suivant l'ordre des colonnes donnés par le mot de code



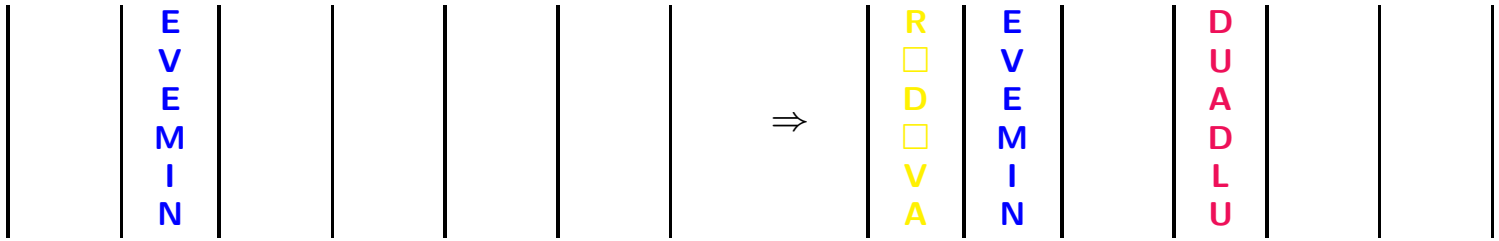
On a affaire à un **code à clef secrète** ou **code symétrique** car la clef de décodage est la même que la clef de codage.

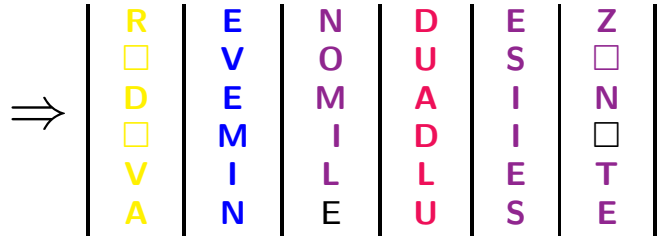
Codes de permutation ou de transposition

Pour décoder

EVEMINR□D□DADUADLUZ□N□TENOMILEESIIES

on range en colonne sur la grille en suivant l'ordre des colonnes donnés par le mot de code





On a affaire à un **code à clef secrète** ou **code symétrique** car la clef de décodage est la même que la clef de codage.

Sécurité d'un cryptosystème

Principe de Kerckhoffs

La sécurité d'un cryptosystème **ne repose pas sur le secret du cryptosystème** mais seulement sur la **clef du cryptosystème** qui est un paramètre facile à transmettre secrètement et à changer, de taille réduite (actuellement de 64 à 2048 bits).

Par exemple dans le cryptosystème précédent on suppose que l'attaquant sait que c'est un **cryptosystème à grille** mais pas la **largeur de la grille** ou la **clef**.

Codes de substitution

Dans les **codes de substitution** par flots ou par blocs l'ordre des lettres est conservé mais on les remplace par des symboles d'un nouvel alphabet suivant un algorithme précis.

Exemple: code de César:

Pour coder on remplace chaque lettre par son rang dans l'alphabet.

A=1, B=2, C=3,....,M=13, N=14,...,S=20,. ..,X=24, Y=25, Z=26

Codes de substitution

Dans les **codes de substitution** par flots ou par blocs l'ordre des lettres est conservé mais on les remplace par des symboles d'un nouvel alphabet suivant un algorithme précis.

Exemple: code de César:

Pour coder on remplace chaque lettre par son rang dans l'alphabet.

A=1, B=2, C=3,....,M=13, N=14,...,S=20,. ..,X=24, Y=25, Z=26

Codes de substitution

Jules César pendant la guerre des Gaules (-50) avait utilisé le code de substitution par flot suivant:

$$\text{lettre codée} = \text{lettre claire} + 3 \text{ modulo } 26$$

Le message en clair

RENDEZ VOUS DEMAIN MIDI VILLETANEUSE

devient

UHQGHC YRXV GHPDLQ PLGL YLOOHWDQHXVH

Codes de substitution

Jules César pendant la guerre des Gaules (-50) avait utilisé le code de substitution par flot suivant:

lettre codée= **lettre claire**+**3** **modulo** 26

Le message en clair

RENDEZ VOUS DEMAIN MIDI VILLETANEUSE

devient

UHQGHC YRXV GHPDLQ PLGL YLOOHWDQHXVH

Codes de substitution

Jules César pendant la guerre des Gaules (-50) avait utilisé le code de substitution par flot suivant:

lettre codée= **lettre claire**+**3** **modulo** 26

Le message en clair

RENDEZ VOUS DEMAIN MIDI VILLETANEUSE

devient

UHQGHC YRXV GHPDLQ PLGL YLOOHWDQHXVH

Codes de substitution

Jules César pendant la guerre des Gaules (-50) avait utilisé le code de substitution par flot suivant:

$$\textbf{lettre codée} = \textbf{lettre claire} + \textbf{3} \ \textbf{modulo}\ 26$$

Le message en clair

RENDEZ VOUS DEMAIN MIDI VILLETANEUSE

devient

UHQGHC YRXV GHPDLQ PLGL YLOOHWDQHXVH

Codes de substitution

Avantage: on peut considérer toute la famille des codes

lettre codée= lettre claire+n modulo 26

où **n** est un entier entre 0 et 25 appelé la **clef** du code.

Avec la clef **n = 7** le texte codé du message précédent devient:

YLUKLG CVBZ KLTHPU TPKP CPSSLAHULBZLBZL

Le **décodage** se fait en utlisant la relation

lettre claire=lettre codée - n mod 26

On a affaire à un **code en continu ou par flots, symétrique** ou à **clef secrète**.

Codes de substitution

Avantage: on peut considérer toute la famille des codes

lettre codée= lettre claire+n modulo 26

où **n** est un entier entre 0 et 25 appelé la **clef** du code.

Avec la clef **n = 7** le texte codé du message précédent devient:

YLUKLG CVBZ KLTHPU TPKP CPSSLAHULBZLBZL

Le **décodage** se fait en utlisant la relation

lettre claire=lettre codée - n mod 26

On a affaire à un **code en continu ou par flots, symétrique** ou à **clef secrète**.

Codes de substitution

Avantage: on peut considérer toute la famille des codes

lettre codée= lettre claire+n modulo 26

où **n** est un entier entre 0 et 25 appelé la **clef** du code.

Avec la clef $\mathbf{n = 7}$ le texte codé du message précédent devient:

YLUKLG CVBZ KLTHPU TPKP CPSSLAHULBZLBZL

Le **décodage** se fait en utlisant la relation

lettre claire=lettre codée - n mod 26

On a affaire à un **code en continu ou par flots, symétrique** ou à **clef secrète**.

Codes de substitution

Avantage: on peut considérer toute la famille des codes

lettre codée= lettre claire+n modulo 26

où **n** est un entier entre 0 et 25 appelé la **clef** du code.

Avec la clef $\mathbf{n = 7}$ le texte codé du message précédent devient:

YLUKLG CVBZ KLTHPU TPKP CPSSLAHULBZLBZL

Le **décodage** se fait en utlisant la relation

lettre claire=lettre codée - n mod 26

On a affaire à un **code en continu ou par flots, symétrique** ou à **clef secrète**.

Cryptanalyse des codes de César

Les codes de César ne résistent pas à une attaque à **texte chiffré connu**.

Exemple: Message codé avec une substitution monoalphabétique:

JTVMNKKTVLDEVVTLWTWITKTXU
TLWJERUTVTWTHDXATLIUNEWV.

JTVIEVWELOWENLVVNOEDJJTVLTPTXYTLWTW
UTSNLITTVQXTVXUJXWEJEWTONKKXLT.

Décodage par **analyse de fréquences**.

Cryptanalyse des codes de César

Les codes de César ne résistent pas à une attaque à **texte chiffré connu**.

Exemple: Message codé avec une substitution monoalphabétique:

JTVMNKKTVLDEVVTLWTWITKTXU
TLWJERUTVTWTHDXATLIUNEWV.

JTVIEVWELOWENLVVNOEDJJTVLTPTXYTLWTW
UTSNLITTVQXTVXUJXWEJEWTONKKXLT.

Décodage par **analyse de fréquences**.

Cryptanalyse des codes de César

Les codes de César ne résistent pas à une attaque à **texte chiffré connu**.

Exemple: Message codé avec une substitution monoalphabétique:

JTVMNKKTVLDEVVTLWTWITKTXU
TLWJERUTVTWTHDXATLIUNEWV.

JTVIEVWELOWENLVVNOEDJJTVLTPTXYTLWTW
UTSNLITTVQXTVXUJXWEJEWTONKKXLT.

Décodage par **analyse de fréquences**.

Cryptanalyse des codes de César

Analyse de fréquences

Lettre	% français	% texte	Lettre	% français	% texte
A	9,4	1	N	7,2	5
B	1,0	0	O	5.1	2,5
C	2,6	0	P	2,9	1
D	3,4	2,5	Q	1,1	1
E	15,9	8	R	6,5	1
F	1	0	S	7,9	1
G	1	0	T	7,3	20
H	0,8	1	U	6,2	4,5
I	8,4	3,8	V	2,1	12
J	0,9	5,1	W	0	9,9
K	0	4,7	X	0,3	6
L	5,3	9	Y	0,2	1
M	3,2	1	Z	0,3	0

Cryptanalyse des codes de César

On peut donc faire l'hypothèse T=E, puis V=S(à cause des lettres doublées) puis que les voyelles A, I, O, U correspondent à D,E, N, X et finalement on obtient la correspondance:

A	B	C	D	E	F	G	H	I	J	K	L	M
D	R	O	I	T	S	H	M	E	F	G	J	K

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
L	N	P	Q	U	V	W	X	Y	Z	A	B	C

Cryptanalyse des codes de César

On peut donc faire l'hypothèse T=E, puis V=S(à cause des lettres doublées) puis que les voyelles A, I, O, U correspondent à D,E, N, X et finalement on obtient la correspondance:

A	B	C	D	E	F	G	H	I	J	K	L	M
D	R	O	I	T	S	H	M	E	F	G	J	K

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
L	N	P	Q	U	V	W	X	Y	Z	A	B	C

Cryptanalyse des codes de César

On peut donc faire l'hypothèse T=E, puis V=S(à cause des lettres doublées) puis que les voyelles A, I, O, U correspondent à D,E, N, X et finalement on obtient la correspondance:

A	B	C	D	E	F	G	H	I	J	K	L	M
D	R	O	I	T	S	H	M	E	F	G	J	K

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
L	N	P	Q	U	V	W	X	Y	Z	A	B	C

Cryptanalyse des codes de César

On peut donc faire l'hypothèse T=E, puis V=S(à cause des lettres doublées) puis que les voyelles A, I, O, U correspondent à D,E, N, X et finalement on obtient la correspondance:

A	B	C	D	E	F	G	H	I	J	K	L	M
D	R	O	I	T	S	H	M	E	F	G	J	K

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
L	N	P	Q	U	V	W	X	Y	Z	A	B	C

Codes polyalphabétiques

• La faiblesse des codes monoalphabétiques où chaque lettre est codé tout au long d'une message par une unique autre lettre, est leur faiblesse à l'analyse des **fréquence des lettres**.

• L'idée est alors de considérer des codes polyalphabétiques où la table de substitution va dépendre **de la position de la lettre considérée dans le texte**.

L'exemple le plus célèbre est certainement **le code de Vigénère** mis au point par Leon Batista Alberti au 15-ième siècle et développés par Blaise de Vigénère. En vérité peu utilisé car complexe à mettre en oeuvre, il fut longtemps considéré comme incassable, puis finalement cryptanalysé par **Charles Babbage** et **Friedrich Wilhelm Kasiski** au 19-ième siècle.

Codes polyalphabétiques

• La faiblesse des codes monoalphabétiques où chaque lettre est codé tout au long d'une message par une unique autre lettre, est leur faiblesse à l'analyse des **fréquence des lettres**.

• L'idée est alors de considérer des codes polyalphabétiques où la table de substitution va dépendre **de la position de la lettre considérée dans le texte**.

L'exemple le plus célèbre est certainement **le code de Vigénère** mis au point par Leon Batista Alberti au 15-ième siècle et développés par Blaise de Vigénère. En vérité peu utilisé car complexe à mettre en oeuvre, il fut longtemps considéré comme incassable, puis finalement cryptanalysé par **Charles Babbage** et **Friedrich Wilhelm Kasiski** au 19-ième siècle.

Codes polyalphabétiques

• La faiblesse des codes monoalphabétiques où chaque lettre est codé tout au long d'une message par une unique autre lettre, est leur faiblesse à l'analyse des **fréquence des lettres**.
• L'idée est alors de considérer des codes polyalphabétiques où la table de substitution va dépendre **de la position de la lettre considérée dans le texte**.
L'exemple le plus célèbre est certainement **le code de Vigénère** mis au point par Leon Batista Alberti au 15-ième siècle et développés par Blaise de Vigénère. En vérité peu utilisé car complexe à mettre en oeuvre, il fut longtemps considéré comme incassable, puis finalement cryptanalysé par **Charles Babbage** et **Friedrich Wilhelm Kasiski** au 19-ième siècle.

Code de Vigénère

La première étape consiste à choisir un entier n secret et à écrire le texte sur n lignes suivant le principe de la scytale. Prenons par exemple le texte: ATTAQUER DEMAIN A L'AUBE LA TOUR EIFFEL avec $n = 5$:

A	U	M	L	L	R	I
T	E	A	A	A	E	L
T	R	I	U	T	F	
A	D	N	B	O	F	
Q	E	A	E	U	E	

Code de Vigénère

La première étape consiste à choisir un entier n secret et à écrire le texte sur n lignes suivant le principe de la scytale. Prenons par exemple le texte: ATTAQUER DEMAIN A L'AUBE LA TOUR EIFFEL avec $n = 5$:

A	U	M	L	L	R	I
T	E	A	A	A	E	L
T	R	I	U	T	F	
A	D	N	B	O	F	
Q	E	A	E	U	E	

Code de Vigénère

On effectue alors un code de César sur chacune des lignes avec un décalage distinct pour chaque ligne: considérons par exemple les décalages $+2, -2, +4, -3, +1$:

$$\begin{array}{ccc} AUMLLRI & \longrightarrow & CWONNTK \\ TEAAAEL & \longrightarrow & RCYYCJ \\ TRIUTF & \longrightarrow & XVMYXJ \\ ADNBOF & \longrightarrow & XZJYKB \\ QEAEUE & \longrightarrow & RFBFVF \end{array}$$

On notera que les A présents sur les lignes 1,2,4,5 sont tous codés différemment ce qui dissimule la fréquence de la lettre A dans le texte original.

Code de Vigénère

On effectue alors un code de César sur chacune des lignes avec un décalage distinct pour chaque ligne: considérons par exemple les décalages $+2, -2, +4, -3, +1$:

$$\begin{array}{ccc} AUMLLRI & \longrightarrow & CWONNTK \\ TEAAAEL & \longrightarrow & RCYYCJ \\ TRIUTF & \longrightarrow & XVMYXJ \\ ADNBOF & \longrightarrow & XZJYKB \\ QEAEUE & \longrightarrow & RFBFVF \end{array}$$

On notera que les A présents sur les lignes 1,2,4,5 sont tous codés différemment ce qui dissimule la fréquence de la lettre A dans le texte original.

Code de Vigénère

On écrit alors le texte dans l'ordre

C	W	O	N	N	T	K
R	C	Y	Y	C	J	
X	V	M	Y	X	J	
X	Z	J	Y	K	B	
R	F	B	F	V	F	

Ce qui donne:

C R X X R W C V Z F O Y M J B N Y Y Y F N C X K V T J J B F K

Cryptanalyse du code de Vigénère

Sachant décoder un code de César par analyse de fréquence, on voit qu'il nous suffit de déterminer l'entier n.

On cherche des répétitions dans le texte chiffré, particulièrement des **trigrammes**, i.e. trois lettres consécutives, dont la présence est due:

- soit à une pure coïncidence, des trigrammes distincts du texte en clair se retrouvent codés de la même façon,
- soit à **un même trigramme du texte original** codé avec la même clef de sorte que ces deux trigrammes sont espacés **d'un multiple de n.**

Dans un texte suffisamment long, de nombreux trigrammes laissent peu de chances aux coïncidences, ce qui nous renseigne sur n qui doit diviser **le pgcd** des distances entre les différents trigrammes.

Cryptanalyse du code de Vigénère

Sachant décoder un code de César par analyse de fréquence, on voit qu'il nous suffit de déterminer l'entier n.
On cherche des répétitions dans le texte chiffré, particulièrement des **trigrammes**, i.e. trois lettres consécutives, dont la présence est due:

- soit à une pure coïncidence, des trigrammes distincts du texte en clair se retrouvent codés de la même façon,
- soit à **un même trigramme du texte original** codé avec la même clef de sorte que ces deux trigrammes sont espacés **d'un multiple de n.**

Dans un texte suffisamment long, de nombreux trigrammes laissent peu de chances aux coïncidences, ce qui nous renseigne sur n qui doit diviser **le pgcd** des distances entre les différents trigrammes.

Cryptanalyse du code de Vigénère

Sachant décoder un code de César par analyse de fréquence, on voit qu'il nous suffit de déterminer l'entier n.
On cherche des répétitions dans le texte chiffré, particulièrement des **trigrammes**, i.e. trois lettres consécutives, dont la présence est due:

- soit à une pure coïncidence, des trigrammes distincts du texte en clair se retrouvent codés de la même façon,
- soit à **un même trigramme du texte original** codé avec la même clef de sorte que ces deux trigrammes sont espacés **d'un multiple de n.**

Dans un texte suffisamment long, de nombreux trigrammes laissent peu de chances aux coïncidences, ce qui nous renseigne sur n qui doit diviser **le pgcd** des distances entre les différents trigrammes.

Indice de coïncidence

• En 1920, W. F. Friedman a introduit la notion **d'indice de coïncidence** d'un texte qui est la probabilité qu'un tirage au hasard de deux lettres du texte donne un doublon. Sa formule est donnée par

$$IC = \sum_{i=1}^{26} \frac{n_i(n_i - 1)}{n(n-1)}$$

où n est la longueur du texte et n_1 le nombre de A, n_2 le nombre de B...

• L'intérêt de cette notion est son indépendance relativement aux substitutions.

Indice de coïncidence

• En 1920, W. F. Friedman a introduit la notion **d'indice de coïncidence** d'un texte qui est la probabilité qu'un tirage au hasard de deux lettres du texte donne un doublon. Sa formule est donnée par

$$IC = \sum_{i=1}^{26} \frac{n_i(n_i - 1)}{n(n-1)}$$

où n est la longueur du texte et n_1 le nombre de A, n_2 le nombre de B...

• L'intérêt de cette notion est son indépendance relativement aux substitutions.

Indice de coïncidence

- En français l'indice de coïncidence avoisine les **7,5%** alors que pour un texte aléatoire il est de $\frac{1}{26}$ soit **3,85 %**.
- Ainsi pour identifier l'entier m d'un code de Vigénère, on peut successivement diviser le texte codé en $2, 3, \cdots$ lignes et calculer les indices de coïncidences respectives jusqu'à s'éloigner des 4%.
- On peut aussi utiliser la formule suivante:

$$IC = \frac{n-m}{m(n-1)}0,075 + \frac{n(m-1)}{(n-1)m}0,0385,$$

afin d'identifier l'entier m.
- Pour un texte court dont la longueur est proche de l'entier m, le code de Vigénère se rapproche d'un code de Vernam dit aussi **à masque jetable** et parfaitement sûr.

Indice de coïncidence

• En français l'indice de coïncidence avoisine les **7,5%** alors que pour un texte aléatoire il est de $\frac{1}{26}$ soit **3,85 %**.

• Ainsi pour identifier l'entier m d'un code de Vigénère, on peut successivement diviser le texte codé en $2, 3, \cdots$ lignes et calculer les indices de coïncidences respectives jusqu'à s'éloigner des 4%.

• On peut aussi utiliser la formule suivante:

$$IC = \frac{n-m}{m(n-1)}0,075 + \frac{n(m-1)}{(n-1)m}0,0385,$$

afin d'identifier l'entier m.

• Pour un texte court dont la longueur est proche de l'entier m, le code de Vigénère se rapproche d'un code de Vernam dit aussi **à masque jetable** et parfaitement sûr.

Indice de coïncidence

• En français l'indice de coïncidence avoisine les **7,5%** alors que pour un texte aléatoire il est de $\frac{1}{26}$ soit **3,85 %**.
• Ainsi pour identifier l'entier m d'un code de Vigénère, on peut successivement diviser le texte codé en $2, 3, \cdots$ lignes et calculer les indices de coïncidences respectives jusqu'à s'éloigner des 4%.
• On peut aussi utiliser la formule suivante:

$$IC = \frac{n-m}{m(n-1)}0,075 + \frac{n(m-1)}{(n-1)m}0,0385,$$

afin d'identifier l'entier m.
• Pour un texte court dont la longueur est proche de l'entier m, le code de Vigénère se rapproche d'un code de Vernam dit aussi **à masque jetable** et parfaitement sûr.

Indice de coïncidence

• En français l'indice de coïncidence avoisine les **7,5%** alors que pour un texte aléatoire il est de $\frac{1}{26}$ soit **3,85 %**.
• Ainsi pour identifier l'entier m d'un code de Vigénère, on peut successivement diviser le texte codé en $2, 3, \cdots$ lignes et calculer les indices de coïncidences respectives jusqu'à s'éloigner des 4%.
• On peut aussi utiliser la formule suivante:

$$IC = \frac{n-m}{m(n-1)}0,075 + \frac{n(m-1)}{(n-1)m}0,0385,$$

afin d'identifier l'entier m.
• Pour un texte court dont la longueur est proche de l'entier m, le code de Vigénère se rapproche d'un code de Vernam dit aussi **à masque jetable** et parfaitement sûr.

Code de VIC: l'apogée du codage manuel

- Le chiffre VIC utilisé par les espions russes pendant la guerre froide, est probablement le cryptosystème classique le plus avancé jamais conçu. Avant la défection en 1957 d'un agent russe, tous les efforts de la NSA pour le casser avaient été infructueux.
- **Un échiquier à diffusion** est un tableau de 3 lignes et 10 colonnes: la première contient les 8 lettres les plus fréquentes soit **RUSTINEA** en français, et deux cases blanches dans un ordre arbitraire et **secret**. Les deux autres lignes contiennent le reste de l'alphabet plus deux caractères, par exemple le point et l'apostrophe.

	0	1	2	3	4	5	6	7	8	9
	T	A		R	U	I	N	E		S
2	B	C	D	F	G	H	J	K	L	M
8	O	.	P	Q	V	'	W	X	Y	Z

Les deux lignes du bas portent le numéro des colonnes qui correspondent aux cases vides en première ligne.

Code de VIC: l'apogée du codage manuel

- Le chiffre VIC utilisé par les espions russes pendant la guerre froide, est probablement le cryptosystème classique le plus avancé jamais conçu. Avant la défection en 1957 d'un agent russe, tous les efforts de la NSA pour le casser avaient été infructueux.
- **Un échiquier à diffusion** est un tableau de 3 lignes et 10 colonnes: la première contient les 8 lettres les plus fréquentes soit **RUSTINEA** en français, et deux cases blanches dans un ordre arbitraire et **secret**. Les deux autres lignes contiennent le reste de l'alphabet plus deux caractères, par exemple le point et l'apostrophe.

	0	1	2	3	4	5	6	7	8	9
	T	A		R	U	I	N	E		S
2	B	C	D	F	G	H	J	K	L	M
8	O	.	P	Q	V	'	W	X	Y	Z

Les deux lignes du bas portent le numéro des colonnes qui correspondent aux cases vides en première ligne.

Code de VIC: l'apogée du codage manuel

- Le chiffre VIC utilisé par les espions russes pendant la guerre froide, est probablement le cryptosystème classique le plus avancé jamais conçu. Avant la défection en 1957 d'un agent russe, tous les efforts de la NSA pour le casser avaient été infructueux.
- **Un échiquier à diffusion** est un tableau de 3 lignes et 10 colonnes: la première contient les 8 lettres les plus fréquentes soit **RUSTINEA** en français, et deux cases blanches dans un ordre arbitraire et **secret**. Les deux autres lignes contiennent le reste de l'alphabet plus deux caractères, par exemple le point et l'apostrophe.

	0	1	2	3	4	5	6	7	8	9
	T	A		R	U	I	N	E		S
2	B	C	D	F	G	H	J	K	L	M
8	O	.	P	Q	V	'	W	X	Y	Z

Les deux lignes du bas portent le numéro des colonnes qui correspondent aux cases vides en première ligne.

Code de VIC: l'apogée du codage manuel

• Le chiffre VIC utilisé par les espions russes pendant la guerre froide, est probablement le cryptosystème classique le plus avancé jamais conçu. Avant la défection en 1957 d'un agent russe, tous les efforts de la NSA pour le casser avaient été infructueux.

• **Un échiquier à diffusion** est un tableau de 3 lignes et 10 colonnes: la première contient les 8 lettres les plus fréquentes soit **RUSTINEA** en français, et deux cases blanches dans un ordre arbitraire et **secret**. Les deux autres lignes contiennent le reste de l'alphabet plus deux caractères, par exemple le point et l'apostrophe.

	0	1	2	3	4	5	6	7	8	9
	T	A		R	U	I	N	E		S
2	B	C	D	F	G	H	J	K	L	M
8	O	.	P	Q	V	'	W	X	Y	Z

Les deux lignes du bas portent le numéro des colonnes qui correspondent aux cases vides en première ligne.

Code de VIC

- Chaque lettre du texte clair est transformée en:
  - le numéro de la colonne correspondante si la lettre apparaît en première ligne: par exemple N est codé avec 6;
  - le numéro de la ligne suivi du numéro de la colonne si la lettre apparaît dans les deux autres lignes: par exemple Q est codé avec 83.

Par exemple ATTAQUER TOUR EIFFEL A L'AUBE est codé par le tableau ci-dessus:

A	T	T	A	Q	U	E	R	T	O	U	R	E	F	F	E	I	L	A	L	'	A	U	B	E
1	0	0	1	83	4	7	3	0	80	4	3	7	5	23	23	7	28	1	28	85	1	4	20	7

- Après la phase d'encodage par l'échiquier à diffusion, le chiffre VIC applique un chiffrement par permutation reposant sur la construction d'une clef secrète de double permutation assez semblable au fonctionnement d'un registre à décalage via la relation de Fibonacci modulo 10 ainsi qu'un IV aléatoire.

Code de VIC

- Chaque lettre du texte clair est transformée en:
  - le numéro de la colonne correspondante si la lettre apparaît en première ligne: par exemple N est codé avec 6;
  - le numéro de la ligne suivi du numéro de la colonne si la lettre apparaît dans les deux autres lignes: par exemple Q est codé avec 83.

Par exemple ATTAQUER TOUR EIFFEL A L'AUBE est codé par le tableau ci-dessus:

A	T	T	A	Q	U	E	R	T	O	U	R	E	F	F	E	I	L	A	L	'	A	U	B	E
1	0	0	1	83	4	7	3	0	80	4	3	7	5	23	23	7	28	1	28	85	1	4	20	7

- Après la phase d'encodage par l'échiquier à diffusion, le chiffre VIC applique un chiffrement par permutation reposant sur la construction d'une clef secrète de double permutation assez semblable au fonctionnement d'un registre à décalage via la relation de Fibonacci modulo 10 ainsi qu'un IV aléatoire.

Code de VIC

- Chaque lettre du texte clair est transformée en:
  - le numéro de la colonne correspondante si la lettre apparaît en première ligne: par exemple N est codé avec 6;
  - le numéro de la ligne suivi du numéro de la colonne si la lettre apparaît dans les deux autres lignes: par exemple Q est codé avec 83.

Par exemple ATTAQUER TOUR EIFFEL A L'AUBE est codé par le tableau ci-dessus:

A	T	T	A	Q	U	E	R	T	O	U	R	E	F	F	E	I	L	A	L	'	A	U	B	E
1	0	0	1	83	4	7	3	0	80	4	3	7	5	23	23	7	28	1	28	85	1	4	20	7

- Après la phase d'encodage par l'échiquier à diffusion, le chiffre VIC applique un chiffrement par permutation reposant sur la construction d'une clef secrète de double permutation assez semblable au fonctionnement d'un registre à décalage via la relation de Fibonacci modulo 10 ainsi qu'un IV aléatoire.

Initialisation des clefs

- L'agent secret retient une date ce qui lui donne 6 chiffres: par exemple le 4 août 1789, soit 481789;
- il choisit un nombre de 5 chiffres au hasard (IV) transmis en clair mais de façon cachée, par exemple 35218 et lui soustrait sans retenue les 5 premiers chiffres du nombre précédent:

$$\begin{array}{rccccc} & 3 & 5 & 2 & 1 & 8 \\ - & 4 & 8 & 1 & 7 & 8 \\ \hline & 9 & 7 & 1 & 4 & 0 \end{array}$$

Remarque: le dernier chiffre 9 de la date secrète servira à transmettre la donnée aléatoire en précisant sa position dans le texte final, ici le 9ième groupe en partant de la fin, sachant que chaque groupe est constitué de 5 chiffres. Il faudra alors éventuellement compléter le message codé afin d'obtenir un nombre de chiffres divisible par 5.

Initialisation des clefs

- L'agent secret retient une date ce qui lui donne 6 chiffres: par exemple le 4 août 1789, soit 481789;
- il choisit un nombre de 5 chiffres au hasard (IV) transmis en clair mais de façon cachée, par exemple 35218 et lui soustrait sans retenue les 5 premiers chiffres du nombre précédent:

$$\begin{array}{rccccc} & 3 & 5 & 2 & 1 & 8 \\ - & 4 & 8 & 1 & 7 & 8 \\ \hline & 9 & 7 & 1 & 4 & 0 \end{array}$$

Remarque: le dernier chiffre 9 de la date secrète servira à transmettre la donnée aléatoire en précisant sa position dans le texte final, ici le 9ième groupe en partant de la fin, sachant que chaque groupe est constitué de 5 chiffres. Il faudra alors éventuellement compléter le message codé afin d'obtenir un nombre de chiffres divisible par 5.

Initialisation des clefs

On étend alors le nombre obtenu via une addition sans retenue à la Fibonacci pour obtenir un nombre de 10 chiffres appelé clef temporelle:

$$9714068546.$$

- Le début d'une chanson lui donne 20 lettres, par exemple à la claire fontaine m'en... qu'il découpe en deux et numérote les lettres relativement à leur position dans l'alphabet, 0 valant 10 ce qui donne les deux clefs musicales

A	L	A	C	L	A	I	R	E	F	O	N	T	A	I	N	E	M	E	N
1	8	2	4	9	3	7	0	5	6	9	6	0	1	4	7	2	5	3	8

- On additionne sans retenue la première clef musicale avec la clef temporelle pour obtenir la clef de préencodage

$$\begin{array}{cccccccccc} & 1 & 8 & 2 & 4 & 9 & 3 & 7 & 0 & 5 & 6 \\ + & 9 & 7 & 1 & 4 & 0 & 6 & 8 & 5 & 4 & 6 \\ \hline & 0 & 5 & 3 & 8 & 9 & 9 & 5 & 5 & 9 & 2 \end{array}$$

Initialisation des clefs

On étend alors le nombre obtenu via une addition sans retenue à la Fibonacci pour obtenir un nombre de 10 chiffres appelé clef temporelle:

$$9714068546.$$

- Le début d'une chanson lui donne 20 lettres, par exemple à la claire fontaine m'en... qu'il découpe en deux et numérote les lettres relativement à leur position dans l'alphabet, 0 valant 10 ce qui donne les deux clefs musicales

A	L	A	C	L	A	I	R	E	F	O	N	T	A	I	N	E	M	E	N
1	8	2	4	9	3	7	0	5	6	9	6	0	1	4	7	2	5	3	8

- On additionne sans retenue la première clef musicale avec la clef temporelle pour obtenir la clef de préencodage

$$\begin{array}{cccccccccc} 1 & 8 & 2 & 4 & 9 & 3 & 7 & 0 & 5 & 6 \\ + & 9 & 7 & 1 & 4 & 0 & 6 & 8 & 5 & 4 & 6 \\ \hline 0 & 5 & 3 & 8 & 9 & 9 & 5 & 5 & 9 & 2 \end{array}$$

Initialisation des clefs

On étend alors le nombre obtenu via une addition sans retenue à la Fibonacci pour obtenir un nombre de 10 chiffres appelé clef temporelle:

$$9714068546.$$

- Le début d'une chanson lui donne 20 lettres, par exemple à la claire fontaine m'en... qu'il découpe en deux et numérote les lettres relativement à leur position dans l'alphabet, 0 valant 10 ce qui donne les deux clefs musicales

A	L	A	C	L	A	I	R	E	F	O	N	T	A	I	N	E	M	E	N
1	8	2	4	9	3	7	0	5	6	9	6	0	1	4	7	2	5	3	8

- On additionne sans retenue la première clef musicale avec la clef temporelle pour obtenir la clef de préencodage

$$\begin{array}{cccccccccc} & 1 & 8 & 2 & 4 & 9 & 3 & 7 & 0 & 5 & 6 \\ + & 9 & 7 & 1 & 4 & 0 & 6 & 8 & 5 & 4 & 6 \\ \hline & 0 & 5 & 3 & 8 & 9 & 9 & 5 & 5 & 9 & 2 \end{array}$$

Construction de l'échiquier à diffusion

- On utilise la deuxième clef musicale

1	2	3	4	5	6	7	8	9	0
9	6	0	1	4	7	2	5	3	8

pour coder la clef de préencodage en la clef génératrice:

0	5	3	8	9	9	5	5	9	2
8	4	0	5	3	3	4	4	3	6

- part le procédé de Fibonacci, on produit une suite de 50 chiffres:

8	4	0	5	3	3	4	4	3	6
2	4	5	8	6	7	8	7	9	8
6	9	3	4	3	5	5	6	7	4
5	2	7	7	8	0	1	3	1	9
7	9	4	5	8	1	4	4	0	6
6	3	9	3	9	5	8	4	6	2

Construction de l'échiquier à diffusion

- On utilise la deuxième clef musicale

1	2	3	4	5	6	7	8	9	0
9	6	0	1	4	7	2	5	3	8

pour coder la clef de préencodage en la clef génératrice:

0	5	3	8	9	9	5	5	9	2
8	4	0	5	3	3	4	4	3	6

- part le procédé de Fibonacci, on produit une suite de 50 chiffres:

8	4	0	5	3	3	4	4	3	6
2	4	5	8	6	7	8	7	9	8
6	9	3	4	3	5	5	6	7	4
5	2	7	7	8	0	1	3	1	9
7	9	4	5	8	1	4	4	0	6
6	3	9	3	9	5	8	4	6	2

Construction de l'échiquier à diffusion

- On utilise la dernière ligne 6393958462 que l'on numérote par ordre lexicographique ce qui donne ici

  6293058471

  qui nous fournit l'échiquier de diffusion

6	2	9	3	0	5	8	4	7	1
R	*U*	*S*	*T*	*I*	*N*	*E*	*A*		

	0	1	2	3	4	5	6	7	8	9
	I		U	T	A	N	R		E	S
1	B	C	D	F	G	H	J	K	L	M
7	O	.	P	Q	V	'	W	X	Y	Z

Codage du message

À ce stade, on peut coder le message, selon le procédé décrit plus haut. Supposons que l'on souhaite écrire

nous sommes heureux d'apprendre que votre candidature a la nsa a ete acceptee. nous vous envoyons de l'argent pour couvrir vos depenses.

On obtient alors le message codé
5702997019198915826728277127546728512672873287470367
281145120124326728418459448384111183887157029747029857
470787059128187546721485370267211702746720672747091288598971

Première permutation des colonnes

• Chaque agent secret possède **un code personnel** compris entre 1 et 16, prenons 10 qu'il additionne au deux derniers chiffres du tableau de 50 chiffres obtenus plus haut, soit dans notre cas 6 et 2 ce qui donne 16 et 12.

• On lit alors les $16 + 12 = 28$ premiers chiffres du tableau de 50 chiffres précédents, colonne par colonne, l'ordre des colonnes étant donné par la clef génératrice 8405334436 pour le produire, soit dans notre cas:

63889 75015 97106 49293 85148 763...

Première permutation des colonnes

• Chaque agent secret possède **un code personnel** compris entre 1 et 16, prenons 10 qu'il additionne au deux derniers chiffres du tableau de 50 chiffres obtenus plus haut, soit dans notre cas 6 et 2 ce qui donne 16 et 12.
• On lit alors les $16 + 12 = 28$ premiers chiffres du tableau de 50 chiffres précédents, colonne par colonne, l'ordre des colonnes étant donné par la clef génératrice 8405334436 pour le produire, soit dans notre cas:

63889 75015 97106 49293 85148 763...

Première permutation des colonnes

Comme notre message est constitué de 166 chiffres, pour en avoir un multiple de 5 on ajoute 4 chiffres, par exemple 1305 (FIN). On répartit alors notre message sur 16 colonnes numérotées par les 16 premiers chiffres

6	3	8	8	9	7	5	0	1	5	9	7	1	0	6	4
5	7	0	2	9	9	7	0	1	9	1	9	8	9	1	5
8	2	6	7	2	8	2	7	7	1	2	7	5	4	6	7
2	8	5	1	2	6	7	2	8	7	3	2	8	7	4	7
0	3	6	7	2	8	1	1	4	5	1	2	0	1	2	4
3	2	6	7	2	8	4	1	8	4	5	9	4	4	8	3
8	4	1	1	1	1	8	3	8	8	7	1	5	7	0	2
9	7	4	7	0	2	9	8	5	7	4	7	0	7	8	7
0	5	9	1	2	8	1	8	7	5	4	6	7	2	1	4
8	5	3	7	0	2	6	7	2	1	1	7	0	2	7	4
6	7	2	0	6	7	2	7	4	7	0	9	1	2	8	8
5	9	8	9	7	1	1	3	0	5						

Première permutation des colonnes

On lit alors les colonnes dans l'ordre indiqué par leur label soit

17848857240 8580450701 72832475579 5774327448 72714891621
91754875175 58203890865 1642808178 98688128271 9722917679
06566149328 27177171709 92222102067 1231574410 07211388773
9471477222

Deuxième permutation des colonnes

Les 12 derniers chiffres 929385148763 de notre série de 28, vont coder la deuxième permutation de colonnes mais le remplissage est un peu différent. On commence par construire deux triangles dans ce tableau de 15×12:

- le premier commence sur la première ligne exactement sur la première colonne qui sera lue, ici la 11ième, part sur la droite en avançant d'un par ligne et colonne;
- le deuxième part sur la ligne suivant celle de la fin du premier triangle sur la colonne qui sera lue en deuxième, ici la 2ième.

Deuxième permutation des colonnes

On commence alors à remplir les lignes mais en évitant les triangles:

9	2	9	3	8	5	1	4	8	7	6	3
1	7	8	4	8	8	5	7	2	4		
0	8	5	8	0	4	5	0	7	0	1	
7	2	8	3	2	4	7	5	5	7	9	5
7											
7	4										
3	2	7									
4	4	8	7								
2	7	1	4	8							
9	1	6	2	1	9						
1	7	5	4	8	7	5					
1	7	5	5	8	2	0	3				
8	9	0	8	6	5	1	6	4			
2	8	0	8	1	7	8	9	8	6		
8	8	1	2	8	2	7	1	9	7	2	
2	9										

Deuxième permutation des colonnes

Puis on remplit les triangles vides avec les chiffres restant:

9	2	9	3	8	5	1	4	8	7	6	3
1	7	8	4	8	8	5	7	2	4	1	7
0	8	5	8	0	4	5	0	7	0	1	6
7	2	8	3	2	4	7	5	5	7	9	5
7	7	9	0	6	5	6	6	1	4	9	3
7	4	2	8	2	7	1	7	7	1	7	1
3	2	7	7	0	9	9	2	2	2	2	1
4	4	8	7	0	2	0	6	7	1	2	3
2	7	1	4	8	1	5	7	4	4	1	0
9	1	6	2	1	9	0	7	2	1	1	3
1	7	5	4	8	7	5	8	8	7	7	3
1	7	5	5	8	2	0	3	9	4	7	1
8	9	0	8	6	5	1	6	4	4	7	7
2	8	0	8	1	7	8	9	8	6	2	2
8	8	1	2	8	2	7	1	9	7	2	2
2	9										

Deuxième permutation des colonnes

On lit alors les colonnes dans l'ordre indiqué par la clef soit

55761905050187 782742471779889 48308774245882
76531130331722 70567267783691 84457921972572
11997221177722 40741214174467 80262008188618
27517274289489 107773429118282 85892781655001

On forme des groupes de 5 et on utilise le 6ième chiffre de la clef temporelle, soit 9, pour placer en 9ième position en partant de la fin la clef aléatoire 35218, ce qui donne finalement le message:

55761 90505 01877 82742 47177 98894 83087 74245 88276
53113 03317 22705 67267 78369 18445 79219 72572
11997 22117 77224 07412 14174 46780 26200 81886 18275
35218 17274 28948 91077 73429 11828 28589 27816 55001

Plan

Les codes modernes

Familles de codes modernes

- Les **codes par flot** (registres à décalage)
- les **codes par blocs** { les codes symétriques (ou à clé secrète). / les codes asymétriques (ou à clé publique)

Les codes à clef publique sont basés sur la notion de **fonction à sens unique**.

Les codes modernes

Familles de codes modernes

- Les **codes par flot** (registres à décalage)
- les **codes par blocs** { les codes symétriques (ou à clé secrète). / les codes asymétriques (ou à clé publique)

Les codes à clef publique sont basés sur la notion de **fonction à sens unique**.

Les codes modernes

Familles de codes modernes

- Les **codes par flot** (registres à décalage)
- les **codes par blocs** $\left\{ \begin{array}{l} \text{les codes symétriques (ou à clé secrète).} \\ \text{les codes asymétriques (ou à clé publique)} \end{array} \right.$

Les codes à clef publique sont basés sur la notion de **fonction à sens unique**.

Les codes modernes

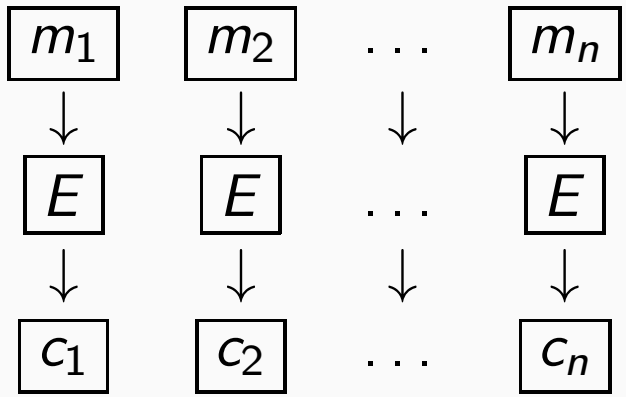
Familles de codes modernes

- Les **codes par flot** (registres à décalage)
- les **codes par blocs** $\begin{cases} \text{les codes symétriques (ou à clé secrète).} \\ \text{les codes asymétriques (ou à clé publique)} \end{cases}$

Les codes à clef publique sont basés sur la notion de **fonction à sens unique**.

Chiffrement en chaîne pour les codes par blocs

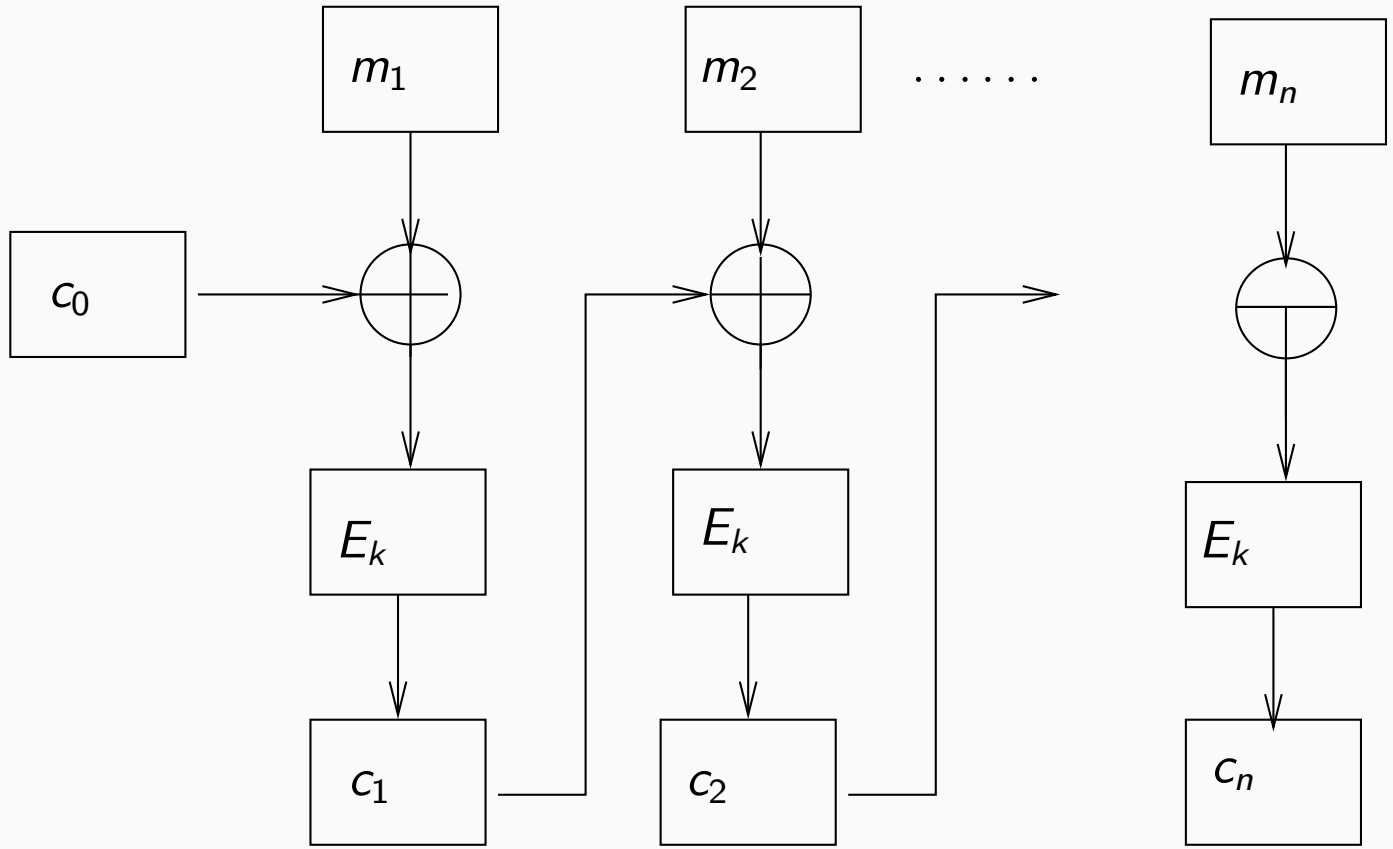
Le mode ECB, Electronic Code Book



Peu sûr, jamais utilisé

Chiffrement en chaîne pour les codes par blocs

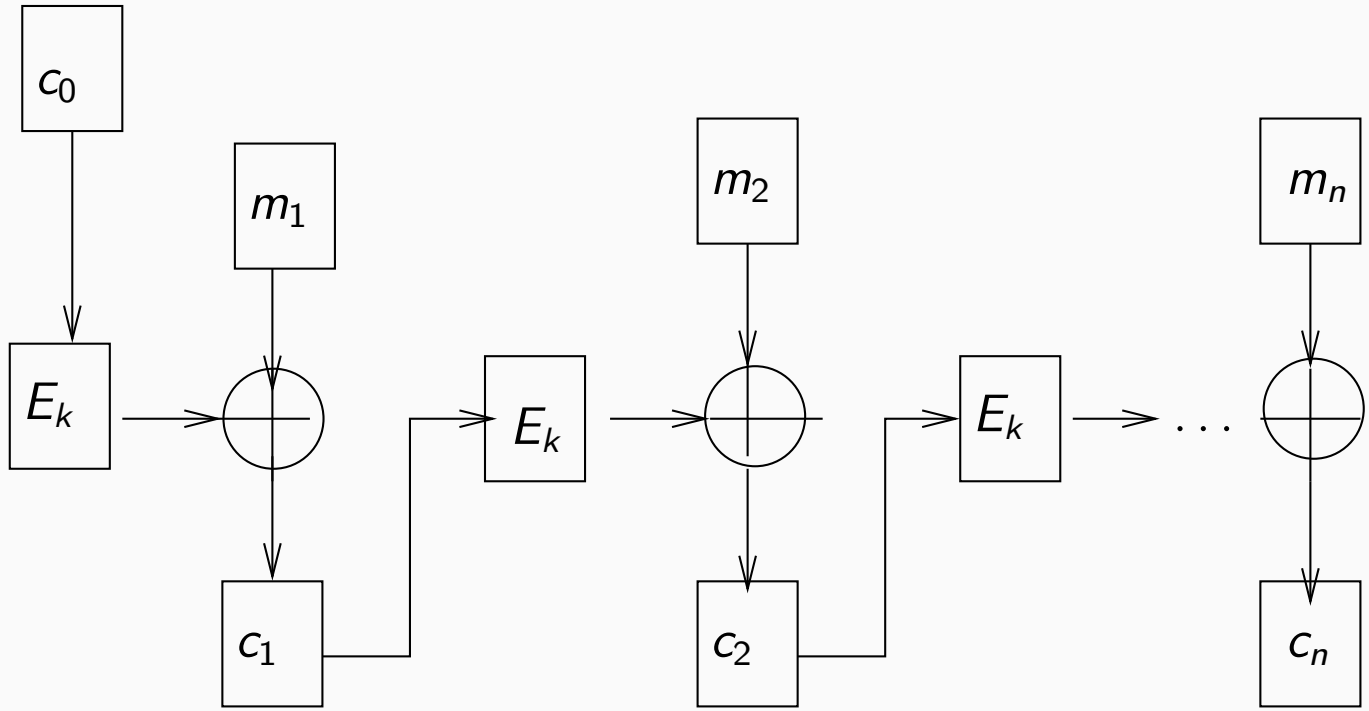
Diagramme du CBC, Cipher Block Chaining



Bonne sécurité, n'affaiblit pas le cryptosystème

Chiffrement en chaîne pour les codes par blocs

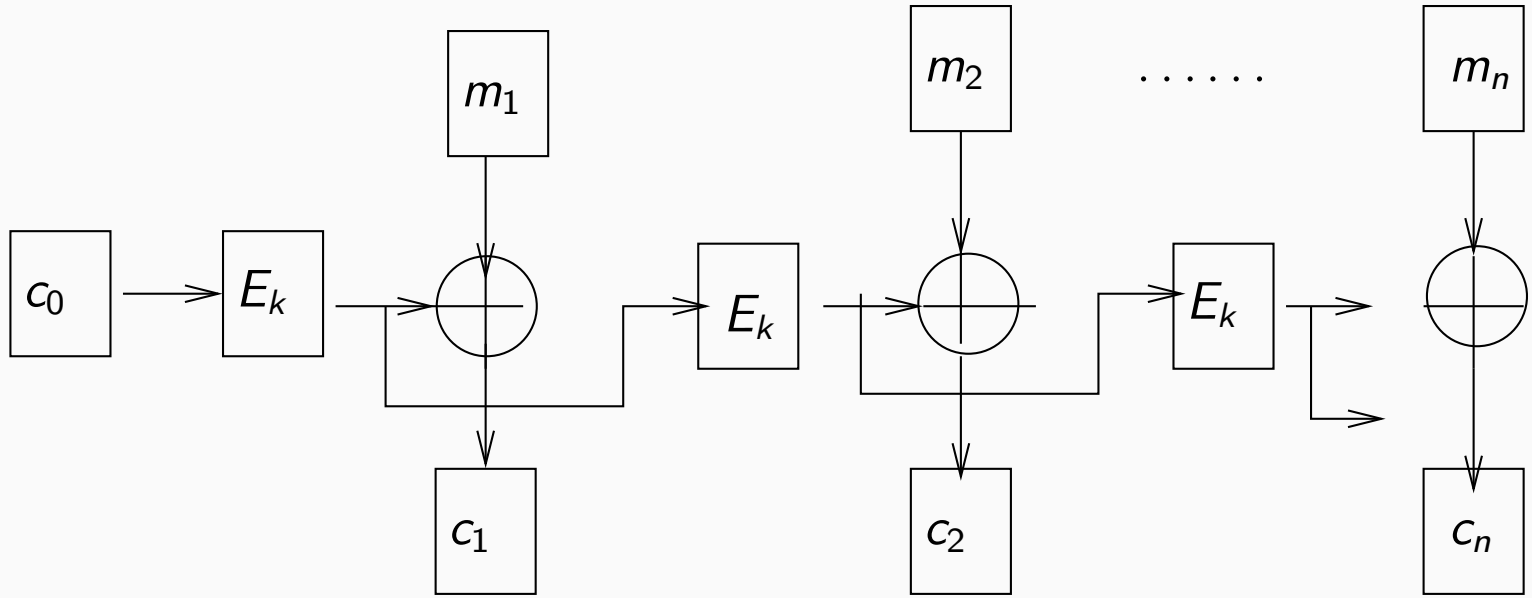
Diagramme du mode CFB, Cipher FeedBack



Un peu moins sûr que CBC, mais plus rapide.

Chiffrement en chaîne pour les codes par blocs

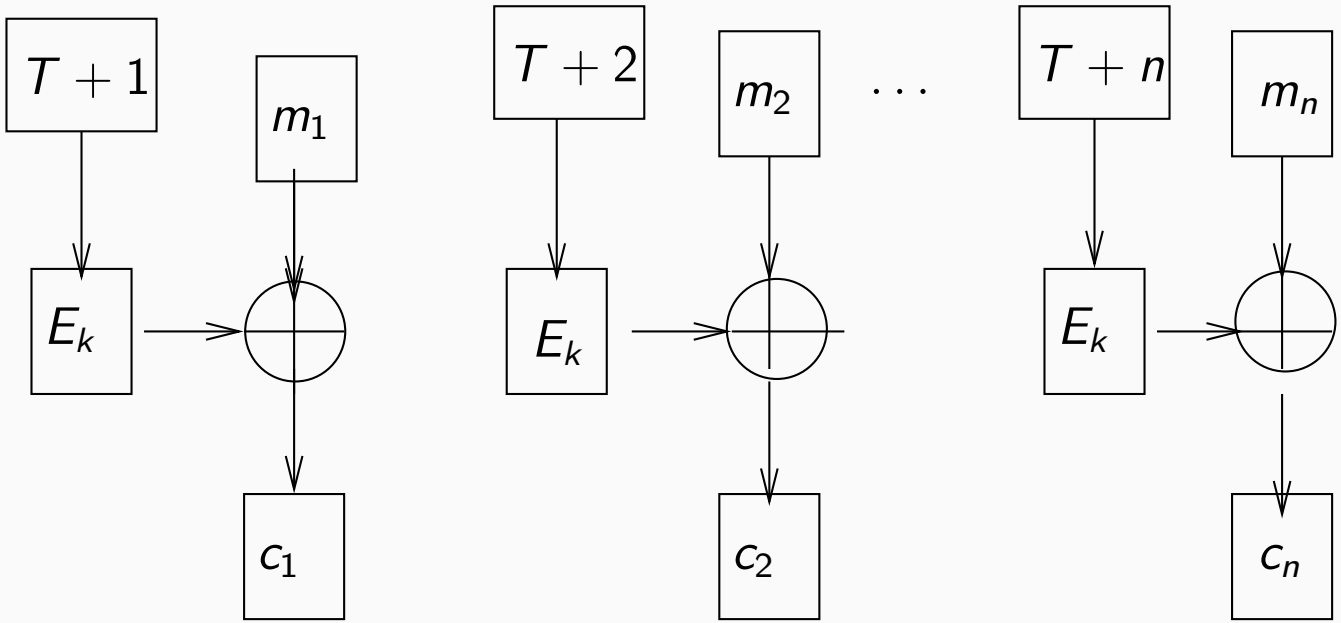
Diagramme du mode OFB, Output FeedBack



Sureté équivalente à celle de CFB

Chiffrement en chaîne pour les codes par blocs

Diagramme du mode CTR, Counter-mode encryption



Sureté équivalente à celle de CFB

Codes à confidentialité parfaite

Il existe des codes à **confidentialité parfaite** si l'on néglige le **problème de la fabrication de clés très longue s au hasard**, de leur stockage et de leur transmission, ce sont les **codes de Vernam** ou **codes à masques jetables**, (1917).

La clé est de la **longueur du texte à coder**, à usage unique, et choisie aléatoirement. Ils sont peu rapides et très lourds à mettre en oeuvre. Rarement utilisés.

Code de Vernam

Exemple de code de Vernam

lettre codée $\equiv$ **lettre claire**+**lettre de la clé** **mod 26**

L'alphabet est codé $A = 1, \ldots, Z = 26$. On envoie un message de 6 lettres

CHIENS + clef **KZUTEG** = **NHDYSZ**

CERISE + clé **KCNPZW** = **NHDYSZ**

Il est donc impossible de remonter au texte clair si on change de clé aléatoirement à chaque fois. **La confidentialité est parfaite** leur sûreté est très élévée. Mais problème de fabrication de **clés au hasard longues et de leur transmission**.

Code de Vernam

Exemple de code de Vernam

lettre codée $\equiv$ **lettre claire**+**lettre de la clé mod 26**

L'alphabet est codé $A = 1, \ldots, Z = 26$. On envoie un message de 6 lettres

CHIENS + **clef** **KZUTEG** = **NHDYSZ**

CERISE + **clé** **KCNPZW** = **NHDYSZ**

Il est donc impossible de remonter au texte clair si on change de clé aléatoirement à chaque fois. **La confidentialité est parfaite** leur sûreté est très élévée. Mais problème de fabrication de **clés au hasard longues et de leur transmission**.

Code de Vernam

Exemple de code de Vernam

lettre codée $\equiv$ **lettre claire**+**lettre de la clé mod 26**

L'alphabet est codé $A = 1, \ldots, Z = 26$. On envoie un message de 6 lettres

CHIENS + **clef** **KZUTEG** = **NHDYSZ**

CERISE + **clé** **KCNPZW** = **NHDYSZ**

Il est donc impossible de remonter au texte clair si on change de clé aléatoirement à chaque fois. **La confidentialité est parfaite** leur sûreté est très élévée. Mais problème de fabrication de **clés au hasard longues et de leur transmission**.

Régistres à décalage ou suites récurrentes linéaires sur un corps fini, LSFR

But: construire des codes analogues aux codes de Vernam mais rapides.

On se donne un entier m, **la longueur de la récurrence**, et des éléments

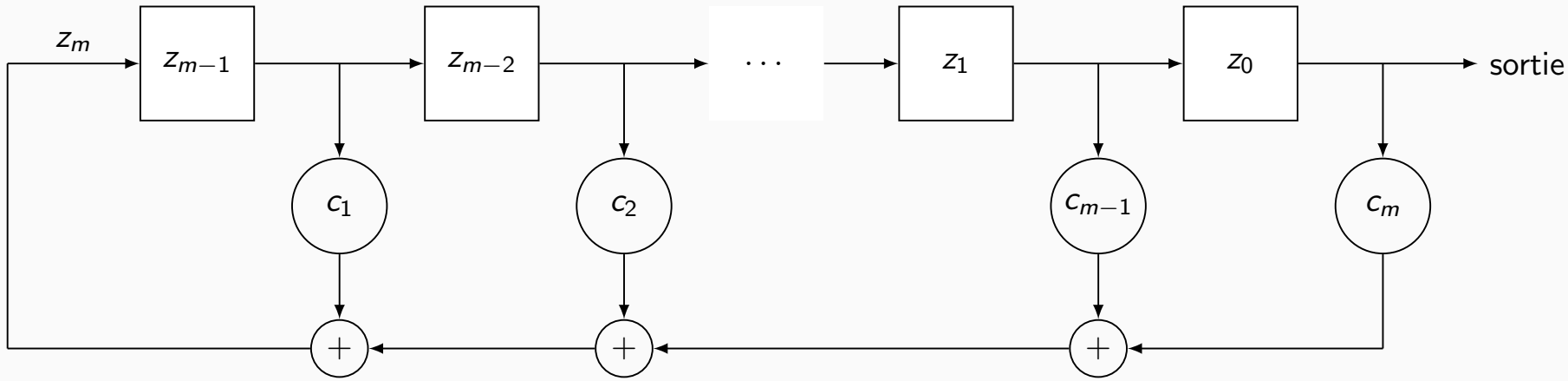
$$k_0, \ldots, k_{m-1} \in \mathrm{Z}/2\mathrm{Z}, \quad \textbf{conditions initiales}$$
$$c_1, \ldots, c_m \in \mathrm{Z}/2\mathrm{Z} \quad \textbf{coefficients de la récurrence}$$

On construit une suite d'éléments $(z_i)_{i\in\mathrm{N}}$ de $\mathrm{Z}/2\mathrm{Z}$:

$$z_0 = k_0, \ldots, \ z_{m-1} = k_{m-1}, \quad z_{m+1} = c_1 z_{m-1} + \cdots + c_m z_0$$
$$z_{m+i} = c_1 z_{m+i-1} + \cdots + c_m z_i = \sum_{j=1}^{m} c_j z_{m+i-j}$$

Régistres à décalage ou LSFR

Diagramme d'un LSFR



Régistres à décalage ou LSFR

Exemple de LFSR

m=4, $(k_1, k_2, k_3, k_4) = (1, 0, 0, 0)$ et

$$z_{i+4} = (z_i + z_{i+1}) \mod 2$$

On vérifie que la période est 15.
Les 14 premiers éléments de la suite sont

$$10001001101111$$

Ensuite la suite se répète.

Régistres à décalage ou LSFR

Chiffrement en chaîne synchrone par XORiration:

message : 1001011101001100 0

clef : 1010011101000101 1

message codé : 0011000000001001 1

Ce type de codage se prète très bien au **codage en temps réel d'images, de sons.** La clef peut être construite par des **registres à décalage** qui peuvent être facilement cablés. Sous la forme indiquée ce système de codage est peu sûr à cause de son caractère linéaire.

Sécurité d'un cryptosystème

Principe de Kerckhoffs

La sécurité d'un cryptosystème **ne repose pas sur le secret du cryptosystème** mais seulement sur la **clef du cryptosystème** qui est un paramètre facile à transmettre secrètement et à changer, de taille réduite (actuellement de 64 à 2048 bits).

Par exemple dans un cryptosystème basé sur des régistres à décalage on suppose que l'attaquant connait **la forme des récurrences linéaires** ainsi que la **fonction de combinaison** mais pas les **conditions intiales des récurrences** qui constituent la clef du code

Régistres à décalage ou LSFR

- Un LSFR est toujours ultimement périodique, i.e. il existe n_0 et T tels que pour tout $n \geq n_0$, on ait $z_{n+T} = z_n$.
- Si $c_m = 1$ alors la suite est périodique, i.e. on peut prendre $n_0 = 0$.
- On a $T \leq 2^m - 1$.

Régistres à décalage ou LSFR

En écrivant le registre comme une matrice colonne $R_i = \begin{pmatrix} z_i \\ z_{i+1} \\ \vdots \\ z_{i+m-1} \end{pmatrix}$,

on a $R_i = AR_{i-1} = A^i R_0$ où

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & & & 0 & 1 & 0 \\ 0 & 0 & \cdots & \cdots & 0 & 1 \\ c_m & c_{m-1} & \cdots & c_3 & c_2 & c_1 \end{pmatrix}.$$

Régistres à décalage ou LSFR

En écrivant le registre comme une matrice colonne $R_i = \begin{pmatrix} z_i \\ z_{i+1} \\ \vdots \\ z_{i+m-1} \end{pmatrix}$,

on a $R_i = AR_{i-1} = A^i R_0$ où

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & & & 0 & 1 & 0 \\ 0 & 0 & \cdots & \cdots & 0 & 1 \\ c_m & c_{m-1} & \cdots & c_3 & c_2 & c_1 \end{pmatrix}.$$

Régistres à décalage ou LSFR

Le polynôme de rétroaction du LFSR est défini par

$$f(X) = 1 + c_1 X + c_2 X^2 + \cdots + c_m X^m.$$

Proposition

La série formelle $z(X) = \sum_{n \geq 0} z_n X^n$ *s'écrit comme une fraction rationnelle*

$$z(x) = \frac{g(X)}{f(X)},$$

avec $g(X) = \sum_{i=0}^{m-1} \left(\sum_{j=0}^{i} c_{i-j} z_j \right) X^i$.

- Le polynôme de rétroaction minimal de la suite $(z_n)_{n\geq 0}$ est l'unique polynôme unitaire $f_0 \in \mathrm{F}_2[X]$ tel qu'il existe g_0 de degré strictement plus petit que celui de f_0 avec

$$z(X) = \frac{g_0(X)}{f_0(X)}.$$

- Le degré de f_0 est appelée la complexité linéaire de $(z_n)_{n\geq 0}$ et correspond à la taille minimal d'un LFSR produisant cette suite.
- Si f_0 de degré r est un diviseur irréductible du polynôme cyclotomique Φ_{2^r-1} alors la période de $(z_n)_{n\geq 0}$ est maximale égale à $2^r - 1$.

- Le polynôme de rétroaction minimal de la suite $(z_n)_{n\geq 0}$ est l'unique polynôme unitaire $f_0 \in \mathrm{F}_2[X]$ tel qu'il existe g_0 de degré strictement plus petit que celui de f_0 avec

$$z(X) = \frac{g_0(X)}{f_0(X)}.$$

- Le degré de f_0 est appelée la complexité linéaire de $(z_n)_{n\geq 0}$ et correspond à la taille minimal d'un LFSR produisant cette suite.
- Si f_0 de degré r est un diviseur irréductible du polynôme cyclotomique Φ_{2^r-1} alors la période de $(z_n)_{n\geq 0}$ est maximale égale à $2^r - 1$.

- Le polynôme de rétroaction minimal de la suite $(z_n)_{n\geq 0}$ est l'unique polynôme unitaire $f_0 \in \mathrm{F}_2[X]$ tel qu'il existe g_0 de degré strictement plus petit que celui de f_0 avec

$$z(X) = \frac{g_0(X)}{f_0(X)}.$$

- Le degré de f_0 est appelée la complexité linéaire de $(z_n)_{n\geq 0}$ et correspond à la taille minimal d'un LFSR produisant cette suite.
- Si f_0 de degré r est un diviseur irréductible du polynôme cyclotomique Φ_{2^r-1} alors la période de $(z_n)_{n\geq 0}$ est maximale égale à $2^r - 1$.

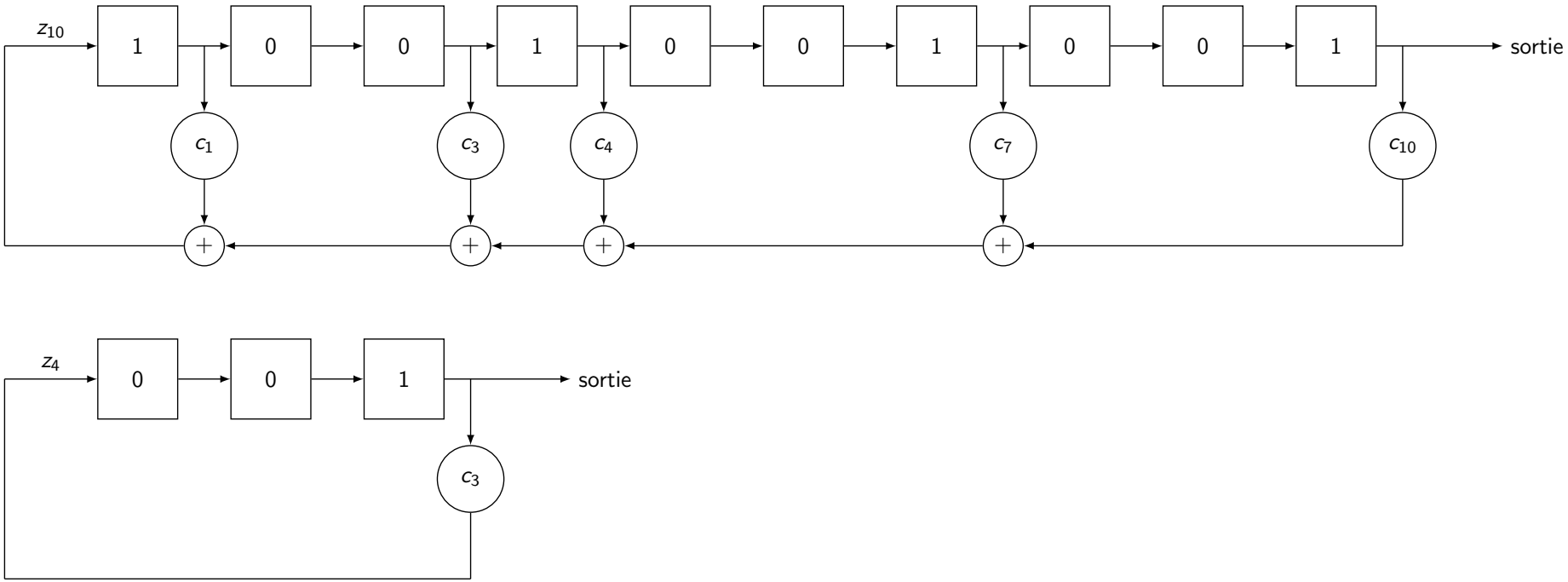
Régistres à décalage ou LSFR

L'égalité

$$\frac{X^7+X+1}{X^{10}+X^7+X^4+X^3+X+1}=\frac{1}{X^3+1}$$

se traduit par une égalité de LFSR.

Régistres à décalage ou LSFR



Régistres à décalage ou LSFR

• On demande à une suite pseudo-alétoire, périodique de période T, de paraître aussi aléatoire que possible ce qui se traduit par trois critères dits de Colomb introduits par celui-ci en 1982. Les LFSR les vérifient.

• L'algorithme de Berlekamp-Massey calcule en temps quadratique la complexité linéaire et un polynôme générateur d'une suite finie. Cet algorithme repose sur l'observation élémentaire qu'une suite récurrente linéaire de complexité L est déterminée par $2L$ de ses termes consécutifs.

$$\begin{pmatrix} z_i & z_{i+1} & z_{i+2} & \cdots & z_{i+L-1} \\ z_{i+1} & z_{i+2} & z_{i+3} & \cdots & z_{i+L} \\ \vdots & \vdots & \ddots & & \vdots \\ z_{i+L-2} & \cdots & \cdots & & z_{i+2L-3} \\ z_{i+L-1} & \cdots & \cdots & \cdots & z_{i+2L-2} \end{pmatrix} \begin{pmatrix} c_L \\ c_{L-1} \\ \vdots \\ c_2 \\ c_1 \end{pmatrix} = \begin{pmatrix} z_{i+L} \\ z_{i+L+1} \\ \vdots \\ c_{i+2L-2} \\ z_{i+2L-1} \end{pmatrix}.$$

Régistres à décalage ou LSFR

• On demande à une suite pseudo-alétoire, périodique de période T, de paraître aussi aléatoire que possible ce qui se traduit par trois critères dits de Colomb introduits par celui-ci en 1982. Les LFSR les vérifient.

• L'algorithme de Berlekamp-Massey calcule en temps quadratique la complexité linéaire et un polynôme générateur d'une suite finie. Cet algorithme repose sur l'observation élémentaire qu'une suite récurrente linéaire de complexité L est déterminée par $2L$ de ses termes consécutifs.

$$\begin{pmatrix} z_i & z_{i+1} & z_{i+2} & \cdots & z_{i+L-1} \\ z_{i+1} & z_{i+2} & z_{i+3} & \cdots & z_{i+L} \\ \vdots & \vdots & \ddots & & \vdots \\ z_{i+L-2} & \cdots & \cdots & & z_{i+2L-3} \\ z_{i+L-1} & \cdots & \cdots & \cdots & z_{i+2L-2} \end{pmatrix} \begin{pmatrix} c_L \\ c_{L-1} \\ \vdots \\ c_2 \\ c_1 \end{pmatrix} = \begin{pmatrix} z_{i+L} \\ z_{i+L+1} \\ \vdots \\ c_{i+2L-2} \\ z_{i+2L-1} \end{pmatrix}.$$

Régistres à décalage ou LSFR

• Même si le polynôme de rétroaction du registre est bien choisi, la complexité linéaire de la suite produit reste généralement trop faible pour se prémunir d'un attaque par l'algorithme de Berlekamp-Massey.

• Afin de s'en prémunir et d'augmenter la taille de l'espace des clefs, on utilise plusieurs LFSR en parallèle et on combine leurs sorties par une fonction booléenne

$$f : \mathrm{F}_2^m \longrightarrow \mathrm{F}_2$$

non linéaire mais équilibrée, i.e. prenant autant de fois la valeur 1 que 0.

Régistres à décalage ou LSFR

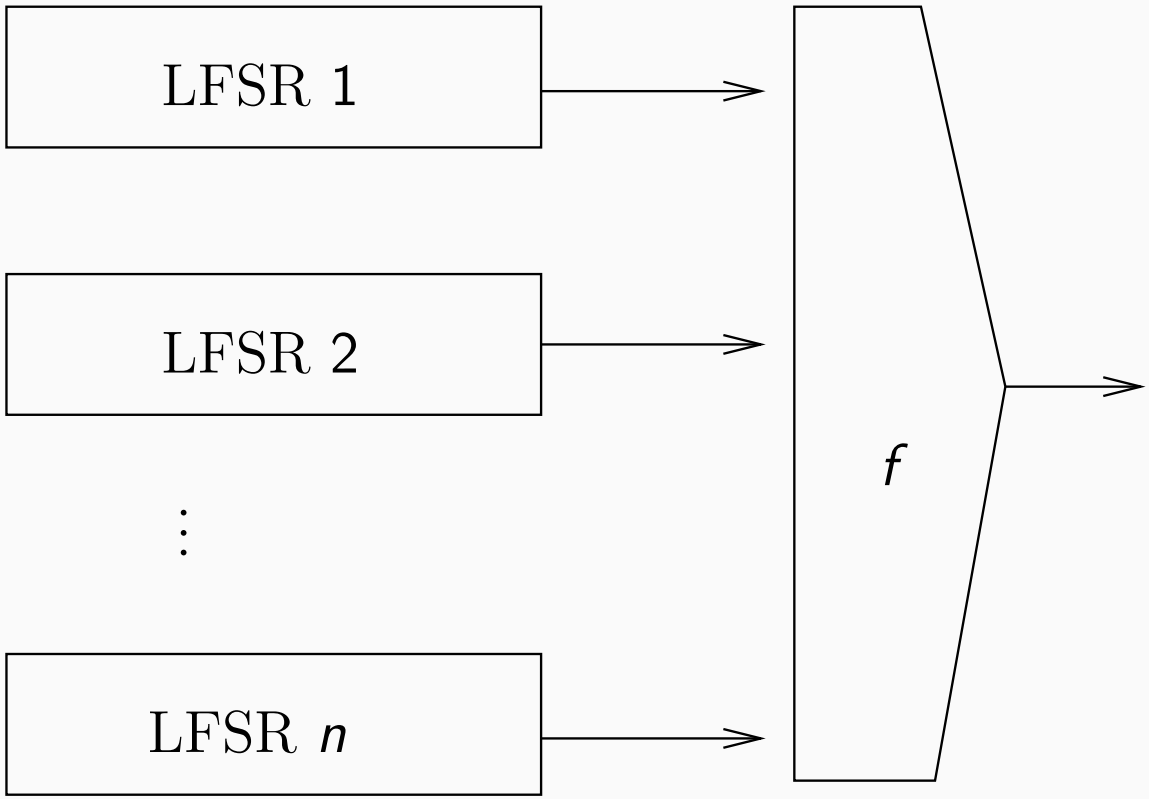
• Même si le polynôme de rétroaction du registre est bien choisi, la complexité linéaire de la suite produit reste généralement trop faible pour se prémunir d'un attaque par l'algorithme de Berlekamp-Massey.
• Afin de s'en prémunir et d'augmenter la taille de l'espace des clefs, on utilise plusieurs LFSR en parallèle et on combine leurs sorties par une fonction booléenne

$$f : \mathrm{F}_2^m \longrightarrow \mathrm{F}_2$$

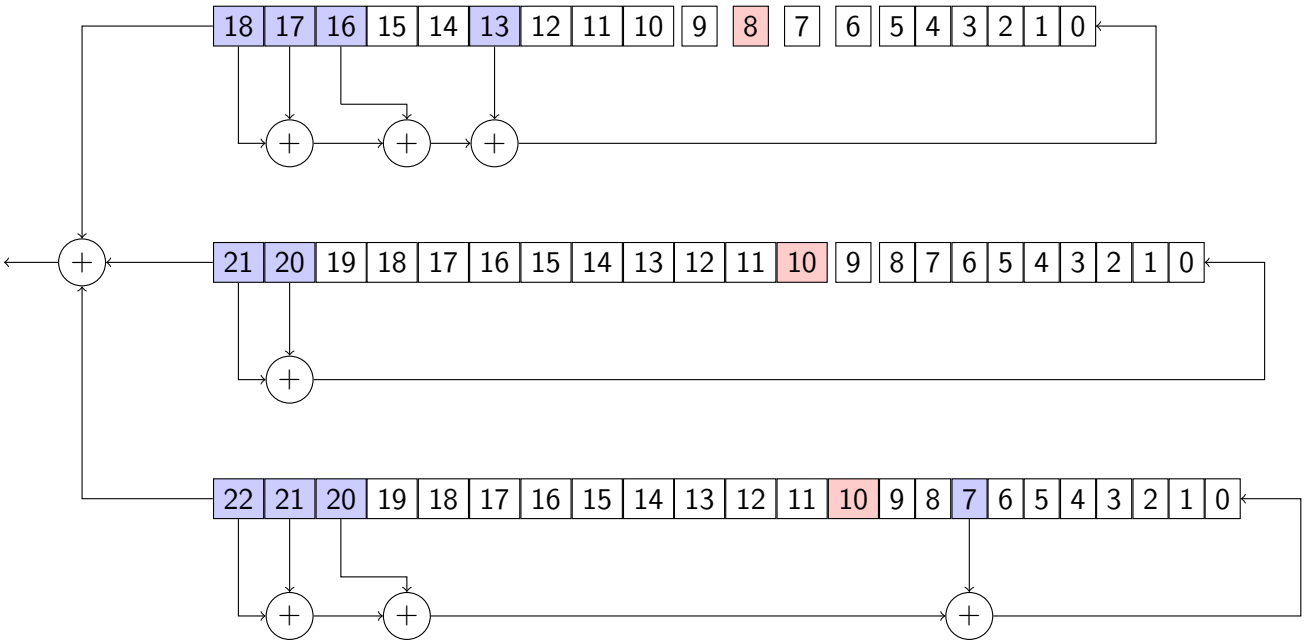
non linéaire mais équilibrée, i.e. prenant autant de fois la valeur 1 que 0.

Régistres à décalage ou LSFR

Diagramme d'un NLCG, Non Linear Combining Generator



A5/1 implémenté dans GSM



A chaque tour d'horloge, les registres sont shiftés suivant la valeur des bits rouges selon la règle de majorité.

RC4

Inventé en 1984 par Ronald Rivest, initialement secret puis dévoilé en 1994, est composé de deux algorithmes:

- KSA qui initialise une permutation S de $\{0, 1, \cdots, 255\}$;
- PRGA qui génère une suite aléatoire d'octets: on chiffre m_i par $c_i = m_i \oplus S[j_i]$.

Remarque: c'est un algorithme proche d'Enigma, un chiffrement par une substitution modifiée à chaque nouveau caractère.
Pour une clef $K = [IV \mid clef]$ de longueur L, on construit la permutation S comme suit: partant de $S = Id$ et $j = 0$, pour i variant de 0 à 255 on échange $S[i]$ et $S[j]$ où

$$j \leftarrow j + S[i] + K[i \mod L] \mod 256.$$

RC4

Inventé en 1984 par Ronald Rivest, initialement secret puis dévoilé en 1994, est composé de deux algorithmes:

- KSA qui initialise une permutation S de $\{0, 1, \cdots, 255\}$;
- PRGA qui génère une suite aléatoire d'octets: on chiffre m_i par $c_i = m_i \oplus S[j_i]$.

Remarque: c'est un algorithme proche d'Enigma, un chiffrement par une substitution modifiée à chaque nouveau caractère.
Pour une clef $K = [IV \mid clef]$ de longueur L, on construit la permutation S comme suit: partant de $S = Id$ et $j = 0$, pour i variant de 0 à 255 on échange $S[i]$ et $S[j]$ où

$$j \leftarrow j + S[i] + K[i \mod L] \mod 256.$$

RC4

Inventé en 1984 par Ronald Rivest, initialement secret puis dévoilé en 1994, est composé de deux algorithmes:

- KSA qui initialise une permutation S de $\{0, 1, \cdots, 255\}$;
- PRGA qui génère une suite aléatoire d'octets: on chiffre m_i par $c_i = m_i \oplus S[j_i]$.

Remarque: c'est un algorithme proche d'Enigma, un chiffrement par une substitution modifiée à chaque nouveau caractère.
Pour une clef $K = [IV \mid clef]$ de longueur L, on construit la permutation S comme suit: partant de $S = Id$ et $j = 0$, pour i variant de 0 à 255 on échange $S[i]$ et $S[j]$ où

$$j \leftarrow j + S[i] + K[i \mod L] \mod 256.$$

RC4

Inventé en 1984 par Ronald Rivest, initialement secret puis dévoilé en 1994, est composé de deux algorithmes:

- KSA qui initialise une permutation S de $\{0, 1, \cdots, 255\}$;
- PRGA qui génère une suite aléatoire d'octets: on chiffre m_i par $c_i = m_i \oplus S[j_i]$.

Remarque: c'est un algorithme proche d'Enigma, un chiffrement par une substitution modifiée à chaque nouveau caractère.

Pour une clef $K = [IV \mid clef]$ de longueur L, on construit la permutation S comme suit: partant de $S = Id$ et $j = 0$, pour i variant de 0 à 255 on échange $S[i]$ et $S[j]$ où

$$j \leftarrow j + S[i] + K[i \mod L] \mod 256.$$

RC4

Inventé en 1984 par Ronald Rivest, initialement secret puis dévoilé en 1994, est composé de deux algorithmes:

- KSA qui initialise une permutation S de $\{0, 1, \cdots, 255\}$;
- PRGA qui génère une suite aléatoire d'octets: on chiffre m_i par $c_i = m_i \oplus S[j_i]$.

Remarque: c'est un algorithme proche d'Enigma, un chiffrement par une substitution modifiée à chaque nouveau caractère.
Pour une clef $K = [IV \mid clef]$ de longueur L, on construit la permutation S comme suit: partant de $S = Id$ et $j = 0$, pour i variant de 0 à 255 on échange $S[i]$ et $S[j]$ où

$$j \leftarrow j + S[i] + K[i \mod L] \mod 256.$$

KSA: un exemple d'exécution

Par exemple pour $N = 8$ au lieu de 256 et $K = [4, 6, 2, 4, 6, 3, 3, 7]$ et donc $L = 8$ on a:

i	0	1	2	3	4	5	6	7	$j \leftarrow j + S[j] + K[i] \mod 8$	échange
init	0	1	2	3	4	5	6	7	0	
0	4	1	2	3	0	5	6	7	$0+0+4=4 \mod 8$	$S[0] \leftrightarrow S[4]$
1	4	2	1	3	0	5	6	7	$4+0+6=2 \mod 8$	$S[1] \leftrightarrow S[2]$
2	4	2	5	3	0	1	6	7	$2+1+2=5 \mod 8$	$S[2] \leftrightarrow S[5]$
3	3	2	5	4	0	1	6	7	$5+1+4=1 \mod 8$	$S[3] \leftrightarrow S[1]$
4	0	2	5	4	3	1	6	7	$1+2+6=1 \mod 8$	$S[4] \leftrightarrow S[1]$
5	0	2	5	4	3	6	1	7	$1+2+4=6 \mod 8$	$S[6] \leftrightarrow S[5]$
6	0	2	1	4	3	6	5	7	$6+1+3=2 \mod 8$	$S[6] \leftrightarrow S[2]$
7	0	2	1	4	3	6	7	5	$2+5+7=6 \mod 8$	$S[7] \leftrightarrow S[6]$

RC4-PRGA

Considérons le message $M = [100, 101, \cdots]$, l'algorithme PRGA tourne alors comme suit:

- Initialisation: $i = 0$, $j = 0$:

k	0	1	2	3	4	5	6	7
$S[k]$	0	2	5	4	3	6	7	1

- $i = 1$, $j = 0 + S[i] = 0 + 2$ et on échange $S[1] \leftrightarrow S[2]$:

k	0	1	2	3	4	5	6	7
$S[k]$	0	5	2	4	3	6	7	1

L'octet masque est donnée par la formule
$S[S[i] + S[j]] = S[5 + 2] = S[7] = 1$ et on chiffre le premier bloc de M $C_1 = [100] \oplus [001] = 101$.

- $i = 2$, $j = 2 + S[2] = 2 + 5 = 7$, on échange $S[2] \leftrightarrow S[7]$:

k	0	1	2	3	4	5	6	7
$S[k]$	0	5	1	4	3	6	7	2

L'octet masque est $S[1 + 2] = 4 = 100$ et $C_2 = [101] \oplus [100] = 001$.

RC4-PRGA

Considérons le message $M = [100, 101, \cdots]$, l'algorithme PRGA tourne alors comme suit:

- Initialisation: $i = 0$, $j = 0$:

k	0	1	2	3	4	5	6	7
$S[k]$	0	2	5	4	3	6	7	1

- $i = 1$, $j = 0 + S[i] = 0 + 2$ et on échange $S[1] \leftrightarrow S[2]$:

k	0	1	2	3	4	5	6	7
$S[k]$	0	5	2	4	3	6	7	1

L'octet masque est donnée par la formule
$S[S[i] + S[j]] = S[5 + 2] = S[7] = 1$ et on chiffre le premier bloc de M $C_1 = [100] \oplus [001] = 101$.

- $i = 2$, $j = 2 + S[2] = 2 + 5 = 7$, on échange $S[2] \leftrightarrow S[7]$:

k	0	1	2	3	4	5	6	7
$S[k]$	0	5	1	4	3	6	7	2

L'octet masque est $S[1 + 2] = 4 = 100$ et $C_2 = [101] \oplus [100] = 001$.

RC4-PRGA

Considérons le message $M = [100, 101, \cdots]$, l'algorithme PRGA tourne alors comme suit:

- Initialisation: $i = 0$, $j = 0$:

k	0	1	2	3	4	5	6	7
$S[k]$	0	2	5	4	3	6	7	1

- $i = 1$, $j = 0 + S[i] = 0 + 2$ et on échange $S[1] \leftrightarrow S[2]$:

k	0	1	2	3	4	5	6	7
$S[k]$	0	5	2	4	3	6	7	1

L'octet masque est donnée par la formule
$S[S[i] + S[j]] = S[5 + 2] = S[7] = 1$ et on chiffre le premier bloc de M $C_1 = [100] \oplus [001] = 101$.

- $i = 2$, $j = 2 + S[2] = 2 + 5 = 7$, on échange $S[2] \leftrightarrow S[7]$:

k	0	1	2	3	4	5	6	7
$S[k]$	0	5	1	4	3	6	7	2

L'octet masque est $S[1 + 2] = 4 = 100$ et $C_2 = [101] \oplus [100] = 001$.

RC4-PRGA

Considérons le message $M = [100, 101, \cdots]$, l'algorithme PRGA tourne alors comme suit:

- Initialisation: $i = 0$, $j = 0$:

k	0	1	2	3	4	5	6	7
$S[k]$	0	2	5	4	3	6	7	1

- $i = 1$, $j = 0 + S[i] = 0 + 2$ et on échange $S[1] \leftrightarrow S[2]$:

k	0	1	2	3	4	5	6	7
$S[k]$	0	5	2	4	3	6	7	1

L'octet masque est donnée par la formule
$S[S[i] + S[j]] = S[5 + 2] = S[7] = 1$ et on chiffre le premier bloc de M $C_1 = [100] \oplus [001] = 101$.

- $i = 2$, $j = 2 + S[2] = 2 + 5 = 7$, on échange $S[2] \leftrightarrow S[7]$:

k	0	1	2	3	4	5	6	7
$S[k]$	0	5	1	4	3	6	7	2

L'octet masque est $S[1 + 2] = 4 = 100$ et $C_2 = [101] \oplus [100] = 001$.

WIFI: protection par WEP

• L'utilisateur choisi une clef K de 40 bits (plus récemment 104 bits) qu'il concatène avec IV de 24 bits pour obtenir $(IV|K)$.

• Il crypte alors M en $C = (M|CRC(M)) \oplus RC4(IV|K)$ et envoie $(IV|C)$.

• Pour le déchiffrement, on récupère en clair IV, puis on forme $(IV|K)$ pour calculer $RC4(IV|K)$ que l'on ajoute à C pour obtenir $(M|CRC(M))$.

• Pour l'authentification, le serveur choisit un aléa R de 128 bits, l'envoi à Alice ave l'IV; Alice renvoie $r' = r \oplus RC4(IV|K)$ et le serveur vérifie si $r' \oplus RC4(IV|K) = r$.

WIFI: protection par WEP

- L'utilisateur choisi une clef K de 40 bits (plus récemment 104 bits) qu'il concatène avec IV de 24 bits pour obtenir $(IV|K)$.
- Il crypte alors M en $C = (M|CRC(M)) \oplus RC4(IV|K)$ et envoie $(IV|C)$.
- Pour le déchiffrement, on récupère en clair IV, puis on forme $(IV|K)$ pour calculer $RC4(IV|K)$ que l'on ajoute à C pour obtenir $(M|CRC(M))$.
- Pour l'authentification, le serveur choisit un aléa R de 128 bits, l'envoi à Alice ave l'IV; Alice renvoie $r' = r \oplus RC4(IV|K)$ et le serveur vérifie si $r' \oplus RC4(IV|K) = r$.

WIFI: protection par WEP

• L'utilisateur choisi une clef K de 40 bits (plus récemment 104 bits) qu'il concatène avec IV de 24 bits pour obtenir $(IV|K)$.
• Il crypte alors M en $C = (M|CRC(M)) \oplus RC4(IV|K)$ et envoie $(IV|C)$.
• Pour le déchiffrement, on récupère en clair IV, puis on forme $(IV|K)$ pour calculer $RC4(IV|K)$ que l'on ajoute à C pour obtenir $(M|CRC(M))$.
• Pour l'authentification, le serveur choisit un aléa R de 128 bits, l'envoi à Alice ave l'IV; Alice renvoie $r' = r \oplus RC4(IV|K)$ et le serveur vérifie si $r' \oplus RC4(IV|K) = r$.

WIFI: protection par WEP

• L'utilisateur choisi une clef K de 40 bits (plus récemment 104 bits) qu'il concatène avec IV de 24 bits pour obtenir $(IV|K)$.
• Il crypte alors M en $C = (M|CRC(M)) \oplus RC4(IV|K)$ et envoie $(IV|C)$.
• Pour le déchiffrement, on récupère en clair IV, puis on forme $(IV|K)$ pour calculer $RC4(IV|K)$ que l'on ajoute à C pour obtenir $(M|CRC(M))$.
• Pour l'authentification, le serveur choisit un aléa R de 128 bits, l'envoi à Alice ave l'IV; Alice renvoie $r' = r \oplus RC4(IV|K)$ et le serveur vérifie si $r' \oplus RC4(IV|K) = r$.

Faiblesse du WEP

- La clé K est statique ce qui n'est pas bon; comme en plus il y a très peu d'IV (16.777.216), une attaque par force brute est possible.
- 802.11 ne rend même pas obligatoire l'incrémentation de l'IV!
- Le CRC utilisé est linéaire!
- Martin sniffant le réseau peut récupérer $(M|CRC(M))$ et C et donc $RC4(IV|K) = (M|CRC(M)) \oplus C$. Lors d'un challenge avec IV, Martin n'a qu'à renvoyer $C' = RC4(IV|K) \oplus (M'|CRC(M'))$ ce qui ne lui coûte aucun calcul.
- Dès 2005, il ne faut pas plus d'une minute pour casser une clé WEP.

Faiblesse du WEP

- La clé K est statique ce qui n'est pas bon; comme en plus il y a très peu d'IV (16.777.216), une attaque par force brute est possible.
- 802.11 ne rend même pas obligatoire l'incrémentation de l'IV!
- Le CRC utilisé est linéaire!
- Martin sniffant le réseau peut récupérer $(M|CRC(M))$ et C et donc $RC4(IV|K) = (M|CRC(M)) \oplus C$. Lors d'un challenge avec IV, Martin n'a qu'à renvoyer $C' = RC4(IV|K) \oplus (M'|CRC(M'))$ ce qui ne lui coûte aucun calcul.
- Dès 2005, il ne faut pas plus d'une minute pour casser une clé WEP.

Faiblesse du WEP

- La clé K est statique ce qui n'est pas bon; comme en plus il y a très peu d'IV (16.777.216), une attaque par force brute est possible.
- 802.11 ne rend même pas obligatoire l'incrémentation de l'IV!
- Le CRC utilisé est linéaire!
- Martin sniffant le réseau peut récupérer $(M|CRC(M))$ et C et donc $RC4(IV|K) = (M|CRC(M)) \oplus C$. Lors d'un challenge avec IV, Martin n'a qu'à renvoyer $C' = RC4(IV|K) \oplus (M'|CRC(M'))$ ce qui ne lui coûte aucun calcul.
- Dès 2005, il ne faut pas plus d'une minute pour casser une clé WEP.

Faiblesse du WEP

- La clé K est statique ce qui n'est pas bon; comme en plus il y a très peu d'IV (16.777.216), une attaque par force brute est possible.
- 802.11 ne rend même pas obligatoire l'incrémentation de l'IV!
- Le CRC utilisé est linéaire!
- Martin sniffant le réseau peut récupérer $(M|CRC(M))$ et C et donc $RC4(IV|K) = (M|CRC(M)) \oplus C$. Lors d'un challenge avec IV, Martin n'a qu'à renvoyer $C' = RC4(IV|K) \oplus (M'|CRC(M'))$ ce qui ne lui coûte aucun calcul.
- Dès 2005, il ne faut pas plus d'une minute pour casser une clé WEP.

Faiblesse du WEP

- La clé K est statique ce qui n'est pas bon; comme en plus il y a très peu d'IV (16.777.216), une attaque par force brute est possible.
- 802.11 ne rend même pas obligatoire l'incrémentation de l'IV!
- Le CRC utilisé est linéaire!
- Martin sniffant le réseau peut récupérer $(M|CRC(M))$ et C et donc $RC4(IV|K) = (M|CRC(M)) \oplus C$. Lors d'un challenge avec IV, Martin n'a qu'à renvoyer $C' = RC4(IV|K) \oplus (M'|CRC(M'))$ ce qui ne lui coûte aucun calcul.
- Dès 2005, il ne faut pas plus d'une minute pour casser une clé WEP.

WPA: 2002

- WPA1: utilise toujours RC4 mais corrige les défauts du WEP:
  - Utilisation de clef de session de 128 bits.
  - Authentification par hachage.
  - Impossibilité de réutiliser un même IV avec la même clé.
  - on remplace le CRC par une fonction de hachage basée sur SHA-1.
- WPA2=WPA1 + AES.

WPA: 2002

- WPA1: utilise toujours RC4 mais corrige les défauts du WEP:
  - Utilisation de clef de session de 128 bits.
  - Authentification par hachage.
  - Impossibilité de réutiliser un même IV avec la même clé.
  - on remplace le CRC par une fonction de hachage basée sur SHA-1.
- WPA2=WPA1 + AES.

codes symétriques commerciaux modernes

Type DES, Data Encrytion Standard

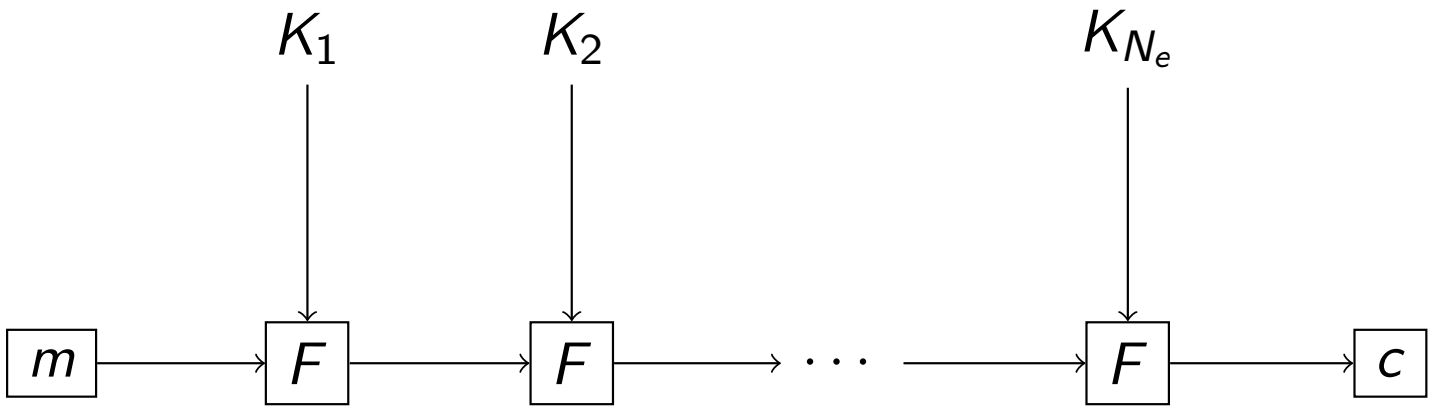
mis au point dans les années 1970 par IBM avec l'aide de la NSA (National Security Agency), ce sont des **hybrides de codes de substitutions et de codes de transpositions**.

Ils restent très sûrs avec des **clés assez courtes de 128 bits à 256 bits**. Leur sûreté est **non prouvée**. Leur cryptanalyse a fait des progrès (**cryptanalyse linéaire et différentielle**), ce qui permet de mieux cerner leur sureté.

Ils ont du mal à réaliser le partage des clefs, la signature, l'authentification, et la non répudiation des messages transmis. Ils s'adjoignent parfois un code asymétrique pour cela, voir le cryptosystème **PGP**.

Réseaux de substitution-permutation

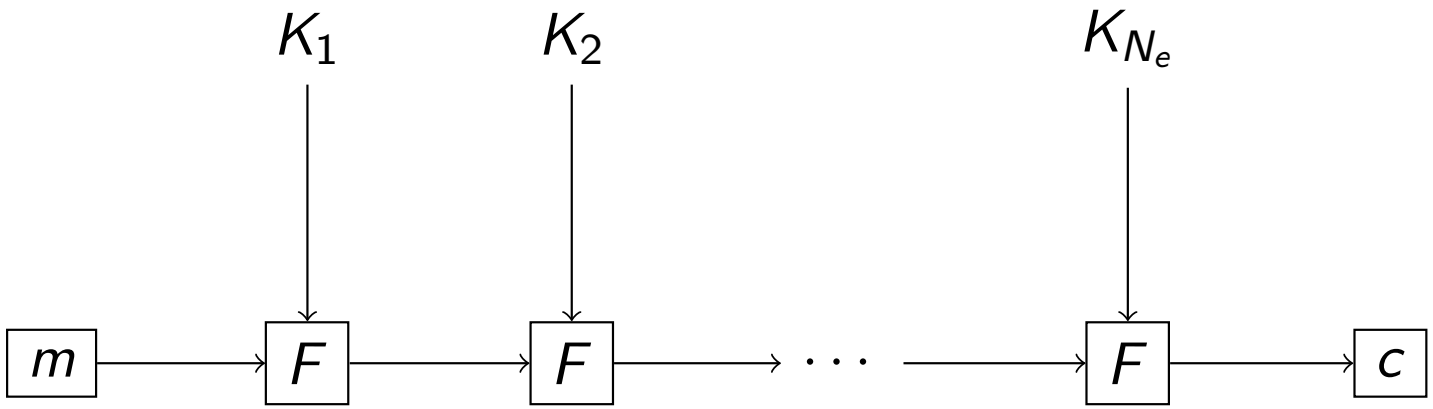
Type particulier d'algorithme de chiffrement itéré; modélise bien les deux exemples de chiffrement itérés que nous décrirons (DES et AES). C'est Algorithme de chiffrement **itératif** par blocs. Il répète N_e fois un algorithme de chiffrement, **fonction d'étage**. On introduit une clef secrète K à partir de laquelle on construit $N_e + 1$ sous-clefs, K^1, K^2,..., K^{N_e+1}, **clefs d'étages**.



Les fonctions de chiffrement associées aux clefs de chiffrement doivent être inversibles, afin de pouvoir déchiffrer. En outre elles doivent répondre aux deux principes fondamentaux introduits par Shannon.

Réseaux de substitution-permutation

Type particulier d'algorithme de chiffrement itéré; modélise bien les deux exemples de chiffrement itérés que nous décrirons (DES et AES). C'est Algorithme de chiffrement **itératif** par blocs. Il répète N_e fois un algorithme de chiffrement, **fonction d'étage**. On introduit une clef secrète K à partir de laquelle on construit $N_e + 1$ sous-clefs, K^1, K^2,..., K^{N_e+1}, **clefs d'étages**.



Les fonctions de chiffrement associées aux clefs de chiffrement doivent être inversibles, afin de pouvoir déchiffrer. En outre elles doivent répondre aux deux principes fondamentaux introduits par Shannon.

Confusion et diffusion

- **Confusion**: il s'agit de rendre la relation entre la clé de chiffrement et le texte chiffré le plus complexe possible: par exemple deux lettres doublées ne doivent pas rester côte à côte. Elle est assurée par l'introduction de **substitutions non-linéaires**.
- **Diffusion**: son but est de faire disparaître tout biais statistique dans le texte clair. L'idée est de permettre à chaque bit du texte en clair d'avoir une influence sur une grande partie du texte chiffré. Elle est assurée par des **permutations linéaires**.

Mise en oeuvre

On se donne deux entiers ℓ et m. Un texte clair et un texte chiffré seront des vecteurs de $\mathrm{F}_2^{\ell m}$ donc des vecteurs de longueur $\ell.m$ constitués de 0 et et 1.

- **La substitution appelée S-boîte** est notée

$$\pi_S : \{0,1\}^\ell \longrightarrow \{0,1\}^\ell.$$

  C'est la seule transformation non linéaire, chargée de la confusion: pour que l'implémentation n'utilise pas trop de mémoire, l doit être petit.

- **La permutation** pour la diffusion est notée

$$\pi_P : \{1,2,\ldots,\ell m\} \longrightarrow \{1,2,\ldots,\ell m\}$$

Mise en oeuvre

On se donne deux entiers ℓ et m. Un texte clair et un texte chiffré seront des vecteurs de $\mathrm{F}_2^{\ell m}$ donc des vecteurs de longueur $\ell.m$ constitués de 0 et et 1.

- **La substitution appelée S-boîte** est notée

$$\pi_S : \{0,1\}^\ell \longrightarrow \{0,1\}^\ell.$$

  C'est la seule transformation non linéaire, chargée de la confusion: pour que l'implémentation n'utilise pas trop de mémoire, l doit être petit.

- **La permutation** pour la diffusion est notée

$$\pi_P : \{1,2,\ldots,\ell m\} \longrightarrow \{1,2,\ldots,\ell m\}$$

Mise en oeuvre

Une chaîne binaire $x = (x_1, x_2, \ldots, x_{\ell m}) \in \{0,1\}^{\ell m}$ peut être considérée comme la concaténation de m sous-chaînes de longueur ℓ

$$x = x_{(1)}||x_{(2)}||\ldots||x_{(m)}$$

et pour $1 \leq i \leq m$ on a

$$x_{(i)} = (x_{(i-1)\ell+1}, \ldots, x_{i\ell}).$$

Schéma de Feistel

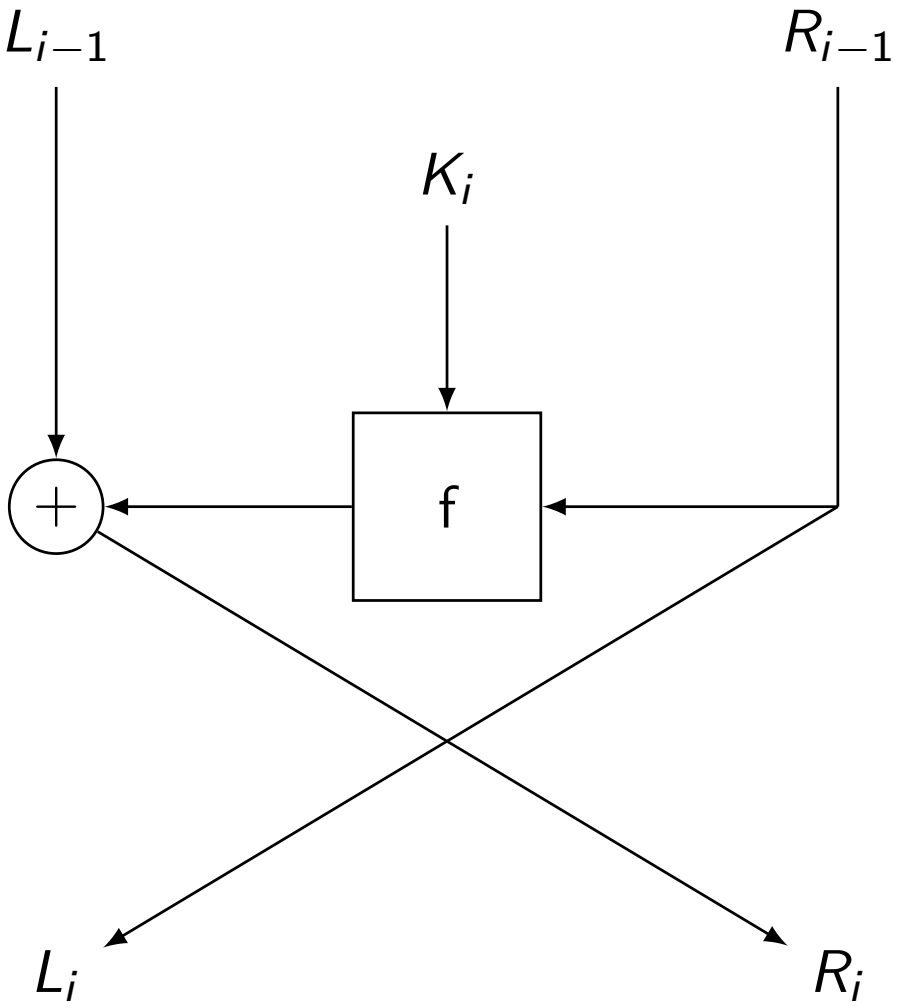
Définition

Un chiffrement de **Feistel** *est un chiffrement itératif par blocs opérant sur des blocs de* $2n$ *bits, la fonction itérée*

$$\begin{array}{rcl} F : \mathrm{F}_2^n \times \mathrm{F}_2^n & \longrightarrow & \mathrm{F}_2^n \times \mathrm{F}_2^n \\ (L_{i-1}, R_{i-1}) & \mapsto & (L_i, R_i) \end{array}$$

avec $L_i = R_{i-1}$ *et* $R_i = L_{i-1} + f(R_{i-1}, K_i)$.

Schéma de Feistel



Description de DES

• **système cryptographique produit**. Il répéte 16 fois un algorithme qui mélange les bits en respectant les principes de C. Shannon: **confusion et diffusion**. C'est un schéma de Feistel.

• DES utilise une clé K de 56 bits, en fait une clé de 64 bits dont les bits 8, 16, 24, 32, 40, 48, 56 ne sont pas utilisés pour la clé.

• Il chiffre par blocs de 64 bits.

Description de DES

• **système cryptographique produit**. Il répéte 16 fois un algorithme qui mélange les bits en respectant les principes de C. Shannon: **confusion et diffusion**. C'est un schéma de Feistel.

• DES utilise une clé **K** de 56 bits, en fait une clé de 64 bits dont les bits 8, 16, 24, 32, 40, 48, 56 ne sont pas utilisés pour la clé.

• Il chiffre par blocs de 64 bits.

Description de DES

1.- La clé **K** donne naissance à 16 sous-clés, les **clefs d'étage**, de 48 bits $\mathbf{K_1}$, $\mathbf{K_2}$,...,$\mathbf{K_{16}}$

2.- Etant donné un bloc de texte clair de 64 bits, **X**, on construit, grâce à la **fonction d'étage 0**, un nouveau texte, $\mathbf{X_0}$, par permutation de l'ordre des bits grâce à la permutation initiale **IP**.

3.- on sépare les 32 bits de gauche, $\mathbf{L_0}$, et les 32 bits de droites, $\mathbf{R_0}$, de $\mathbf{X_0}$: $\mathbf{X_0 = L_0 R_0}$

Description de DES

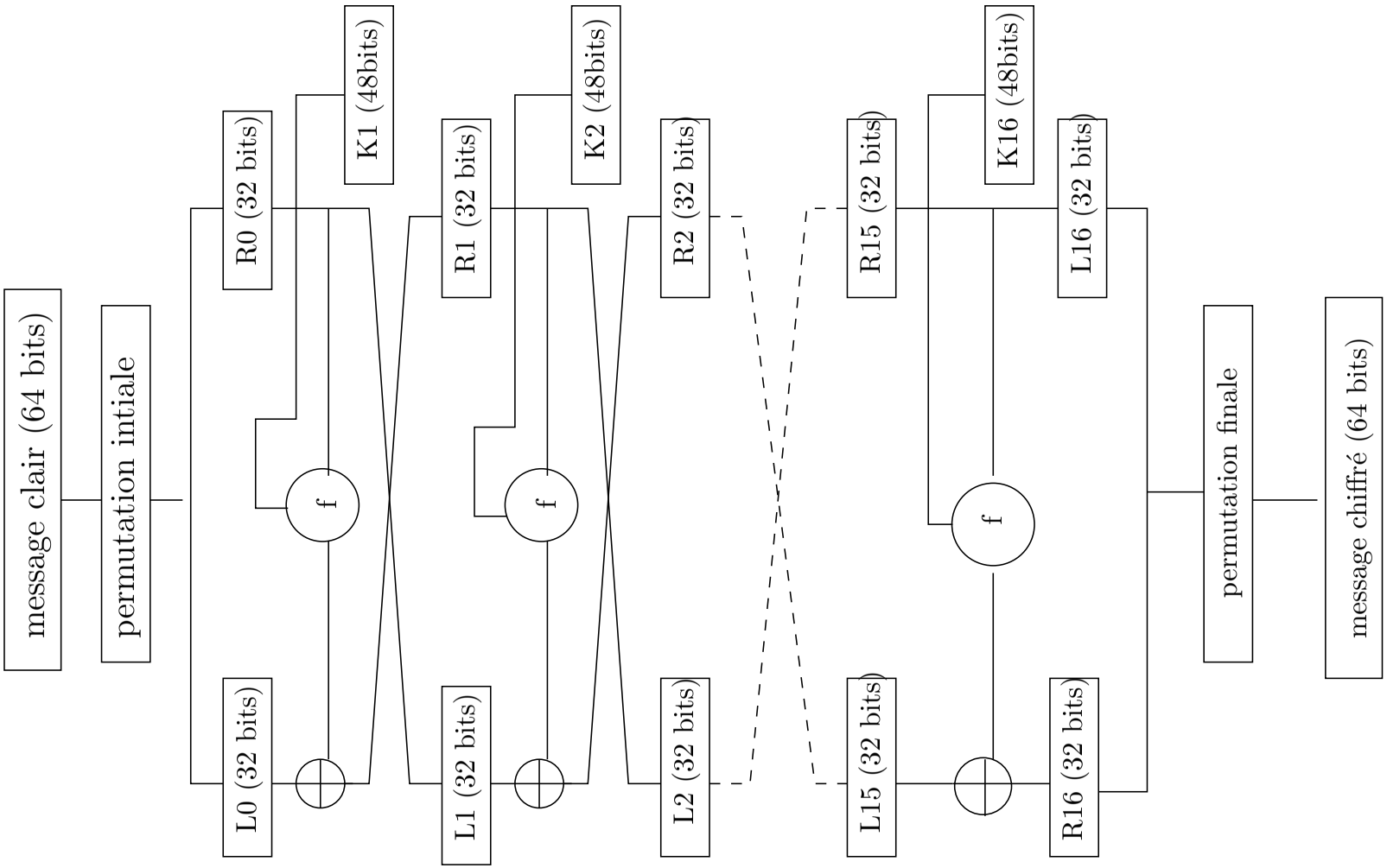
4.- 16 itérations (ou tours ou étages). On calcule la valeur de la fonction d'étage $\mathbf{g(X_{i-1}, K^i) = L_i R_i}$, $1 \leq i \leq 16$ suivant la règle

$$\mathbf{L_i = R_{i-1}}$$
$$\mathbf{R_i = L_{i-1} \oplus f(R_{i-1}, K_i)}$$

où $\oplus$ est la somme dans $\mathbb{Z}/2\mathbb{Z}$ de L_{i-1} et $f(R_{i-1}, K_i)$ et f une fonction **non linéaire** qui participe à la diffusion et qui est décrite par les **S-boites**

5.- Le permutation inverse de la permutation initiale, IP^{-1}, est appliquée à $\mathbf{R_{16} L_{16}}$

Architecture du DES



Description de DES

- **La fonction f**:

$$\begin{array}{rcl} f : \mathrm{F}_2^{32} \times \mathrm{F}_2^{48} & \longrightarrow & \mathrm{F}_2^{32} \\ (R_{i-1}, K_i) & \mapsto & f(R_{i-1} K_i) \end{array}$$

qui se compose:

- une augmentation E de R_{i-1} pour en faire un bloc de 48 octets: $E(R_{i-1})$ est composé de tous les bits de R_{i-1}, 16 d'entre eux apparaissant deux fois;
- on calcule $E(R_{i-1}) \oplus K_i$ et on le découpe en 8 sous-chaînes de 6 bits;
- chacune des sous-chaînes de 6 bits est transformée par une fonction **non linéaire** fixée en une sous-chaîne de 4 bits;
- les sous-chaînes de 4 bits sont réordonnées suivant une permutation fixée.

- Le DES a été choisi comme norme aux Etats Unis en 1975 et est devenu le système cryptographique le plus utilisé dans le monde.
- Le DES a été critiqué à cause de la taille trop faible de ses clés.
- En 1998, un défi a été résolu en quelques jours par une machine de 250.000 dollars spécialement construite pour retrouver la clé par une attaque exhaustive.
- Triple DES consiste à appliquer DES muni de la clef K, DES^{-1} muni de la clef K' et DES muni de la clef K.

- Le DES a été choisi comme norme aux Etats Unis en 1975 et est devenu le système cryptographique le plus utilisé dans le monde.
- Le DES a été critiqué à cause de la taille trop faible de ses clés.
- En 1998, un défi a été résolu en quelques jours par une machine de 250.000 dollars spécialement construite pour retrouver la clé par une attaque exhaustive.
- Triple DES consiste à appliquer DES muni de la clef K, DES^{-1} muni de la clef K' et DES muni de la clef K.

- Le DES a été choisi comme norme aux Etats Unis en 1975 et est devenu le système cryptographique le plus utilisé dans le monde.
- Le DES a été critiqué à cause de la taille trop faible de ses clés.
- En 1998, un défi a été résolu en quelques jours par une machine de 250.000 dollars spécialement construite pour retrouver la clé par une attaque exhaustive.
- Triple DES consiste à appliquer DES muni de la clef K, DES^{-1} muni de la clef K' et DES muni de la clef K.

- Le DES a été choisi comme norme aux Etats Unis en 1975 et est devenu le système cryptographique le plus utilisé dans le monde.
- Le DES a été critiqué à cause de la taille trop faible de ses clés.
- En 1998, un défi a été résolu en quelques jours par une machine de 250.000 dollars spécialement construite pour retrouver la clé par une attaque exhaustive.
- Triple DES consiste à appliquer DES muni de la clef K, DES^{-1} muni de la clef K' et DES muni de la clef K.

Description d'AES

Le principe de l'AES, est aussi un **système cryptographique produit** constitué d'une suite d'opérations de permutations et de substitutions. AES travaille sur des blocs de 128 bits. AES résiste à toutes les attaques connues. Ce n'est pas un schéma de Feistel.

Fait d'expérience pas une preuve

Système de chiffrement itéré constitué d'un algorithme de chiffrement, la **fonction d'étage**, répété de 10 à 14 fois et d'une clef d'une longueur de 128, 192 ou 256 bits. Chaque étape s'appelle un étage d'AES. Un **algorithme de diversification de clef**, **ExpandKey[i]**, crée une clef pour chacun des étages, i.

Description d'AES

Le principe de l'AES, est aussi un **système cryptographique produit** constitué d'une suite d'opérations de permutations et de substitutions. AES travaille sur des blocs de 128 bits. AES résiste à toutes les attaques connues. Ce n'est pas un schéma de Feistel.

Fait d'expérience pas une preuve

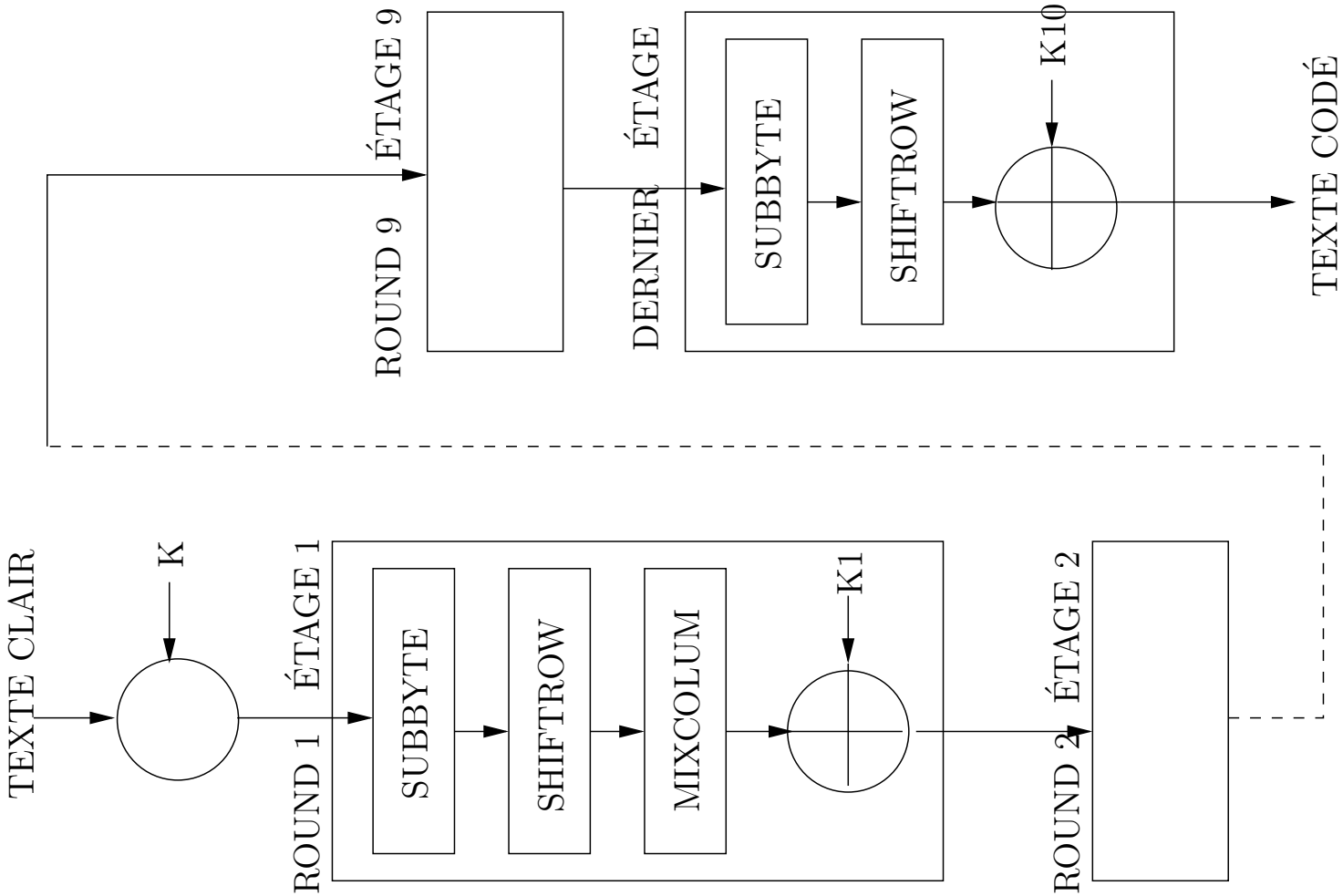
Système de chiffrement itéré constitué d'un algorithme de chiffrement, la **fonction d'étage**, répété de 10 à 14 fois et d'une clef d'une longueur de 128, 192 ou 256 bits. Chaque étape s'appelle un étage d'AES. Un **algorithme de diversification de clef**, **ExpandKey[i]**, crée une clef pour chacun des étages, i.

Description d'AES

Sécurité d'AES

Le passage à une clé de 128 bits minimum rend impossible dans le futur prévisible les recherches exhaustives de clefs. Si on suppose que l'on a un algorithme capable de comparer en une seconde 2^{56} clefs (i.e de casser DES en une seconde) il lui faudra 149 mille milliards d'année pour casser AES.

Diagramme du codage en AES



Description d'AES

On notera N_e le nombre d'étages. La fonction d'étage, g, est l'ensemble des opérations que l'on fait subir au texte qui arrive à l'étage i. Elle est la même pour tous les étages sauf le dernier dans un soucis de simplicité.

La fonction g consiste en l'application successive de 4 opérations **SubBytes**, **ShiftRows**, **MixColumns**, **AddRoundKey**.

- A partir de la clef secrète, **K**, on génére des sous clefs d'étage par l'**algorithme de diversification** de la clef, **ExpandKey[i]**. On obtient les sous-clefs $\mathbf{K}^1, \ldots, \mathbf{K}^{N_e}$.

On note x le texte initial (texte en clair) et w^i le texte utilisé en entrée à l'étage i et y sera le texte crypté final.

Description d'AES

On notera N_e le nombre d'étages. La fonction d'étage, g, est l'ensemble des opérations que l'on fait subir au texte qui arrive à l'étage i. Elle est la même pour tous les étages sauf le dernier dans un soucis de simplicité.

La fonction g consiste en l'application successive de 4 opérations **SubBytes**, **ShiftRows**, **MixColumns**, **AddRoundKey**.

- A partir de la clef secrète, $\mathbf{K}$, on génére des sous clefs d'étage par l'**algorithme de diversification** de la clef, **ExpandKey[i]**. On obtient les sous-clefs $\mathbf{K}^1, \ldots, \mathbf{K}^{N_e}$.

On note x le texte initial (texte en clair) et w^i le texte utilisé en entrée à l'étage i et y sera le texte crypté final.

Description d'AES

On notera N_e le nombre d'étages. La fonction d'étage, g, est l'ensemble des opérations que l'on fait subir au texte qui arrive à l'étage i. Elle est la même pour tous les étages sauf le dernier dans un soucis de simplicité.

La fonction g consiste en l'application successive de 4 opérations **SubBytes**, **ShiftRows**, **MixColumns**, **AddRoundKey**.

• A partir de la clef secrète, **K**, on génére des sous clefs d'étage par l'**algorithme de diversification** de la clef, **ExpandKey[i]**. On obtient les sous-clefs $\mathbf{K^1}, \ldots, \mathbf{K^{N_e}}$.

On note x le texte initial (texte en clair) et w^i le texte utilisé en entrée à l'étage i et y sera le texte crypté final.

Description d'AES

On notera N_e le nombre d'étages. La fonction d'étage, g, est l'ensemble des opérations que l'on fait subir au texte qui arrive à l'étage i. Elle est la même pour tous les étages sauf le dernier dans un soucis de simplicité.

La fonction g consiste en l'application successive de 4 opérations **SubBytes**, **ShiftRows**, **MixColumns**, **AddRoundKey**.

• A partir de la clef secrète, **K**, on génére des sous clefs d'étage par l'**algorithme de diversification** de la clef, **ExpandKey[i]**. On obtient les sous-clefs $\mathbf{K^1}, \ldots, \mathbf{K^{N_e}}$.

On note x le texte initial (texte en clair) et w^i le texte utilisé en entrée à l'étage i et y sera le texte crypté final.

Diagramme d'un tour d'AES

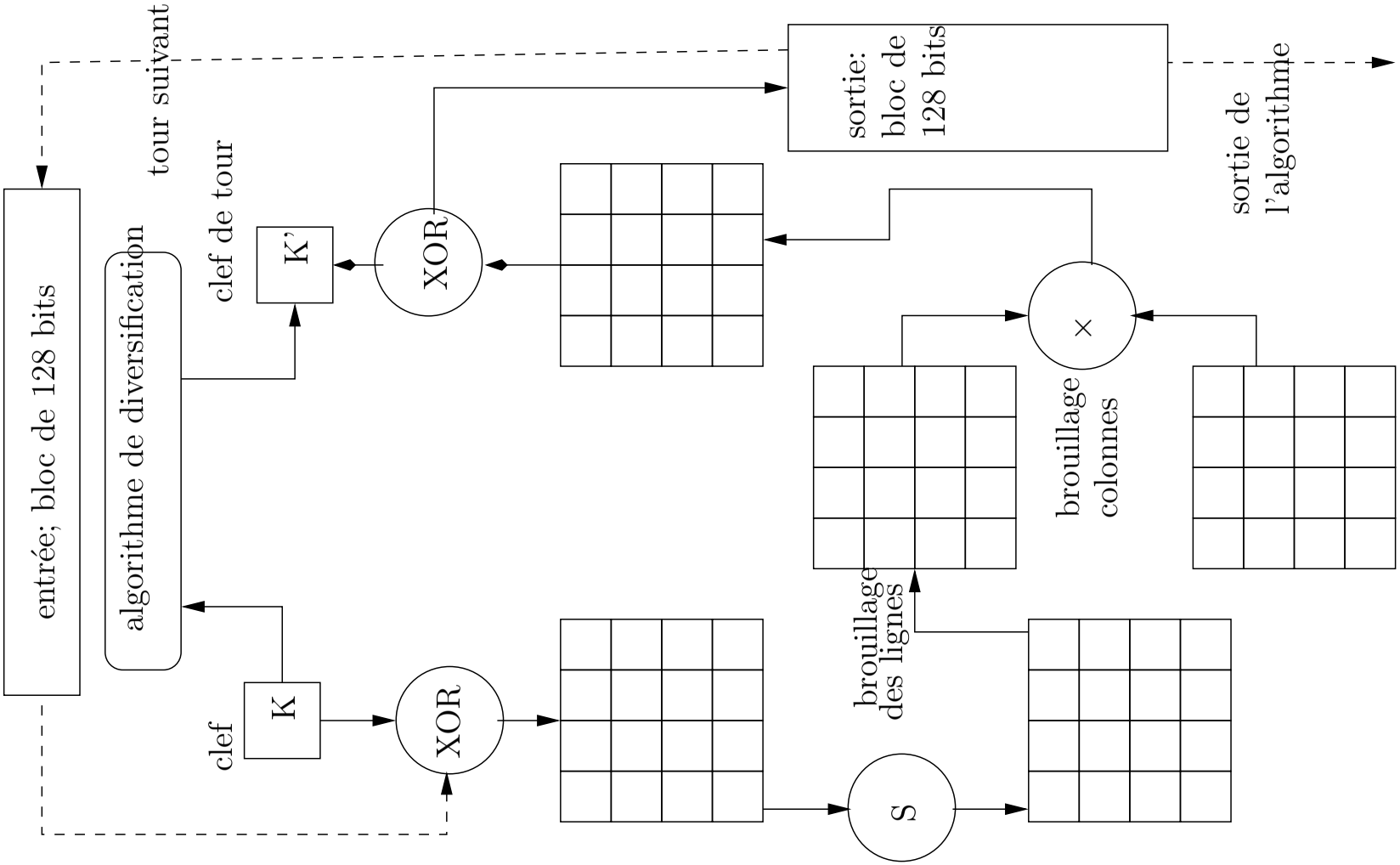
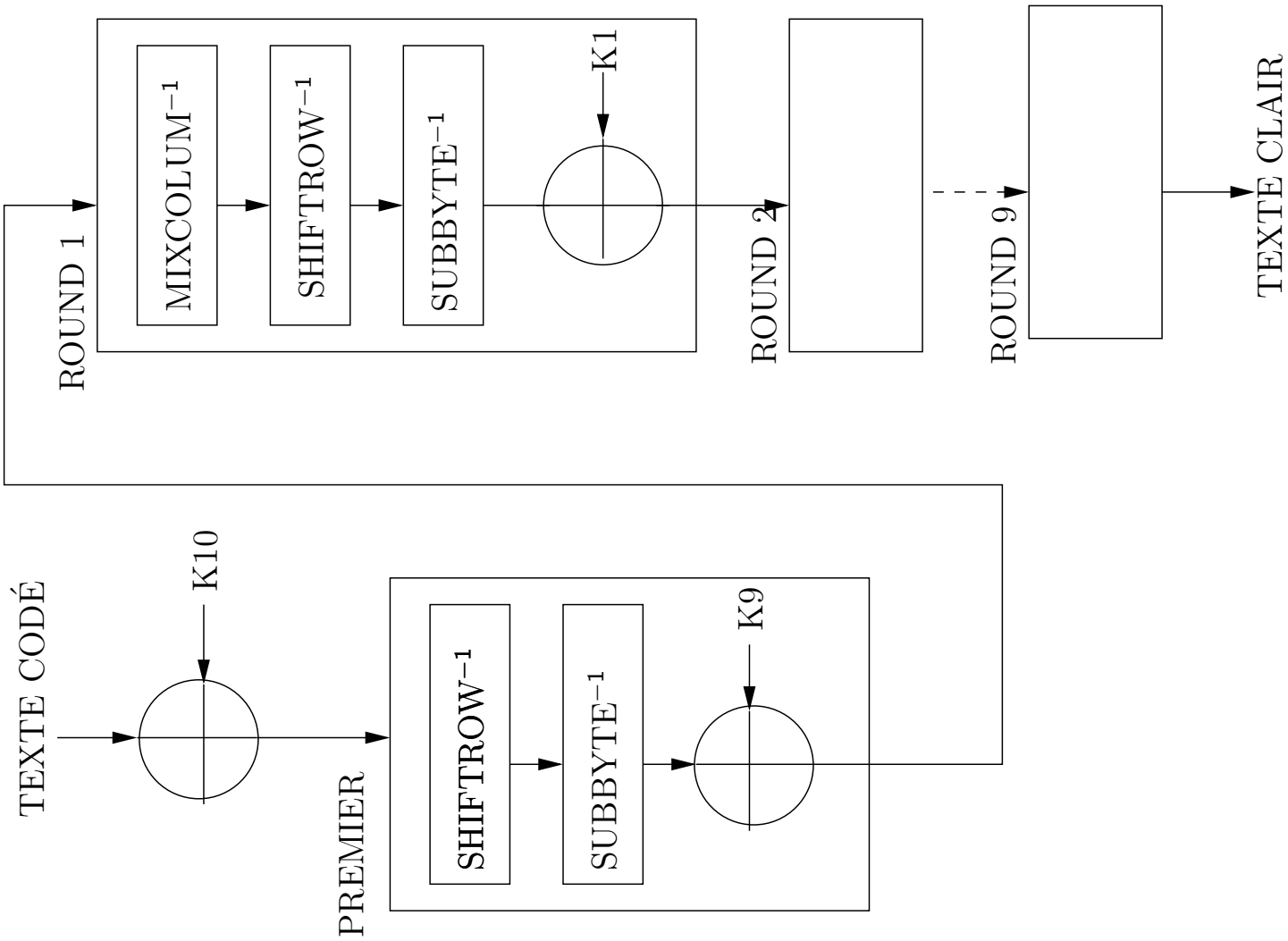


Diagramme du décodage en AES



Description schématique d'AES I

- On initialise w^0 à x et et on effectue l'opération **AddRoundKey**

$$W^0 \oplus K$$

- Pour chacun des $1 < i \leq N_e - 1$ étages suivants on effectue sur w^i les opérations suivantes

Description schématique d'AES II

1. l'opération de substitution **SubBytes**
2. sur le résultat de **SubBytes** une permutation notée **ShiftRows** (permutation circulaire sur les éléments des lignes de la matrice des octets $s_{i,j}$)
3. sur le résultat de **ShiftRows** une opération notée **MixColumns** (application linéaire sur l'espace des matrices sur F_{2^8})
4. Sur le résultat de **MixColumns** l'opération **AddRoundKey** avec la sous-clé de l'étage i

- Pour le dernier étage N_e on supprime l'opération **MiXColumns**

L'opération ExpandKey

L'opération **ExpandKey** dépend de la taille de clé choisie, $N_k \times 32$, (128, 160, 192, 224 ou 256 bits). On supposera $N_k = 6$.

Extension de la clef secrète K. On écrit la clef K sous forme d'une matrice de N_k colonne de 4 **octets** chacune dénotés k_0,...,k_5

k_0	k_1	k_2	k_3	k_4	k_5

L'opération ExpandKey

Ensuite cette matrice est étendue en une matrice de taille $4N_r + 1$ où N_r est le nombre d'étages par l'algorithme suivant

• Si i **n'est pas un multiple de N_k** (la longueur de la clef) alors la colonne d'indice i, k_i, est la **XORisation** de la colonne d'indice $i - N_k$, k_{i-N_k}, et de la colonne d'indice $i-1$, k_{i-1}. C'est donc l'**addition bit à bit sans retenue** de la colonne k_{i-N_k} et de la colonne k_{i-1}, $k_i = k_{i-N_k} \oplus k_{i-1}$.

L'opération ExpandKey

• Si i est un **multiple de N_k** on applique à chacun des bytes de la colonne k_{i-1} la permutation π_S décrite au paragraphe suivant suivie d'une rotation **Rotword** de la nouvelle colonne k_{i-1} et de l'addition d'une constante d'étage définie par **Fieldto Binary**$(x^{j-1}000000)$ pour l'étage j. On XOR le résultat obtenu avec la colonne k_{i-N_k}.
On obtient ainsi

k_0	k_1	k_2	k_3	k_4	k_5	...	$k_{i \cdot N_b}$	...	$k_{(i+1)N_b-1}$	...
clé d'étage 0						...	clé d'étage i			...

Exemple d'expansion de clé

$$
\begin{aligned}
\textbf{RoundKey}[00] &= 00000000000000000000000000000000 \\
\textbf{RoundKey}[01] &= 62636363626363636263636362636363 \\
\textbf{RoundKey}[02] &= 9b9898C9f9fbfbaa9b9898c9f9fbfbaa \\
\textbf{RoundKey}[03] &= 90973450696ccffaf2f457330b0fac99 \\
\textbf{RoundKey}[04] &= ee06da7b87a1581759e42b27e91ee2b \\
\textbf{RoundKey}[05] &= 7f2e2b88f8443e098dda7cbbf34b9290 \\
\textbf{RoundKey}[06] &= ec614b851425758c99ff09376ab49ba7 \\
\textbf{RoundKey}[07] &= 21751787355062bacaf6b3cc61bf09b \\
\textbf{RoundKey}[08] &= 0ef903333ba9613897060a04511dfa9f \\
\textbf{RoundKey}[09] &= b1d4d8e28a7db9da1d7bb3de4c664941 \\
\textbf{RoundKey}[10] &= b4ef5bcb3e92e21123e951cf6f8f188e
\end{aligned}
$$

L'opération SubBytes

L'opération **SubBytes** est la seule opération non linéaire. On suppose que le message w^i est un nombre de 128 bits=16 octets, $(s_{i,j})_{0\leq i,j\leq 3}$, écrit en base 2. On le réécrit sous la forme d'une matrice S_i, où les $s_{i,j}$ sont des octets.

$$S_i = \begin{matrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{matrix}$$

L'opération **SubBytes** consiste en une permutations π_S sur $\{0,1\}^8$, agissant indépendamment sur chacun des octets $s_{i,j}$ de l'étage i.

L'opération SubBytes

Pour décrire π_S on travaille dans un surcorps du corps à 2 éléments, F_2, le corps fini à 256 éléments, F_{2^8}, que l'on identifie avec les polynômes de degré < 8, munis de l'addition et de la multiplication modulo le polynôme irréductible de $\mathrm{F}[x]$, $x^8 + x^4 + x^3 + x + 1$

$$\mathrm{F}_{2^8} \simeq \mathrm{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)\mathrm{F}_2[x]$$

On associe à l'octet $a_7 a_6 \dots a_1 a_0$ l'élément

$$\mathbf{BinarytoField}(a_7 a_6 \dots a_1 a_0) = \sum_{i=0}^{7} a_i x^i$$

du corps F_{2^8} dont l'application inverse est **FieldtoBinary**. On a dans le corps F_{2^8} l'opération **FieldInv** qui à un élément $z \neq 0$ du corps associe z^{-1}.

L'opération SubBytes

Pour décrire π_S on travaille dans un surcorps du corps à 2 éléments, F_2, le corps fini à 256 éléments, F_{2^8}, que l'on identifie avec les polynômes de degré < 8, munis de l'addition et de la multiplication modulo le polynôme irréductible de $\mathrm{F}[x]$, $x^8 + x^4 + x^3 + x + 1$

$$\mathrm{F}_{2^8} \simeq \mathrm{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)\mathrm{F}_2[x]$$

On associe à l'octet $a_7 a_6 \dots a_1 a_0$ l'élément

$$\textbf{BinarytoField}(a_7 a_6 \dots a_1 a_0) = \sum_{i=0}^{7} a_i x^i$$

du corps F_{2^8} dont l'application inverse est **FieldtoBinary**. On a dans le corps F_{2^8} l'opération **FieldInv** qui à un élément $z \neq 0$ du corps associe z^{-1}.

L'opération SubBytes

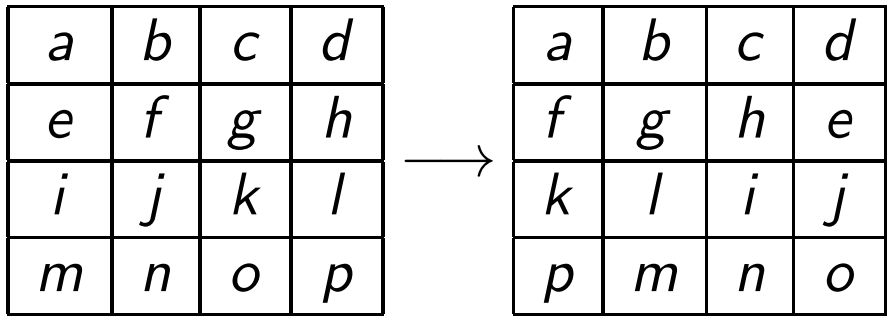
L'opération **SubBytes**$(a_7 a_6 \dots a_1 a_0) = (b_7 b_6 \dots b_1 b_0)$ s'écrit alors:

- application de **BinarytoField** à $(a_7 a_6 \dots a_1 a_0) = (b_7 b_6 \dots b_1 b_0)$ qui donne $a \in \mathrm{F}_{2^8}$
- application **FieldInv** à a qui donne a^{-1} si $a \neq 0$ ou 0 si $a = 0$
- ajout de l'élément $c = (c_7 c_6 \dots c_1 c_0) =$ **BinarytoField**(01100011) à a^{-1} pour obtenir $b = (b_7 b_6 \dots b_1 b_0)$
- application de **FieldtoBinary** à b

c est une constante d'AES.

L'opération ShiftRows

C'est une **permutation cyclique** dans les lignes de la matrice S_i



L'opération MixColumns

C'est une **transformation linéaire** dans les colonnes; la colonne j étant transformée de la manière suivante (les calculs sont faits dans F_{2^8})

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

L'opération MixColumns

Polynomialement cette opération s'interprète de la manière suivante Chacune des colonnes de la matrice **State** est considérée comme un polynôme de degré 3 à coefficients dans F_{256}.

L'opération **MixColumns** consiste alors à effectuer pour chaque colonne une multiplication du polynôme correspondant à la colonne par un polynôme fixe $c(x) = 03x^3 + x^2 + x + 02$ suivie d'une réduction modulo le polynôme $x^4 + 1$ de façon à obtenir un polyôme de degré inférieur ou égal à 3 de $\mathrm{F}_{256}[X]$ qui est interprété comme une colonne d'une matrice de $M_4(\mathrm{F}_{256})$.

L'opération AddRoundKey

AddRoundKey[i] est l'addition de la clef obtenue à l'étage i par l'algorithme de diversification des clés, **ExpandKey[i]**, au texte obtenu à l'issue de l'étape **MixColumns**.

On écrit la clef **ExpandKey[i]** de 128 bits sous la forme d'une matrice de 4×4 bytes et on l'ajoute au résultat de l'étape précédente par un XOR

L'opération AddRoundKey

$$\begin{array}{|c|c|c|c|} \hline a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ \hline a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ \hline a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ \hline a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \\ \hline \end{array} \oplus \begin{array}{|c|c|c|c|} \hline k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ \hline k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ \hline k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ \hline k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \\ \hline \end{array}$$

$$= \begin{array}{|c|c|c|c|} \hline b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ \hline b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ \hline b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ \hline b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \\ \hline \end{array}$$

Cryptanalyse linéaire

La cryptanalyse linéaire peut être appliquée contre tout algorithme de chiffrement itéré comme les réseaux de substitution-permutation.

- Elle consiste à **simplifier l'algorithme de chiffrement en faisant une approximation linéaire**.
- Attaque à texte clair connu: à l'aide d'un grand nombre de paires de textes clairs et du texte correspondant chiffré correspondant on améliore la précision de l'approximation.
- La cryptanalyse linéaire s'intéresse aux relations linéaires entre les bits au cours de l'algorithme.

Cryptanalyse linéaire

La cryptanalyse linéaire peut être appliquée contre tout algorithme de chiffrement itéré comme les réseaux de substitution-permutation.

• Elle consiste à **simplifier l'algorithme de chiffrement en faisant une approximation linéaire**.

• Attaque à texte clair connu: à l'aide d'un grand nombre de paires de textes clairs et du texte correspondant chiffré correspondant on améliore la précision de l'approximation.

• La cryptanalyse linéaire s'intéresse aux relations linéaires entre les bits au cours de l'algorithme.

Cryptanalyse linéaire

On suppose qu'il existe une F_2-combinaison linéaire des bits d'entrée et de sortie qui ait lieu avec une probabilité nettement supérieure ou nettement inférieure à $\frac{1}{2}$.
On suppose qu'il existe $i_1, i_2, \ldots, i_u$ et $j_1, j_2, \ldots, j_v$ tels que si $x = (X_1, X_2, \ldots X_N$ sont les bits d'entrée (du message en clair) considérés comme des variables aléatoires définies sur $\{0,1\}$ et $Y_1, Y_2, \ldots, Y_m$ sont les bits de la sortie (du message chiffré) considérés comme des variables aléatoires définies sur $\{0,1\}$ on ait

$$\mathrm{Prob}\{X_{i_1} \oplus X_{i_2} \oplus \cdots \oplus X_{i_n} \oplus Y_{j_1} \oplus Y_{j_2} \oplus \cdots \oplus Y_{j_m} = 0\} >> \frac{1}{2}$$

Le principe est alors d'approximer une partie de l'algorithme de chiffrement par cette combinaison linéaire sur F_2.

Plan

Codes à clefs publiques

Diffie et Hellman ont dégagé vers 1976 la notion de **code à clef publique** fondée sur la notion de **fonction à sens unique**.

Le système RSA

RSA: du nom des inventeurs Rivest, Shamir, Adleman.

• Chacun des correspondants A_i prend **deux nombres premiers p_i et q_i distincts**.

En pratique on les choisit de taille comparable de telle sorte que leur produit $\mathbf{n_i = p_i q_i}$ soit un nombre d'au moins **300 chiffres** en base 10 et plutôt **500** pour une protection longue.

- A_i choisit ensuite un **nombre e_i** premier avec **$\varphi(n_i)$**, $e_i \wedge \varphi(n_i) = 1$ où $\varphi(n_i) = (p_i - 1)(q_i - 1)$, en pratique on impose de plus $1 \leq e_i \leq \varphi(n_i) - 1$.

- Puis il calcule **l'inverse d_i** de **e_i modulo $\varphi(n_i)$**

c'est à dire en pratique, on cherche $d_i \in \mathrm{N}$ tel qu'il existe $k_i \in \mathrm{N}$ avec

$$d_i \times e_i = 1 + k_i \varphi(n_i) \text{ et } 1 \leq d_i \leq \varphi(n_i) - 1$$

- Chacun des correspondants A_j, A_i publie dans un **annuaire public n_j** et **e_j** qui forment sa **clé publique de chiffrement** et **garde secret p_j**, **q_j** et **d_j** qui forment sa **clé secrète de déchiffrement**.

Protocole d'envoi d'un message en RSA

Alice veut envoyer un message $\mathcal{M}$ à Bob. La clef publique d'Alice (resp Bob) est $(\mathbf{n_a}, \mathbf{e_a})$, (resp. $\mathbf{n_b}, \mathbf{e_b}$), sa clef secrète est $(\mathbf{p_a}, \mathbf{q_a}, \mathbf{d_a})$ (resp. $\mathbf{p_b}, \mathbf{q_b}, \mathbf{d_b}$)

• Alice commence par **transformer le message en nombres** par exemple en remplaçant les lettres et les différents symboles utilisés par des chiffres (de 0 à 255 dans le cas du code ASCII).

• Alice regarde dans l'annuaire public la clé de chiffrement $(\mathbf{n_b}, \mathbf{e_b})$ de Bob. Elle découpe le message $\mathcal{M}$ en blocs M de taille approximativement $\mathbf{n_b}$.

- Elle calcule

$$\mathbf{f_b(M) = M' \equiv M^{e_b} \mod n_b}$$

et envoie $M' = f_b(M)$ à Bob.

- Pour récupérer le texte en clair Bob calcule

$$f_b^{-1}(M') = (M')^{d_b} = f_b(M)^{d_b} \equiv \mathbf{M^{e_b d_b} = M^{1+k_b \varphi(n_b)} \mod n_b}$$

D'après le théorème d'Euler (cf. infra)

$$\mathbf{M^{\varphi(n_b)} \equiv 1 \mod n_b}$$

Signature RSA

Le code RSA permet aussi facilement de réaliser l'authentification du signataire (en admettant que sa clef privée n'a pas été volée).
Alice commence par calculer avec sa **clé secrète d_a**

$$M'' = f_a^{-1}(M) \equiv \mathbf{M^{d_a} \mod p_a q_a}$$

puis avec la clé publique de Bob, elle calcule M'

$$M' = f_b(M'') = \mathbf{f_b(f_a^{-1}(M))}$$

Elle envoie à Bob le couple

$$(\mathbf{f_b(M)}, \mathbf{M'})$$

Bob décode en calculant $\mathbf{f_a(f_b^{-1}(M'))} = f_a(f_b^{-1}(f_b(f_a^{-1}(M)))) = \mathbf{M}$ et $\mathbf{f_b^{-1} f_b(M) = M}$

Il constate que $\mathbf{M} = \mathbf{M}$. Il est sûr alors que le message vient d'Alice car seule Alice connait

$$\mathbf{f_a^{-1}}$$

De plus il est sûr que le message n'a pas été altéré.

Exemple de code RSA

On choisit un alphabet à 40 lettres

A=0, **B=1**, **C=2**, **D=3**, **E=4**, **F=5**,..., **X=23**, **Y=24**, **Z=25**, □=26, **.=27**,

?=28, **euros=29**, **0=30**, **1=31**,..., **8=38**, **9=39**

On choisit comme clé publique le couple $\mathbf{n} = 2047$, $\mathbf{e = 179}$, on a $40^2 \leq 2047 \leq 40^3$ on choisit donc de découper **les messages en clair en blocs de deux lettres** et **les messages chiffrés en blocs de trois lettres**.

On écrit chaque bloc de **deux symboles clairs $\mathbf{X_1}X_2$** sous la forme $\mathbf{40}x_1 + x_2$ où x_i est le nombre entre 1 et 39 correspondant à X_i. On écrit chaque bloc de **trois symboles codés $\mathbf{Y_1}Y_2Y3$** sous la forme $\mathbf{40^2}y_1 + 40y_2 + y_3$ où y_i est le nombre entre 0 et 39 correspondant à Y_i.

Le message clair est

ENVOYEZ euros2500.

Pour le coder on considère les blocs

$$\begin{aligned} \text{EN} &= \mathbf{4} \times 40 + 13 = 173, & \text{VO} &= \mathbf{21} \times 40 + 14 = 854, \\ \text{YE} &= \mathbf{24} \times 40 + 4 = 964, & \text{Z}\square &= \mathbf{1026}, \\ \text{euros2} &= \mathbf{1192}, & 50 &= \mathbf{1430}, \\ 0. &= \mathbf{1110} \end{aligned}$$

codage

$$\mathrm{EN} \mapsto \mathbf{(173)^{179}} \ \mathbf{mod\ 2047}$$
$$= 854 = 0 \times 40^2 + 21 \times 40 + 14 = AVO$$

$$\mathrm{VO} \mapsto \mathbf{(854)^{179}} \ \mathbf{mod\ 2047}$$
$$= 1315 = 0 \times 40^2 + 32 \times 40 + 35 = A25$$

$$\mathrm{YE} \mapsto \mathbf{(964)^{179}} \ \mathbf{mod\ 2047}$$
$$= 452 = 0 \times 40^2 + 11 \times 40 + 12 = ALM$$

$$\mathrm{Z}\square \mapsto \mathbf{(1026)^{179}} \ \mathbf{mod\ 2047}$$
$$= 1295 = 0 \times 40^2 + 32 \times 40 + 15 = A2P$$

$$\text{euros2} \mapsto (\mathbf{1192})^{\mathbf{179}} \ \mathbf{mod\ 2047}$$
$$= 511 = 0 \times 40^2 + 12 \times 40 + 31 = AM1$$

$$50 \mapsto (\mathbf{1430})^{\mathbf{179}} \ \mathbf{mod\ 2047}$$
$$= 1996 = 1 \times 40^2 + 9 \times 40 + 36 = BJ6$$

$$0. \mapsto (\mathbf{1110})^{\mathbf{179}} \ \mathbf{mod\ 2047 = 1642}$$
$$= 1 \times 40^2 + 1 \times 40 + 2 = BBC$$

Le message codé est donc

BAVOA25ALMA2PAM1BJ6BBC

Pour décoder on factorise $\mathbf{2}047 = 23 \times 89$
donc $\varphi(2047) = 22 \times 88 = 1936$ et $\mathbf{d} = 411$.

En fait les paramètres de ce code sont mal choisis car 22 divise 88 et donc on peut prendre pour d n'importe quel inverse de 179 modulo 88 comme par exemple 59. On constate que $411 = 59 + 4 \times 88$

Sécurité du système RSA

Le cryptosystème El Gamal

On considère le cryptosystème suivant:

Soit **p** un nombre premier tel que le problème du **logarithme discret** dans $\mathrm{Z}/p\mathrm{Z}$ soit difficile.

Soit **g** une **racine primitive** modulo p, cf. infra.

Soit $\mathcal{P} = (\mathbf{Z}/\mathbf{pZ})^*$, les messages en clair, $\mathcal{C} = (\mathbf{Z}/\mathbf{pZ})^* \times (\mathbf{Z}/\mathbf{pZ})^*$ les messages cryptés et soit

$$\mathcal{K} = \{(\mathbf{p}, \mathbf{g}, \alpha, \beta); \beta \equiv \mathbf{g}^{\alpha} \mod \mathbf{p}\}$$

l'espace des clefs

$\mathbf{p}, \mathbf{g}, \beta$ sont publiques, α est secret.

Alice veut transmettre un message, **M**, à Bob

1.–Bob **choisit un grand nombre premier p_b** (grand devant M) et **deux nombres g_b et α_b** inférieurs à p

2.–Bob calcule $\beta_b = g^{\alpha_b}$; alors **(β_b, g_b, p_b)** est sa clef publique, **α_b** est sa clef secrète.

3.–Alice choisit **k_a** et calcule

$$y_1 \equiv g_b^{k_a} \pmod{p_b} \text{ et } y_2 = \beta_b^{k_a} M$$

La paire $\mathbf{e_K(M, k_a) = (y_1, y_2)}$ est le **message codé** qu'Alice envoie à Bob.

Pour décoder Bob calcule $\mathbf{M \equiv y_2 (y_1^{\alpha_b})^{-1} \pmod{p_b}}$. Comme $y_1^{\alpha_b} = g^{k_a \alpha_b} \pmod{p}$ on a:

$$\mathbf{d_K(y_1, y_2) = y_2 (y_1^{\alpha_b})^{-1} \equiv \beta_b^k M (g_b^{k\alpha_b})^{-1} \equiv}$$
$$\mathbf{\equiv g_b^{\alpha_b k_a} M g^{-\alpha_b k_a} \equiv M \mod p}$$

La sécurité du cryptosystème EL Gamal repose sur le fait que pour de bon choix du nombre premier p_b le temps de calcul du logarithme discret est en $\mathbf{O\left(e^{C\sqrt{\log_2(p) \log_2 \log_2(p)}}\right)}$

Signature El Gamal

On reprend les notations précédentes. Soit M un message qu'Alice veut transmettre en le signant à Bob.
Pour $\mathbf{K_a} = (\mathbf{p_a}, \mathbf{g_a}, \beta_\mathbf{a})$ la clef publique d'Alice, $\alpha_\mathbf{a}$ sa clef secrète et $\mathbf{k'_a} \in (\mathbf{Z}/\mathbf{pZ})^*$ (secret) et premier à $(\mathbf{p_a} - \mathbf{1})$ on définit

$$\mathrm{sign}_{\mathbf{K_a}}(\mathbf{M}, \mathbf{k'_a}) = (\gamma_\mathbf{a}, \delta_\mathbf{a})$$

avec

$$\gamma_\mathbf{a} \equiv \mathbf{g_a}^{\mathbf{k'_a}} \mod \mathbf{p_a}$$
$$\delta_\mathbf{a} \equiv (\mathbf{M} - \alpha_\mathbf{a}\gamma_\mathbf{a})\mathbf{k'_a}^{-1} \mod (\mathbf{p_a} - \mathbf{1})$$

Pour M, $\gamma_a \in (\mathrm{Z}/p_a\mathrm{Z}_p)^*$ et $\delta_a \in (\mathrm{Z}/(p_a-1)\mathrm{Z})$ on définit

$$\mathbf{ver_{K_a}(M,\gamma_a,\delta_a) = (vrai) \Leftrightarrow \beta_a^{\gamma_a}\gamma_a^{\delta_a} \equiv g_a^M \mod p_a}$$

On constate que cette procédure réalise bien une signature du message M car

$$\begin{aligned}\beta_a^{\gamma_a}\gamma_a^{\delta_a} &\equiv g_a^{\alpha_a\gamma_a} g_a^{k'_a(M-\alpha_a\gamma_a)k_a'^{-1}} \mod p_a \\ &\equiv g_a^M \mod p_a\end{aligned}$$

Le cryptosystème Menezes-Vanstone

Ce système basé sur le groupe des points d'une **courbe elliptique définie sur un corps fini** F_q est une variante du cryptosystème El Gamal.

Une courbe elliptique est l'ensemble des points de $K \times K$ vérifiant l'équation

$$\mathbf{y}^2 = x^3 + ax + b, \quad a, b \in K$$

où K est un corps, par exemple $K = \mathrm{Q}, \mathrm{R}, \mathrm{C}, \mathrm{F}_q, \ldots$. En cryptographie K est un corps fini.

Le calcul de $\#E$ est difficile mais il existe un algorithme performant pour le faire **l'algorithme de Schoof**.

Ce système basé sur le groupe des points d'une **courbe elliptique définie sur un corps fini** F_q est une variante du cryptosystème El Gamal.

Une courbe elliptique est l'ensemble des points de $K \times K$ vérifiant l'équation

$$\mathbf{y}^2 = x^3 + ax + b, \quad a, b \in K$$

où K est un corps, par exemple $K = \mathrm{Q}, \mathrm{R}, \mathrm{C}, \mathrm{F}_q, \ldots$. En cryptographie K est un corps fini.
Le calcul de $\#E$ est difficile mais il existe un algorithme performant pour le faire **l'algorithme de Schoof**.

Soit E une courbe elliptique définie sur le corps fini $\mathrm{F}_p \simeq \mathrm{Z}/p\mathrm{Z}$ ($p > 3$ premier) telle que E contienne un sous-groupe cyclique H, engendré par un point $\mathcal{B}$ de E, dans lequel le problème du logarithme discret soit difficile. Soit

$$P = \mathrm{F}_p^* \times \mathrm{F}_p^*, \text{ les textes en clair}$$
$$C = E \times \mathrm{F}_p^* \times \mathrm{F}_p^*, \text{ les textes chiffrés}$$

Posons, $\sharp H$ la cardinal de H, et posons

$$K = \left\{(E, \alpha, a, \beta) : \beta = a \cdot \alpha\right\}$$

l'espace des clefs où $\alpha \in E$. Les valeurs α et β sont publiques et $a \in \mathrm{Z}/\sharp H\mathrm{Z}$ est secret et choisi au hasard.

Pour $K = (E, \alpha, a, \beta)$, et pour un nombre aléatoire secret $k \in \mathrm{Z}/\sharp H\mathrm{Z}$, et pour $x = (x_1, x_2) \in P = \mathrm{F}_p^* \times \mathrm{F}_p^*$, on chiffre le message x à l'aide de la clef K et du nombre aléatoire k en posant

$$\mathcal{H}_K(x, k) = (y_0, y_1, y_2),$$

où

$$y_0 = k \cdot \alpha, \qquad (c_1, c_2) = k \cdot \beta,$$
$$y_1 = c_1 x_1 \pmod{p}, \qquad y_2 = c_2 x_2 \pmod{p}$$

Pour déchiffrer un message chiffré

$$y = (y_0, y_1, y_2)$$

on effectue les opérations

$$\mathcal{D}_K(y) = (y_1 c_1^{-1} \pmod p, \quad y_2 c_2^{-1} \pmod p),$$

où $a \cdot y_0 = (c_1, c_2)$.

Pour pouvoir utiliser le groupe des points d'une courbe elliptique sur un corps fini, E, il faut ensuite trouver un **sous-groupe cyclique** aussi gros que possible afin de pouvoir transposer le schéma de codage El-Gamal. Il y a des algorithmes efficaces pour cela.

Avec des sous-groupes cycliques de E à 2^{160} éléments on a une très bonne sécurité si l'on prend quelques précautions pour éliminer des courbes elliptiques indésirables pour lesquelles le problème du logarithme discret est facile.

L'avantage des cryptosystèmes basés sur les courbes elliptiques est, entre autre, la possibilité d'avoir des **clés courtes** pour une bonne sécurité.

Plan

Fonctions de Hachage

Remarque: les procédures de signature précédentes ont un coût prohibitif pour signer des longs messages.
On introduit donc la notion de fonction de **hachage cryptographique**, **h**.
Cette fonction est publique.

message	x	longueur arbitraire
	$\downarrow$	
empreinte numérique	$z = h(x)$	160 bits
	$\downarrow$	
signature	$y = \mathrm{sign}_K(z)$	320 bits

En réalité une fonctions de hachage prend un message x de taille inférieure à N fixé qu'elle transforme en une empreinte de taille 160 bits (ou 256 ou 384 ou 512).

Il existe des techniques classiques pour étendre les fonctions de hachage de domaine de départ fini $\leq N$ en des fonctions de hachage dont le domaine de départ est constitué de messages de longueur arbitraire. Elles sont appelées alors **fonction de compression**. On peut supposer que cette extension des fonctions de hachage est déjà faite.

Lorsqu'Alice souhaite envoyer un message signé, x, elle calcule d'abord l'empreinte numérique, $z = h(x)$, elle signe avec $y = \text{sign}_K(z)$ et transmet le **couple (x, y) par le canal de communication**.

Tout le monde peut vérifier la signature en calculant l'empreinte $z = h(x)$ et en utilisant le procédé de vérification de la signature $\text{ver}_K(z, y)$.

Critères de conception

La fonction de hachage doit:

- réduire la taille des entrées (**compression**);
- être **facile à calculer** mais difficile à inverser, en particulier ce n'est pas une bonne idée de prendre AES!

- Cependant elle doit être construite avec soin pour qu'elle n'affaiblisse pas le procédé de signature. On lui demande en particulier de **se comporter de façon aléatoire**, de ne pas comporter de relation entre le haché et le message initial.
- Elle peut être liée ou pas à **une primitive à clef secrète**.

Critères de conception

La fonction de hachage doit:

- réduire la taille des entrées (**compression**);
- être **facile à calculer** mais difficile à inverser, en particulier ce n'est pas une bonne idée de prendre AES!

• Cependant elle doit être construite avec soin pour qu'elle n'affaiblisse pas le procédé de signature. On lui demande en particulier de **se comporter de façon aléatoire**, de ne pas comporter de relation entre le haché et le message initial.

• Elle peut être liée ou pas à **une primitive à clef secrète**.

Critères de conception

La fonction de hachage doit:

- réduire la taille des entrées (**compression**);
- être **facile à calculer** mais difficile à inverser, en particulier ce n'est pas une bonne idée de prendre AES!

• Cependant elle doit être construite avec soin pour qu'elle n'affaiblisse pas le procédé de signature. On lui demande en particulier de **se comporter de façon aléatoire**, de ne pas comporter de relation entre le haché et le message initial.

• Elle peut être liée ou pas à **une primitive à clef secrète**.

Sécurité

• Comme une fonction de hachage n'est évidemment pas injective, il existe des couples de messages x' et x tels que $h(x') = h(x)$. L'attaque la plus évidente consiste pour Martin à partir d'un message signé (x, y) authentique ($y = \text{sign}_K(x)$) précedemment calculé par Alice à **calculer $z = h(x)$** et à chercher $x' \neq x$ tel que $h(x) = h(x')$. Si Martin y parvient **(x', y) est un message valide** .

Définition

Une fonction de hachage h est à **collisions faibles difficiles** *si étant donné un message x, il calculatoirement difficile d'obtenir un message $x' \neq x$ tel que $h(x) = h(x')$.*

Définition

Une fonction de hachage h est à **collisions fortes difficiles** *s'il est calculatoirement difficile d'obtenir deux messages différents x et x' tels que $h(x) = h(x')$*

Définition

Une fonction de hachage est **à sens unique** *si étant donnée une empreinte numérique z, il est calculatoirement difficile de trouver un message x tels $h(x) = z$*

Si une fonction de hachage est à collisions fortes difficiles elle est bien sûr à collisions faibles difficiles, mais aussi à sens unique.

Définition

Une fonction de hachage h est à **collisions fortes difficiles** *s'il est calculatoirement difficile d'obtenir deux messages différents x et x' tels que $h(x) = h(x')$*

Définition

Une fonction de hachage est **à sens unique** *si étant donnée une empreinte numérique z, il est calculatoirement difficile de trouver un message x tels $h(x) = z$*

Si une fonction de hachage est à collisions fortes difficiles elle est bien sûr à collisions faibles difficiles, mais aussi à sens unique.

Définition

Une fonction de hachage h est à **collisions fortes difficiles** *s'il est calculatoirement difficile d'obtenir deux messages différents x et x' tels que $h(x) = h(x')$*

Définition

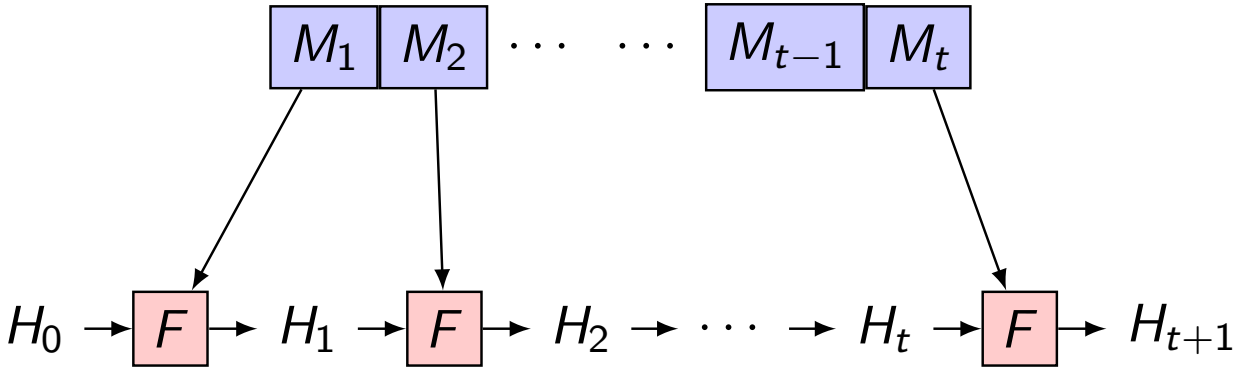
Une fonction de hachage est **à sens unique** *si étant donnée une empreinte numérique z, il est calculatoirement difficile de trouver un message x tels $h(x) = z$*

Si une fonction de hachage est à collisions fortes difficiles elle est bien sûr à collisions faibles difficiles, mais aussi à sens unique.

En particulier dans une assemblée de **23 personnes la probabilité d'avoir deux dates d'anniversaire égale est supérieure à** $1/2$.

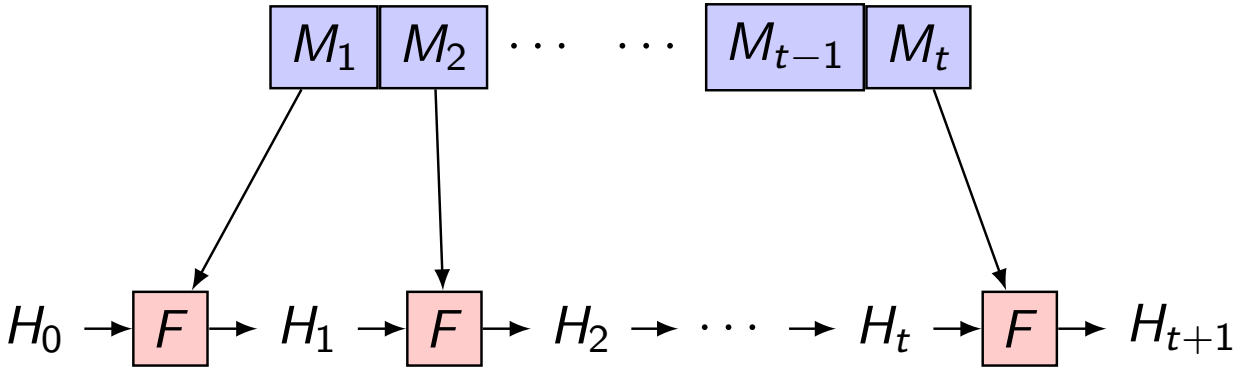
Une empreinte de 40 bits est vulnérable à une attaque des anniversaires avec probabilité supérieure à $1/2$ en utilisant seulement 2^{20} messages aléatoires . Pour une empreinte de 128 bits il faut 2^{64} messages aléatoires. Le choix d'une **empreinte de 160 bits** donne une bonne résistance aux attaques anniversaires.

Fonctions itératives



- Les H_i sont appelés **les hachés intermédiaires**. Le haché de M est **le dernier haché intermédiaire,** H_{t+1}.
- M_t s'appelle **le padding** et contient comme information, en particulier, la taille du message à hacher.

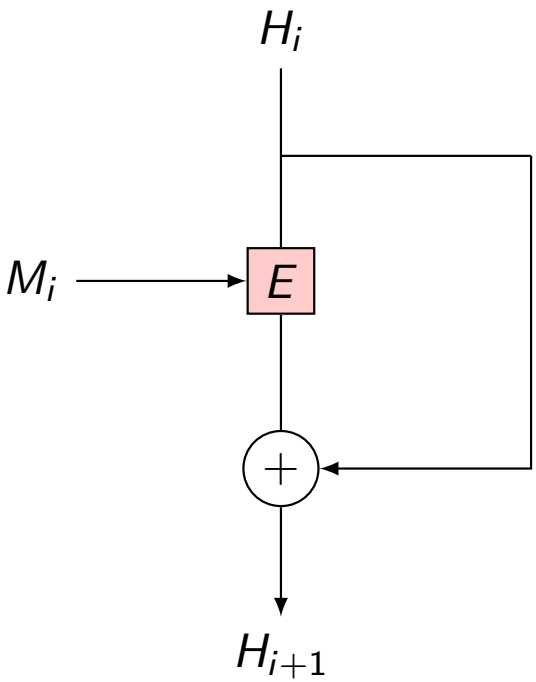
Fonctions itératives



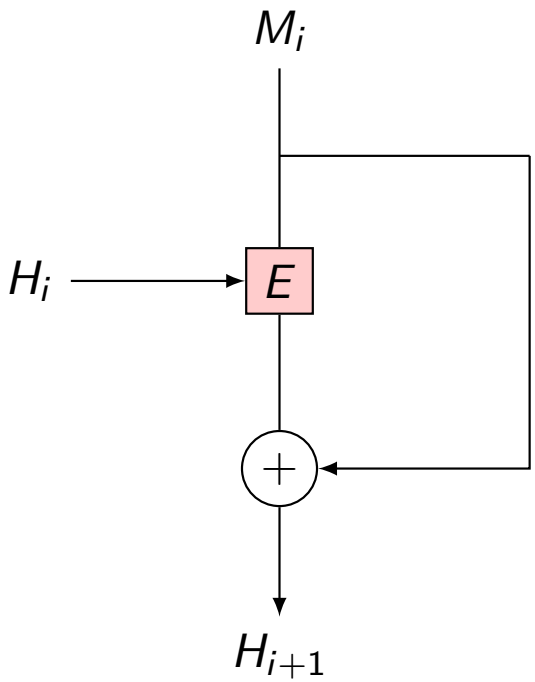
- Les H_i sont appelés **les hachés intermédiaires**. Le haché de M est **le dernier haché intermédiaire,** H_{t+1}.
- M_t s'appelle **le padding** et contient comme information, en particulier, la taille du message à hacher.

Davies-Meyer

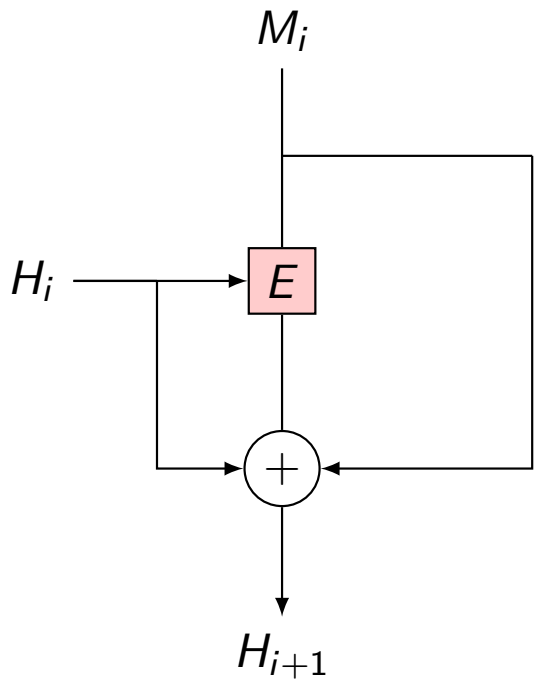
Pour obtenir une fonction à sens unique, i.e. difficile à inverser, on utilise le schéma de Davies-Meyer:



Davies-Meyer



Matyas-Meyer-Oseas



Miyaguchi-Preneel

• Un standard actuel en matière de fonction de hachage est **SHA-1** (Secure Hash Algorithm) avec une empreinte de taille 160 bits, faisant suite aux standards MD4, (1990), MD5 (1992), SHA (1995).

• Depuis 2001, une nouvelle version de SHA-1, SHA-2, ainsi que les versions SHA-256, SHA-384 et SHA-512 sont en cours de validation (256, 384, 512 est la taille de l'empreinte).

• Depuis 2005, des faiblesses sur SHA-1 ont été constatées; progressivement son utilisation est remplacée par SHA-256.

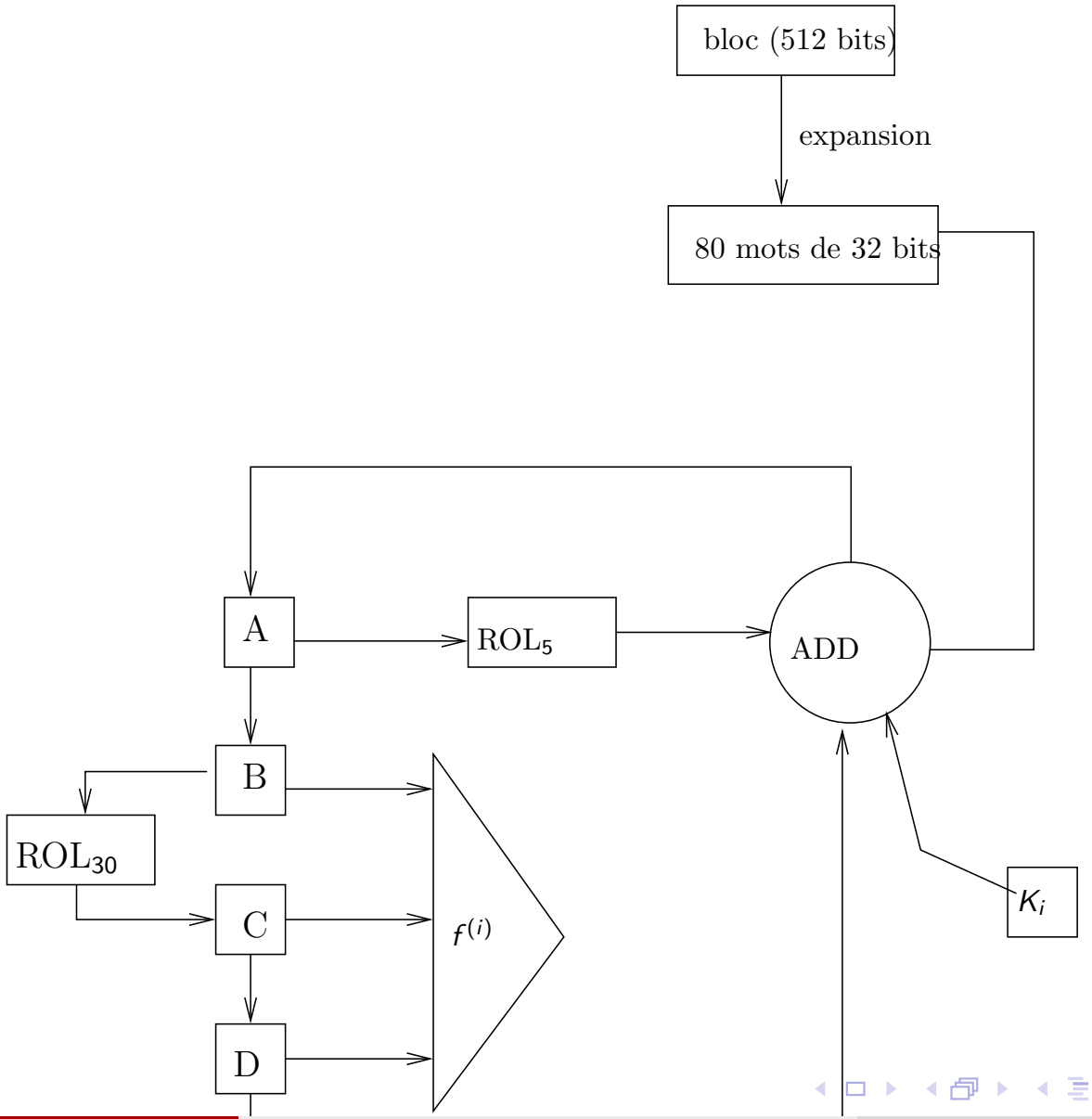
• La conception de SHA-2 étant proche de celle de SHA-1, un concours international a été organisé pour SHA-3 dont le résultat devait être connu en décembre 2012.

• Un standard actuel en matière de fonction de hachage est **SHA-1** (Secure Hash Algorithm) avec une empreinte de taille 160 bits, faisant suite aux standards MD4, (1990), MD5 (1992), SHA (1995).
• Depuis 2001, une nouvelle version de SHA-1, SHA-2, ainsi que les versions SHA-256, SHA-384 et SHA-512 sont en cours de validation (256, 384, 512 est la taille de l'empreinte).
• Depuis 2005, des faiblesses sur SHA-1 ont été constatées; progressivement son utilisation est remplacée par SHA-256.
• La conception de SHA-2 étant proche de celle de SHA-1, un concours international a été organisé pour SHA-3 dont le résultat devait être connu en décembre 2012.

• Un standard actuel en matière de fonction de hachage est **SHA-1** (Secure Hash Algorithm) avec une empreinte de taille 160 bits, faisant suite aux standards MD4, (1990), MD5 (1992), SHA (1995).
• Depuis 2001, une nouvelle version de SHA-1, SHA-2, ainsi que les versions SHA-256, SHA-384 et SHA-512 sont en cours de validation (256, 384, 512 est la taille de l'empreinte).
• Depuis 2005, des faiblesses sur SHA-1 ont été constatées; progressivement son utilisation est remplacée par SHA-256.
• La conception de SHA-2 étant proche de celle de SHA-1, un concours international a été organisé pour SHA-3 dont le résultat devait être connu en décembre 2012.

• Un standard actuel en matière de fonction de hachage est **SHA-1** (Secure Hash Algorithm) avec une empreinte de taille 160 bits, faisant suite aux standards MD4, (1990), MD5 (1992), SHA (1995).
• Depuis 2001, une nouvelle version de SHA-1, SHA-2, ainsi que les versions SHA-256, SHA-384 et SHA-512 sont en cours de validation (256, 384, 512 est la taille de l'empreinte).
• Depuis 2005, des faiblesses sur SHA-1 ont été constatées; progressivement son utilisation est remplacée par SHA-256.
• La conception de SHA-2 étant proche de celle de SHA-1, un concours international a été organisé pour SHA-3 dont le résultat devait être connu en décembre 2012.

Architecture du SHA-1



SHA-1: algorithme

L'algorithme SHA-1 est découpé en deux phases:

- **le prétraitement**:
  - on commence par compléter le message en ajoutant un 1 puis une série de 0 et enfin la longueur du message initial codé sur 64 bits (les messages ont un maximum de 2^{64} bits) de façon à obtenir une longueur multiple de 512 bits.
  - On découpe ensuite en blocs de 512 bits et
  - on initialise les variables de travail.

$$K_i = \begin{cases} 0x5a827999 & 0 \leq i \leq 19 \\ 0x6ed9eba1 & 20 \leq i \leq 39 \\ 0x8f1bbcdc & 40 \leq i \leq 59 \\ 0xca62c1d6 & 60 \leq i \leq 79 \end{cases}$$

- Le calcul du haché de $5 \times 32 = 160$ bits, décrit dans le transparent suivant.

SHA-1: algorithme

L'algorithme SHA-1 est découpé en deux phases:

- **le prétraitement**:
  - on commence par compléter le message en ajoutant un 1 puis une série de 0 et enfin la longueur du message initial codé sur 64 bits (les messages ont un maximum de 2^{64} bits) de façon à obtenir une longueur multiple de 512 bits.
  - On découpe ensuite en blocs de 512 bits et
  - on initialise les variables de travail.

$$K_i = \begin{cases} 0x5a827999 & 0 \leq i \leq 19 \\ 0x6ed9eba1 & 20 \leq i \leq 39 \\ 0x8f1bbcdc & 40 \leq i \leq 59 \\ 0xca62c1d6 & 60 \leq i \leq 79 \end{cases}$$

- Le calcul du haché de $5 \times 32 = 160$ bits, décrit dans le transparent suivant.

SHA-1: algorithme

L'algorithme SHA-1 est découpé en deux phases:

- **le prétraitement**:
  - on commence par compléter le message en ajoutant un 1 puis une série de 0 et enfin la longueur du message initial codé sur 64 bits (les messages ont un maximum de 2^{64} bits) de façon à obtenir une longueur multiple de 512 bits.
  - On découpe ensuite en blocs de 512 bits et
  - on initialise les variables de travail.

$$K_i = \begin{cases} 0x5a827999 & 0 \leq i \leq 19 \\ 0x6ed9eba1 & 20 \leq i \leq 39 \\ 0x8f1bbcdc & 40 \leq i \leq 59 \\ 0xca62c1d6 & 60 \leq i \leq 79 \end{cases}$$

- Le calcul du haché de $5 \times 32 = 160$ bits, décrit dans le transparent suivant.

SHA-1: tour élémentaire

- **Expansion**: à partir d'un mot de 512 bits on construit 80 mots de 32 bits comme suit:

  $W_i = M_i$ pour $i = 1, \cdots, 16$

  $W_i = ROT_1(W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16})$ pour $i = 17, \cdots, 80$

- On effectue alors 80 tours selon le schéma précédent avec

$$A_0 = 0x67452301$$
$$B_0 = 0xefcdab89$$
$$C_0 = 0x98badcfe$$
$$D_0 = 0x10325476$$
$$E_0 = 0xc3d2e1f0$$

- Les fonctions $f_i(x, y, z)$ sont données par:

$$\begin{array}{ll} (x \wedge y) \vee (\neg x \wedge z) & 0 \leq i \leq 19 \\ x \oplus y \oplus z & 20 \leq i \leq 39 \\ (x \wedge y) \vee (y \wedge z) \vee (z \wedge x) & 40 \leq i \leq 59 \\ x \oplus y \oplus z & 60 \leq i \leq 79 \end{array}$$

SHA-1: tour élémentaire

- **Expansion**: à partir d'un mot de 512 bits on construit 80 mots de 32 bits comme suit:

  $W_i = M_i$ pour $i = 1, \cdots, 16$

  $W_i = ROT_1(W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16})$ pour $i = 17, \cdots, 80$

- On effectue alors 80 tours selon le schéma précédent avec

$$A_0 = 0x67452301$$
$$B_0 = 0xefcdab89$$
$$C_0 = 0x98badcfe$$
$$D_0 = 0x10325476$$
$$E_0 = 0xc3d2e1f0$$

- Les fonctions $f_i(x, y, z)$ sont données par:

$$(x \wedge y) \vee (\neg x \wedge z) \qquad 0 \leq i \leq 19$$
$$x \oplus y \oplus z \qquad 20 \leq i \leq 39$$
$$(x \wedge y) \vee (y \wedge z) \vee (z \wedge x) \qquad 40 \leq i \leq 59$$
$$x \oplus y \oplus z \qquad 60 \leq i \leq 79$$

SHA-1: tour élémentaire

- **Expansion**: à partir d'un mot de 512 bits on construit 80 mots de 32 bits comme suit:

  $W_i = M_i$ pour $i = 1, \cdots, 16$
  $W_i = ROT_1(W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16})$ pour $i = 17, \cdots, 80$

- On effectue alors 80 tours selon le schéma précédent avec

$$A_0 = 0x67452301$$
$$B_0 = 0xefcdab89$$
$$C_0 = 0x98badcfe$$
$$D_0 = 0x10325476$$
$$E_0 = 0xc3d2e1f0$$

- Les fonctions $f_i(x, y, z)$ sont données par:

$(x \wedge y) \vee (\neg x \wedge z)$	$0 \leq i \leq 19$
$x \oplus y \oplus z$	$20 \leq i \leq 39$
$(x \wedge y) \vee (y \wedge z) \vee (z \wedge x)$	$40 \leq i \leq 59$
$x \oplus y \oplus z$	$60 \leq i \leq 79$

Protocoles

Plan

Signature

Signature d'un document par un **cryptosystème à clef publique**.

Le protocole est simple

1. Alice chiffre le document d'une part avec sa **clé privée**, signant ainsi le document et d'autre part avec **la clé publique de Bob**
2. Alice **envoie les deux document à Bob**.
3. Bob déchiffre le premier document avec la clé publique d'Alice et le second avec sa clé privée. Si les **deux sont identiques** il est sûr qu'Alice est l'expéditeur.

Sûreté du protocole

1. La signature est **authentique**: quand Bob vérifie le message avec la clé publique d'Alice, il est sûr que seule Alice pouvait l'avoir crypté avec sa clé privée.
2. La signature est **infalsifiable**. Seule Alice connait sa clé privée.
3. La signature **n'est pas réutilisable**. La signature est une fonction du document et elle ne peut pas être transférée sur n'importe quel autre document.
4. Le document est **immuable**. Si Bob falsifiait le message après l'avoir reçu, il ne pourrait pas le signer car la clé privée d'Alice n'est connue que d'elle seule.
5. La signature **ne peut pas être reniée**. Bob n'a pas besoin de l'aide d'Alice pour vérifier sa signature.

Sûreté du protocole

1. La signature est **authentique**: quand Bob vérifie le message avec la clé publique d'Alice, il est sûr que seule Alice pouvait l'avoir crypté avec sa clé privée.
2. La signature est **infalsifiable**. Seule Alice connait sa clé privée.
3. La signature **n'est pas réutilisable**. La signature est une fonction du document et elle ne peut pas être transférée sur n'importe quel autre document.
4. Le document est **immuable**. Si Bob falsifiait le message après l'avoir reçu, il ne pourrait pas le signer car la clé privée d'Alice n'est connue que d'elle seule.
5. La signature **ne peut pas être reniée**. Bob n'a pas besoin de l'aide d'Alice pour vérifier sa signature.

Sûreté du protocole

1. La signature est **authentique**: quand Bob vérifie le message avec la clé publique d'Alice, il est sûr que seule Alice pouvait l'avoir crypté avec sa clé privée.
2. La signature est **infalsifiable**. Seule Alice connait sa clé privée.
3. La signature **n'est pas réutilisable**. La signature est une fonction du document et elle ne peut pas être transférée sur n'importe quel autre document.
4. Le document est **immuable**. Si Bob falsifiait le message après l'avoir reçu, il ne pourrait pas le signer car la clé privée d'Alice n'est connue que d'elle seule.
5. La signature **ne peut pas être reniée**. Bob n'a pas besoin de l'aide d'Alice pour vérifier sa signature.

Sûreté du protocole

1. La signature est **authentique**: quand Bob vérifie le message avec la clé publique d'Alice, il est sûr que seule Alice pouvait l'avoir crypté avec sa clé privée.
2. La signature est **infalsifiable**. Seule Alice connait sa clé privée.
3. La signature **n'est pas réutilisable**. La signature est une fonction du document et elle ne peut pas être transférée sur n'importe quel autre document.
4. Le document est **immuable**. Si Bob falsifiait le message après l'avoir reçu, il ne pourrait pas le signer car la clé privée d'Alice n'est connue que d'elle seule.
5. La signature **ne peut pas être reniée**. Bob n'a pas besoin de l'aide d'Alice pour vérifier sa signature.

Sûreté du protocole

1. La signature est **authentique**: quand Bob vérifie le message avec la clé publique d'Alice, il est sûr que seule Alice pouvait l'avoir crypté avec sa clé privée.
2. La signature est **infalsifiable**. Seule Alice connait sa clé privée.
3. La signature **n'est pas réutilisable**. La signature est une fonction du document et elle ne peut pas être transférée sur n'importe quel autre document.
4. Le document est **immuable**. Si Bob falsifiait le message après l'avoir reçu, il ne pourrait pas le signer car la clé privée d'Alice n'est connue que d'elle seule.
5. La signature **ne peut pas être reniée**. Bob n'a pas besoin de l'aide d'Alice pour vérifier sa signature.

Raffinements

- Connaissant les signatures s_1, s_2 de m_1 et m_2 alors

$$m_1 \times m_2 \text{ est signé par } s_1 \times s_2.$$

Afin de casser **cette propriété de multiplicativité**, on introduit **une encapsulation**, i.e. on ajoute au début du message un certain nombre de zéros (ou une suite déterminée d'octets).

- Les algorithmes à clef publique sont **trop lents** pour signer de longs documents.

Pour gagner du temps les protocoles de **signature numérique** sont souvent réalisés avec des **fonctions de hachage à sens unique**.

Au lieu de signer le document Alice signe l'**empreinte du document**.

Raffinements

- Connaissant les signatures s_1, s_2 de m_1 et m_2 alors

$$m_1 \times m_2 \text{ est signé par } s_1 \times s_2.$$

Afin de casser **cette propriété de multiplicativité**, on introduit **une encapsulation**, i.e. on ajoute au début du message un certain nombre de zéros (ou une suite déterminée d'octets).

- Les algorithmes à clef publique sont **trop lents** pour signer de longs documents.

Pour gagner du temps les protocoles de **signature numérique** sont souvent réalisés avec des **fonctions de hachage à sens unique**.

Au lieu de signer le document Alice signe l'**empreinte du document**.

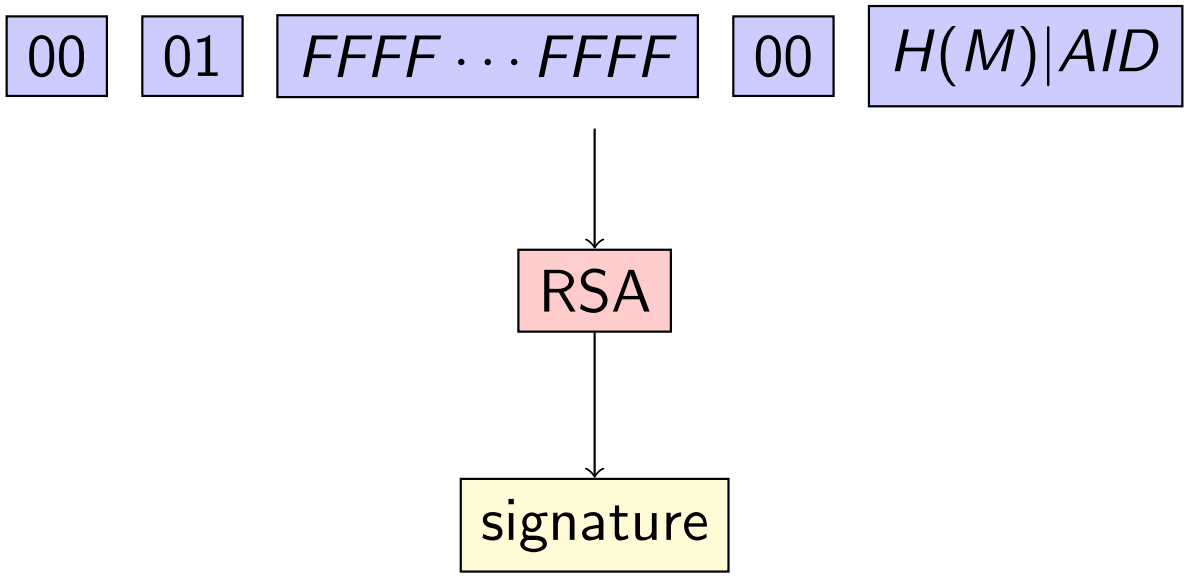
1. Alice calcule à l'aide de la fonction de hachage à sens unique, l'**empreinte du document**.
2. Alice chiffre, à l'aide de l'**algorithme de signature numérique**, cette empreinte avec sa clef privée, signant par la même occasion le document.
3. Alice envoie le **document et l'empreinte signée** à Bob (à l'aide de la clef publique de Bob).
4. Bob calcule, à l'aide de la fonction de hachage à sens unique, l'**empreinte du document qu'Alice lui a envoyé**. Ensuite à l'aide de l'algorithme de signature numérique, il **déchiffre l'empreinte signée** avec la clef publique d'Alice. La signature est valide si l'empreinte de la signature est la même que l'empreinte qu'il a produite.

Avantage de ce procédé:

• **rapidité** de la transmission et de la comparaison des empreinte car une empreinte ne comporte que 160 bits.

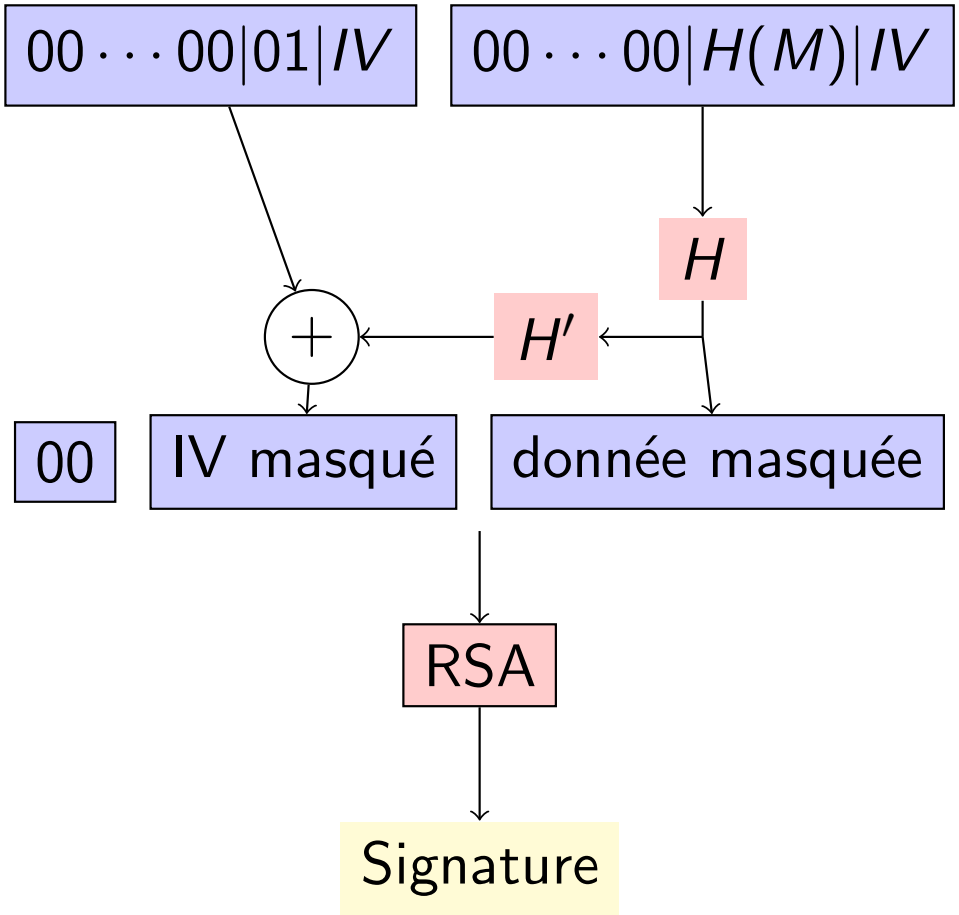
• **confidentialité** car la signature peut être gardée à part du message. On peut donc vérifier l'existence du document sans stocker son contenu.

Signature RSA PKCS-1 V1.5



Signature RSA PSS

L'algorithme de signature PSS utilise des nombres aléatoires de sorte que la signature d'un même message peut être différente:



Signature El Gamal

- On commence par choisir un premier p et un générateur g de $\mathrm{F}_p^\times$.
- On choisit $0 < x < p-1$ et on calcule $y = g^x \mod p$. La clé publique est (p, g, y), la clé privée est x.
- Pour signer un message m:
  - on choisit $k < p-1$ **aléatoirement** et premier avec $p-1$ et on calcule

$$r = g^k \mod p \text{ et } s = k^{-1}\Big(H(m) - xr\Big) \mod p-1.$$

  - La signature de m est (r, s).
- **Pour vérifier une signature**:
  - on vérifie que $r < p$;
  - on calcule $u = y^r r^s \mod p$ et $v = g^{H(m)} \mod p$.
  - On vérifie alors que $u = v$ puisque

$$u = y^r r^s = g^{rs} g^{kk^{-1}(H(m)-xr)} = g^{xr+H(m)-xr} = v.$$

Signature El Gamal

- On commence par choisir un premier p et un générateur g de $\mathrm{F}_p^\times$.
- On choisit $0 < x < p-1$ et on calcule $y = g^x \mod p$. La clé publique est (p, g, y), la clé privée est x.
- Pour signer un message m:
  - on choisit $k < p-1$ **aléatoirement** et premier avec $p-1$ et on calcule

$$r = g^k \mod p \text{ et } s = k^{-1}\Big(H(m) - xr\Big) \mod p-1.$$

  - La signature de m est (r, s).
- **Pour vérifier une signature**:
  - on vérifie que $r < p$;
  - on calcule $u = y^r r^s \mod p$ et $v = g^{H(m)} \mod p$.
  - On vérifie alors que $u = v$ puisque

$$u = y^r r^s = g^{rs} g^{kk^{-1}(H(m)-xr)} = g^{xr+H(m)-xr} = v.$$

Signature El Gamal

- On commence par choisir un premier p et un générateur g de $\mathrm{F}_p^\times$.
- On choisit $0 < x < p-1$ et on calcule $y = g^x \mod p$. La clé publique est (p, g, y), la clé privée est x.
- Pour signer un message m:
  - on choisit $k < p-1$ **aléatoirement** et premier avec $p-1$ et on calcule

$$r = g^k \mod p \text{ et } s = k^{-1}\Big(H(m) - xr\Big) \mod p-1.$$

  - La signature de m est (r, s).
- **Pour vérifier une signature**:
  - on vérifie que $r < p$;
  - on calcule $u = y^r r^s \mod p$ et $v = g^{H(m)} \mod p$.
  - On vérifie alors que $u = v$ puisque

$$u = y^r r^s = g^{rs} g^{kk^{-1}(H(m)-xr)} = g^{xr+H(m)-xr} = v.$$

Signature El Gamal

- On commence par choisir un premier p et un générateur g de $\mathrm{F}_p^\times$.
- On choisit $0 < x < p-1$ et on calcule $y = g^x \mod p$. La clé publique est (p, g, y), la clé privée est x.
- Pour signer un message m:
  - on choisit $k < p-1$ **aléatoirement** et premier avec $p-1$ et on calcule

$$r = g^k \mod p \text{ et } s = k^{-1}\Big(H(m) - xr\Big) \mod p-1.$$

  - La signature de m est (r, s).
- **Pour vérifier une signature**:
  - on vérifie que $r < p$;
  - on calcule $u = y^r r^s \mod p$ et $v = g^{H(m)} \mod p$.
  - On vérifie alors que $u = v$ puisque

$$u = y^r r^s = g^{rs} g^{kk^{-1}(H(m)-xr)} = g^{xr+H(m)-xr} = v.$$

Signature El Gamal

Il faut veiller à modifier k sinon pour deux messages m_1 et m_2:

$$\begin{aligned} s_1 &= k^{-1}(H(m_1) - xr) \mod p-1 \\ s_2 &= k^{-1}(H(m_2) - xr) \mod p-1 \\ (s_1 - s_2)k &= H(m_1) - H(m_2) \mod p-1 \\ k &= (s_1 - s_2)^{-1}(H(m_1) - H(m_2)) \mod p-1 \end{aligned}$$

et on obtient $r = g^k$ puis **la clé secrète** x via la formule

$$sk = H(m) - xr.$$

Signature El Gamal

Il faut veiller à modifier k sinon pour deux messages m_1 et m_2:

$$\begin{aligned} s_1 &= k^{-1}(H(m_1) - xr) \mod p-1 \\ s_2 &= k^{-1}(H(m_2) - xr) \mod p-1 \\ (s_1 - s_2)k &= H(m_1) - H(m_2) \mod p-1 \\ k &= (s_1 - s_2)^{-1}(H(m_1) - H(m_2)) \mod p-1 \end{aligned}$$

et on obtient $r = g^k$ puis **la clé secrète** x via la formule

$$sk = H(m) - xr.$$

Signature El Gamal

- Si on n'utilise pas **de fonction de hachage** alors pour $s = k(m - xr) \mod p - 1$:
  - pour β premier avec $p - 1$,
  - on calcule $r = g^\alpha y^\beta \mod p$ et $s = -r\beta^{-1} \mod p - 1$

et (r, s) est une signature du message $m = s\alpha$ puisque

$$u = y^r r^s = y^r (g^\alpha y^\beta)^s) = g^{s\alpha} y^{r+s\beta} = g^m = v.$$

- Si on autorise des $r > p$, il serait facile de produire des messages signés m' dès que l'on dispose d'une signature (r, s) de m:
  - on calcule $u' = H(m')H(m)^{-1}$ et on pose $s' = su'$;
  - par le théorème chinois on trouve $r' < p^2$ tel que
    - $r' = ru' \mod p - 1$;
    - $r' = r \mod p$.

  (r', s') est alors une signature de m' car $r = r' = g^k \mod p$ et

  $$s' = k^{-1}(H(m) - xr) \times H(m')H(m)^{-1} = k^{-1}(H(m') - xr') \mod p - 1.$$

Signature El Gamal

• Si on n'utilise pas **de fonction de hachage** alors pour $s = k(m - xr) \mod p - 1$:

- pour β premier avec $p - 1$,
- on calcule $r = g^{\alpha} y^{\beta} \mod p$ et $s = -r\beta^{-1} \mod p - 1$

et (r, s) est une signature du message $m = s\alpha$ puisque

$$u = y^r r^s = y^r (g^{\alpha} y^{\beta})^s) = g^{s\alpha} y^{r+s\beta} = g^m = v.$$

• Si on autorise des $r > p$, il serait facile de produire des messages signés m' dès que l'on dispose d'une signature (r, s) de m:

- on calcule $u' = H(m')H(m)^{-1}$ et on pose $s' = su'$;
- par le théorème chinois on trouve $r' < p^2$ tel que
  - $r' = ru' \mod p - 1$;
  - $r' = r \mod p$.

  (r', s') est alors une signature de m' car $r = r' = g^k \mod p$ et

$$s' = k^{-1}(H(m) - xr) \times H(m')H(m)^{-1} = k^{-1}(H(m') - xr') \mod p - 1.$$

Signature DSA

Les données sont

- un premier p (1500 bits) et un autre q (160 bits) divisant $p-1$;
- un élément $g \in \mathrm{F}_p^\times$ d'ordre q.
- Pour $x < q$ on calcule $y = g^x \mod p$ et on publie (p, q, g, y) alors que x reste secret.

La signature d'un message m est (r, s) avec

$$\begin{cases} r = g^k \mod p \\ s = k^{-1}(H(m) + xr) \mod q \end{cases}$$

La vérification demande $r < p$ et $s < q$ puis on calcule

$$u = s^{-1}H(m) \mod q \text{ et } v = rs^{-1} \mod q$$

et $r = g^u y^v \mod p$.

- La signature est rapide en revanche la vérification est plus coûteuse

Signature DSA

Les données sont

- un premier p (1500 bits) et un autre q (160 bits) divisant $p-1$;
- un élément $g \in \mathrm{F}_p^\times$ d'ordre q.
- Pour $x < q$ on calcule $y = g^x \mod p$ et on publie (p, q, g, y) alors que x reste secret.

La signature d'un message m est (r, s) avec

$$\begin{cases} r = g^k \mod p \\ s = k^{-1}(H(m) + xr) \mod q \end{cases}$$

La vérification demande $r < p$ et $s < q$ puis on calcule

$$u = s^{-1}H(m) \mod q \text{ et } v = rs^{-1} \mod q$$

et $r = g^u y^v \mod p$.

- La signature est rapide en revanche la vérification est plus coûteuse

Signature DSA

Les données sont

- un premier p (1500 bits) et un autre q (160 bits) divisant $p-1$;
- un élément $g \in \mathrm{F}_p^\times$ d'ordre q.
- Pour $x < q$ on calcule $y = g^x \mod p$ et on publie (p, q, g, y) alors que x reste secret.

La signature d'un message m est (r, s) avec

$$\begin{cases} r = g^k \mod p \\ s = k^{-1}(H(m) + xr) \mod q \end{cases}$$

La vérification demande $r < p$ et $s < q$ puis on calcule

$$u = s^{-1}H(m) \mod q \text{ et } v = rs^{-1} \mod q$$

et $r = g^u y^v \mod p$.

• La signature est rapide en revanche la vérification est plus coûteuse.

Documents datés et signés

Dans certaines circonstances Bob peut duper Alice.

Il peut **réutiliser le document** (par exemple un chèque signé) et le présenter plusieurs fois à la banque. Pour l'éviter les signatures numériques comportent souvent une **datation (date+heure)**. Cette datation de la signature est attachée au message et signée avec lui.

La banque stocke ces datations dans une base de données.

Protocoles de datation

On peut confier la datation des documents à un **service officiel de datation**. Bob veut dater une signature du message x. Il dispose d'une **fonction de hachage à sens unique, h**.

1. Bob calcule $z = h(x)$ et $y = sig_K(z)$
2. Bob soumet (z, y) au service de datation.
3. Le service de datation ajoute la date D et signe le triplet (z, y, D)

Bob peut aussi **dater un document, x, seul**.
Il collecte un certain nombre d' **informations publiques récentes** (qui n'auraient pas pu être prédites auparavant), notée pub. Par exemple les derniers résultats des courses hippiques et les cours actuels de la bourse.
On dispose aussi d'une **fonction de hachage publique à sens unique, h**.

1. Bob calcule $z = h(x)$
2. Bob calcule $z' = h(z||\text{pub})$
3. Bob calcule $y = sig_K(z')$
4. Bob publie (z, pub, y) dans le prochain quotidien

La date de la signature de Bob est comprise entre la date des informations pub et la date de parution du quotidien.

Attaque de l'homme du milieu

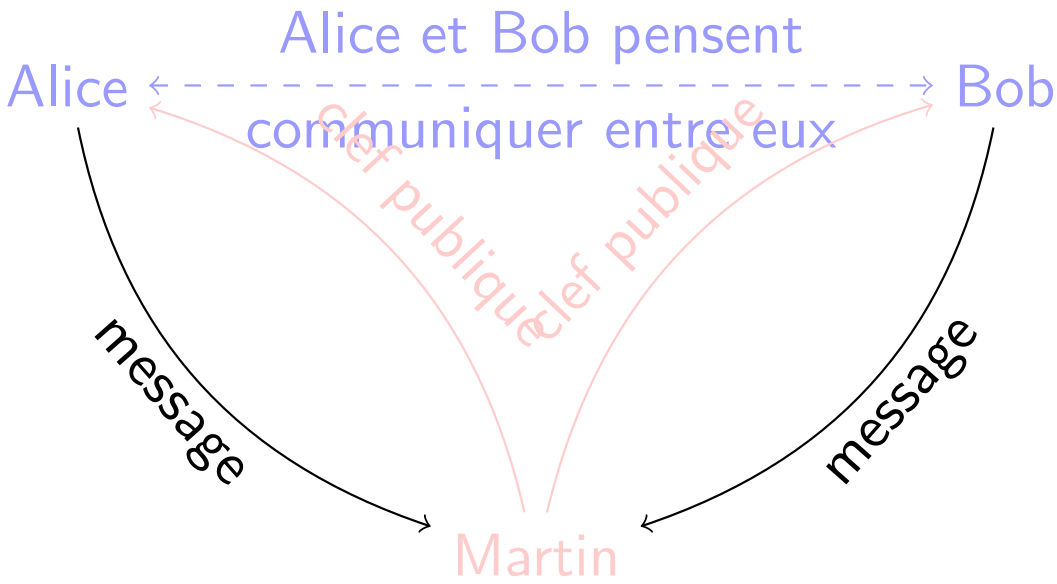
- Supposons qu'Alice et Bob veuillent communiquer via un cryptosystème **à clefs publiques**,
- mais que **Martin** parvienne à se faire passer pour Alice à Bob et pour Bob à Alice.
- Martin envoie sa clef publique à Alice et Bob de sorte que lorsque Alice crypte son message à l'attention de Bob, elle l'envoie à Martin qui le décode via sa clef secrète puis renvoie le message à Bob, à l'identique ou modifié et ainsi de suite pour toute les communications.

Attaque de l'homme du milieu

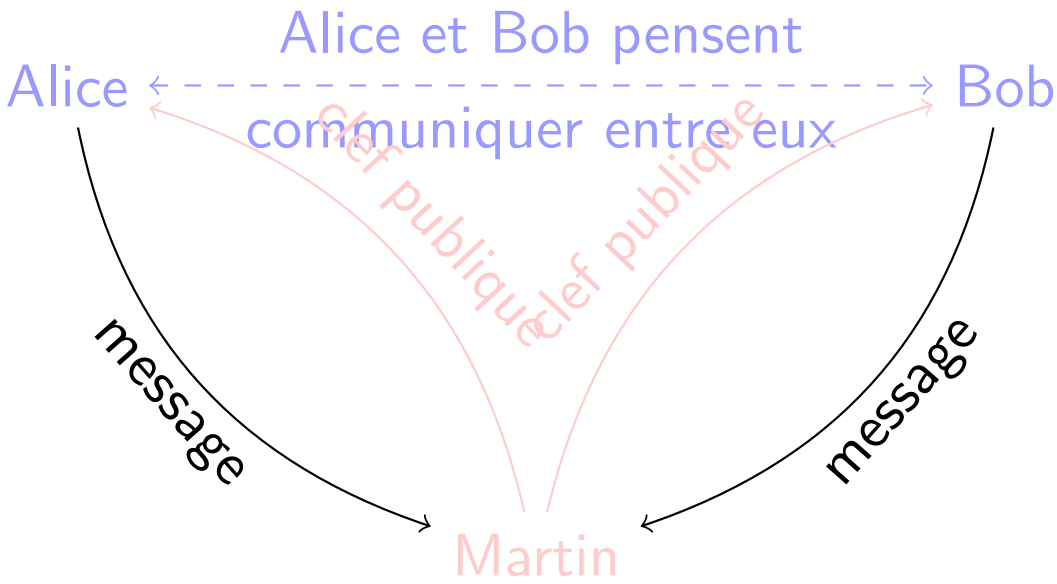
- Supposons qu'Alice et Bob veuillent communiquer via un cryptosystème **à clefs publiques**,
- mais que **Martin** parvienne à se faire passer pour Alice à Bob et pour Bob à Alice.
- Martin envoie sa clef publique à Alice et Bob de sorte que lorsque Alice crypte son message à l'attention de Bob, elle l'envoie à Martin qui le décode via sa clef secrète puis renvoie le message à Bob, à l'identique ou modifié et ainsi de suite pour toute les communications.

Attaque de l'homme du milieu

- Supposons qu'Alice et Bob veuillent communiquer via un cryptosystème **à clefs publiques**,
- mais que **Martin** parvienne à se faire passer pour Alice à Bob et pour Bob à Alice.
- Martin envoie sa clef publique à Alice et Bob de sorte que lorsque Alice crypte son message à l'attention de Bob, elle l'envoie à Martin qui le décode via sa clef secrète puis renvoie le message à Bob, à l'identique ou modifié et ainsi de suite pour toute les communications.



Alice et Bob ne se rendent pas compte qu'ils sont **espionnés** et les informations transmises ne sont même pas sûres.



déchiffre, modifie et transmet

Alice et Bob ne se rendent pas compte qu'ils sont **espionnés** et les informations transmises ne sont même pas sûres.

PKI (Public Key Infrastructure)

A l'instar des pièces d'identité dans la vie courante, on a recours aux certificats numériques validés par **une autorité de certification (PKI)** qui délivre des certificats numériques contenant divers types d'information:

- le nom du propriétaire du certificat, email, adresse...
- la date de validité,
- le système cryptographique préféré,
- la clé publique associée.

Ces renseignements sont **signés par l'autorité à l'aide de sa clef secrète** et tout un chacun peut en vérifier l'authenticité en utilisant la clef publique de l'autorité.
Le certificats les plus répandus sont **X.509 et PGP** lequel est plus souple via la notion **de parrainage**.

PKI (Public Key Infrastructure)

A l'instar des pièces d'identité dans la vie courante, on a recours aux certificats numériques validés par **une autorité de certification (PKI)** qui délivre des certificats numériques contenant divers types d'information:

- le nom du propriétaire du certificat, email, adresse...
- la date de validité,
- le système cryptographique préféré,
- la clé publique associée.

Ces renseignements sont **signés par l'autorité à l'aide de sa clef secrète** et tout un chacun peut en vérifier l'authenticité en utilisant la clef publique de l'autorité.

Le certificats les plus répandus sont **X.509 et PGP** lequel est plus souple via la notion **de parrainage**.

PKI (Public Key Infrastructure)

A l'instar des pièces d'identité dans la vie courante, on a recours aux certificats numériques validés par **une autorité de certification (PKI)** qui délivre des certificats numériques contenant divers types d'information:

- le nom du propriétaire du certificat, email, adresse...
- la date de validité,
- le système cryptographique préféré,
- la clé publique associée.

Ces renseignements sont **signés par l'autorité à l'aide de sa clef secrète** et tout un chacun peut en vérifier l'authenticité en utilisant la clef publique de l'autorité.
Le certificats les plus répandus sont **X.509 et PGP** lequel est plus souple via la notion **de parrainage**.

MAC: Message Authentification Codes

• Contrairement à la signature, **un MAC** est un mécanisme **à clé secrète** dont le but est la vérification par **un tiers ciblé**, de l'authentification de l'émetteur ou de l'intégrité des données.

• Considérons l'exemple naïf suivant: étant donnée une fonction pseudo-aléatoire F_k dépendant d'une clef k et un message de n blocs à authentifier on calcule:

$$MAC(m) = F_k(m_1) \oplus \cdots \oplus F_k(m_n).$$

• Notons alors que:

- $MAC(m)$ ne dépend pas de l'ordre des blocs,
- $(m|m, 0)$ est toujours valide,
- connaissant (m_1, t_1) et (m_2, t_2) alors $(m_1|m_2, t_1 \oplus t_2)$ est valide.

MAC: Message Authentification Codes

• Contrairement à la signature, **un MAC** est un mécanisme **à clé secrète** dont le but est la vérification par **un tiers ciblé**, de l'authentification de l'émetteur ou de l'intégrité des données.
• Considérons l'exemple naïf suivant: étant donnée une fonction pseudo-aléatoire F_k dépendant d'une clef k et un message de n blocs à authentifier on calcule:

$$MAC(m) = F_k(m_1) \oplus \cdots \oplus F_k(m_n).$$

• Notons alors que:
- $MAC(m)$ ne dépend pas de l'ordre des blocs,
- $(m|m, 0)$ est toujours valide,
- connaissant (m_1, t_1) et (m_2, t_2) alors $(m_1|m_2, t_1 \oplus t_2)$ est valide.

MAC: Message Authentification Codes

• Contrairement à la signature, **un MAC** est un mécanisme **à clé secrète** dont le but est la vérification par **un tiers ciblé**, de l'authentification de l'émetteur ou de l'intégrité des données.
• Considérons l'exemple naïf suivant: étant donnée une fonction pseudo-aléatoire F_k dépendant d'une clef k et un message de n blocs à authentifier on calcule:

$$MAC(m) = F_k(m_1) \oplus \cdots \oplus F_k(m_n).$$

• Notons alors que:

- $MAC(m)$ ne dépend pas de l'ordre des blocs,
- $(m|m, 0)$ est toujours valide,
- connaissant (m_1, t_1) et (m_2, t_2) alors $(m_1|m_2, t_1 \oplus t_2)$ est valide.

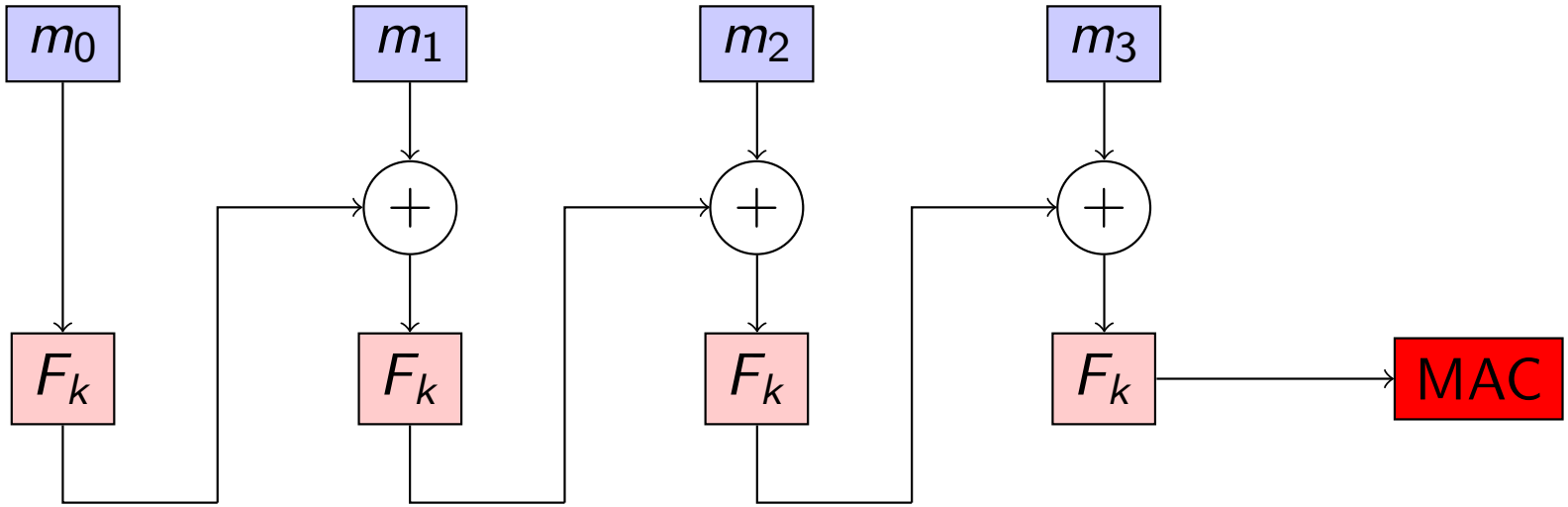
Or on voudrait pour le moins empêcher un attaquant:

- de produire un MAC valide (**forge existentielle**)

et bien entendu

- d'être capable de produire un MAC sur tout message (**forge universelle**).

CBC-MAC



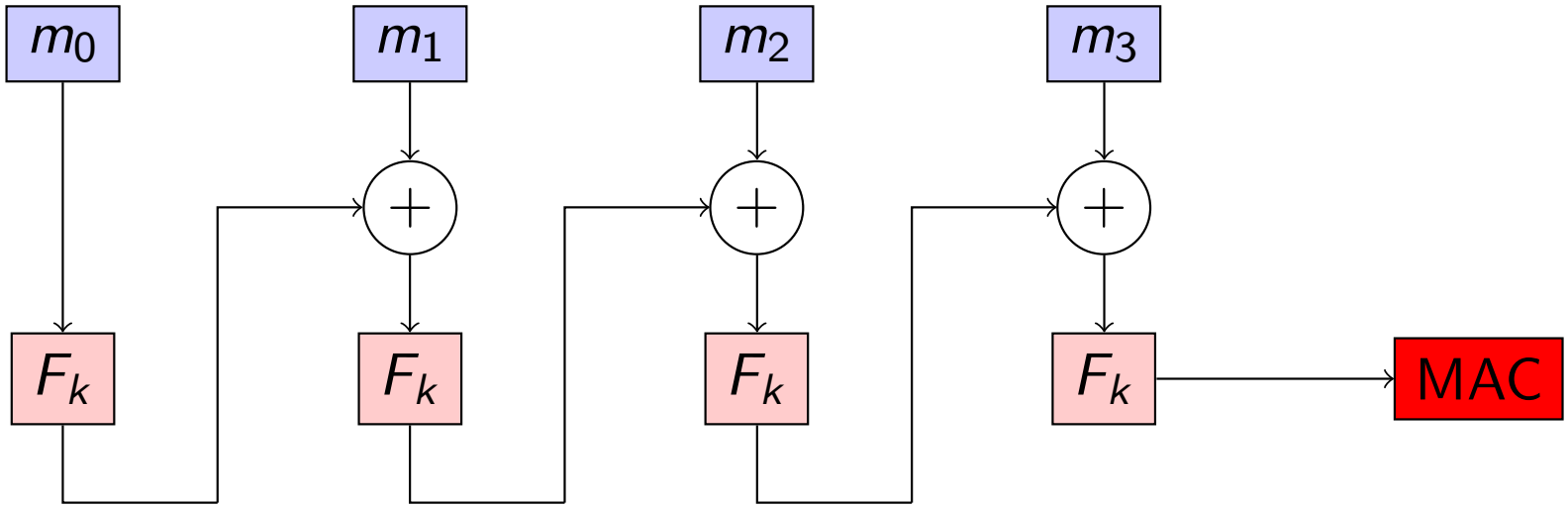
- En revanche il est vital de ne pas utiliser d'**IV** sinon pour

$$t = F(m_2 \oplus F(m_1 \oplus IV))$$

la valeur $IV \oplus m_1 \oplus m'_1$ peut être utilisée pour authentifier le message $m'_1|m_2$!

- En outre il faut absolument **fixer la longueur des messages** puisque si t est le MAC de $m = m_1|m_2|m_3$ et t' celui de $m' = m'_1|m'_2$ alors t' est aussi le MAC de $m_1|m_2|m_3|m'_1 \oplus t|m_2$.

CBC-MAC



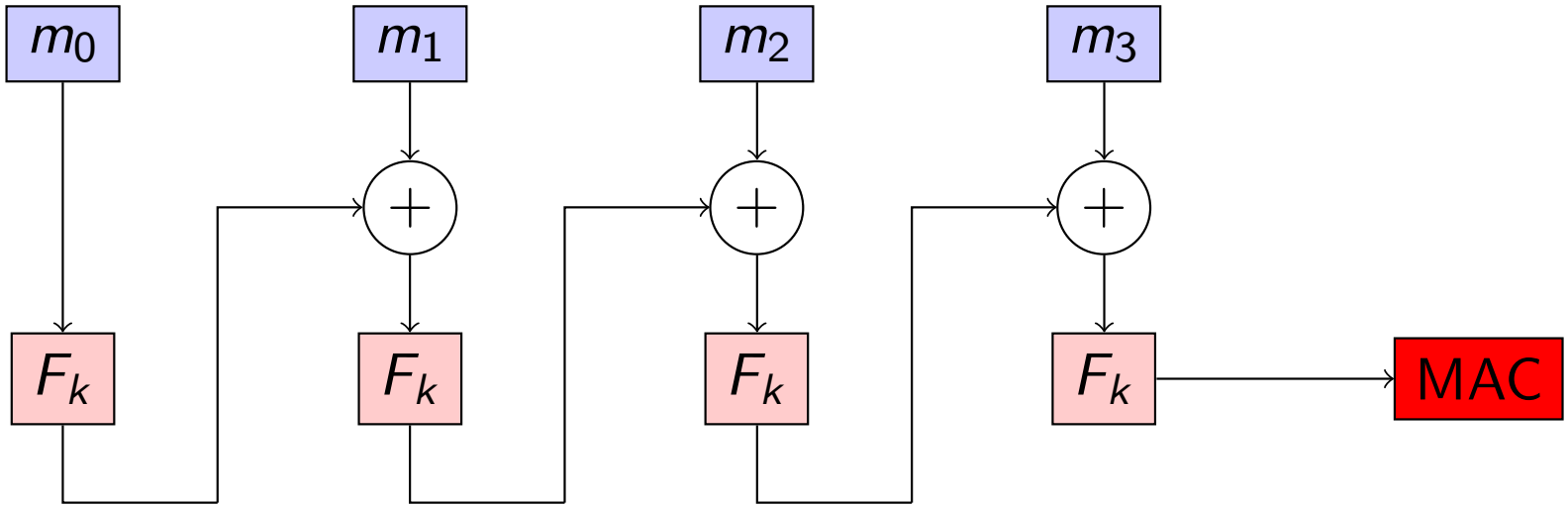
- En revanche il est vital de ne pas utiliser d'**IV** sinon pour

$$t = F(m_2 \oplus F(m_1 \oplus IV))$$

la valeur $IV \oplus m_1 \oplus m'_1$ peut être utilisée pour authentifier le message $m'_1|m_2$!

- En outre il faut absolument **fixer la longueur des messages** puisque si t est le MAC de $m = m_1|m_2|m_3$ et t' celui de $m' = m'_1|m'_2$ alors t' est aussi le MAC de $m_1|m_2|m_3|m'_1 \oplus t|m_2$.

CBC-MAC



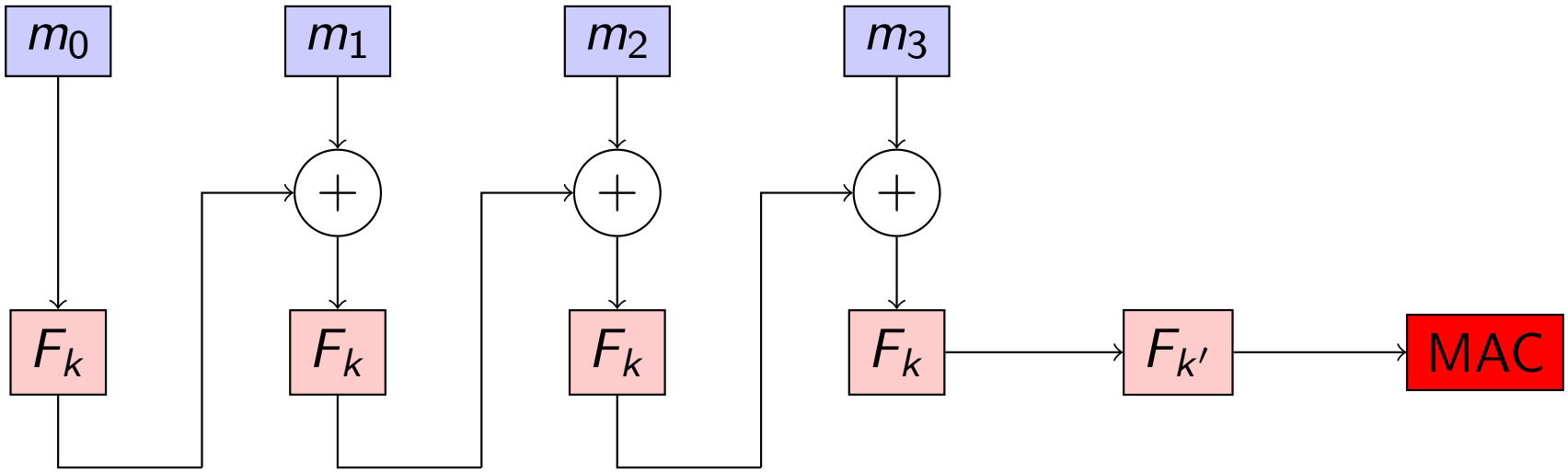
- En revanche il est vital de ne pas utiliser d'**IV** sinon pour

$$t = F(m_2 \oplus F(m_1 \oplus IV))$$

la valeur $IV \oplus m_1 \oplus m'_1$ peut être utilisée pour authentifier le message $m'_1|m_2$!

- En outre il faut absolument **fixer la longueur des messages** puisque si t est le MAC de $m = m_1|m_2|m_3$ et t' celui de $m' = m'_1|m'_2$ alors t' est aussi le MAC de $m_1|m_2|m_3|m'_1 \oplus t|m_2$.

EMAC=CBC-MAC amélioré



en utilisant une clef k' distincte de k. Cette version jouit alors d'une **preuve de sécurité**.

HMAC

HMAC est défini par

$$HMAC_k(m) = \mathcal{H}\Big(k \oplus opad | \mathcal{H}(k \oplus ipad | m)\Big)$$

où:

- ipad (inner padding)=0x36 répété sur la longueur de $\mathcal{H}$;
- opad (outer padding)=0x5C répété sur la longueur de $\mathcal{H}$.

Authentification serveur

Dans une authentification asymétrique, **le serveur met à l'épreuve le client** en lui faisant effectuer une opération que seul le client légitime est en mesure de mener à bien correctement.

Echange de clefs

Supposons qu'Alice et Bob souhaite s'échanger de nombreux documents en utilisant un cryptosystème symétrique (plus rapide) mais qu'ils ne puissent pas convenir d'une clef K par une voie sécurisée (au fond d'un café...).

- En utilisant un cryptosystème à clefs publiques, Bob peut utiliser la clef publique d'Alice pour envoyer la clef K mais ce n'est pas vraiment satisfaisant puisque Bob **seul** a décidé de la clef.
- Ce protocole est utilisé dans SSH v1 et SSL.

Echange de clefs

Supposons qu'Alice et Bob souhaite s'échanger de nombreux documents en utilisant un cryptosystème symétrique (plus rapide) mais qu'ils ne puissent pas convenir d'une clef K par une voie sécurisée (au fond d'un café...).

- En utilisant un cryptosystème à clefs publiques, Bob peut utiliser la clef publique d'Alice pour envoyer la clef K mais ce n'est pas vraiment satisfaisant puisque Bob **seul** a décidé de la clef.
- Ce protocole est utilisé dans SSH v1 et SSL.

Echange de clefs

Supposons qu'Alice et Bob souhaite s'échanger de nombreux documents en utilisant un cryptosystème symétrique (plus rapide) mais qu'ils ne puissent pas convenir d'une clef K par une voie sécurisée (au fond d'un café...).

- En utilisant un cryptosystème à clefs publiques, Bob peut utiliser la clef publique d'Alice pour envoyer la clef K mais ce n'est pas vraiment satisfaisant puisque Bob **seul** a décidé de la clef.
- Ce protocole est utilisé dans SSH v1 et SSL.

Diffie-Hellman

En 1976 Diffie et Hellman ont proposé le protocole suivant:

- ils choisissent selon un canal non sécurisé un nombre premier p et $g \in \mathrm{F}_p^\times$;
- Alice et Bob choisissent secrètement chacun de leur coté des entiers a, b et calculent respectivement g^a et g^b;
- Alice et Bob s'échangent, via le canal non sécurisé, les nombres g^a et g^b;
- la clef symétrique sera alors $K = g^{ab}$ qu'Alice et Bob sont capables de calculer sans connaître l'entier secret de l'autre.

- La sécurité de ce protocole repose sur la difficulté du log discret.
- Il demande **la simultanéité de la communication**, il est par exemple utilisé dans le protocole du bluetooth.

Diffie-Hellman

En 1976 Diffie et Hellman ont proposé le protocole suivant:

- ils choisissent selon un canal non sécurisé un nombre premier p et $g \in \mathrm{F}_p^\times$;
- Alice et Bob choisissent secrètement chacun de leur coté des entiers a, b et calculent respectivement g^a et g^b;
- Alice et Bob s'échangent, via le canal non sécurisé, les nombres g^a et g^b;
- la clef symétrique sera alors $K = g^{ab}$ qu'Alice et Bob sont capables de calculer sans connaître l'entier secret de l'autre.

- La sécurité de ce protocole repose sur la difficulté du log discret.
- Il demande **la simultanéité de la communication**, il est par exemple utilisé dans le protocole du bluetooth.

Diffie-Hellman

En 1976 Diffie et Hellman ont proposé le protocole suivant:

- ils choisissent selon un canal non sécurisé un nombre premier p et $g \in \mathrm{F}_p^\times$;
- Alice et Bob choisissent secrètement chacun de leur coté des entiers a, b et calculent respectivement g^a et g^b;
- Alice et Bob s'échangent, via le canal non sécurisé, les nombres g^a et g^b;
- la clef symétrique sera alors $K = g^{ab}$ qu'Alice et Bob sont capables de calculer sans connaître l'entier secret de l'autre.

• La sécurité de ce protocole repose sur la difficulté du log discret.
• Il demande **la simultanéité de la communication**, il est par exemple utilisé dans le protocole du bluetooth.

Bluetooth

L'appariement entre deux périphériques bluetooth commence par la désignation d'un maître M et d'un esclave E puis:

- le propriétaire des deux objets entre **un code PIN** sur chacun des objets (pas très pratique pour une oreillette!);
- M choisit un nombre n **au hasard** et l'envoie à E; chacun calcule $K_i = E22(N, PIN)$ où $E22$ est une fonction décrite par le protocole bluetooth;
- M et S s'envoie **un nombre aléatoire** a_M et a_E, calculent $c_M = K_i \oplus a_M$ et $c_E = K_i \oplus a_E$;
- M (resp. S) envoie c_E (resp. c_M) et vérifient la justesse du message reçu qui prouve que le périphérique possède bien **le même code PIN**.
- Ils calculent enfin à l'aide de a_M, a_E et de leurs adresses physiques crées à la sortie de l'usine, **une clef d'authentification commune** K_{auth}.

Bluetooth

L'appariement entre deux périphériques bluetooth commence par la désignation d'un maître M et d'un esclave E puis:

- le propriétaire des deux objets entre **un code PIN** sur chacun des objets (pas très pratique pour une oreillette!);
- M choisit un nombre n **au hasard** et l'envoie à E; chacun calcule $K_i = E22(N, PIN)$ où $E22$ est une fonction décrite par le protocole bluetooth;
- M et S s'envoie **un nombre aléatoire** a_M et a_E, calculent $c_M = K_i \oplus a_M$ et $c_E = K_i \oplus a_E$;
- M (resp. S) envoie c_E (resp. c_M) et vérifient la justesse du message reçu qui prouve que le périphérique possède bien **le même code PIN**.
- Ils calculent enfin à l'aide de a_M, a_E et de leurs adresses physiques crées à la sortie de l'usine, **une clef d'authentification commune** K_{auth}.

Bluetooth

L'appariement entre deux périphériques bluetooth commence par la désignation d'un maître M et d'un esclave E puis:

- le propriétaire des deux objets entre **un code PIN** sur chacun des objets (pas très pratique pour une oreillette!);
- M choisit un nombre n **au hasard** et l'envoie à E; chacun calcule $K_i = E22(N, PIN)$ où $E22$ est une fonction décrite par le protocole bluetooth;
- M et S s'envoie **un nombre aléatoire** a_M et a_E, calculent $c_M = K_i \oplus a_M$ et $c_E = K_i \oplus a_E$;
- M (resp. S) envoie c_E (resp. c_M) et vérifient la justesse du message reçu qui prouve que le périphérique possède bien **le même code PIN**.
- Ils calculent enfin à l'aide de a_M, a_E et de leurs adresses physiques crées à la sortie de l'usine, **une clef d'authentification commune** K_{auth}.

Bluetooth

L'appariement entre deux périphériques bluetooth commence par la désignation d'un maître M et d'un esclave E puis:

- le propriétaire des deux objets entre **un code PIN** sur chacun des objets (pas très pratique pour une oreillette!);
- M choisit un nombre n **au hasard** et l'envoie à E; chacun calcule $K_i = E22(N, PIN)$ où $E22$ est une fonction décrite par le protocole bluetooth;
- M et S s'envoie **un nombre aléatoire** a_M et a_E, calculent $c_M = K_i \oplus a_M$ et $c_E = K_i \oplus a_E$;
- M (resp. S) envoie c_E (resp. c_M) et vérifient la justesse du message reçu qui prouve que le périphérique possède bien **le même code PIN**.
- Ils calculent enfin à l'aide de a_M, a_E et de leurs adresses physiques crées à la sortie de l'usine, **une clef d'authentification commune** K_{auth}.

Bluetooth

L'appariement entre deux périphériques bluetooth commence par la désignation d'un maître M et d'un esclave E puis:

- le propriétaire des deux objets entre **un code PIN** sur chacun des objets (pas très pratique pour une oreillette!);
- M choisit un nombre n **au hasard** et l'envoie à E; chacun calcule $K_i = E22(N, PIN)$ où $E22$ est une fonction décrite par le protocole bluetooth;
- M et S s'envoie **un nombre aléatoire** a_M et a_E, calculent $c_M = K_i \oplus a_M$ et $c_E = K_i \oplus a_E$;
- M (resp. S) envoie c_E (resp. c_M) et vérifient la justesse du message reçu qui prouve que le périphérique possède bien **le même code PIN**.
- Ils calculent enfin à l'aide de a_M, a_E et de leurs adresses physiques crées à la sortie de l'usine, **une clef d'authentification commune** K_{auth}.

Bluetooth

L'appariement entre deux périphériques bluetooth commence par la désignation d'un maître M et d'un esclave E puis:

- le propriétaire des deux objets entre **un code PIN** sur chacun des objets (pas très pratique pour une oreillette!);
- M choisit un nombre n **au hasard** et l'envoie à E; chacun calcule $K_i = E22(N, PIN)$ où $E22$ est une fonction décrite par le protocole bluetooth;
- M et S s'envoie **un nombre aléatoire** a_M et a_E, calculent $c_M = K_i \oplus a_M$ et $c_E = K_i \oplus a_E$;
- M (resp. S) envoie c_E (resp. c_M) et vérifient la justesse du message reçu qui prouve que le périphérique possède bien **le même code PIN**.
- Ils calculent enfin à l'aide de a_M, a_E et de leurs adresses physiques crées à la sortie de l'usine, **une clef d'authentification commune** K_{auth}.

Bluetooth

• Lors d'une communication ultérieure, au lieu de répéter le processus précédent, le maître propose **un chalenge** à l'esclave en lui envoyant un nombre m au hasard.

• L'esclave calcule $R = E1(K_{auth}, m)$ et le retourne à M qui peut vérifier ainsi que S possède bien K_{auth}, où E_1 est un algorithme du protocole bluetooth.

Bluetooth

• Lors d'une communication ultérieure, au lieu de répéter le processus précédent, le maître propose **un chalenge** à l'esclave en lui envoyant un nombre m au hasard.
• L'esclave calcule $R = E1(K_{auth}, m)$ et le retourne à M qui peut vérifier ainsi que S possède bien K_{auth}, où E_1 est un algorithme du protocole bluetooth.

Diffie-Hellman signé

• Le protocole de Diffie-Hellman est sujet à l'attaque de l'homme du milieu.

• La première solution consiste **à signer** les messages: Alice envoie $(A, g^x, sign_A(g^x))$ et Bob répond par $(B, g^y, sign_B(g^y))$. En revanche **un rejeu** est possible puisqu'un couple $(g^x, sign_A(g^x))$ capturé permet de s'authentifier indéfiniment comme Alice même si on ne peut pas calculer la valeur de la clé.

• Une autre solution consiste à ce qu'Alice envoie (A, g^x), Bob répond par $(B, g^y, sign_B(g^y, g^x))$ et qu'enfin Alice confirme par $(A, sign_A(g^x, g^y)$ ce qui évite le rejeu.

Diffie-Hellman signé

• Le protocole de Diffie-Hellman est sujet à l'attaque de l'homme du milieu.

• La première solution consiste **à signer** les messages: Alice envoie $(A, g^x, sign_A(g^x))$ et Bob répond par $(B, g^y, sign_B(g^y))$. En revanche **un rejeu** est possible puisqu'un couple $(g^x, sign_A(g^x))$ capturé permet de s'authentifier indéfiniment comme Alice même si on ne peut pas calculer la valeur de la clé.

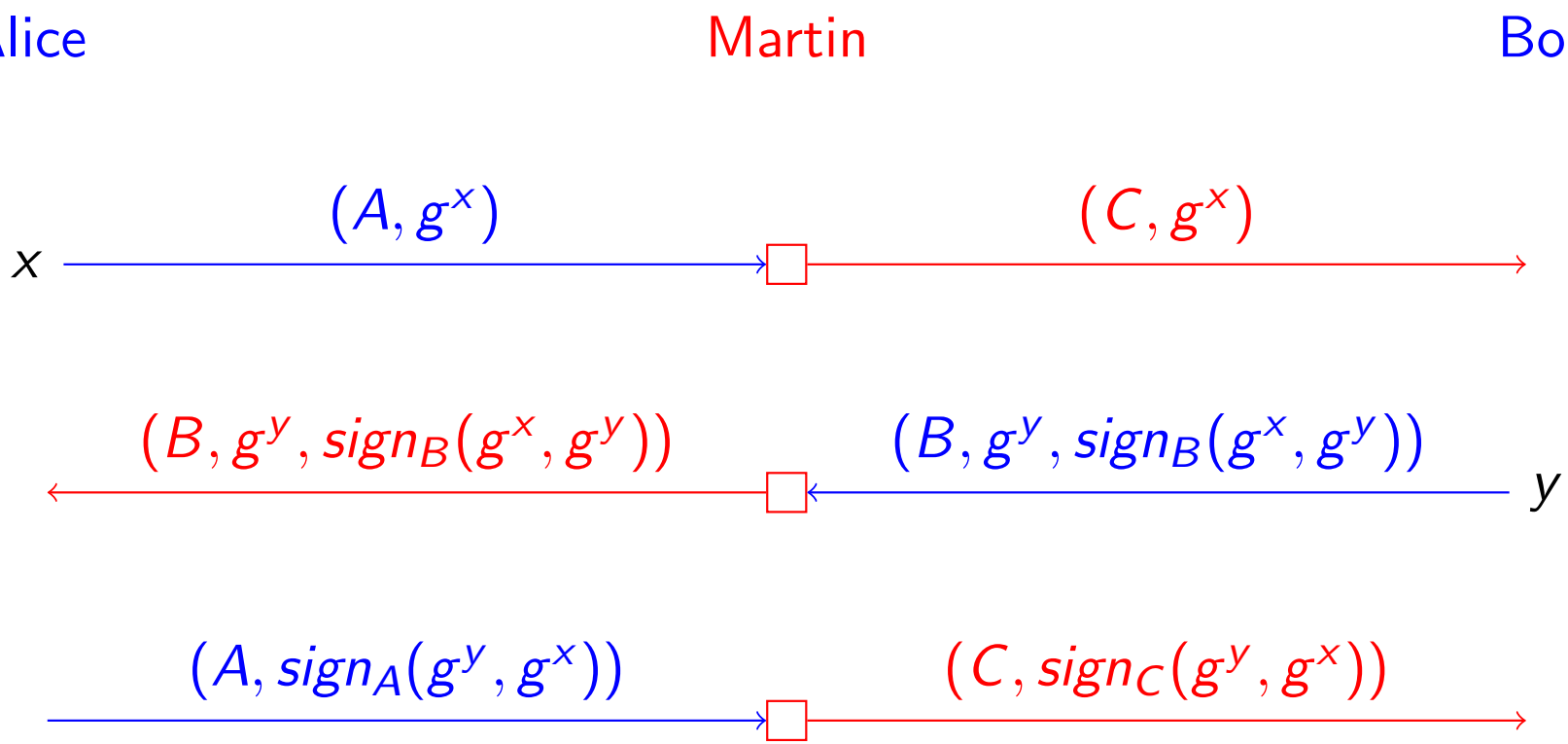
• Une autre solution consiste à ce qu'Alice envoie (A, g^x), Bob répond par $(B, g^y, sign_B(g^y, g^x))$ et qu'enfin Alice confirme par $(A, sign_A(g^x, g^y))$ ce qui évite le rejeu.

Diffie-Hellman signé

• Le protocole de Diffie-Hellman est sujet à l'attaque de l'homme du milieu.
• La première solution consiste **à signer** les messages: Alice envoie $(A, g^x, sign_A(g^x))$ et Bob répond par $(B, g^y, sign_B(g^y))$. En revanche **un rejeu** est possible puisqu'un couple $(g^x, sign_A(g^x))$ capturé permet de s'authentifier indéfiniment comme Alice même si on ne peut pas calculer la valeur de la clé.
• Une autre solution consiste à ce qu'Alice envoie (A, g^x), Bob répond par $(B, g^y, sign_B(g^y, g^x))$ et qu'enfin Alice confirme par $(A, sign_A(g^x, g^y)$ ce qui évite le rejeu.

Usurpation d'identité

Le protocole précédent est sujet à **l'usurpation d'identité**:



Usurpation d'identité

• Tous les messages envoyés par Alice sont vus par Bob comme venant de Martin même si ce dernier ne connait pas la clef g^{xy}, ce qui peut être gênant si Alice demande à sa banque (Bob) de créditer son compte...

• La solution dont la sécurité a été prouvée, consiste à ce qu'Alice envoie (A, g^x), que Bob renvoie $(B, g^y, sign_B(B, g^y, g^x, A))$ et qu'enfin Alice confirme par $(A, sign_A(A, g^x, g^y, B))$.

Usurpation d'identité

• Tous les messages envoyés par Alice sont vus par Bob comme venant de Martin même si ce dernier ne connait pas la clef g^{xy}, ce qui peut être gênant si Alice demande à sa banque (Bob) de créditer son compte...
• La solution dont la sécurité a été prouvée, consiste à ce qu'Alice envoie (A, g^x), que Bob renvoie $(B, g^y, sign_B(B, g^y, g^x, A))$ et qu'enfin Alice confirme par $(A, sign_A(A, g^x, g^y, B))$.

Fonction de hachage et mot de passe

Pour accéder à un ordinateur (ou à un distributeur de billet) on utilise souvent un mot de passe qui doit être reconnu par l'ordinateur. S'il est **stocké dans l'ordinateur il y a danger** pour la sécurité car un ordinateur peut être facilement infiltré (virus). Grâce à la fonction de hachage on peut pallier à ce problème:

On dispose d'une **fonction de hachage à sens unique**, **h**. L'ordinateur ne stocke pas le mot de passe **mp** d'Alice mais le résultat de la fonction de hachage à sens unique appliquée à **mp**.

1. Alice envoie son mot de passe **mp** à l'ordinateur.
2. L'ordinateur calcule **h(mp)**
3. L'ordinateur compare le résultat de ce calcul à celui qu'il a dans sa base de données.

Ce protocole permet de se défendre contre le vol de la base des données des mots de passe des utilisateurs.
Pour renforcer la sécurité et par exemple que deux personnes avec **le même mot de passe** s'en rende compte en constatant l'égalité de leur haché, la fonction de hachage dépend d'un **IV tiré au hasard et stocké en clair** par le serveur.

1. Alice envoie son mot de passe **mp** à l'ordinateur.
2. L'ordinateur calcule **h(mp)**
3. L'ordinateur compare le résultat de ce calcul à celui qu'il a dans sa base de données.

Ce protocole permet de se défendre contre le vol de la base des données des mots de passe des utilisateurs.

Pour renforcer la sécurité et par exemple que deux personnes avec **le même mot de passe** s'en rende compte en constatant l'égalité de leur haché, la fonction de hachage dépend d'un **IV tiré au hasard et stocké en clair** par le serveur.

1. Alice envoie son mot de passe **mp** à l'ordinateur.
2. L'ordinateur calcule **h(mp)**
3. L'ordinateur compare le résultat de ce calcul à celui qu'il a dans sa base de données.

Ce protocole permet de se défendre contre le vol de la base des données des mots de passe des utilisateurs.
Pour renforcer la sécurité et par exemple que deux personnes avec **le même mot de passe** s'en rende compte en constatant l'égalité de leur haché, la fonction de hachage dépend d'un **IV tiré au hasard et stocké en clair** par le serveur.

Preuve sans transfert de connaissance

Pile ou face par Téléphone

Alice et Bob viennent de divorcer et habitent dans des villes différentes et veulent par téléphone tirer à pile ou face pour savoir à qui va échoir la voiture, la machine à laver, les livres, etc...

Mais Bob hésite à dire à Alice **face** pour s'entendre dire **voilà je jette une pièce**,..., **elle retombe**,..., **pas de chance c'est pile**.

Ils veulent un protocole qui **évite toute tricherie**.

Pour cela ils se mettent d'accord sur une **fonction à sens unique f** de E dans F et on se donne une **partition** $\mathbf{E = X_0 \cup X_1}$ (exemple $E = \{0, 1, \ldots, n\}$, X_0 sera les entiers pairs de E et X_1 les entiers impairs de E.

On considère le protocole suivant

1. Alice choisit un $\mathbf{x} \in E$ **aléatoirement**, calcule $y = f(x)$ et communique y à Bob
2. Bob choisit son bit aléatoire $\mathbf{b} \in \{0, 1\}$ et l'annonce à Alice
3. Alice **déclare qui a gagné** suivant que $x \in X_b$ ou non: elle **prouve sa bonne foi** en révélant x
4. Bob se convainc qu'il n'y a **pas eu tricherie** en vérifiant que $y = f(x)$

Remarques

1. Si la fonction f est bien choisie la **donnée de $f(x)$ n'apprend rien à Bob sur x**.
2. Pour qu'Alice ne puisse pas **tricher**, il faut s'assurer qu'elle ne puise pas fabriquer deux entiers $x_0 \in X_0$ et $x_1 \in X_1$ tels que $\mathbf{f}(x_0) = f(x_1)$, par exemple prendre f bijective.
3. f doit être à sens unique en un **sens fort** pour la connaissance de $f(x)$ n'apprenne rien à Bob sur l' **appartenance de x à $\mathbf{X}_0$ ou $\mathbf{X}_1$**.
4. Ce protocole met en évidence la notion d'**engagement**: Alice **s'engage** sur x en publiant $f(x)$. Cela **ne révèle rien sur x** mais force Alice à **ne pas modifier son choix**.

Protocole de preuve sans transfert de connaissance

Pour accéder à un ordinateur on **donne son mot de passe** qui est reconnu par l'ordinateur et donc **stocké dedans**.
Si l'ordinateur auquel on se connecte est lointain, le mot de passe circule (crypté) sur des canaux qui **risquent d'être espionnés**.
Pour la sécurité il **faudrait le changer souvent**. Si la connexion est coupée accidentellement, il faut le redonner **sans avoir le temps d'en changer**.
Le clavier sur lequel on compose le mot de passe **peut être espionné**.
Il y a donc un intérêt à avoir un système de reconnaissance **sans envoi de mot de passe même crypté**.

Protocole d'identification de Fiat-Shamir

• L'utilisateur commence par construire deux grands nombres premiers p, q et un entier $1 \leq s \leq pq - 1$ pris au hasard. Il calcule alors $v = s^2 \mod n$. Le couple (n, v) est sa clé publique et s sa clé secrète.

• Le protocole consiste à persuader le serveur que l'on connait s **sans le divulguer** et repose sur le fait qu'extraire une racine carrée modulo $n = pq$ est aussi difficile que factoriser n.

- L'utilisateur choisit un entier r au hasard et envoie $x = r^2$ au serveur;
- le serveur choisit un bit $e \in \{0, 1\}$ au hasard et l'envoie à l'utilisateur;
- l'utilisateur renvoie $y = rs^e$ au serveur;
- le serveur vérifie que $y^2 = xv^e$.

Protocole d'identification de Fiat-Shamir

• L'utilisateur commence par construire deux grands nombres premiers p, q et un entier $1 \leq s \leq pq - 1$ pris au hasard. Il calcule alors $v = s^2 \mod n$. Le couple (n, v) est sa clé publique et s sa clé secrète.

• Le protocole consiste à persuader le serveur que l'on connait s **sans le divulguer** et repose sur le fait qu'extraire une racine carrée modulo $n = pq$ est aussi difficile que factoriser n.

- L'utilisateur choisit un entier r au hasard et envoie $x = r^2$ au serveur;
- le serveur choisit un bit $e \in \{0, 1\}$ au hasard et l'envoie à l'utilisateur;
- l'utilisateur renvoie $y = rs^e$ au serveur;
- le serveur vérifie que $y^2 = xv^e$.

Protocole d'identification de Fiat-Shamir

• L'utilisateur commence par construire deux grands nombres premiers p, q et un entier $1 \leq s \leq pq - 1$ pris au hasard. Il calcule alors $v = s^2 \mod n$. Le couple (n, v) est sa clé publique et s sa clé secrète.

• Le protocole consiste à persuader le serveur que l'on connait s **sans le divulguer** et repose sur le fait qu'extraire une racine carrée modulo $n = pq$ est aussi difficile que factoriser n.

- L'utilisateur choisit un entier r au hasard et envoie $x = r^2$ au serveur;
- le serveur choisit un bit $e \in \{0, 1\}$ au hasard et l'envoie à l'utilisateur;
- l'utilisateur renvoie $y = rs^e$ au serveur;
- le serveur vérifie que $y^2 = xv^e$.

L'espion ne connaissant pas s a alors deux choix:

- soit il ne triche pas à la première étape et envoie $x = r^2$ et répond toujours r si le serveur envoie $e = 0$ mais ne sait que répondre à $e = 1$;
- soit il envoie $x = r^2 v^{-1}$ à la première étape pariant sur $e = 1$ au retour du serveur.

En répétant ce protocole une vingtaine de fois, le serveur s'assurera de l'identité de l'utilisateur.

L'espion ne connaissant pas s a alors deux choix:

- soit il ne triche pas à la première étape et envoie $x = r^2$ et répond toujours r si le serveur envoie $e = 0$ mais ne sait que répondre à $e = 1$;
- soit il envoie $x = r^2 v^{-1}$ à la première étape pariant sur $e = 1$ au retour du serveur.

En répétant ce protocole une vingtaine de fois, le serveur s'assurera de l'identité de l'utilisateur.

Transfert inconscient

Alice dispose d'**un ensemble de m secrets** $\{s_1, s_2, \ldots, s_m\}$. Alice est **prête à en donner un** à Bob. Bob aimerait **en obtenir un, s_i**, mais il ne souhaite pas **révéler à Alice lequel** des secrets l'intéresse. Ce protocole

1. permet à Bob de **disposer de s_i**
2. ne lui donne aucune information **sur les autres secrets s_j, $j \neq i$**
3. ne permet pas à Alice de **savoir quel secret elle a livré à Bob**

Transfert inconscient

Supposons que chaque secret est un bit $s_i \in \{0,1\}$.

- Alice choisit p, q de grands nombres premiers et a qui n'est pas un résidu quadratique modulo $n = pq$ et de symbole de Jacobi égal à 1.
- Pour chaque $i \in \{1, \cdots, m\}$, Alice choisit aléatoirement $xi \neq 0 \mod n$ et calcule
$$y_i = x_i^2 a^{s_i} \mod n,$$
et donne à Bob les entiers n, a ainsi que la liste $(y_i)_{i=1,\cdots,m}$.
- Bob choisit aléatoirement $r \neq 0 \mod n$ et un bit aléatoire $b \in \{0,1\}$. Souhaitant connaître le secret s_i, Bob transmet à A l'entier
$$q = y_i r^2 a^b \mod n,$$
qui est sa question.
- Alice dit à Bob si q est un carré modulo n ou non auquel cas le bit secret s_i est égal à b et sinon c'est l'autre.

Le protocole manque de fiabilité (Alice peut tricher en utilisant un a qui soit un résidu quadratique, ou bien Bob peut envoyer $q = y_i y_j r^2 a^b$ et en déduire $s_i + s_j$). Dans les faits il doit être raffiné.

Partage de secret

En Russie au début des années 1990 pour utiliser l'arme nucléaire il fallait la participation de deux personnes parmi le président, le ministre de la défense et le chef des armées.

Dans un tel contexte on parle **de partage de secret à n participants avec seuil k** si le secret est partagé par n personnes de sorte que:

- k personnes prises parmi ces n participants peuvent reconstituer l'information secrète;
- $(k-1)$ ne le peuvent pas.

Partage de secret

En Russie au début des années 1990 pour utiliser l'arme nucléaire il fallait la participation de deux personnes parmi le président, le ministre de la défense et le chef des armées.
Dans un tel contexte on parle **de partage de secret à n participants avec seuil k** si le secret est partagé par n personnes de sorte que:

- k personnes prises parmi ces n participants peuvent reconstituer l'information secrète;
- $(k-1)$ ne le peuvent pas.

Partage de secret: version simple

• Supposons que le secret soit un nombre de 4 chiffres à partager entre n participants avec seuil n. On donne aux $n-1$ premiers un nombre de 4 chiffres aléatoires et au dernier le nombre tel que l'addition sans retenue de tous les nombres soit le secret.

• Pour un seuil égal à 2, supposons que le secret soit les coordonnées d'un point du plan que l'on écrit comme l'intersection de deux droites. Chacun des participants reçoit l'équation de la première droite ainsi qu'un point au hasard sur la deuxième.

Partage de secret: version simple

• Supposons que le secret soit un nombre de 4 chiffres à partager entre n participants avec seuil n. On donne aux $n-1$ premiers un nombre de 4 chiffres aléatoires et au dernier le nombre tel que l'addition sans retenue de tous les nombres soit le secret.

• Pour un seuil égal à 2, supposons que le secret soit les coordonnées d'un point du plan que l'on écrit comme l'intersection de deux droites. Chacun des participants reçoit l'équation de la première droite ainsi qu'un point au hasard sur la deuxième.

Partage de secret: protocole de Shamir

Soit S un secret à partager que l'on suppose être un nombre réel; il y a n participants et on voudrait un seuil égal à t.

- On se donne des réels choisis au hasard $a_1, \cdots, a_{t-1}$ et on considère le polynôme
$$P(X) = S + \sum_{i=1}^{t-1} a_i X^i.$$
- Chaque participant reçoit un nombre x_i et $y_i = P(x_i)$.

Par la théorie des polynômes interpolateurs de Lagrange, on sait reconstituer P dès que l'on dispose de t couples (x_i, y_i) et la connaissance de $t-1$ d'entre eux ne donne aucun renseignement sur S.

Partage de secret: protocole de Shamir

Soit S un secret à partager que l'on suppose être un nombre réel; il y a n participants et on voudrait un seuil égal à t.

- On se donne des réels choisis au hasard $a_1, \cdots, a_{t-1}$ et on considère le polynôme
$$P(X) = S + \sum_{i=1}^{t-1} a_i X^i.$$
- Chaque participant reçoit un nombre x_i et $y_i = P(x_i)$.

Par la théorie des polynômes interpolateurs de Lagrange, on sait reconstituer P dès que l'on dispose de t couples (x_i, y_i) et la connaissance de $t-1$ d'entre eux ne donne aucun renseignement sur S.

CB: protocole original (1990)

- **Authentification de la carte** introduite dans le terminal: sur la carte à puce se trouve en clair et crypté par la clef secrète du groupement des cartes bancaires, les données concernant le détenteur de la carte (nomn prénom, banque, numéro de compte...). En utilisant la clef publique, le terminal peut vérifier qu'il est en présence d'une vraie CB.
- Le client tape en clair **son code confidentiel**; s'il est correct la carte retourne OK au terminal.
- Lors d'un paiement supérieur à 100 euros, dans 20 % des cas, le terminal interroge un centre de contrôle à distance qui envoie à la carte un challenge via sa propre clef secrète inscrite dans une partie illisible de la carte à l'aide d'un chiffrement symétrique (DES puis triple DES et AES).

CB: paiement par internet

Le numéro de carte bancaire:

- le premier chiffre identifie le type de carte (3 pour American Express, 4 pour Visa, 5 pour Mastercard...);
- les trois suivants identifient la banque;
- les 7 suivants identifient le numéro de compte, chaque banque ayant son propre système d'identification.
- Le dernier chiffre est **la clé de Luhn** et se calcule comme suit: on multiplie par 2 les chiffres impairs de la carte et on fait la somme de tous les chiffres modulo 10; la clé est le complément à 10 du chiffre obtenu.

Un algorithme secret de chaque banque, calcule le cryptogramme de sécurité (non stocké sur la partie magnétique de la carte).

CB: paiement par internet

Le numéro de carte bancaire:

- le premier chiffre identifie le type de carte (3 pour American Express, 4 pour Visa, 5 pour Mastercard...);
- les trois suivants identifient la banque;
- les 7 suivants identifient le numéro de compte, chaque banque ayant son propre système d'identification.
- Le dernier chiffre est **la clé de Luhn** et se calcule comme suit: on multiplie par 2 les chiffres impairs de la carte et on fait la somme de tous les chiffres modulo 10; la clé est le complément à 10 du chiffre obtenu.

Un algorithme secret de chaque banque, calcule le cryptogramme de sécurité (non stocké sur la partie magnétique de la carte).

CB: l'affaire Humpich 1998

- **Faille logique**: comme le couple (Données,S(Données)) est inscrit en clair sur la puce, il est possible de le lire et de reproduire une carte valide. En outre il est possible de construire des **Yescard** qui répondent OK au terminal même si le code confidentiel est erroné.
- **Faille cryptographique**: la clef RSA utilisée pour le chiffrement asymétrique (inchangée depuis 1990!) ne comportait que 320 bits et en 1998 on savait factoriser des nombres de 512 bits. Humpich, avec un simple logiciel de factorisation, avait réussi à calculer la clef secrète du groupement des cartes bancaires.

A ce stade seule l'étape d'authentification en ligne par un challenge résistait à la méthode de Humpich mas celle-ci n'est utilisée que rarement en France.

CB: l'affaire Humpich 1998

- **Faille logique**: comme le couple (Données,S(Données)) est inscrit en clair sur la puce, il est possible de le lire et de reproduire une carte valide. En outre il est possible de construire des **Yescard** qui répondent OK au terminal même si le code confidentiel est erroné.
- **Faille cryptographique**: la clef RSA utilisée pour le chiffrement asymétrique (inchangée depuis 1990!) ne comportait que 320 bits et en 1998 on savait factoriser des nombres de 512 bits. Humpich, avec un simple logiciel de factorisation, avait réussi à calculer la clef secrète du groupement des cartes bancaires.

A ce stade seule l'étape d'authentification en ligne par un challenge résistait à la méthode de Humpich mas celle-ci n'est utilisée que rarement en France.

CB: nouveau protocole

- Augmentation de la taille de la clé secrète et changement de l'algorithme de chiffrement dans l'authentification en ligne.
- **DDA**: un couple de clés secrètes et publiques cryptées par la clef publique du groupement. Sur la partie lisible de la carte à puce on trouve donc (données,S(données), S(Pcarte)) et sur la partie illisible S(Scarte). Le terminal peut alors demander un challenge à la carte en utilisant ces clefs.
- **3D Secure**: pour finaliser une transaction sur internet, le client doit entrer un code d'authentification communiquer (par SMS) par sa banque.
- **e-carte bleue**: la banque fournit en temps réel un numéro de carte, une date d'expiration et un cryptogramme valides pour chaque transaction.

SSL: Secure Socket Layer

SSL est une sur-couche sécurisée des protocoles courants (HTTP, POP, IMAP...) introduit par Netscape pour sécuriser les transactions. La version 2 déployé en 1994 est désormais considérée comme peu sûre, on utilise plutôt la version 3 puis plus récemment TSL en version 1.2 depuis 2008.

L'établissement (**handshake**) d'une session protégée par TLS se déroule en trois phases:

- négociation d'un algorithme à utiliser,
- échange de clef de session,
- chiffrement et authentification du flux.

SSL: Secure Socket Layer

SSL est une sur-couche sécurisée des protocoles courants (HTTP, POP, IMAP...) introduit par Netscape pour sécuriser les transactions. La version 2 déployé en 1994 est désormais considérée comme peu sûre, on utilise plutôt la version 3 puis plus récemment TSL en version 1.2 depuis 2008. L'établissement (**handshake**) d'une session protégée par TLS se déroule en trois phases:

- négociation d'un algorithme à utiliser,
- échange de clef de session,
- chiffrement et authentification du flux.

SSH

- Le client envoie une requête au serveur qui lui envoie sa clé publique avec son certificat X.509.
- Le client stocke la clé et vérifie sa validité via le certificat.
- Pour vérifier qu'il n'y a pas d'usurpation d'identité, le client choisi un nombre n au hasard et l'envoie au serveur via sa clef publique et attend du serveur qu'il lui renvoie une réponse correcte au challenge via sa clef secrète.
- S'ensuit alors un échange de clé à la Diffie-Hellman pour AES (en général).

En ce qui concerne SSL, le handshake permet de choisir le protocole d'échange de clés (RSA, Diffie-Hellman fixe, éphémère, anonyme ou Fortezza), l'algorithme de compression/décompression et introduit des nombres aléatoires de session. L'intégrité des messages est assuré par HMAC.

SSH

- Le client envoie une requête au serveur qui lui envoie sa clé publique avec son certificat X.509.
- Le client stocke la clé et vérifie sa validité via le certificat.
- Pour vérifier qu'il n'y a pas d'usurpation d'identité, le client choisi un nombre n au hasard et l'envoie au serveur via sa clef publique et attend du serveur qu'il lui renvoie une réponse correcte au challenge via sa clef secrète.
- S'ensuit alors un échange de clé à la Diffie-Hellman pour AES (en général).

En ce qui concerne SSL, le handshake permet de choisir le protocole d'échange de clés (RSA, Diffie-Hellman fixe, éphémère, anonyme ou Fortezza), l'algorithme de compression/décompression et introduit des nombres aléatoires de session. L'intégrité des messages est assuré par HMAC.

PGP: Pretty Good Privacy

- **Compression**: le texte (email) est compressé afin de réduire le temps de transmission mais surtout pour améliorer la sécurité puisque la compression détruit les modèles du texte (fréquence des lettres, mots répétés...);
- **Chiffrement du message**: une clef de session est générée et le message chiffré par un algorithme symétrique (IDEA puis CAST);
- **Chiffrement de la clé de session**: la clé de session est chiffrée (RSA) en utilisant la clé publique du destinataire;
- **Envoi et réception des messages**: l'expéditeur envoie le couple message chiffré et la clé de session. Le destinataire récupère la clé de session en utilisant sa clé privée puis déchiffre le message.

Le succès du PGP est essentiellement dû à son caractère autonome (les clés publiques et privées sont générées aléatoirement à partir des mouvements de la souris de l'utilisateur) et la souplesse de ces certificats via la notion de parrainage.

PGP: Pretty Good Privacy

- **Compression**: le texte (email) est compressé afin de réduire le temps de transmission mais surtout pour améliorer la sécurité puisque la compression détruit les modèles du texte (fréquence des lettres, mots répétés...);
- **Chiffrement du message**: une clef de session est générée et le message chiffré par un algorithme symétrique (IDEA puis CAST);
- **Chiffrement de la clé de session**: la clé de session est chiffrée (RSA) en utilisant la clé publique du destinataire;
- **Envoi et réception des messages**: l'expéditeur envoie le couple message chiffré et la clé de session. Le destinataire récupère la clé de session en utilisant sa clé privée puis déchiffre le message.

Le succès du PGP est essentiellement dû à son caractère autonome (les clés publiques et privées sont générées aléatoirement à partir des mouvements de la souris de l'utilisateur) et la souplesse de ces certificats via la notion de parrainage.

Rappels d'arithmétique

7 Rappels mathématiques

- Théorie de la complexité
- Division euclidienne
- Congruences
- Corps finis
- Petit théorème de Fermat and co.
- Sur les nombres premiers
- Méthode de factorisation
- Polynômes
- Courbes elliptiques

Théorie de la complexité

Algorithmique

Donner des procédés de **calculs efficaces sur ordinateur** et d'évaluer les temps de calculs, les capacités de stockage nécessaires, etc..., en **fonction de la taille des données**

Théorie de la complexité

Classifier les problèmes algorithmiques en fonction de leur complexité.

- Algorthimes polynomiaux en fonction de la taille des données, **classe P**
- Algorthimes non-polynomiaux en fonction de la taille des données, **classe NP**

Une grande conjecture **P$\neq$ NP**

Théorie de la complexité

Algorithmique

Donner des procédés de **calculs efficaces sur ordinateur** et d'évaluer les temps de calculs, les capacités de stockage nécessaires, etc..., en **fonction de la taille des données**

Théorie de la complexité

Classifier les problèmes algorithmiques en fonction de leur complexité.

- Algorthimes polynomiaux en fonction de la taille des données, **classe P**
- Algorthimes non-polynomiaux en fonction de la taille des données, **classe NP**

Une grande conjecture **P**$\neq$ **NP**

Théorie de la complexité

Algorithmique

Donner des procédés de **calculs efficaces sur ordinateur** et d'évaluer les temps de calculs, les capacités de stockage nécessaires, etc..., en **fonction de la taille des données**

Théorie de la complexité

Classifier les problèmes algorithmiques en fonction de leur complexité.

- Algorthimes polynomiaux en fonction de la taille des données, **classe P**
- Algorthimes non-polynomiaux en fonction de la taille des données, **classe NP**

Une grande conjecture **P**$\neq$ **NP**

Théorie de la complexité

Algorithmique

Donner des procédés de **calculs efficaces sur ordinateur** et d'évaluer les temps de calculs, les capacités de stockage nécessaires, etc..., en **fonction de la taille des données**

Théorie de la complexité

Classifier les problèmes algorithmiques en fonction de leur complexité.

- Algorthimes polynomiaux en fonction de la taille des données, **classe P**
- Algorthimes non-polynomiaux en fonction de la taille des données, **classe NP**

Une grande conjecture **P**$\neq$ **NP**

Coût d'un algorithme

Complexité algorithmique

Coût d'un algorithme

Si on veut exécuter un algorithme sur une donnée x deux coûts sont à envisager

• le coût en temps $\mathcal{C}_{\text{temps}}(x)$, le nombre d'opérations effectuées pour obtenir le résultat final, plus formellement le nombre de **déplacement de la tête de lecture/écriture avant arrêt** de la machine de Turing modélisant l'ordinateur.

• le coût en espace, $\mathcal{C}_{\text{espace}}(x)$ est la taille de la mémoire utilisée, plus formellement le **nombre de case de la bande écrite au moins une fois**.

P $\neq$ NP

Complexité polynomiale et non polynomiale

Quel est le coût acceptable pour un problème donné. Il n'y a pas de réponse claire ni absolue. En 2000

- 2^{40} **opérations élémentaires** est accessible à un particulier s'il est patient.
- 2^{56} **opérations élémentaires** est accessible avec de gros moyens.
- 2^{80} **opérations élémentaires** est hors de portée de quiconque.

Temps d'exécution

Soit un algorithme dont le nombre d'opérations élementaires en fonction de la taille n de l'entrée est décrit par la fonction f.
On dispose d'un ordinateur **faisant 10^9 opérations élémentaires par secondes**.
Le temps pris par l'algorithme suivant les instances de f est:

f	$\log_2(n)$	$\log_2^3(n)$	n	$n\log_2(n)$	n^2	2^n
$n = 100$	$6,6 \cdot 10^{-9}$ s	$4,4 \cdot 10^{-7}$ s	10^{-7} s	$6,7 \cdot 10^{-7}$ s	10^{-5} s	$4 \cdot 10^{13}$ ans
$n = 10^5$	$1,7 \cdot 10^{-8}$ s	$5,2 \cdot 10^{-6}$ s	10^{-4} s	$1,7 \cdot 10^{-3}$ s	10 s	$\geq 10^{30086}$ ans
$n = 10^{10}$ s	$3,3 \cdot 10^{-8}$	$3,5 \cdot 10^{-5}$ s	10 s	330 s	30 siècles	$\geq 10^{3 \cdot 10^9}$ ans
$n = 10^{20}$	$6,6 \cdot 10^{-8}$ s	$2,3 \cdot 10^{-4}$ s	30 siècles	$\geq 10^5$ ans	$\geq 10^{23}$ ans	!!!

Opérations élémentaires

Algorithmes polynomiaux

(en fonction de la taille des données)

On décompose tout algorithme arithmétique en opérations élémentaires. L'opération élémentaire est l'addition de 3 chiffres en base 2 et le report de la retenue

$$\textbf{retenue} + \textbf{chiffre 1} + \textbf{chiffre 2} = \textbf{résultat} + \textbf{retenue}$$

Définition

On dit qu'un algorithme est **polynomial en fonction de la taille des données** *s'il peut être décomposé en un* **nombre d'opérations élémentaires majoré par une fonction polynomiale du nombre de chiffres des données.**

Fonctions polynomiales

Montrons que l'addition de deux nombres de tailles $\leq k$ est une **fonction polynomiale de k**:

	1	1	1	1				*retenue*
		1	1	1	1	0	0	0 *ligne 1*
+		0	0	1	1	1	1	0 *ligne 2*
	1	0	0	1	0	1	1	0 *résultat*

On suppose que le plus grand des entiers a k chiffres.

Opération élémentaire

1. Regarder dans la colonne i $(1 \leq i \leq k+1)$ les chiffres des lignes *lignes 1, 2 et retenue*
2. S'il y a 0 dans les *lignes 1, 2 et retenue* mettre 0 dans la ligne *résultat* et aller à la colonne $i+1$
3. S'il y a un 0 dans deux des *lignes 1 et 2 et retenue* on reporte 1 dans la ligne *résultat* et on passe à la colonne $i+1$
4. S'il y a un 1 dans deux des *lignes 1 et 2 et retenue* on reporte 0 dans la ligne *résultat* on met 1 dans la ligne *retenue* à la colonne $i+1$ et on passe à la colonne $i+1$
5. S'il y a un 1 dans les 3 *lignes 1 et 2 et retenue* on reporte 1 dans la ligne *résultat* on met 1 dans la ligne *retenue* à la colonne $i+1$ et on passe à la colonne $i+1$

On admet en première approximation que **le temps nécessaire pour faire k opérations élémentaires est proportionnel à k** avec une constante dépendant de l'ordinateur et de l'implantation de l'algorithme.

Exemples d'algorithmes polynomiaux

• **L'addition de deux entiers** $n \leq m$ de longueur $\leq k$ est polynomial en fonction de la taille des entrées
car (en première approximation) il faut

$$C \cdot k \stackrel{\text{déf}}{=} O(k) = 0(log_2(m)) \text{ opérations}$$

Ajouter deux entiers de 100 chiffres en base 10 avec un ordinateur faisant **10^9 opérations par secondes** nécessite $\sim 300 \cdot 10^{-9} \sim 3\mu$sec.

• **la multiplication de deux entiers** $n \leq m$ de taille k et ℓ nécessite

$$2 \cdot k\ell = O\big(\log_2^2(m)\big) \text{ opérations}$$

Multiplier deux entiers de 100 chiffres en base 10 nécessite
$\sim 90000 \cdot 10^{-9} \sim 10\mu$ sec.

- **Calcul du PGCD de deux entiers**, $n \geq m$ de taille $k \geq \ell$ nécessite

$$O(k^3) = O\big(\log_2^3(n)\big) \text{ opérations}$$

Trouver le PGCD de deux entiers de 100 chiffres en base 10 nécessite $\sim 27000000 \cdot 10^{-9} \sim 0,027$ sec.

- **Calculer b^m mod n** nécessite $O\big(\log_2^3(n)\big)$ opérations.

- **Décider si un nombre entier n de taille k est premier ou non**

nécessite

$$O(k^{6+\varepsilon}) = O\big(\log_2^{6+\varepsilon}(n)\big) \text{ opérations}$$

Tester si un entier de 30 chiffres en base 10 est premier nécessite avec l'algorithme polynomial à peu près une journée. Il y a des algorithmes pour tester la primalité d'un entier non polynômiaux et/ou non déterministes qui sont plus efficaces pour des nombres pas trop grands.

Exemples d'algorithmes non polynomiaux

- **Calcul de n**! $= 1 \cdot 2 \ldots (n-1) \cdot n$,

si l'on calcule brutalement il faut environ

$$C \cdot n^2 \log_2^2 n = O(n^2 \log_2^2 n) \text{ opérations}$$

Calculer $\left(10^{100}\right)!$ nécessite $\sim 10^{190}\mu$ sec $\sim 10^{180}$ années!!!.

- **Recherche d'un diviseur d'un entier n**,

les meilleurs algorithmes nécessitent

$$0\left(e^{C\sqrt{\log_2 n \log_2 \log_2 n}}\right) \text{ opérations}, \quad C \leq 2$$

Trouver un diviseur d'un nombre de 100 chiffres en base 10 nécessite $\sim e^{105} \cdot 10^{-9} \sim 410^{36}$ sec $\sim 10^{26}$siècles.

La division euclidienne

Sur les entiers naturels $\mathrm{N} = \{1, 2, 3 \dots\}$ on définit la **division euclidienne**: Si a et b sont deux entiers ($b \neq 0$) il existe un unique couple d'entiers q et r tel que

$$\mathbf{a = bq + r} \quad \mathbf{et} \begin{cases} \mathbf{ou\ bien} & \mathbf{r = 0} \\ \mathbf{ou\ bien} & \mathbf{1 \leq r \leq b - 1} \end{cases}$$

on dit que b est **un diviseur** de a s'il existe $q \in \mathrm{N}$ tel que $a = bq$.
Tout entier a au moins deux diviseurs triviaux 1 et lui-même. Mais il peut en avoir d'autres par exemple:

$$6 = 2 \cdot 3,\ 28 = 4 \cdot 7 = 2 \cdot 2 \cdot 7 = 2 \cdot 14.$$

La division euclidienne implique que $\mathbf{Z}$ **est un anneau principal**, i.e. **tout idéal de** $\mathbf{Z}$ **est principal**

Décomposition en facteurs premiers

Parmi les entiers naturel on distingue

- **1** qui est une unité
- les **nombres premiers** qui n'ont pas d'autres diviseurs que les triviaux i.e. 1 et lui même (e.g. 2,3,5,7,...,37,....).

Théorème (théorème fondamental de l'arithmétique)

Tout nombre entier différent de 1 *possède une unique décomposition en facteurs premiers à l'ordre près des termes*

Exemples: $10780 = 2^2 \cdot 5 \cdot 7^2$, $4200 = 2^3 \cdot 3 \cdot 5^2 \cdot 7 \cdot 11$

Plus Grand Commun Diviseur; PGCD

Deux nombres entiers a et b ont un **plus grand commun diviseur**, un PGCD, qui s'obtient

- soit à l'aide de leur **décomposition en facteurs premiers**

$$a = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_r^{\alpha_r}, \ b = p_1^{\beta_1} p_2^{\beta_2} \dots p_r^{\beta_r}, \text{ avec } \alpha_i, \beta_i \geq 0$$

alors

$$a \wedge b = p_1^{\inf\{\alpha_1,\beta_1\}} p_2^{\inf\{\alpha_2,\beta_2\}} \dots p_r^{\inf \alpha_r,\beta_r}$$

Cet algorithme n'est pas efficace car il nécessite la décomposition en facteurs premiers de a et b et les meilleurs algorithmes connus pour décomposer a et facteurs premiers sont en $O(e^{C\sqrt{\log_2(a)\log_2\log_2(a)}}$.

- soit par l'algorithme de la **division euclidienne** beaucoup plus efficace.

Exemple: Le PGCD de $4200 = 2^3 \cdot 3 \cdot 5^2 \cdot 7$ et de $10780 = 2^2 \cdot 5 \cdot 7^2 \cdot 11$ est

$$\mathrm{PGCD}(4200, 10780) = 2^2 \cdot 5 \cdot 7 = 140$$

Algorithme du PGCD

Soit $a \geq b$ deux entiers de taille k au plus.
Trouver le PGCD$(\mathbf{a}, \mathbf{b}) = \mathbf{a} \wedge \mathbf{b}$ **et évaluer le nombre d'opérations élémentaires**.

Écrire

$$a = bq_0 + r_1 \qquad b = q_1 r_1 + r_2$$
$$r_1 = q_2 r_2 + r_3 \qquad \ldots$$
$$\vdots \qquad r_{j-2} = q_{j-1} r_{j-1} + r_j$$
$$r_{j-1} = q_j r_j + r_{j+1} \qquad r_j = q_{j+1} r_{j+1}$$

avec $0 \leq r_1 < b$, $0 \leq r_2 < r_1$, $0 \leq r_3 < r_2$,..., $0 \leq r_j < r_{j+1}$.
On arrête l'algorithme dès qu'un **reste est nul** et alors

$$\mathbf{a \wedge b = r_{j+1}}$$

Cet algorithme nécessite $\mathbf{O}\left(\log_2^3(n)\right)$ opérations élémentaires.

Exemple: Trouver le PGDCD de 4200 et 10780:

$$10780 = 2 \times 4200 + 2380;\ \ 4200 = 2380 + 1820;$$
$$2380 = 1 \times 1820 + 560, 1820 = 3 \times 560 + 140,\ \ 560 = 4 \times 140$$
$$\mathrm{PGCD}(10780, 4200) = 140$$

L'algorithme de la division euclidienne donne aussi une version effective du théorème de Bézout, ainsi qu'une version effective du calcul du générateur de l'idéal $\langle a, b \rangle$ engendré par a et b

Algorithme d'Euclide

Entrée 2 entiers positifs **a** et **b** non tous deux nuls
Sortie $\mathbf{a} \wedge \mathbf{b}$
Tant que $b \neq 0$ faire

$$\mathbf{a} \wedge \mathbf{b} := \mathbf{b} \wedge (\mathbf{a} \mod \mathbf{b})$$

si $b = 0$ retourner **a** fin

Théorème de Bézout

Théorème (théorème de Bézout)

Si $d = pgcd(a, b)$, avec $a \geq b$. Il existe alors des entiers u et v tels que $au + bv = d$.

Proof.

on reprend l'algorithme du PGCD à partir de la fin. On écrit avec les notations précédentes

$$
\begin{aligned}
d = r_{j+1} &= r_{j-1} - q_j r_j = r_{j-1} u_{j-1}(1) - r_j v_j(q_j) \\
&= r_{j-1} - q_j \left(r_{j-2} - q_{j-1} r_{j-1} \right) \\
&= r_{j-1} \left(1 + q_j q_{j-1} \right) - q_j r_{j-2} \\
&= -u_{j-2}(q_j) r_{j-2} + r_{j-1} v_{j-1}(q_{j-1}, q_j) \\
&= \left(r_{j-3} - q_{j-2} r_{j-2} \right) \left(1 + q_j q_{j-1} \right) - q_j r_{j-2} \\
&= r_{j-3} \left(1 + q_j q_{j-1} \right) - r_{j-2} \left(q_{j-2} \left(1 + q_j q_{j-1} \right) + q_j \right)
\end{aligned}
$$

suite de la preuve

$$d = r_{j-3}u_{j-3}(q_j, q_{j-1}) - r_{j-2}v_{j-2}(q_{j-2}, q_{j-1}, q_j)$$

$$\vdots$$

$$= (-1)^{j+1}au_0(q_1, \ldots, q_j) + (-1)^j bv_0(q_0, \ldots, q_j)$$

On a une relation de récurrence facile sur u_j et v_j.

$$u_{j+1} = u_{j-1} - q_{j+1}u_j$$
$$v_{j+1} = v_{j-1} - q_{j+1}v_j$$

□

Exemple:

Relation de Bézout entre 10780 et 4200:

$$\begin{aligned}
140 &= 1820 - 3 \times 560 = 1820 - 3 \times (2380 - 1820) \\
&= 4 \times 1820 - 3 \times 2380 = 4 \times (4200 - 2380) - 3 \times 2380 \\
&= 4 \times 4200 - 7 \times 2380 = 4 \times 4200 - 7 \times (10780 - 2 \times 4200) \\
&= 18 \times 4200 - 7 \times 10780 = 75600 - 75460
\end{aligned}$$

Algorithme d'Euclide étendu

Entrée 2 entiers positifs a et b non tous nuls
Sortie 3 entiers u, v, d tels que $au + bv = a \wedge b = d$
Variables entières u_1, v_1, u_2, v_2, u_3, v_3, q, r

Si $a = 0$ retourner $(0, 1, b)$ et **fin**
$Au_1 + Bv_1 = a$ et $Au_2 + Bv_2 = b$ où A et B sont les valeurs initiales de a et b

Faire

$$(u_1, v_1) := (1, 0) \qquad (u_2, v_2) := (0, 1)$$

Tant que $\mathbf{b \neq 0}$ faire

$\mathbf{(q, r)}$:= **quotient et reste de la division euclidienne de a** par **b**

$\mathbf{(u_3, v_3) := (u_1 - qu_2, v_1 - qv_2)}$

$\mathbf{(u_1 v_1, a) := (u_2, v_2, b)}$

$\mathbf{(u_2, v_2, b) := (u_3, v_3, r)}$

si $\mathbf{b = 0}$ retourner $\mathbf{(u_1, v_1, a)}$.

Congruences

Définition

Soit a, b et m trois entiers. On dit que **a est congru à b modulo m** *et on écrit* $\mathbf{a \equiv b \mod m}$, *si la* **différence a − b est divisible par m**

On montre facilement que

Proposition

La relation de congruence est une **relation d'équivalence** *sur les entiers.*

Proposition

L'addition et la multiplication sont compatibles à la relation de conguence sur les entiers; autrement dit:

$$\Big(\big(a \pmod m\big) + \big(b \pmod m\big) \pmod m \Big) = \big(a + b \pmod m\big)$$

$$\Big(\big(a \pmod m\big) \times \big(b \pmod m\big) \pmod m \Big) = \big(a \times b \pmod m\big)$$

$\mathbb{Z}/m\mathbb{Z}$

On notera $\mathbb{Z}/m\mathbb{Z}$ les classes d'équivalences de $\mathbb{Z}$ modulo m munies des **deux lois $+$ et $\times$**.

Corollaire

Les éléments de $\mathbb{Z}/m\mathbb{Z}$ qui ont un **inverse multiplicatif** *sont ceux dont les représentants dans $\mathbb{Z}$ sont* **premiers à** *m*

Proof.

Facile par Bézout □

Théorème

Si p est premier $\mathbb{Z}/p\mathbb{Z}$ est un **corps fini** *à p éléments. On le note $\mathbb{F}_p$*

Proof.

facile d'après le corollaire précédent. □

Théorème des restes chinois

Théorème

Soient m_1, m_2,...m_r des entiers naturels **deux à deux premiers** *(i. e. $m_i \wedge m_j = 1$ si $i \neq j$) et soient a_1, a_2,...a_r des entiers relatifs. Le système de congruences*

$$\begin{array}{ll} \mathbf{x \equiv a_1 \mod m_1,} & \mathbf{x \equiv a_2 \mod m_2} \\ \vdots & \vdots \\ \ldots & \mathbf{x \equiv a_r \mod m_r} \end{array}$$

a toujours une solution *et deux solutions quelconques sont congrues modulo $M = m_1 m_2 \ldots m_r$*

Proof.

Unicité $\pmod{M}$ de la solution au système de congruences. Si x' et x'' sont deux solutions alors $x = x' - x'' \equiv 0 \pmod{m_i}$ pour $1 \leq i \leq r$ et donc $x \equiv 0 \pmod{M}$.

Soit $M_i = \frac{M}{m_i}$, clairement $M_i \wedge m_i = 1$ donc il existe un entier $\mathbf{N_i}$ **tel que** $\mathbf{M_i N_i \equiv 1 \pmod{m_i}}$ et de plus $m_i \mid M_j$ si $i \neq j$. Alors

$$\mathbf{x = \sum_{i=1}^{r} a_i M_i N_i \Longrightarrow x \equiv a_i \pmod{m_i}}, \quad 1 \leq i \leq r$$

Les corps finis

Pour chaque entier $r \geq 1$ il existe un **seul corps fini à $q = p^r$ éléments** à isomorphisme près, noté $\mathrm{F}_{p^r} = \mathrm{F}_q$. Ils s'obtiennent en considérant les quotients

$$\mathbf{F_q \simeq F_p[X]/P(x)F_p[X]}, \quad P(X) \in \mathrm{F}_p[X], \text{irréductible, de degré r}$$

Remarque: on utilise essentiellement que F_{2^r} est un F_2-espace vectoriel de dimension 2 muni d'un endomorphisme particulier correspondant à la multiplication par X.

Les corps finis

Pour chaque entier $r \geq 1$ il existe un **seul corps fini à $q = p^r$ éléments** à isomorphisme près, noté $\mathrm{F}_{p^r} = \mathrm{F}_q$. Ils s'obtiennent en considérant les quotients

$$\mathbf{F_q \simeq F_p[X]/P(x)F_p[X]}, \quad P(X) \in \mathrm{F}_p[X], \text{irréductible, de degré r}$$

Remarque: on utilise essentiellement que F_{2^r} est un F_2-espace vectoriel de dimension 2 muni d'un endomorphisme particulier correspondant à la multiplication par X.

Le petit théorème de Fermat

Théorème

Soit p un nombre premier. Tout entier a vérifie la congruence $\mathbf{a^p \equiv a \mod p}$, *et si* $\mathbf{a \wedge p = 1}$ *on a aussi* $\mathbf{a^{p-1} \equiv 1 \mod p}$

Exemple: $p = 7$, $a = 3$ alors

$$\begin{aligned}\left(3^6 \pmod 7\right) &= \left(3^{2+4} \pmod 7\right)\\ &= \left(3^2 \pmod 7\right)\left((3^2)^2 \pmod 7\right)\\ &= \left(2 \times 4 \pmod 7\right) = \left(1 \pmod 7\right)\end{aligned}$$

remarquer la manière de calculer une puissance: écrire l'exposant en base 2 et procéder par élévations au carré successives

Cas des corps finis

Théorème

Soient p est premier et $q = p^r$. Le groupe multiplicatif

$$\left(\mathrm{F}_q^*\right)^* = \{\textit{éléments inversibles de } \left(\mathrm{F}_q^*, \times\right)\}$$

est **cyclique**.

Remarque: ainsi pour tout $x \in \mathrm{F}_q^*$, on a $x^{q^r} = x$.

Théorème d'Euler

Définition

Si m est un entier positif on note **$\varphi(m)$ le nombre d'entiers $b < m$ et premiers à m**, *ou de manière équivalente $\varphi(m)$ est le nombre d'entiers $b < m$ inversibles modulo m*

Montrons que si $m = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_r^{\alpha_r}$ alors

$$\varphi(m) = p_1^{\alpha_1 - 1}(p_1 - 1) p_2^{\alpha_2 - 1}(p_2 - 1) \dots p_r^{\alpha_r - 1}(p_r - 1)$$

Il est clair que $\varphi(1) = 1$, si p est premier $\varphi(p) = p - 1$ et $\varphi(p^\alpha) = (p-1)p^{\alpha-1}$. Par les restes chinois si m est premier à n alors $\varphi(mn) = \varphi(m) \times \varphi(n)$.

Théorème d'Euler

Définition

Si m est un entier positif on note $\varphi(\mathbf{m})$ **le nombre d'entiers b $<$ m et premiers à m**, *ou de manière équivalente $\varphi(m)$ est le nombre d'entiers $b < m$ inversibles modulo m*

Montrons que si $\mathbf{m} = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_r^{\alpha_r}$ alors

$$\varphi(\mathbf{m}) = \mathbf{p_1^{\alpha_1 - 1}(p_1 - 1) p_2^{\alpha_2 - 1}(p_2 - 1) \dots p_r^{\alpha_r - 1}(p_r - 1)}$$

Il est clair que $\varphi(1) = 1$, si p est premier $\varphi(p) = p - 1$ et $\varphi(p^\alpha) = (p-1)p^{\alpha-1}$. Par les restes chinois si m est premier à n alors $\varphi(mn) = \varphi(m) \times \varphi(n)$.

Théorème d'Euler

Théorème (Théorème d'Euler)

Soit $m > 1$ un entier et soit a un entier premier à m, alors on a:
$\mathbf{a^{\varphi(m)} \equiv 1 \mod m}$

Proof.

Même preuve que pour le *petit théorème de Fermat* □

En particulier si p et q sont premiers, $\varphi(\mathbf{pq}) = \mathbf{(p-1)(q-1)}$ et

$$\mathbf{a^{(p-1)(q-1)} \equiv 1 \mod pq}$$

Carrés modulo p

Définition (Symbole de Legendre)

On pose si p est premier impair

$$\left(\frac{a}{p}\right) = \begin{cases} +1 & \text{si } a \text{ est premier à } p \text{ et est un carré} \pmod p \\ -1 & \text{si } a \text{ est premier à } p \text{ et n'est pas un carré} \pmod p \\ 0 & \text{si } a \text{ n'est pas premier à } p \end{cases}$$

Le symbole $\left(\frac{\mathbf{a}}{\mathbf{p}}\right)$ *est appelé le symbole de Legendre.*

Théorème d'Euler

C'est un caractère du groupe multiplicatif $(\mathrm{Z}/p\mathrm{Z})^*$ à valeur dans $\{\pm 1\} \cup \{0\}$ de période p donc

$$\left(\frac{a}{p}\right)\left(\frac{b}{p}\right) = \left(\frac{ab}{p}\right)$$

$$\left(\frac{a}{p}\right) = \left(\frac{a+p}{p}\right)$$

Théorème (Euler)

Si p est premier impair et a premier à p alors

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \mod p, \quad \left(\frac{-1}{p}\right) = (-1)^{\frac{p-1}{8}}, \quad \left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}}$$

Loi de réciprocité quadratique

Théorème (Loi de réciprocité quadratique)

Soit p et q des premiers impairs alors

$$\left(\frac{q}{p}\right)\left(\frac{p}{q}\right) = (-1)^{\frac{p-1}{2}\cdot\frac{q-1}{2}}$$

Ce théorème (un des plus célèbres de l'arithmétique) est du à C. F. Gauss, il se généralise au symbole de Jacobi ci-dessous. La preuve est délicate.

Symbole de Jacobi

Définition (Symbole de Jacobi)

On pose si $m = \prod_{p_i \mid m} p_i^{\alpha_i}$ *est impair et* $n \in \mathbb{Z}$

$$\left(\frac{n}{m}\right) = \begin{cases} \prod_{p_i \mid m} \left(\frac{n}{p_i}\right)^{\alpha_i} & \text{si } m \wedge n = 1 \\ 1 & \text{si } m = 1 \\ 0 & \text{si } m \wedge n \neq 1 \end{cases}$$

Le symbole de Jacobi possède les propriétés suivantes

$$n_1 \equiv n_2 \mod m \Longrightarrow \left(\frac{n_1}{m}\right) = \left(\frac{n_2}{m}\right)$$

Si m est un entier impair

$$\left(\frac{2}{m}\right) = \begin{cases} +1 & \text{si } m \equiv \pm 1 \pmod 8 \\ -1 & \text{si } m \equiv \pm 3 \pmod 8 \end{cases}$$

Si m est un entier impair

$$\left(\frac{n_1 n_2}{m}\right) = \left(\frac{n_1}{m}\right)\left(\frac{n_2}{m}\right)$$

Loi de réciprocité quadratique: Si m et n entiers impairs:

$$\left(\frac{\mathbf{n}}{\mathbf{m}}\right) = \begin{cases} -\left(\frac{\mathbf{m}}{\mathbf{n}}\right) & \text{si } \mathbf{m} \equiv \mathbf{n} \equiv \mathbf{3} \pmod{\mathbf{4}} \\ +\left(\frac{\mathbf{n}}{\mathbf{m}}\right) & \text{sinon} \end{cases}$$

Tests de primalité

Pour tester si n est premier on pourrait essayer de le diviser par tout les entiers $< \sqrt{n}$, c'est la méthode du **crible d'Eratosthenes**. C'est impraticable dès que n est grand car $\sqrt{n}$ n'est pas majoré uniformément par un polynôme en $\log n$.

Le temps de calcul devient vite prohibitif. On a vu en effet qu'une division de n par un entier inférieur demande $O(\log^3 n)$ opération élémentaire donc cette méthode nécessitera

$$\mathbf{O}\Big(\sum_{i=1}^{\sqrt{n}} \log^3 n\Big) = O(\sqrt{n}\log^3 n) \text{ opérations}$$

Parmi les tests de primalités certains parmi les plus utilisés sont basés sur le petit théorème de Fermat, ou des analogues comme les résidus quadratiques.

Il existe maintenant (2002) des tests de primalité, basés sur le petit théorème de Fermat, qui sont **polynomiaux en temps** (actuellement en $\mathbf{0}(\log^{12}(n))$ et même $0(\log^{6}(n)))$. Mais pour l'instant ils sont moins performants que des **tests probabilistes** comme ceux décrits ci-dessous.

Tests de primalités probabilistes

Pour tester si n est premier on procède en pratique de la manière suivante.

1. Vérifier que **n n'est pas divisible par des petits facteurs premiers**.
2. Puis pour des **a choisis au hasard** et tels que $1 \leq a \leq n$ et $a \wedge n = 1$ calculer $\mathbf{a^{n-1} \mod n}$.
3. Si $\mathbf{a^{n-1} \not\equiv 1 \mod n}$ **pour au moins un a** alors **n n'est pas premier**.
4. Sinon on ne peut pas conclure (nombres de Carmichaël, par exemple 561). On dit que n est pseudo premier.

Pour avoir un résultat sûr on peut utiliser le **test de Lucas**

Test de Lucas

Théorème (Lucas)

Si a est premier à n et si pour tout diviseur premier p de $n-1$ on a $a^{\frac{n-1}{p}} \not\equiv 1 \mod n$ alors n est premier

La preuve de ce résultat est immédiate.

On peut aussi utiliser les propriétés du symbole de **Legendre et de Jacobi**

Pour tester si p est premier, on choisit **b premier à** n avec $1 \leq b \leq n-1$ et on calcule $\mathbf{b}^{\frac{\mathbf{p-1}}{\mathbf{2}}} \pmod{\mathbf{p}}$. Si p est premier doit avoir $b^{\frac{p-1}{2}} \equiv \left(\frac{b}{p}\right) \pmod{p}$ pour tout b. Si **p n'est pas premier** alors on a $\mathbf{b}^{\frac{\mathbf{p-1}}{\mathbf{2}}} \not\equiv \left(\frac{\mathbf{b}}{\mathbf{p}}\right) \pmod{\mathbf{p}}$ pour au moins 50% des b premiers à n.

Test de primalité de Solovay-Strassen

1. Tirer un entier aléatoire a, $1 \leq a \leq n-1$
2. Si $\left(\frac{a}{n}\right) \equiv a^{\frac{n-1}{2}} \pmod{n}$ alors réponds **n est premier**
   sinon réponds **n est décomposable**

La **loi de réciprocité quadratique** rend ce test probabiliste très efficace. Elle évite en effet pour calculer le symbole de Jacobi d'avoir à factoriser a et n.

Exemple

Testons la primalité de 547. On choisit $a = 5$.

On calcule d'une part

$$\left(\frac{5}{547}\right) = \left(\frac{547}{5}\right) = \left(\frac{2}{5}\right) = (-1)^{\frac{5^2-1}{8}} = -1$$

d'autre part

$$5^{\frac{547-1}{2}} \mod 547 \equiv 5^{1+2^4+2^8} \mod 547 \equiv 5 \cdot 113 \cdot 395 \mod 547$$
$$\equiv -1 \mod 547$$

Donc 547 a une chance d'être premier. Pour se conforter on recommence avec 7 au lieu de 5, etc.

Actuellement on est capable de prouver la primalité d'un nombre sans symétrie particulière de **2000 à 3000 chiffres en base 10** en un mois de CPU sur une station de travail. On prouve aussi la primalité de nombres tests comme les nombres de Mersenne $(2^p - 1)$ ayant plusieurs millions de chiffres.

Construction de grands nombres premiers

Pour construire de grands nombres premiers, on couple les résultats précédents avec des théorèmes sur la répartition des nombres premiers.

On sait par exemple que le nombre de nombres premiers inférieurs à x, $\pi(x) \sim \frac{x}{\ln x}$.

On en déduit en particulier qu'en moyenne l'écart entre deux nombres premiers consécutifs autour de n est $\ln n$, de sorte que l'algorithme consistant à tester si n puis $n+2$ etc, sont premiers doit en moyenne aboutir en un temps polynomial en la taille de n.

Si on veut un résultat exact, signalons le théorème de Bertrand qui montre qu'il y a toujours un nombre premier entre n et $2n$.

Construction de grands nombres premiers

Pour construire de grands nombres premiers, on couple les résultats précédents avec des théorèmes sur la répartition des nombres premiers.

On sait par exemple que le nombre de nombres premiers inférieurs à x, $\pi(x) \sim \frac{x}{\ln x}$.
On en déduit en particulier qu'en moyenne l'écart entre deux nombres premiers consécutifs autour de n est $\ln n$, de sorte que l'algorithme consistant à tester si n puis $n+2$ etc, sont premiers doit en moyenne aboutir en un temps polynomial en la taille de n.
Si on veut un résultat exact, signalons le théorème de Bertrand qui montre qu'il y a toujours un nombre premier entre n et $2n$.

Construction de grands nombres premiers

Pour construire de grands nombres premiers, on couple les résultats précédents avec des théorèmes sur la répartition des nombres premiers.

On sait par exemple que le nombre de nombres premiers inférieurs à x, $\pi(x) \sim \frac{x}{\ln x}$.
On en déduit en particulier qu'en moyenne l'écart entre deux nombres premiers consécutifs autour de n est $\ln n$, de sorte que l'algorithme consistant à tester si n puis $n+2$ etc, sont premiers doit en moyenne aboutir en un temps polynomial en la taille de n.
Si on veut un résultat exact, signalons le théorème de Bertrand qui montre qu'il y a toujours un nombre premier entre n et $2n$.

Une méthode de factorisation

Principe d'un algorithme très utilisé pour factoriser de grand nombres entiers: **le crible quadratique**.
Il est fondé sur la proposition suivante

Proposition (Fermat)

Soit n un entier positif impair. Il y a une **bijection** *entre les* **décompositions de n sous la forme $n = ab$** *avec $a \geq b \geq 0$ entiers et les* **représentations de n sous la forme $n = t^2 - s^2$**, *où s et t sont des entiers positifs ou nuls. la bijection est donnée par les équations*

$$t = \frac{a+b}{2}, \quad s = \frac{a-b}{2}, \quad a = t+s, \quad b = t-s$$

Exemple

Soit à factoriser $\mathbf{n = 23360947609}$

$$n = 23360947609 \Longrightarrow E(\sqrt{n}) = 152843$$

on pose $\mathbf{t = 152843}$. On teste s'il existe i pas trop grand tel que

$$\mathbf{(t+i)^2 - n = (152843+i)^2 - 23360947609}$$

est un carré.
On trouve que **pour i=2** on a

$$152845^2 - 23360947609 = 804^2$$

et donc $\mathbf{n = pq}$ avec $\begin{cases} \mathbf{p = 152845 + 804 = 152649} \\ \mathbf{q = 152845 - 804 = 152041} \end{cases}$. On vérifie aisément que p et q sont premiers.

Estimation du temps d'exécution

La factorisation des entiers reste un problème difficile. En 1994 il a fallu **le travail combiné de 600 personnes pendant 8 mois pour factoriser un nombre de 129 chiffres** proposé 17 ans plus tôt et pour cela il fallu développer dans l'intervalle de nouvelles méthodes de factorisation.

En 1999 il a fallu 8 mois de travail pour factoriser un nombre de 155 chiffres. Il a fallu en plus de l'augmentation de puissance des ordinateurs développer et perfectionner de nouvelles méthodes de factorisation.

Pour des généralisations des codes RSA ainsi que pour des procédés de factorisation et des tests de primalité, on utilise l'**arithmétique sur les courbes elliptiques** qui est une sorte de généralisation de l'arithmétique modulaire .

Anneau des polynôme sur un corps

Soit K un corps on définit l'anneau des polynômes formels en une variable $\mathrm{K}[X]$. Sur $\mathrm{K}[X]$ on a une **division euclidienne**:

Définition

Si $A(X)$ et $B(X)$ sont deux polynômes de $\mathrm{K}[X]$ ($B \neq 0$) il existe un unique couple de polynômes $Q(X)$ et $R(X)$ tel que

$$\mathbf{A(X) = B(X)Q(X) + R(X)} \text{ et } \begin{cases} \textbf{ou bien} & \mathbf{R = 0} \\ \textbf{ou bien} & \mathbf{0 \leq \deg R \leq \deg B - 1} \end{cases}$$

On dit que $B(X)$ est **un diviseur** de $A(X)$ s'il existe $Q(X) \in \mathrm{K}[X]$ tel que $A(X) = B(X)Q(X)$.
Tout polynôme $A(X)$ a au moins comme diviseurs triviaux les éléments de K^* et lui-même. Mais il peut en avoir d'autres par exemple:

$$X^2 - 1 = (X - 1)(X + 1)$$

$X^2 - 1$ a donc comme diviseurs les constantes non nulles ($=\mathrm{K}^*$), les polynômes $X - 1$ et $X + 1$ à une constante multiplicative non nulle près et lui-même à une constante multiplicative non nulle près.

Un polynôme est appelé un **polynôme trivial ou une unité** s'il est réduit à un élément de K.

L'arithmétique de $\mathrm{K}[X]$ est tout à fait parallèle à celle de Z.

- Les deux sont des anneaux munis d'une **division euclidienne**
- on définit sur $\mathrm{K}[X]$ une notion de **factorisation en facteurs premiers**,
- on définit le **PGCD de deux polynômes**
- Il y a sur $\mathrm{K}[X]$ une relation de Bezout, un **algorithme d'Euclide étendu** pour calculer le PGCD

etc..

Définition

On dit qu'un polynôme $A(X) \in \mathrm{K}[X]$ est **irréductible sur** K *s'il n'existe pas de couple $(B(X), C(X)) \in\in \mathrm{K}[X]^2$ avec B et C non triviaux tels que $A = BC$.*

Proposition

Tout polynôme de $\mathrm{K}[X]$ est décomposable de manière unique (à l'ordre près) en un produit de polynômes unitaires et d'un polynôme trivial

Exemple: Dans $\mathrm{Q}[X]$ le polynôme $X^2 + 1$ est irréductible, par contre dans $\mathrm{F}_2[X]$ il ne l'est car sur F_2 on a $X^2 + 1 = (X + 1)^2$.

PGCD

Définition

Le PGCD de deux polynômes $A(X)$ et $B(X)$ de $\mathrm{K}[X]$ est le polynôme unitaire de plus grand degré $P(X)$ qui divise à la fois $A(X)$ et $B(X)$. On note $PGCD(A, B) = A \wedge B$

On a la proposition

Proposition (Bézout)

Soient $A(X), B(X) \in \mathrm{K}[X]$, alors il existe des polynômes $U(X), V(X) \in \mathrm{K}[X]$ tels que

$$A(X)U(X) + B(X)V(X) = A(X) \wedge B(X)$$

Le calcul pratique se fait par l'algorithme d'Euclide ou d'Euclide étendu pour les polynômes qui est formellement le même que pour les entiers

Définition

Si $P(X) \in \mathrm{K}[X]$ et si $A(X)$, $B(X)$ appartiennent à $\mathrm{K}[X]$ on dit que A et B sont **congrus modulo** P *s'il existe $Q \in \mathrm{K}[X]$ tel que $A - B = PQ$. On écrira $A \equiv B \mod P$.*

On vérifie aisément que c'est une **relation d'équivalence** sur l'anneau des polynômes qui est compatible aux opérations sur les polynômes. L'anneau des polynômes modulo cette relation d'équivalence, noté $\mathrm{K}[X]/P(X)$, est un anneau et c'est un corps si P est irréductible.

Si K est un corps fini et si P est un **polynôme irréductible** sur K alors $\mathrm{K}[X]/P(X)$ est un **corps fini** qui contient K. Sur tout corps fini il y a des polynômes irréductibles de degré n pour tout n.

Courbes elliptiques

Les courbes elliptiques définies sur un corps K sont des courbes décrites par l'ensemble des solutions de certaines équations polynomiales à deux inconnues $f(x, y) \in \mathrm{K}[X, Y]$

Définition

Soit R *le corps des réels et soit* $a, b \in \mathrm{R}$ *tels que* $4a^3 + 27q^2 \neq 0$. *Une* **courbe elliptique non singulière définie sur** R *est l'ensemble* E *des points* P *de* R^2 *dont les coordonnées* (x, y) *sont des solutions de l'équation*

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

plus un point spécial $\mathcal{O}$ *appelé* **point à l'infini**.

La condition $4a^3 + 27q^2 \neq 0$ assure que la courbe est sans points doubles. Si $4a^3 + 27q^2 = 0$ la courbe est dite singulière.

Exemple la courbe $y^2 = x^3 - 4x$

Si E est une courbe elliptique non-singulière on définit sur les points de E une addition notée $+$ qui fera de l'ensemble des points de E un **groupe abélien**

- $\forall\, P \in E$, par définition $\mathbf{P + \mathcal{O} = \mathcal{O} + P = P}$
- Si $P, Q \in E$ avec $P = (x_1, y_1)$, $Q = (x_2, y_2)$ on considère les 3 cas
  1. $x_1 \neq x_2$
  2. $x_1 = x_2$ et $y_1 = -y_2$
  3. $x_1 = x_2$ et $y_1 = y_2$

Dans le cas 1, on note L la **droite passant par P et Q**. Elle coupe E en 3 points dont deux, P et Q, sont déjà connus, on note $R' = (x', y')$ le troisième point d'intersection. On prend le **symétrique de R′ par rapport à l'axe des abscisses**. On obtient une point $R = (x', -y') = (x, y)$ qui est encore sur E on pose

$$P + Q = R$$

Par un calcul sans mystère on trouve l'équation de L

$$y = \lambda x + \nu = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)x + \left(y_1 - \frac{y_2 - y_1}{x_2 - x_1}x_1\right)$$

on obtient alors les coordonnées de R

$$x = \lambda^2 - x_1 - x_2 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2$$

$$y = \lambda(x_1 - x_2) - y_1 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_2) - y_1$$

Dans le cas 2, on définit si $P = (x, y) \in E$, $-P = (x, -y)$

$$(x, y) + (x, -y) = \mathcal{O}$$

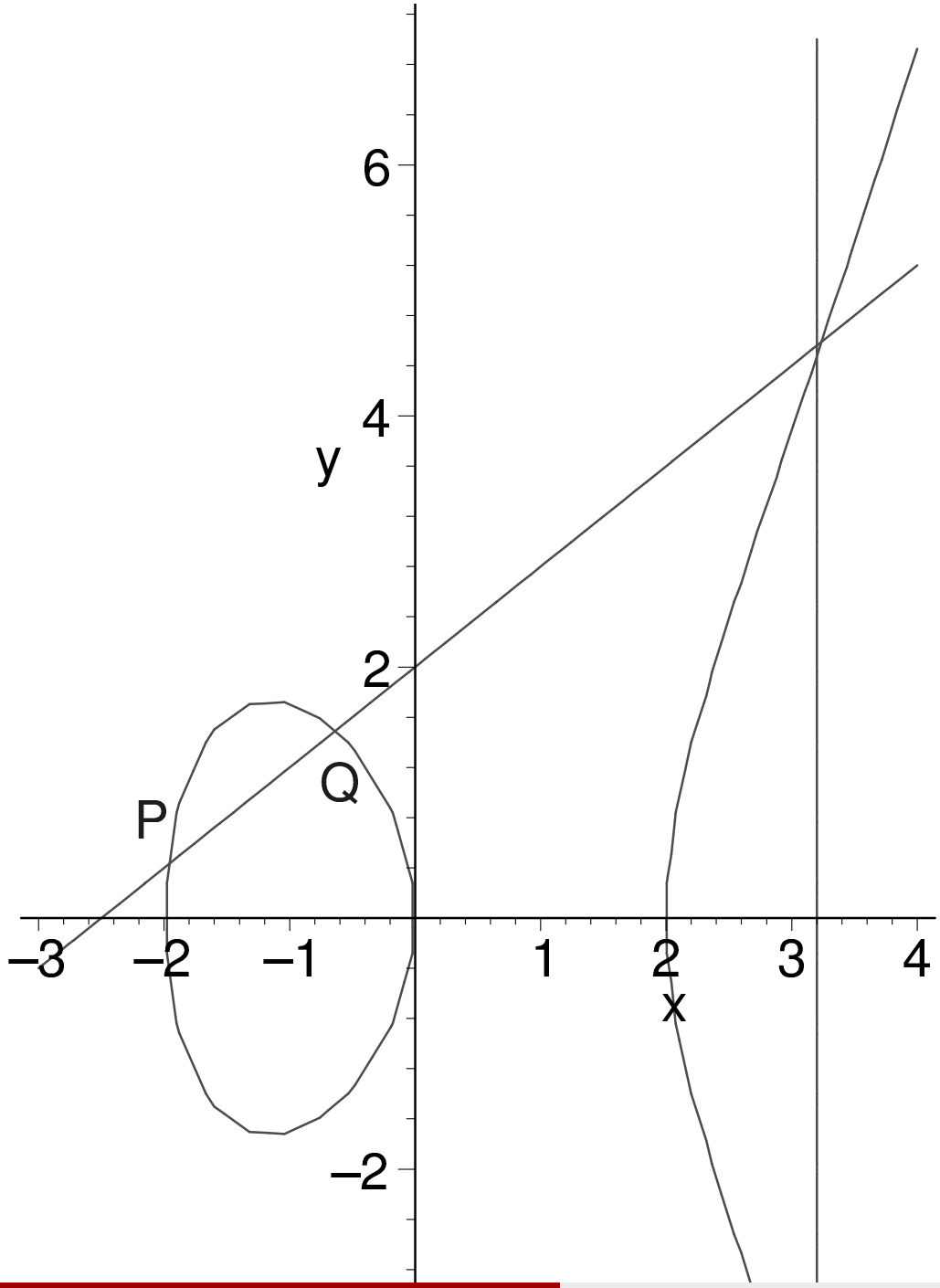
Le cas 3 se ramène au cas 2 en considérant que la droite L est la tangente en P à E. On trouve que l'équation de L est

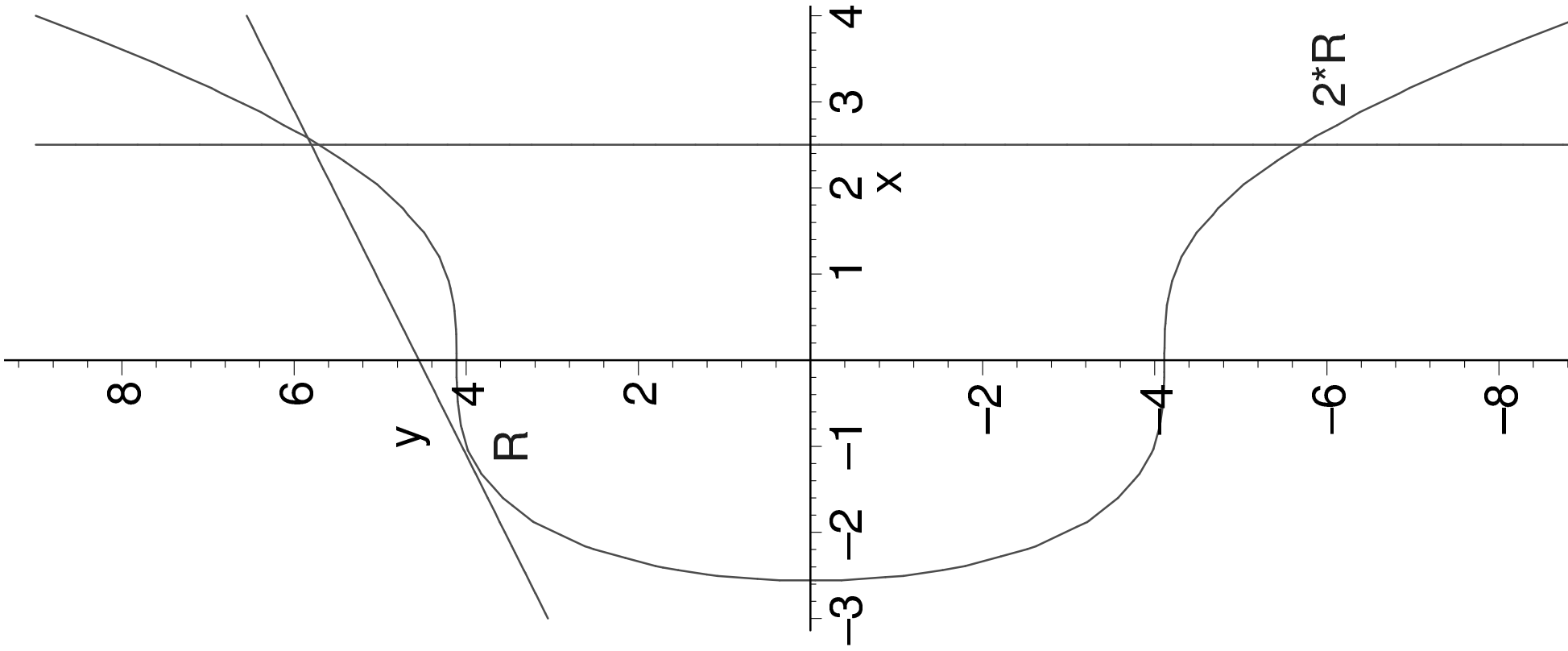
$$y = \lambda x + \nu = \left(\frac{3x_1^2 + a}{2y_1}\right)x + \left(y_1 - \left(\frac{3x_1^2 + a}{2y_1}\right) \cdot x_1\right)$$

ensuite le calcul de R est identique.

On montre que la loi ainsi définie est

- stable sur E
- commutative
- possède un élément neutre, $\mathcal{O}$
- Chaque point de E admet un inverse pour cette addition
- associative


6
4
y
2
P
Q
−3
−2
−1
1
2
3
4
x
−2



courbe elliptique sur R

$$y^2 = x^3 + 17$$

Courbes Elliptiques sur un corps fini

Les fait que le corps de base soit R ne joue pas grand rôle dans les calculs précédents (sauf pour pouvoir dessiner les courbes).

Si maintenant on a un corps fini F_q où $q = p^f$ avec p un nombre premier on peut refaire la théorie en supposant que l'on cherche des solutions dans F_q^2 (on peut prendre $\mathrm{F}_p = \mathrm{Z}/p\mathrm{Z}$).

Définition

Soit F_q *un corps fini avec* $p \geq 5$ *et soit* $a, b \in \mathrm{F}_q$ *tels que* $4a^3 + 27q^2 \neq 0$. *Une* **courbe elliptique non singulière définie sur** F_q *est l'ensemble* E *des points* P *de* F_q^2 *dont les coordonnées* (x, y) *sont des solutions de l'équation*

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

plus un point spécial $\mathcal{O}$ *appelé* **point à l'infini**.

Calculons par exemple les points de la courbe elliptique $y^2 = x^3 + x + 6$ dans F_{11}

x	$x^3 + x + 6 \mod 11$	résidu quadratique?	y
0	6	non	
1	8	non	
2	5	oui	4,7
3	3	oui	5,6
4	8	non	
5	4	oui	2,9
6	8	non	
7	4	oui	2,9
8	9	oui	3,8
9	7	non	
10	4	oui	2,9

Cette courbe sur F_{11} admet 13 points y compris celui à l'infini.

On a le théorème important

Théorème (Hasse)

Soit p un nombre premier supérieur à 3 et soit $q = p^f$. Soit E une courbe elliptique définie sur le corps fini F_q. On note $\#E$ le nombre de points de E définis sur F_q. On a

$$q + 1 - 2\sqrt{q} \leq \#E \leq q + 1 + 2\sqrt{q}$$

Le calcul de $\#E$ est difficile mais il existe un algorithme performant pour le faire **l'algorithme de Schoof**.