Distance de hamming : $d_h(x,y)$ nombre de composantes ou x et y diffèrent. $d_h(x,y) \leq d_h(x,z) + d_h(z,y)$.

Code parfait : Un code est dit parfait si les boules (disjointes) de rayon $\lfloor \frac{d-1}{2} \rfloor$ centrées aux mots du code recouvrent l'ensemble des mots. Par exemple, le code $\{(0,0,0,0,0),(1,1,1,1,1)\}$, est parfais tous les mots sont recouvert par les boules (on sait à partir de n'importe quel mot retrouver son mot du code le plus proche), avec le même code mais une répétition de 4 ce n'est plus le cas, si on reçoit $(0,0,1,1)$ on ne sais pas à quel mot du code il appartient.

Code en bloc : n est la longueur du code. c est émis et r est reçus. Code à simple contrôle de parité : somme des bits pairs (impair -> 1, pair -> 0). Détecte 1 erreur mais pas 2 ($d = 2$). Ne corrige aucune erreur. Code à répétition : (pour $repet = 4$, 4 répétition du même bit) détecte 4 erreurs et corrige 2 erreurs ($d = 5$). d est la distance minimal entre deux mots différents du code. Un code en bloc détecte $d-1$ erreurs et corrige $\frac{d-1}{2}$ (arrondis inférieur) erreurs. Une boule de rayon $d-1$ centré sur un mot du code ne contient pas d'autre mot du code (si centré en d alors au moins 1 mot du code).

Nombre premier : un nombre est premier seulement si il est divisible par 1 et par lui-même. Le nombre de nombre premier entre 1 et n est donné par la formule $\pi(n) \approx \frac{1}{\ln(x)}$ (pas pareil que $\phi(n)$!). Si p premier, $\phi(p) = p-1$. Si $p \neq q$ premiers alors $\phi(pq) = (p-1)(q-1)$.

Petit théorème de Fermat : si p premier, $pgcd(a,p) = 1 \rightarrow a^{\phi(p)} = a^{p-1} \equiv 1 \mod p$ et $a^p \equiv a \mod p$. Par contre, si on ne sait pas si p est premier ou non, on ne peut pas dire que p est composé si $a^n \approx a \mod n$. Il existe des nombres (de Carmichaël) qui ont cette propriété (par exemple $561 = 3 \cdot 11 \cdot 17$).

Division euclidienne : $\frac{14}{5} = 2 \cdot 5 + 4 \rightarrow a = qb + r,\ r = a \mod b$. Tout nombre est un facteur unique de nombres premiers. Pour trouver un nombre premier : crible d'Érathosthène (tester jusqu'à $t \geq \sqrt{n}$). Infinité de nombre premier, $\pi(x) \approx \frac{1}{\ln(x)}$.

PGCD : Plus grand commun diviseur, $ppcm(a,b)$ plus petit commun multiple. $pgcd(a,b) \cdot ppcm(a,b) = a \cdot b\ \forall a,b > 0$. $pgcd(a,b) = pgcd(b, a \mod b)$. Pour calculer le PPCM, il suffit de calculer $\frac{a \cdot b}{pgcd(a,b)}$.

Euclide étendu : $pgcd(a,b) = ua + vb$: démarrer avec $r_1 = a, r_2 = b, q_1 = \emptyset, u_1 = 1, u_2 = 0, v_1 = 0, v_2 = 1$. Ensuite, répéter tant que $r \neq 0$:

- $r_i = r_{i-2} \mod r_{i-1}$
- $q_{i-1} = r_{i-2} \div r_{i-1}$ (division euclidienne)
- $u_i = u_{i-1} \cdot -q_{i-1} + u_{i-1}$
- $v_i = v_{i-1} \cdot -q_{i-1} + v_{i-1}$

- $pgcd(a,b) = ua + bv$
- $m \cdot pgcd(a,b) = m \cdot ua + m \cdot bv$
- $sa + tb = s\alpha pgcd(a,b) + t\beta pgcd(a,b) = (s\alpha + t\beta) pgcd(a,b)$
- Un multiple entier de $pgcd(a,b)$ est une combinaison linéaire de a et b

Congruence modulo m : $m \in \mathbb{N}, m \geq 1, a, b \in \mathbb{Z}$. a congru à b modulo m : $a \equiv b \mod m \rightarrow a \mod m = b \mod m$,$a - b \mod m = 0$ $a = b + km, k \in \mathbb{Z}$.

- $a \equiv b \mod m, c \equiv d \mod m \rightarrow -a \equiv -b \mod m, a + c \equiv b + d \mod m, ac \equiv bd \mod m$
- $ab \equiv ac \mod m \wedge pgcd(a,m) = 1 \rightarrow b \equiv c \mod m$
- $a \equiv b \mod m \wedge a \equiv b \mod n \rightarrow a \equiv b \mod mn$

Inverse modulo m : Calculer l'inverse de a modulo m : Euclide étendue $\rightarrow ua + vm = 1$, u est l'inverse modulo m de a.

Indicatrice de Euler : $\phi(n)$ nombre de nombre premier avec n. Si p premier, alors $\phi(p) = p - 1$, si $p \neq q$ premiers, alors $\phi(pq) = (p-1)(q-1)$.

Thèorème d'Euler : $a \in \mathbb{Z}, m \geq 1, pgcd(a,m) = 1 \rightarrow a^{\phi(m)} \equiv 1 \mod m$. $a^{\phi(m)-1}$ est donc l'inverse de a modulo m si a est premier à m.

RSA : Choisir p et q premiers. $n = pq$, $\phi(n) = (p-1)(q-1)$, choisir e premier à $\phi(n)$ et $e < \phi(n)$ ($pgcd(e, \phi(n)) = 1$, calculer d comme l'inverse de e modulo $\phi(n)$ avec Euclide étendu. Clé publique : (n,e), clé privé : (n,d). Chiffre un message M : $C = M^e \mod n$. Déchiffrer un message : $M = C^d \mod n$.

Théorème des restes chinois : Résoudre le système $x \equiv a_1 \mod a_2, x \equiv b_1 \mod b_2, x \equiv c_1 \mod c_2$. $\bowtie (a,b,c)$ est l'inverse de ab modulo c. $x = a_1 \cdot \bowtie (b_2,c_2,a_2) \cdot b_2c_2 + b_1 \cdot \bowtie (a_2,c_2,b_2) \cdot a_2c_2 + c_1 \cdot \bowtie (a_2,b_2,c_2) \cdot a_2b_2$

Anneau $\mathbb{Z}/\mathbb{Z}$: $m \in \mathbb{N}^*$,$\mathbb{Z}/m\mathbb{Z} = \mathbb{Z}/m = \{0,1,...,m-1\}$,$0 = m = -m = 2m = ...$ Addition et multiplication change pas. Tous les éléments (différent de 0) on un inverse modulo m (Euclide étendue). Règle de calcul usuel (associativité,...). $m \cdot a = 0$.

Anneau $\mathbb{Z}/p\mathbb{Z}$, p premier : $a^{\phi(p)} = a^{p-1} = 1 \forall a \in (\mathbb{Z}/p\mathbb{Z})^*$. Ordre de $a \in (\mathbb{Z}/p\mathbb{Z})^*$ plus petit exposant entier e tel que $e \geq 1$ et $a^e = 1$, l'ordre de a divise toujours $p-2$. Ordre de $a^l = ppcm(l,e_a)/pgcd(e_a,l)$ et divise donc l'ordre de a. Il existe toujours un élément α dont les puissances successives engendre tous les éléments du corps. Un corps fini compte p^ν éléments avec p premier. Il existe $\phi(p-1)$ générateurs. Si α est générateur : a^l générateur si et seulement si $l \in 1,2,...,p-1$ premier à $p-1$. **Pour construire les tables : faire addition + multiplication le tout modulo p.**

Éléments inversibles d'un corps : tous les éléments, sauf 0, sont inversibles dans un corps.

Savoir si un polynôme est irréductible dans $\mathbb{F}_2$: Est-ce qu'il s'annule pour 0 ou 1 ? Est-ce qu'il est carré d'un polynôme (racine impair -> non) ? Divisible par x^2+x+1 ? Dans $\mathbb{C}$ il n'existe pas de polynôme irréductible de degré ≥ 1.
Pour trouver les polynômes irréductibles, partir des polynômes de degrés 1 et calculer les suivants. Il ne doivent s'annuler ni en 0 ni en 1 donc le terme constant vaut 1 et il doit y avoir un nombre impair de termes.

Structure additive de $\mathbb{F}_{p^\nu}$: Ensemble des vecteurs $a = (a_0,a_1,...,a_{\nu-1})$ à ν composantes. $a+b$: addition composante par composante. λa : multiplication de chaque composante par λ. $a = (a_0,a_1,...,a_{\nu-1} = a_0 + a_1x + a_2x^2 + ... + a_{\nu-1}x^{\nu-1} = a(x)$.

Table d'addition et multiplication dans $\mathbb{F}_{2^\nu}$: Addition composante par composante avec un XOR. Donc la diagonale vaut 0. Les éléments sont des vecteurs de longueur ν. Dans $\mathbb{F}_{2^3}^*$ tous les éléments sont générateurs, on obtient donc $x = 2, x^3 = 3, x^2 = 4, x^6 = 5, x^4 = 6, x^5 = 7$ codé en binaire. Ensuite, pour l'addition, on utilise la formule $x^kx^l = x^{k+l}$. On a $m(x) = x^3 + x + 1$, donc $x^3 = x + 1 = 2 + 1 = 3$. On construit les puissances des colonnes et des lignes à partir de 0, 1, x et $m(x)$.

$F[x]$: Ensemble des polynômes en x à coefficient $\in F$. $a(x_0) = 0$. Des polynômes différents peuvent être égales sur F. $a(x) = q(x)b(x) + r(x)$ (division euclidienne). $r(x) = a(x) \mod b(x)$. Tout polynôme de F est un produit unique de $\alpha \in F$ et de polynôme unitaires irréductibles. $x^2+x+1 \in F[x]$ seul polynôme irréductible de degré 2. Pour le $pgcd(a(x),b(x)$ ne change pas de la façon normale. La congruence ne change pas de la congruence dans $\mathbb{Z}$.

Exemple $\mathbb{F}_4 = \mathbb{F}_{2^2}$: x^2+x+1 est irréductible, on a donc $x^2 = x+1$. Donc $x \cdot x = x^2 = x+1$.

Code linéaire : $F = \mathbb{F}_{p^\nu}$, souvent $\mathbb{F}_2$. n est la longueur du code, k la dimension. Pour un mot x, $w(x)$ est le poids du mot (nombre de composante non-nulle), $d_h(x,y) = w(x-y)$. Un code est linéaire si les mots s'exprime sous la forme $c = aG$.

Exemple : $C = \{0000, 1011, 0110, 1101\}$, $k = 2$ (0 ou 1),$n = 4$,$d = w_{min} = w(0110) = 2$. Code de type $(n = 4, k = 2, d = 2)$. Matrice génératrice : $[1011; 0110]$. La matrice est systématique si elle contient une sous-matrice unitaire. Si une matrice n'est pas systématique : transformation par l'algorithme de Gauss.

Code engendrer par des vecteurs : On à b_0 et b_1 comme vecteur, on a donc un code avec $\{0, b_0, b_1, b_0+b_1\}$ (addition avec un XOR). La matrice G est créer avec ces vecteurs, essayer d'avoir une matrice systématique avec par exemple $G = [b_0; b_0+b_1]$.

Matrice systématique : Pour simple contrôle de parité : G est une matrice identité avec une colonne de 1. Pour un code à répétition, G est un vecteur de longueur n de 1.

Produit scalaire de polynôme : $x = (x_0,...,x_{n-1}), y = (y_0,...,y_{n-1}) \in F^n$, $\langle x,y \rangle = xy^T = \sum_{l=0}^{n-1} x_l y_l$. Les mots d'un code linéaire sont caractériser par les mots auxquels ils sont tous perpendiculaires.

Code dual : Si G est une matrice génératrice et $G = (I_k|P)$ systématique, alors $H = (-P^T|I_{n-k})$ matrice génératrice de C^T. $GH^T = O$ car lignes de G perpendiculaire à lignes de H. $x \in F^n$, H matrice de contrôle de $C : x \in X \leftrightarrow xH^T = 0 \in F^{n-k}$. Un code est de type (n, k, d).**Pour déterminer le code C à partir de G : ajouter** 0 **et addition des deux vecteurs de** G.

Syndrome : $S(x) = xH^T$, $S(x) = 0$ si x est un mot du code. Mot c du code émis : $r = c + e \rightarrow S(r) = S(c) + S(e) = S(e)$, ne dépend que de l'erreur ajoutée. Pour calculer le syndrome à partir de H, on commence par 0, puis les colonnes de h et pour finir on additionne les syndromes des en-têtes qui formes les en-têtes en dehors des mots du code. **Pour attribuer les syndromes, toujours commencer par l'en-tête avec le** 1 **le plus à droite et seul**.

Tableau standard : $G = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}, H = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$. Construction du tableau standard : 1ère ligne $\rightarrow$ mots du code en commençant par 0. Ensuite, prendre un mot de poids minimal pour la tête de ligne et remplir le reste de la ligne avec un XOR à l'intersection. Pour décoder par le tableau standard : le mot reçus auquel on soustrait la tête de ligne. **Mots hors du code :** distance plus grandes que les boules de rayon souhaités (voir la première colonne). **Pour trouver H :** à partir de G, prendre la partie non-systématique de G, la transposer et la mettre à droite dans H, remplir le reste avec une matrice identité.

Décodage par le syndrome : Calculer les syndromes des têtes de lignes. Syndrome des têtes de lignes : 00, 11, 10, 01. On reçoit $r = 1111$, on calcule le syndrome $rH^T = 10$, tête de la troisième ligne : $1111 - 0100 = 1111 \oplus 0100 = 1011$. Code de type (4,2,2), ne corrige aucune erreur.

Code de Hamming bianire : $H = 3, 5, 6, 7, 4, 2, 1$ écrits en binaire, $d = 3$, $G = 8, 4, 2, 1, 7, 11, 13$ écrits en binaire. C'est un code parfait : les boules disjointes de rayon 1 centrées aux 2^4 mots du code, à 8 mots chacune, recouvrent bien les $2^4 \cdot 8 = 2^7$ mots. Idem pour $H(8, 4, 4)$.

Codes cycliques (Anneau $F[x]/(x^n - 1)$) : On considère des polynômes $u(x) \in F[x]/(x^n - 1) : u(x) = u(x) \mod (x^n - 1)$, on a $x^n = 1, x^N = x^{N \mod n}$. Tout mot $\in F[x]/(x^n - 1)$ est représenté par un polynôme de degré $< n$. $v(x) = v_0 + v_1x + ... + v_{n-2}x^{n-2} \in F[x]/(x^n - 1)$, quand on multiplie par $x : xv(x) = v_0x + v_1x^2 + ... + v_{n-2}x^{n-1}$, c'est donc un décalage circulaire des composantes de v vers la droite.

Simplification de polynômes : Prendre les puissances du polynôme modulo la puissance de $m(x)$ dans $\mathbb{F}_2[x]/(x^n - 1)$. Si deux fois la même puissance (par exemple $x^4 + x^4$, ça s'annule).

Factorisation de $x^n - 1$ dans $\mathbb{F}_\nu$: Trouver une racine, factoriser et recommencer jusqu'à n'avoir que des facteurs irréductibles. Exemple : $\nu = 2$ $x^3 - 1 = (x + 1)(x^2 + x + 1)$.

Matrice de contrôle d'un code cyclique (Exemple) : Déterminer une matrice de contrôle du code cyclique ternaire de longueur 4 engendré par $g(x) = x^2 + 1$: $h(x) = (x^4 - 1)/(x^2 + 1) = x^2 - 1 = x^2 + 2 \in \mathbb{F}_3[x]$. $H = \begin{pmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \end{pmatrix}$. $d = 2$ donc le code est de type $(4, 2, 2)$.

Polynôme générateur du code : $x^n - 1 = (x - 1)(...)$. On choisis un diviseur unitaire $g(x)$ de $x^n - 1$ dans $F[x]$. On aura donc un contrôle avec $h(x) = (x^n - 1)/g(x)$. Exemple pour $F = \mathbb{F}_2$,$n = 7 : x^7 - 1 = (x - 1)(x^3 + x + 1)(x^3 + x^2 + 1)$, $g(x) = x^3 + x + 1$ et $h(x) = (x - 1)(x^3 + x^2 + 1) = x^4 + x^2 + x + 1$.

Matrice génératrice : Chaque mot du code est représenter par son reste modulo $x^n - 1 \rightarrow a(x)g(x) \in F[x]/(x^n - 1)$ pour un seul $a(x) = a_0 + a_1x + ... + a_{k-1}x^{k-1} \in F[x]$ de degré $< k = n - deg(g(x))$. $g(x) = g_0 + g_1x + ... + g_{n-k}x^{n-k}$ unitaire de degré $n - k$. $G = \begin{pmatrix} g_0 & g_1 & g_2 & ... & g_{n-k} & 0 & ... & ... & 0 \\ 0 & g_0 & g_1 & g_2 & ... & g_{n-k} & 0 & ... & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & ... & 0 & g_0 & g_1 & g_2 & ... & g_{n-k} & 0 \\ 0 & ... & ... & 0 & g_0 & g_1 & g_2 & ... & g_{n-k} \end{pmatrix}$. **c=aG**

Exemple : Code cyclique binaire de longueur 7, $g(x) = 1 + x + x^3$, $h(x) = x^4 + x^2 + x + 1$.

$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$, $H = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$: Hamming(7,4).

Matrice génératrice systématique : $g(x) = x^3 + x + 1$ dans $\mathbb{F}_2[x]/(x^7 - 1)$:

- $x^3 = 1 \cdot g(x) + reste$
- $x^4 = xg(x) + reste$
- $x^5 = (x^2 + 1)g(x) + reste$
- $x^6 = (x^3 + x + 1)g(x) + reste$

$$G_s = \begin{pmatrix} x^3 \\ x^4 \\ x^5 \\ x^6 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Polynôme syndrome : $\sigma(v(x)) = v(x) \mod g(x) \in F[x]$: reste de la division du mot $v(x)$ par $g(x)$. Si $\sigma(v(x)) = 0 \leftrightarrow v(x) \in C$. Ne dépend que du polynôme d'erreur ajouté lors de la transmission. Le décodage est correct si $r(x) \leq \lfloor \frac{d-1}{2} \rfloor$.

Corps $\mathbb{F}_{2^\nu}$ (Exemple) : $\nu = 4$, polynôme irréductible primitif $m(x) = x^4 + x + 1 \in \mathbb{F}_{\not\in}[\curvearrowleft]$ de degré ν. $\mathbb{F}_{16} = \mathbb{F}_{2^4} = \mathbb{F}_2[x]/(m(x))$. $\alpha = x \in \mathbb{F}_{16}$: $\mathbb{F}_{16} = \{0, \alpha, \alpha^2, ..., \alpha^{15} = 1\}$. $\mathbb{F}_{2^4}$ est un espace vectoriel de dimension 4 sur $\mathbb{F}_2$ avec comme base $1, \alpha, \alpha^2, \alpha^3$. On peut coder $0, 1, \alpha, \alpha^2, ..., \alpha^{15}$ comme des combinaisons linéaires de $1, \alpha, \alpha^2, \alpha^3$. On a $m(x) = x^4 + x + 1$ donc $\alpha^4 = 1 + \alpha$.

Polynômes minimaux : les polynômes minimaux sont les polynômes irréductibles d'un corps. Par exemple : $x^15 - 1 = (x - 1)(x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1)(x^4 + x^3 + 1)$. Le degré des polynômes minimaux est $\leq \nu$: $(\alpha^l)^{2^\nu} = (\alpha^{2^\nu})^l = \alpha^l$.

One-time pad : Soit un message x de l bits, on génère une clef de l bits complètement aléatoire. On transmet la clef à Bob et on calcule le texte chiffré $y = x \oplus k$, où $\oplus$ est un XOR. Ce code est parfaitement sûr.

Un chiffrement est dit parfaitement sûr si le texte clair x et le texte chiffré y sont statistiquement indépendant :

$$P[X|Y] = P[X]$$

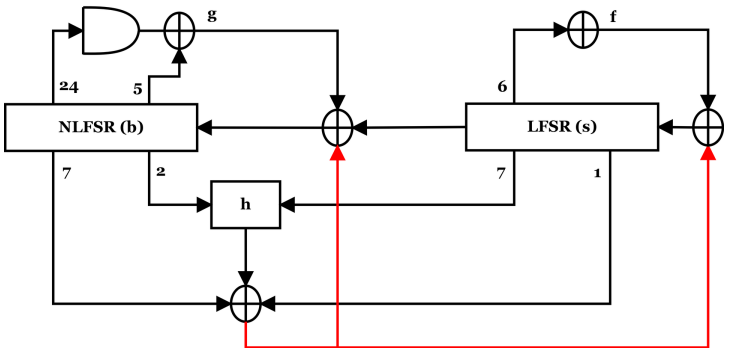
Ce qui vaut

$$P[X|Y] = \frac{P[X \wedge Y]}{P[Y]}$$

Inconvénient : clef unique, le texte chiffré peut être manipulé, la transmission des clefs posent problèmes et la génération d'une clef aléatoire est problématique.

RC4 : RC4 n'est pas recommandé en pratique, on initialise une clé puis on utilise un générateur de flot pseudo-aléatoire en essayant de reproduire le "one-time pad".

Grain : Algorithme qui prend une clef de 80 bits et un vecteur d'initialisation de 64 bits. Économe en ressource si implémenter sur FPGA et encore incassé à ce jour.



Chiffrement par bloc : Étant donné un chiffrement par bloc F qui chiffre des textes clairs $X = \{x_1, ..., x_l\}$ de l bits au moyen d'une clef K, où $x_i \in \mathbb{Z}_2$, on peut écrire F sous la forme d'un système d'équations booléennes :

$$y_1 = F_K^1(x_1, ..., x_l)$$
$$...$$
$$y_l = F_K^l(x_1, ..., x_l)$$

Confusion : les fonctions $F_K^1, ..., F_K^l$ doivent être mathématiquement complexes car en pratique on peut trouver la clef K au moyen du texte clair et du texte chiffré. Les fonctions ne doivent pas être linéaire.

Diffusion : Une modification d'un seul bit du texte clair devrait changer en moyenne la moitié des bits du texte chiffré.

Recherche exhaustive de clef : Le principe est d'essayer toutes les clefs possibles. La recherche exige un critère d'arrêt : comment savoir que la clef est bonne ? Pour une clef de l bits, il faut au minimum 1 essai, au pire des cas 2^l essais et en moyenne 2^{l-1} essais. En 2015, une taille de clef de 64 bits est considéré comme insuffisante. 80 bits en minimal. Moore : la puissance de calcule double environ tous les 18 mois.

Chiffrement itéré : Itérer une fonction de ronde un certain nombre de fois. Chaque fonction de ronde prend comme paramètre un sous-clef de la clef principale au moyen de cadencement de clef. Le but est de trouver un équilibre entre nombre d'itération et force du code.

DES Prend une clef de 64 bits dont 8 bits ignorés (donc 56 bits) et est bâti sur un schéma de Feistel à 16 rondes.

IP est une matrice de permutation sur $\mathbb{Z}_2$ de taille 64×64 et FP est son inverse. E est une fonction linéaire qui étend son entrée de 32 bits en une sortie de 48 bits en dupliquant certains bits. P est une matrice de permutation de 32×32. $PC1$ est une fonction linéaire qui passe de 64 bits à 56 bits. $PC2$ est une fonction linéaire qui passe de 56 bits à 48 bits

Triple DES Triple DES à deux clefs : $Y = DES_{K1}(DES_{K2}^{-1}(DES_{K1}(X)))$
Triple DES à trois clefs : $Y = DES_{K3}(DES_{K2}^{-1}(DES_{K1}(X)))$

IDEA Possède une clef de 128 bits, sa fonction de ronde est itérée 8.5 fois et qui chiffre des blocs de données de 64 bits. IDEA repose sur l'utilisation de 3 opérations incompatibles algébriquement :

— l'addition sur $\mathbb{Z}_2^{16}$ notée $\oplus$
— l'addition sur $\mathbb{Z}_{2^{16}}$ notée $\boxtimes$
— la multiplication sur $\mathbb{Z}_{2^{16}+1}$, notée $\odot$ et où 0 est identifié à la valeur 2^{16}.

IDEA est basé sur un schéma de Lai-Nassem.

AES Chiffre des données de 128 bits qui supporte des clefs de 128 bits (10 rondes), 192 bits (12 rondes) et 256 bits (14 rondes). Très efficace sur les plateformes embarquées, CPU et hardware.

Mode opératoire Un algorithme de chiffrement par flot peut chiffrer des données de n'importe quelle longueur contrairement à un algorithme de chiffrement par bloc (64 ou 128 bit typiquement).

Un mode opératoire est une manière de chiffrer des données plus grandes que la taille d'un bloc. Modes classiques :

— "Electronic Code Block" ECB
— "Cipher Block Chaining" CBC
— "Counter Mode" CTR
— CFB, OFB,...

ECB Les données à chiffrer sont divisées en blocs de la taille du bloc de l'algorithme de chiffrement, chaque bloc est chiffré indépendamment des autres.

Problème : Deux blocs de textes clairs identiques seront chiffrés de manières identiques. On a donc une perte potentielle de confidentialité. Le mode ECB résiste donc très mal aux attaques visant à modifier l'intégrité du texte chiffré. C'est un algorithme déterministe.

CBC

Principe :

— les données sont divisées en blocs de la taille du bloc de l'algorithme de chiffrement
— avant le chiffrement, un bloc est combiné avec un XOR avec le texte chiffré précédent
— Le premier bloc est combiné avec un bloc de chiffrement aléatoire appelé **vecteur d'initialisation** (IV)

Chaque bloc de texte chiffré est dépendant de tous les blocs précédents. Le chiffrement est séquentiel mais le déchiffrement peut être parallèle.

CTR

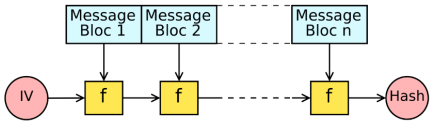
— Un compteur est chiffré par l'algorithme de chiffrement par bloc
— Le flux est combiné au moyen d'un XOR avec le texte clair

Algorithme parallèle, pas besoin de couper des données si pas de la bonne taille, la sécurité repose sur la qualité de l'IV, mauvaise résistance aux adversaires attaquant l'intégrité du texte chiffré.

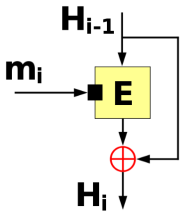
Fonction de hachage Fonction définis comme $H : \{0,1\}* \to \{0,1\}^l$ qui possède les propriétés suivantes :

— le calcul de l'empreinte (digest) $h = H(m)$ d'une message m est une opération rapide
— étant donnée une image h, il est impossible en pratique de trouver un message m tel que $h = H(m)$ (**résistance à la première préimage**)
— étant donnée un message m et son empreinte $h = H(m)$, il est impossible en pratique de trouver un message $m' \neq m$ tel que $h = H(m')$ (**Résistance à la seconde préimage**)
— Il est impossible en pratique de trouver deux messages $m \neq m'$ tels que $H(m) = H(m')$

Construction de Merkle-Damgård Cette construction permet de transformer une fonction de compression en une fonction de hachage.



Construction de Davies-Meyer La construction de Davies-Meyer permet de transformer un algorithme de chiffrement par bloc en une fonction de compression.



Quelques fonctions de hachage

MD5 : Empreintes de 128 bits, **cassée !** Il est théoriquement possible de trouver des collisions en quelques secondes

SHA-1 : Empreintes de 160 bits, **cassée !** Il théoriquement possible de trouver des collisions en environ 2^64 opérations

SHA-256, SHA-512 : Empreintes de 256 et 512 bits et "sûre"

SHA-3 : Empreintes de 256 et 512 bits, similaire à AES

Authentification symétrique La cryptographie symétrique peut garantir l'authenticité d'un canal de communication une famille de fonction est définis comme $h_k : \{0,1\} * \times \kappa \longrightarrow \{0,1\}^l$. A éviter : $SHA256(X) \oplus K$. Exemple de familles sûre : HMAC, CBC-MAC,...

Protocole :

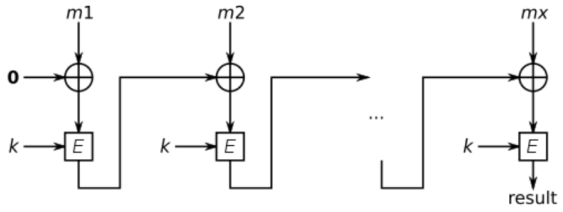
1. Alice et Bob partage un secret K
2. Alice envoie la liste X ainsi que la valeur de MAC $\tau = h_k(X)$ à Bob
3. Bob calcule $\tau = h_k(X')$ avec le message X' reçu, et l'accepte comme étant authentique $\iff \tau = \tau'$ où τ' est la valeur du MAC attaché au message.

HMAC Construit un MAC à partir de n'importe quelle fonction de hachage cryptographiquement sûre, HMAC-MD5 et HMAC-SHA1 sont très largement utilisées en pratique (TLS, IPSec, ...).

Fonction de hachage H
Clef secrète K
Message à authentifier X
Constantes $opad = 0x5c5c5c...5c$ $ipad = 0x363636363...36$

$$HMAC_k(X) = H((K \oplus opad)||H((K \oplus ipad)||X))$$

CBC-MAC Seulement sûr pour des messages de taille fixe !



Protocole d'identification : un mot de passe et un username permet d'identifier un utilisateur.

Protocole d'authentification : e-banking, authentification via un certificat et le protocole TLS.
Il existe trois type d'authentification :
— authentification faible
— authentification forte
— authentification "zero knowledge"
Un mot de passe est une authentification faible (si mdp espionné -> "**replay attack**").
Challenge-response est une authentification forte :

1. Alice et Bob partage un secret symétrique commun
2. Alice s'annonce auprès du serveur
3. Bob génère un challenge R (valeur aléatoire) à usage unique, stocke la pair ("*Alice*", R) et envoie R à Alice.
4. Alice calcule $\tau = MAC_K(\text{"}Alice\text{"}||R)$ et envoie la pair ("Alice",τ) à Bob.
5. Bob reçoit la pair (X, τ'), récupère le secret K ainsi que la valeur R correspondant à l'identité X, et calcule $\tau = MAC_K(X||R)$.
6. Si $\tau = \tau'$ alors Bob considère que Alice s'est authentifié avec succès.

Protocole de Diffie-Hellman : Le but de ce protocole est de **partagé un secret** entre deux entités au moyen d'un canal non-confidentiel. Par contre, le canal doit être authentique ("man in the middle attack"). L'idée principale consiste à utiliser une fonction à sens unique.

Logarithme discret

$p = 2q + 1$, p est premier et q en général est premier
$g \in G \to \mathbb{Z}_p^*$
$f : x \mapsto g^x$
$x \longrightarrow y = g^x \mod p$

Protocole de Diffie-Hellman :

1. Choisir $\mathbb{Z}_p^*$ et g générateur de $\mathbb{Z}_p^*$
2. Alice génère une valeur a tel que $a \in \{1, 2, ..., p-2\}$, calcule $y_a = g^a \mod p$ et envoie y_a à Bob
3. Bob génère une valeur b tel que $b \in \{1, 2, ..., p-2\}$, calcule $y_b = g^b \mod p$ et envoie y_b à Alice
4. Alice calcule $k = y_b^a \mod p$ et Bob calcule $k = y_a^b \mod p$
5. La clef secrète partagée est k.

Chiffrement RSA : L'idée est d'utiliser une fonction à sens unique munie d'une trappe.

Génération de clef RSA :

On définis P comme l'ensemble des nombres premiers
$p \in P$ $q \in P$
$n = pq$
On définis l'opérateur $\oplus$ comme étant $e \oplus d$: e est premier avec d
$e \oplus \phi(n)$
$\phi(n) = (p-1)(q-1)$
$d = e^{-1} \mod \phi(n)$
(n, e) : clef publique
(n, d) : clef privée

Utilisation : on chiffre un message m avec $c = m^e \mod n$, on déchiffre avec $m = c^d \mod n$

Signature RSA : Étant données la clef privée (n, d) et un message m à signer, on obtient la signature s en calculant $s = m^d \mod n$. Étant donnés la clef publique (n, e) et un message m' muni d'une signature s, on vérifie la signature en calculant $m = s^e \mod n$ et on accepte la signature si et seulement si $m = m'$.

En pratique, on ne signe pas un message mais une empreinte h de ce message : $h = H(m)$ où $H(.)$ est une fonction de hachage cryptographiquement sûr. Étant donné deux signatures $s = m^d \mod n$ et $s = m'^d \mod n$ on peut créer une nouvelle signature :

$$s \cdot s' \equiv m^d \cdot m'^d \equiv (m \cdot m')^d \mod n$$

Signature RSA rapides :

1. On calcule $d_p = d \mod (p-1)$, ainsi que $d_q = d \mod (q-1)$
2. On calcule $q^{-1} \mod p$ et $p^{-1} \mod q$ via Euclide étendu
3. Signature partielle :
   — $s_p = m^{d_p} \mod p$
   — $s_q = m^{d_q} \mod q$
4. Signature finale : $s = (s_p(q^{-1} \mod p)q + s_q(p^{-1} \mod q)p) \mod n$

Signature d'El-Gamal : Repose sur la difficulté de résoudre le problème du logarithme discret.
— On génère un nombre premier $p = 2q + 1$ et un générateur $g \in \mathbb{Z}_p^*$
— On génère $a \in \{1, ..., p-2\}$
— On calcule $A = g^a \mod p$
— Clef publique : (p, g, A)
— Clef privé : (p, g, a)

Signature : On utilise une fonction de hachage cryptographiquement sûre $h : \{0,1\}^* \longrightarrow \{1, ..., p-2\}$. On calcule un nombre $k \in \{1, ..., p-2\}$ premier avec $p-1$. pour signer un message m, on calcule $r = g^k \mod p$ ainsi que $s = k^{-1}(h(m) - ar) \mod p-1$. La signature du message est (r, s).

***** Vérification : $r \in 1, ..., p-2$ et $A^r r^s \equiv g^{h(m) \mod p}$

Choix de la taille des clés : Difficile de choisir les paramètres cryptographiques. 1ère chose à prendre en compte : la difficulté supposée de casse un problème difficile ($n = pq$, ou logarithme discret).

Nombres premiers : Pour trouver un nombre premier de grande taille : générer un nombre impair aléatoire, tester si il est bon sinon recommencer, tester si il convient à l'algorithme sinon recommencer. En moyenne : $\ln(p)$ fois pour obtenir un nombre.

Test de Fermat : utiliser le petit théorème de Fermat pour tester le nombre si premier ou non (problème avec les nombres de Carmichael (infinité de nombre !)).

Test de Miller-Rabin : soit n un nombre entier et $n \geq 3$ et k un paramètre de sécurité :

- écrire $n-1 = 2^s d$ avec d impair
  - générer un nombre a entre 1 et $n-1$ (compris)
  - calculer $x = a^d \mod n$, si $x \equiv 1 \mod n$ ou $x \equiv n-1 \mod n$ continue ;
  - Pour $1 \leq r \leq s-1$, calculer $x_r \leftarrow x_{r-1}^2 \mod n$, avec $x_0 = a^d \mod n$. Si $a^{2^r d}$ n'est pas congrue à $-1 \mod n$ pour $1 \leq r \leq s-1$, alors n n'est pas premier et stopper. Sinon, continuer.
- n est probablement premier.

La probabilité que le test de Miller-Rabin annonce n comme étant premier de manière erronnée est inférieure à 4^{-k}.

Exponentiation rapide : crucial de calculer $a^b \mod c$ rapidement. On a donc $a^b \equiv (a^{b_0})^{2^0}(a^{b_1})^{2^1}(a^{b_2})^{2^2}...(a^{b_{n-1}})^{2^{n-1}}$. Par exemple, pour $b = 37 = 100101$ on a donc : $a^b \equiv (a^1)^{2^0}(a^0)^{2^1}(a^1)^{2^2}(a^0)^{2^3}(a^0)^{2^4}(a^1)^{2^5} \mod c$. On écrit les nombres de droite à gauche (on commence par le bit de poid faible).

Courbes elliptiques : une courbe elliptique C définie sur $\mathbb{F}$ avec les coefficients $a, b \in \mathbb{F}$ est définie comme étant l'ensemble des points $(x, y) \in \mathbb{F}^2$ qui satisfont l'équation suivante dans $\mathbb{F}[x]$: $y^2 = x^3 + ax + b$ et on a en plus un point O appelé **point à l'infini**.

Addition de points : soit $P = (x_1, y_1), Q = (x_2, y_2)$ sur une courbe elliptique avec $x_1 \neq x_2$, on calcule $R = (x_3, y_3) = P + Q$: $x_3 = (\frac{y_2 - y_1}{x_2 - x_1})^2 - x_1 - x_2$ et $y_3 = -y_1 + (\frac{y_2 - y_1}{x_2 - x_1})(x_1 - x_3)$.

Doublement des points : soit $P = (x_1, y_1)$ sur une courbe elliptique, on calcule on calcule $R = (x_2, y_2) = 2P$: $x_3 = (\frac{3x_1^2 - 1}{2y_1})^2 - 2x_1$ et $y_3 = (\frac{3x_1^2 - 1}{2y_1})(x_1 - x_2) - y_1$.

Courbes elliptiques sur $GF(2^r)$:

- $y^2 + b_1 y = x^3 + a_1 x + a_0$ avec $b_1 \neq 0$
- $y^2 + xy = x^3 + a_2 x^2 + a_0$ avec $a_0 \neq 0$

Théorème de Hasse : Le théorème de Hasse garantie que n'importe quelle courbe elliptique définie sur un corps suffisament grand aura un grand nombre de points. Si N est le nombre de points sur une courbe elliptique définie sur un corps fini à q éléments, alors N est borné par $|N - (q+1) \leq 2\sqrt{q}|$.

Courbes Elliptiques et Extensions de Corps : Une courbe elliptique est dite être définie sur le corps fini $GF(q)$ avec $q = p^n$ pour un entier naturel n si ses coefficients sont dans $GF(q)$.La courbe définie par toutes les solutions dans l'extension de corps $GF(q^r)$ est noté E_r. Soit $s = q + 1 - N$. On note α et β les deux racines du polynôme caractéristique $T^2 - sT + q$ d'une courbe elliptique E. Pour une courbe elliptique E définie sur $GF(q)$, le nombre N_r de points qu'elle possède lorsqu'elle est définie sur $GF(q^r)$ avec $r \geq 1$ est donnée par $N_r = -\alpha^r - \beta^r + q^r + 1$.

Avantages des courbes elliptiques :

- Le problème du logarithme discret est transposable à une courbe elliptique
- Les algo efficace pour le logarithme discret dans $GF(q^n)$ ne sont pas applicable aux courbes
- Les courbes elliptiques définis sur $GF(2^n)$ sont efficacement implémenté en hardware

ECDH : Le protocole de Diffie-Hellman peut être implémenté sur le groupe des points d'une courbe elliptique :

1. Se mettre d'accord sur une courbe elliptique cryptographiquement sûre et sur un point G d'ordre n. Le nombre $h = N/n$ est appelé le co-facteur et devrait être petit, de préférence égal à 1.
2. Alice génère une valeur secrète $a \in \{1, ..., n-1\}$ uniformément au hasard, calcule $Y_a = aG$, et envoie Y_a à Bob via le canal authentique.
3. Bob génère une valeur secrète $b \in \{1, ..., n-1\}$ uniformément au hasard, calcule $Y_b = bG$, et envoie Y_b à Alice via le canal authentique.
4. Alice calcule $K = aY_b$ et Bob $K = bY_a$.
5. La clé secrète partagée est K.

El-Gamal :

1. Se mettre d'accord sur une courbe elliptique cryptographiquement sûre et sur un point G d'ordre n.
2. Clé privé : $a \in \{1, ..., n-1\}$
3. Clé publique : aG
4. Pour chiffrer un message M, on génère un nombre $k \in \{1, ..., n-1\}$ secret.
5. Le texte chiffré est la pair $(kG, M + k(aG))$
6. Pour déchiffrer, on calcule $a(kG)$ puis $M + k(aG) - a(kG) = M$