

1. Introduction

Le développement des transactions électroniques dans un nombre croissant de domaines économiques pose de manière de plus en plus critique le problème de la sécurité de telles transactions, notamment vis-à-vis des questions de confidentialité, d'authentification et de certification qui s'y rattachent. Pour tous ces aspects, la cryptographie est appelée à devenir une technique de plus en plus fondamentale pour la protection des informations possédées et/ou échangées par des individus ou des organisations.

2. Définitions

Méthode permettant de rendre illisible des informations afin de garantir l'accès à un seul destinataire authentifié. La conversion des données s'effectue au moyen d'une clé [Cryp].

Autrement dit, la cryptographie est une étude des techniques dédiée à fournir des services de sécurité (ou propriétés de sécurité) à mettre en œuvre pour la sécurité informatique. Ces propriétés de sécurité sont la confidentialité, l'authentification, la non répudiation, et l'intégrité des données.

3. Terminologie

- ✓ **Coder** : Transformer un texte, une information en remplaçant les mots dans une écriture faite de signes prédéfinis.
- ✓ **Chiffrer** : Transformer un texte, une information en remplaçant les lettres dans une écriture faite de signes prédéfinis.
- ✓ **Cryptologie** : est une science mathématique qui comporte deux branches: la cryptographie et la cryptanalyse.
- ✓ **Cryptographie** : science qui utilise les mathématiques pour le cryptage et le décryptage de données

C'est aussi l'étude des techniques mathématiques en rapport avec les aspects de la sécurité informatique (confidentialité, intégrité et authenticité)

- ✓ **Cryptanalyse** : c'est l'étude des informations cryptées, afin d'en découvrir le secret. Les Cryptanalystes sont également appelés des « pirates ».

- ✓ **Texte en clair (Plaintext):** données lisibles et compréhensible sans intervention spécifique.
- ✓ **Cryptage (chiffrement):** méthode permettant de dissimuler le texte en clair en masquant son contenu. Cette opération permet s'assurer que seules les personnes auxquelles les infos. Sont destinées +pourront y accéder.
- ✓ **Texte chiffré (Ciphertext):** texte nintelligible résultant du cryptage.
- ✓ **Cryptogramme :** Message chiffre
- ✓ **Clé :** une clé est un paramètre utilise en entrée d'une opération cryptographique (chiffrement, déchiffrement, ...)
- ✓ **Décryptage (déchiffrement):** processus inverse de transformation du texte chiffré en texte clair.
- ✓ **Algorithme cryptographique :** fonction mathématique utilisée pour le chiffrement et le déchiffrement. On parle d'algorithmes de chiffrement et de déchiffrement
- ✓ Pour une vraie sécurité, tous les algorithmes modernes utilise une clé. Cette clé peut prendre une des valeurs parmi un grand nombre de valeurs possible (espace des clés)
- ✓ La valeur de la clé « K » affecte les algorithmes de chiffrement et de déchiffrement, et donc les fonctions correspondantes « eK » et « dK » [Buc06].

4. Propriétés de sécurité

Dans cette section, on étudie des propriétés de sécurité classique qui sont vérifiées dans les différentes communications des réseaux.

4.1. Confidentialité :

On peut l'appeler aussi *secret*, qui est un mécanisme qui sert à transmettre des données de telle sorte que seul le destinataire (participant honnête) autorisé, puisse les lire.

4.2. Authentification :

L'authentification est un mécanisme permettant d'identifier des personnes ou des entités et de certifier leur identité. Au moins deux variantes importantes existent de cette dernière propriété, appelées *authentification faible* et *authentification forte* [Hor07].

La propriété de *l'authentification faible* exige qu'un agent puisse vérifier l'authenticité d'un message, mais il ne peut pas être sûr que ce message a été émis lors de la session

courante. L'intrus peut par exemple le stocker et rejouer lors d'une nouvelle session. Un participant d'un protocole qui satisfait la propriété de l'authentification faible peut donc être sûr que son interlocuteur a participé dans une session du protocole, mais il ne sait pas dans laquelle.

Un protocole qui satisfait la propriété de *l'authentification forte* donne plus de garanties. Un participant qui exécute un tel protocole peut être sûr que le message en question a été émis dans la session courante par celui qu'il croit être son interlocuteur. Si un intrus lui envoie un message émis lors d'une autre session, il peut le détecter, même s'il a été émis par l'interlocuteur dont il l'attend.

4.3. Non-répudiation :

La non-répudiation notifie la possibilité de vérifier que les participants honnêtes (émetteur/récepteur) sont bien les parties qui disent avoir respectivement envoyé ou reçu le message. En autre façon, la non-répudiation de l'origine prouve que les données ont été envoyées, et la non-répudiation de l'arrivée prouve qu'elles ont été reçues.

4.4. Intégrité :

L'intégrité est un mécanisme mis pour s'assurer que les données reçues n'ont pas été modifiées durant la transmission.[Chi10].

5. Les primitives cryptographiques :

Dans la communication entre les participants ou des machines dans le réseau, les messages créés et transmis sont souvent appliqués sur eux des primitives cryptographiques. Dans cette section, on présente les principes de chiffrement symétrique et le chiffrement asymétrique. Les autres primitives, on détaille dans les sections suivantes.

5.1. Chiffrement symétrique :

Le principe du chiffrement symétrique, ou chiffrement à clé secrète est : la clé utilise pour le cryptage du texte clair (*plaintext*) est la même pour décryptage du texte chiffré (*ciphertext*). Le problème se pose dans les algorithmes symétriques est la gestion de clé.

Il existe deux classes d'algorithmes de chiffrement symétrique: chiffrement en continu « *Block-cipher* » et chiffrement par blocs « *Stream Ciphers* ». Les algorithmes

principales de type block-cipher sont : DES (Data Encryption Standard), AES (Advanced Encryption Standard), IDEA (International Data Encryption Algorithm), et RC5 (Rivest Cipher). Le chiffrement par bloc, crypter un bloc complet de données à un moment, alors que la taille d'un tel bloc de données est habituellement de 64, 128, 192 ou 256 bits et la transformation de cryptage est fixe [Chi10].

Dans ce cas, pour un message « m », on écrit $e_K(m) = c$, $d_K(c) = m$ et $d_K(e_K(m)) = m$ (voir la figure N° I.1) [ChiC11].

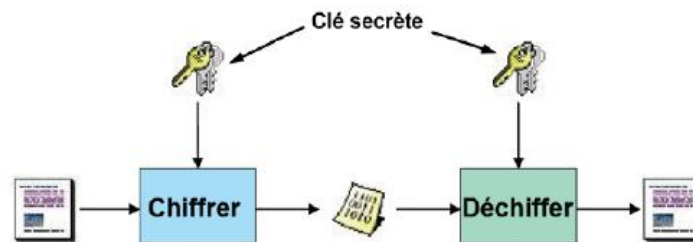


Figure N° I.1: chiffrement symétrique[ChiC11].

5.2. Chiffrement asymétrique :

On peut l'appeler chiffrement par clé publique, dans ce chiffrement, la clé de cryptage et la clé de décryptage sont différentes. Chaque personne possède deux clés distinctes (une privée, une publique) avec impossibilité de déduire la clé privée à partir de la clé publique. Une clé publique diffusée à tout le monde, utilisée pour chiffrer le message. Une clé privée tenue secrète, utilisée pour déchiffrer le message. Dans ce cas, on écrit : $e_{K1}(m) = c$, $d_{K2}(c) = m$ et $d_{K2}(e_{K1}(m)) = m$ (voir figure N° 02)

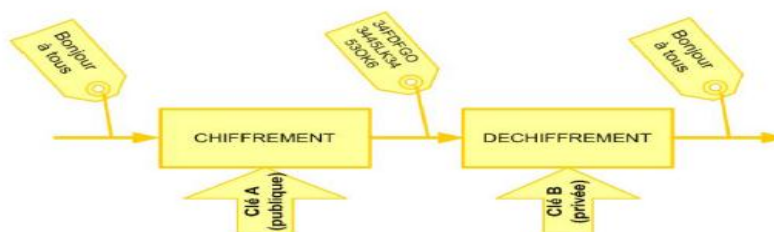


Figure N° I.2: chiffrement à clé publique[ChiC11].

On peut utiliser des algorithmes à clé publique en : chiffrement/déchiffrement (cela fournit le secret), signatures numériques (cela fournit l'authentification), et échange de clés (ou des clefs de session).

Ces algorithmes reposent sur des principes mathématiques qui sont : (1) basée sur la factorisation de grands nombres (e.g. RSA), (2) basée sur les logarithmes discrets (e.g. Diffie-Hellman, ElGamal), (3) basée sur les courbes elliptiques (e.g. ECC).

6. Génération des nombres pseudo-aléatoires :

6.1. Historique du développement des générateurs pseudo-aléatoires :

Le développement des algorithmes générant des nombres pseudo-aléatoires est très lié à celui de la cryptographie, l'importance militaire et économique de cette science ayant motivé de nombreuses recherches au cours de l'histoire.

Les chiffrements utilisés traditionnellement jusqu'au XIX^e siècle reposaient essentiellement sur le secret autour de la méthode utilisée, et sur l'absence de traitement de masse. De nos jours, de telles méthodes sont impraticables car il existe de nombreuses théories statistiques qui permettent de retrouver l'algorithme de génération à partir de ses résultats. En outre, les techniques de chiffrement ne peuvent plus être gardées secrètes, et en 1883, Auguste Kerckhoffs exposera une règle fondamentale de la cryptographie moderne : la sécurité d'un système ne doit pas reposer sur la méconnaissance de la méthode de chiffrement. Cette règle, accompagnée par le développement des algorithmes de chiffrement par clé, marquera le début de l'essor des générateurs pseudo-aléatoires.

Leur importance est toutefois restée limitée tant que les moyens physiques de calcul n'ont pu supporter les longs et répétitifs calculs qu'ils nécessitent. C'est pourquoi leur émergence n'a vraiment commencé qu'en 1946, quand John von Neumann publie son générateur middle-square. C'est durant les années qui suivirent que la plupart des algorithmes rapides et populaires virent le jour : en 1948 D. H. Lehmer introduit les générateurs congruentiels linéaires qui seront améliorés en 1958 par G.J. Mitchell et D.P. Moore ; et deviendront par la suite extrêmement répandus, la plupart des fonctions de chiffrement basique y ayant recours.

Ces premiers générateurs pseudo-aléatoires possèdent malgré leur large popularité des propriétés statistiques assez mauvaises, et ne répondaient pas aux besoins des cryptographes. Plus récemment, des algorithmes robustes vis-à-vis des analyses statistiques ont été élaborés, comme l'algorithme Mersenne Twister (1997) ou encore la Méthode de Fibonacci lacunaire.

Mais aucun algorithme pseudo-aléatoire ne peut vraiment générer de suite à l'abri de toute analyse statistique, en particulier car la « graine » doit en théorie être elle-même aléatoire, et l'algorithme utilisé ne peut s'initialiser lui-même. Les générateurs cryptographiques actuels sont donc obligés de faire intervenir une part de hasard qui n'est pas générée par un moyen déterministe : on s'oriente vers des générateurs hybrides, possédant un algorithme de génération de nombres pseudo-aléatoires robuste, et s'initialisant sur un moyen physique de production de hasard[Géné] .

6.2. Qu'est-ce qu'une variable aléatoire ?

Une variable aléatoire est une variable dont la suite des valeurs n'est pas prédictible. Aussi longue que soit la série générée, il n'est pas possible de trouver une équation permettant de prédire le reste de la série à partir de celles déjà générées. Par exemple X est une variable aléatoire. Et $x_1, x_2, x_3, x_i \dots$ est la suite des valeurs générées successivement.

Toute variable aléatoire suit une loi de probabilité. Cette loi définit la répartition des valeurs au sein d'un intervalle. Par exemple, la loi « uniforme » définit une répartition uniforme des valeurs dans un intervalle, c'est à dire que toutes les valeurs de l'intervalle ont la même probabilité d'apparaître dans une longue série. [Hug]

6.3. Quelle est leur utilité ?

Il est fréquent d'avoir à utiliser des variables aléatoires. Les principaux domaines utilisateurs de ces variables sont les jeux, les simulations et la cryptographie. Qu'il s'agisse de déterminer la valeur d'un lancé de dés où de positionner un personnage dans un jeu vidéo, les besoins en hasard sont nombreux. Les logiciels de simulation numérique, permettant par exemple de tester virtuellement la résistance d'une installation nucléaire ont recours de façon massive aux variables aléatoires. Les logiciels de

cryptographie sont eux aussi très gourmands en nombres aléatoires pour la génération de clés secrètes incassables (ou presque).

6.4. Comment les générer ?

Le hasard vrai n'existe que dans la nature. Il est très difficile de le reproduire dans un ordinateur, sauf à relier celui-ci à un dispositif physique de mesure d'un phénomène naturel. Ces dispositifs sont très complexes et coûteux.

En revanche, il existe des algorithmes basés sur l'heure système qui fournissent un pseudo-hasard suffisant pour les applications courantes.

L'heure système possède une précision de l'ordre de la milliseconde, elle change donc très souvent de valeur. Elle est utilisée dans les générateurs aléatoires des langages de programmation les plus courants. Des calculs successifs sur les valeurs précédemment générées et sur cette heure système fournissent donc les valeurs des variables aléatoires.

Dans certains systèmes, d'autres paramètres entrent en ligne de compte. Par exemple les mouvements de la souris et les frappes au clavier peuvent être mémorisés afin d'influer sur la génération de nombres aléatoires. Mais ces paramètres ne sont pas parfaits : certaines touches sont beaucoup plus utilisées que les autres et la souris pointe souvent les mêmes zones. D'autres encore, comme l'activité du réseau, des supports de stockage, des processus du système, etc. peuvent être couplés à l'heure système. [Hug]

6.5. Les méthodes de générateur pseudo-aléatoire :[Géné]

6.5.1. La méthode de Von Neumann

En 1946, John von Neumann propose un générateur pseudo-aléatoire connu sous le nom de la méthode *middle-square* (carré médian). Très simple, elle consiste à prendre un nombre, à l'élever au carré et à prendre les chiffres au milieu comme sortie. Celle-ci est utilisée comme graine pour l'itération suivante.

Exemple :

Soit le nombre « 1111 ».

1. $1111^2 = 1234321$
2. on récupère les chiffres du milieu : 3432. C'est la sortie du générateur.
3. $3432^2 = 11778624$
4. on récupère les chiffres du milieu : 7786. et ainsi de suite.

Défauts : Von Neumann utilisa des nombres comportant 10 chiffres, le principe restant le même. Toutefois, la période du *middle-square* est faible. La qualité des sorties dépend de la graine, « 0000 » produit toujours la même séquence et constitue un « état absorbant » de l'algorithme. Von Neumann en était conscient, mais il craignait que des retouches *a priori* nécessaires n'apportent d'autres vices cachés. Sur l'ordinateur ENIAC qu'il utilisait avec sa méthode, il obtenait une génération 200 fois plus rapide que les résultats obtenus avec des cartes perforées. Selon Von Neumann, les générateurs basés sur du matériel ne pouvaient pas fonctionner correctement car il ne stockait pas les résultats (et on ne pouvait donc pas les vérifier). La méthode de Von Neumann montra vite ses limites lors d'applications utilisant des méthodes statistiques comme celle de Monte Carlo.

6.5.2. Méthode de Fibonacci :

Cette méthode est basée sur la suite de Fibonacci modulo la valeur maximale désirée :

$$x_n = (x_{n-1} + x_{n-2}) \bmod M \quad \text{avec } x(0) \text{ et } x(1) \text{ en entrée.}$$

On peut employer une variante :

$$x_n = (x_{n-1} + x_{n-k}) \bmod M \quad \text{avec } x(1) \dots x(k-1) \text{ en entrée.}$$

La qualité du générateur dépend de k et des nombres utilisés pour initialiser la suite. Ce générateur est par contre très simple à implémenter et ne consomme que peu de ressources.

6.5.3. Générateurs congruentiels linéaires

Introduits en 1948 par D. H. Lehmer sous une forme réduite (incrément nul), ils vont être généralisés et seront largement utilisés ensuite. Ils reposent sur une simple formule de récurrence :

$$x_{n+1} = (a \cdot x_n + c) \bmod m$$

Avec x_0 la graine (*seed* en anglais : un nombre employé pour produire une suite pseudo-aléatoire habituellement plus longue). En général, la graine est un nombre premier, mais

les contraintes exactes à son sujet dépendent de l'algorithme. Certaines graines peuvent conduire à des séquences dégénérées.

La période de ce générateur est au maximum de m , c'est-à-dire qu'elle est relativement courte puisque m est souvent choisi de manière à être de l'ordre de la longueur des mots sur l'ordinateur (par exemple : 2^{32} sur une machine 32 bits). Mais cette méthode présente un avantage : on connaît les critères sur les nombres a , c et m qui vont permettre d'obtenir une période maximale (égale à m).

Algorithme	A	C	M	Remarques
Randu	65539	0	2^{31}	Biaisé et fortement déconseillé
générateur de Robert Sedgewick	31415821	1	10^8	Intéressant mais déconseillé (essayer avec $X_0 = 0$)
Standard minimal	16807	0	$2^{31}-1$	

Tableau I.1 : Exemples d'algorithmes

6.5.4. Générateurs pseudo-aléatoires cryptographiques

Certains générateurs pseudo-aléatoires peuvent être qualifiés de *cryptographiques* quand ils font preuve de certaines propriétés nécessaires pour qu'ils puissent être utilisés en cryptologie. Ils doivent être capables de produire une sortie suffisamment peu discernable d'un aléa parfait et doivent résister à des attaques par exemple l'injection de données forgées de manière à produire des imperfections dans l'algorithme, ou encore des analyses statistiques qui permettraient de prédire la suite.

Dans la catégorie des générateurs suffisamment sûrs, on trouve :

- Yarrow
- Fortuna
- Blum Blum Shub (sûr mais lent)
- ISAAC

Une méthode courante pour générer de l'aléa en cryptographie consiste à « accumuler de l'entropie » (via diverses sources disponibles sur un ordinateur : temps entre deux accès au disque, taille de la mémoire, mouvements du pointeur de la souris...) et à faire passer le résultat dans une fonction de hachage cryptographique comme MD5 ou SHA-1. Ce principe est utilisé par Yarrow et Fortuna qui ne génèrent un nombre que lorsque l'entropie est suffisante.

7. Fonction de hachage : [InCr-09]

7.1. Principe

Une fonction de hachage est une fonction qui fait subir une succession de traitements à une donnée quelconque fournie en entrée pour en produire une « empreinte » servant à identifier la donnée initiale sans que l'opération inverse de décryptage soit possible. Le terme hachage évite l'emploi de l'anglicisme hash. Le résultat de cette fonction est par ailleurs aussi appelé somme de contrôle, empreinte, résumé de message, condense, condensat ou encore empreinte cryptographique.

Les fonctions de hachage sont conçues pour effectuer un traitement de données rapide : calculer l'empreinte d'une donnée ne doit coûter qu'un temps négligeable. Une fonction de hachage doit aussi éviter le plus possible les collisions (deux empreintes identiques alors que les données différaient). Selon l'emploi de la fonction de hachage, il peut être souhaitable qu'un infime changement de la donnée en entrée (un seul bit, par exemple) entraîne une perturbation conséquente de l'empreinte correspondante, rendant une recherche inverse par approximations successives impossible : on parlera d'effet avalanche.

7.2. MD5

En 1991, Ronald Rivest améliore l'architecture de MD4 et crée MD5 (Message Digest 5). C'est une fonction de hachage cryptographique qui permet d'obtenir pour chaque message une chaîne de 32 caractères (soit 128 bits) hexadécimaux avec une probabilité très forte que, pour deux messages différents, leurs empreintes soient différentes. Quelle que soit la taille de l'information en entrée (de 0 octets à plusieurs gigas). Cette transformation est donc irréversible (Dans le sens où on ne peut pas trouver l'information en entrée à partir d'une somme MD5).

En 1996, une faille grave (possibilité de créer des collisions à la demande) est découverte. En 2004, une équipe chinoise découvre des collisions complètes. MD5 n'est donc plus considéré comme sûr au sens cryptographique.

MD5 reste encore utilisé comme outil de vérification lors des téléchargements (par exemple, en FTP). Les sites affichent encore souvent la signature en MD5 de leurs fichiers. [InCr-09]

Le programme John the ripper permet de casser les MD5 triviaux par force brute. Des serveurs de "tables inverses" (à accès direct, et qui font parfois plusieurs giga-octets) permettent de les craquer souvent en moins d'une seconde.

Aujourd'hui, il est par exemple possible de créer des pages HTML aux contenus très différents et ayant pourtant le même MD5. La présence de codes de "bourrage" placés en commentaires, visibles seulement dans la source de la page web, trahit toutefois les pages modifiées pour usurper le MD5 d'une autre.

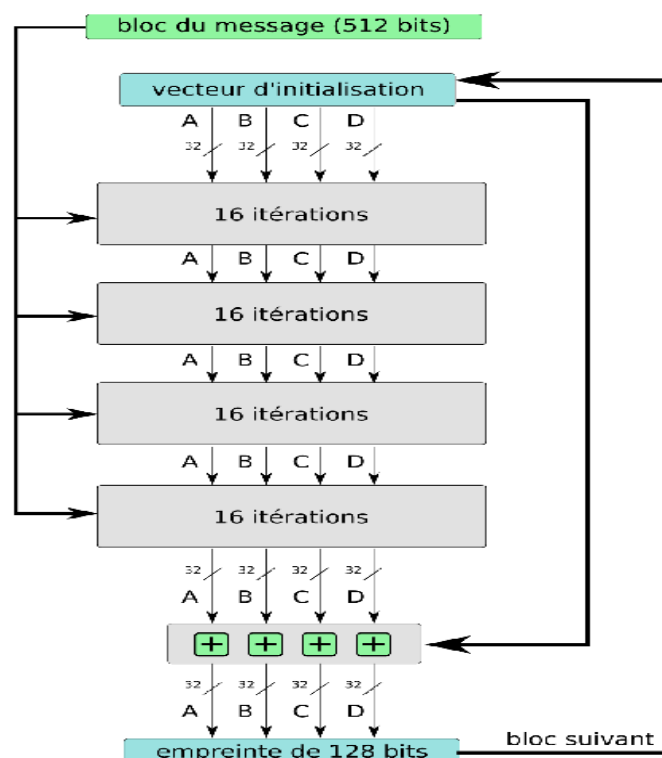


Figure N° I.3: Principe de MD-5 [InCr-09]

7.3. SHA-1

SHA-1 (*Secure Hash Algorithm*) est une fonction de hachage cryptographique conçue par la NSA (1995), et publiée par le gouvernement des Etats-Unis comme un standard fédéral de traitement de l'information. Elle produit un résultat de 160 bits (40 caractères). Même si on arrive à générer des collisions avec SHA-1. C'est-à-dire que l'on peut trouver deux messages au contenu aléatoire qui produisent la même signature. On ne sait toujours pas, à partir d'une signature donnée, forger un second message qui génère la même valeur. Or, c'est ce type d'attaque qui pourrait mettre en péril les applications comme PGP et l'authenticité des données.

Des versions offrant plus de sécurité sont également disponibles : SHA-256, SHA-384 et SHA-512. Comme leur nom l'indique, ces versions fournissent des signatures de 256, 384 et 512 bits.

Exemple :

SHA-1(Les poules se sont échappées des qu'on avait ouvert la porte) =
a187da360890f111566557aa6c197aa238dc533a

SHA-1(Les poules se sont échappées des **con** avait ouvert la porte) =
15166056114addee4bd717a1c45ae51389920a61

Comme vous le constatez, les deux phrases n'ont rien à voir entre elles ...

8. Opérateur ou-exclusif

Le primitive «ou exclusif » dénoté par *xor* et le symbole \oplus , est utilisé dans nombreux importants protocoles cryptographiques et dans des domaines différents, comme le protocole WEP (*Wired Equivalent Privacy*) pour les réseaux sans fil, il permet de protéger les échanges de données lors de communication wifi. Il existe nombreux importants des protocoles d'authentification dans les systèmes RFID qui exigent l'opérateur xor (voir chapitre 4).

Les caractéristiques de l'opérateur *xor* sont : (C1) d'associativité, (C2) de commutativité, (C3) l'existence d'un élément neutre 0 et (C4) nilomètre.

$$(x \oplus y) \oplus z = x \oplus (y \oplus z) \rightarrow (C1)$$

$$x \oplus y = y \oplus x \rightarrow (C2)$$

$$x \oplus 0 = x \rightarrow (C3)$$

$$x \oplus x = 0 \rightarrow (C4)$$

L'opérateur binaire XOR combine l'état de 2 bits selon le tableau suivant : [Opé]

Table de vérité XOR		
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

L'opérateur C est ^. Il agit sur chaque bit de la valeur :

```
unsigned a = 0xF0F0;
unsigned b = 0x00FF;
unsigned c = a ^ b; /* c == 1111 0000 0000 1111 soit
0xF00F */
```

9. Concaténation

Le terme concaténation (substantif féminin), du latin *cum* (« avec ») et *catena* (« chaîne, liaison »), désigne l'action de mettre bout à bout au moins deux chaînes. En programmation, on appelle la concaténation de deux chaînes de caractères, la chaîne formée de ces deux chaînes mises bout à bout.

Exemple : La concaténation de «Hello», « », «world !» est «Hello world !»

Le terme « concaténation » désigne également l'opération de concaténer des chaînes, la concaténation est une figure fondée sur une énumération de termes ordonnés selon une intensité croissante ou décroissante, il s'agit d'une idée d'expansion.[Con]

10. Protocoles cryptographiques :

Un protocole cryptographique ou protocole de sécurité est un ensemble de règles d'échange entre les participants d'un réseau (entités ou agents), basé sur les notions de crypto-systèmes qui permettent de sécuriser les communications dans un environnement hostile afin de réaliser une certaine fonctionnalité.

Le domaine d'Internet, de carte à puce, de réseau ad-hoc, de réseau de capteur, de réseau sans fil et systèmes RFID ont des buts plus spécifiques reliés par l'objectif du protocole et les caractéristiques de réseau.

10.1. Protocoles d'authentification:

Le protocole d'authentification est un protocole cryptographique qui assure la propriété d'authenticité. Cette authentification est, soit unilatérale ou mutuelle. On cite quelques protocoles largement utilisés dans le monde de communication des réseaux (surtout Internet) : EAP (*Extensible Authentication Protocol*), Kerberos, NSPK, PGP (*Pretty Good Privacy*).

On présente par exemple, le protocole d'authentification *Needham-Schroeder Public Key* (NSPK) [NS78]:

A et B disposent des clés publiques, K_a et K_b respectivement ;

$$A \rightarrow B : \{A, Na\}_{K_b} \quad (1)$$

$$B \rightarrow A : \{Na, Nb\}_{K_a} \quad (2)$$

$$A \rightarrow B : \{Nb\}_{K_b} \quad (3)$$

(1) l'agent A envoie son nom A et nonce Na . Ce message est chiffré par clé publique de B

(2) B reçoit le message $\{A, Na\}_{K_b}$ envoyé par A. Comme il a la clé privée lui permettant d'ouvrir le message, il comprend qu'A veut lui parler et renvoie le nonce Na ainsi qu'un autre nonce Nb qu'il vient d'engendrer, le tout chiffré avec la clé publique K_a d'agent A. B envoie donc à A le message $\{Na, Nb\}_{K_a}$.

(3) A reçoit le message $\{Na, Nb\}_{K_a}$, le déchiffre et reconnaît son nonce Na . Elle en déduit que l'agent B lui a répondu et elle lui renvoie son nonce Nb crypté avec sa clé publique pour lui signifier qu'elle connaît maintenant le message Nb . Elle envoie donc $\{Nb\}_{K_b}$.

Donc, quand A déchiffre (2), elle vérifie que Na est la bonne valeur et en déduit que B connaît Na et est bien authentifié. B fait de même avec (3).

10.2. Protocoles d'échange de clé :

D'autres types de protocoles assurent la confidentialité de clé symétrique générée et partagée par plusieurs participants, tels que : le protocole IKE (*Internet Key Exchange*), TLS (*Transport Layer Security*), et protocole AKA (*Authentication and Key Agreement*).

On présente le protocole Diffie-Hellman [DH76] qui été un protocole d'échange de clés entre deux agents A et B. Une variante simple de ce protocole, en trois étapes est présentée ci-dessous :

$$A \rightarrow B : g^{Na} \bmod p \quad (1)$$

$$B \rightarrow A : g^{Nb} \bmod p \quad (2)$$

$$A \rightarrow B : \{Nsec\}_K \quad (3)$$

(1) A génère le nonce Na et calcule l'exponentiation modulaire de g par Na , g^{Na} , où g et p sont des nombres adaptés à Diffie-Hellman connus de tous les participants. A envoie ensuite le message g^{Na} à l'agent B.

(2) l'agent B génère lui aussi un nonce Nb et calcule g^{Nb} puis $K = (g^{Na})^{Nb}$. Il envoie le premier à A et conserve le second comme clé partagée avec A. Dès que A reçoit le message g^{Nb} , il est capable de calculer $(g^{Nb})^{Na}$ et il a donc lui aussi la clé partagée. En effet, d'après les propriétés de l'exponentiation modulaire $K = (g^{Na})^{Nb} = (g^{Nb})^{Na}$.

(3) le message $\{Nsec\}_K$ est envoyé par A à l'agent B avec $Nsec$ une donnée qui doit rester secrète entre les agents A et B.

Le protocole de Diffie-Hellman souffre ainsi d'une attaque bien connue de type homme au milieu (*man in the middle*).