
# INTRODUCTION A LA SECURITE DES SYSTEMES D'INFORMATION

Année scolaire 2012 - 2013

Cours

FORMATION

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE LYON

**Auteur de la Ressource Pédagogique**
**Paul Ferrand**

membre de UNIVERSITÉ DE LYON

INSA LYON

# Introduction à la sécurité des systèmes d'information

Paul Ferrand

2012-2013

# Chapitre 1

# Notions de cryptographie

Avant de s'attaquer à la sécurité des systèmes, il est bon de connaitre un peu les différentes techniques existantes pour assurer la sécurité des communications. La science étudiant les *codes secrets* s'appelle la cryptographie, et celle étudiant les faiblesses de ces codes la *cryptanalyse*. Elles sont toutes les deux intimement liées, bien évidemment, et l'expertise dans l'une implique une bonne connaissance de l'autre. Ce principe reste général à tous les problèmes de sécurité que vous pourrez rencontrer, informatiques ou non ; vous ne pouvez pas être un bon expert en sécurité sans être potentiellement un bon voleur. J'insiste sur le *potentiellement*, le but n'étant pas d'être un voleur littéral, mais plutôt de maîtriser les techniques et le mode de pensée dans un cadre légal.

La cryptographie porte donc, au sens large, sur les techniques de *masquage* de l'information. C'est une discipline très ancienne, qui a été historiquement liée aux domaines diplomatiques et militaires pour lesquels le secret des correspondances est crucial. Aujourd'hui, tout le monde utilise une forme de cryptographie, en général sans le savoir. Votre carte bancaire, votre compte mail, votre page de commande Amazon et même votre mot de passe d'ouverture de session sont basés sur ce qu'on appelle des *primitives cryptographiques*, c'est à dire des fonctions mathématiques optimisées pour le masquage de l'information.

Quelques points de vocabulaire :

- Le *chiffrement* ou le *chiffre* est le terme technique du « code secret », et consiste à transformer un message pour le rendre indéchiffrable par un tiers.
- On parle de *texte clair* et de *texte chiffré*, et leurs équivalents anglais *plaintext* et *ciphertext*.
- La cryptanalyse cherche à *casser* les algorithmes de chiffrement et les primitives cryptographiques.
- *Crypter* n'existe pas en français, il s'agit d'un anglicisme. Néanmoins, il est passé dans le langage courant et je fais moi-même souvent cette erreur.

## 1.1 Chiffrements mono-alphabétiques

Historiquement, une des plus anciennes méthodes de chiffrement connue s'appelle le *code de César*. On dit que Jules César utilisait ces algorithmes pour chiffrer ses communications durant les guerres, mais en réalité, il utilisait ce chiffre pour ses communications personnelles ! Le fonctionnement du chiffre de César est extrêmement simple. Il suffit de prendre l'alphabet usuel, et de décaler les lettres de $k \leq 25$ pour obtenir l'alphabet chiffré correspondant. Ainsi, pour $k = 3$, on obtient l'équivalence $A \rightarrow D$, $B \rightarrow E$ ... $Z \rightarrow C$. On parle de chiffrement *mono-alphabétique*, car à chaque lettre correspond une et une seule substitution. La lettre chiffrée $D$ dans notre exemple précédent correspond, dans le message chiffré, uniquement à $A$. Il est donc possible de n'utiliser qu'un seul alphabet de *substitution* :

| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C |

On peut facilement exprimer un chiffre de César sous forme mathématique. Pour cela, on associe chaque lettre de l'alphabet dans l'ordre lexicographique usuel à un nombre de 0 à 25, et l'on utilise les principes de l'arithmétique entière *modulo* $N = 26$. Dans l'exemple que nous avons donné précédemment, à partir d'une lettre de l'alphabet $x$, on obtient la lettre chiffrée $c$ par l'opération :

$$c = x + k \mod N \qquad N = 26$$

Dans cette équation, la variable $x$ correspond au *texte clair*, la variable $c$ au *texte chiffré* et la variable $k$ à la *clé* du chiffrement. C'est cette clé qui indique comment déchiffrer le message. Si l'on inverse l'équation précédente, on a :

$$x = c + (N - k) \mod N \qquad N = 26$$

Un chiffre de César a donc 25 clés possibles. Ce nombre est extrêmement faible, et sous réserve qu'un cryptanalyste sache que l'émetteur a utilisé un chiffre de César, il ne lui faudrait pas très longtemps pour obtenir par un test exhaustif le décalage $k$ utilisé !

**Principe de Kerckhoffs**

Pendant des siècles, la cryptographie a été le fief gardé des gouvernements et des organisations secrètes, qui cachaient jalousement les algorithmes utilisés pour chiffrer leurs communications. En pratique, comme on le voit dans le cas du chiffre de César, connaitre l'algorithme utilisé permet de réduire *grandement* la difficulté de la cryptanalyse. Partant de ce principe, un professeur d'HEC au XIX$^{e}$siècle a formulé le principe suivant :

*La sécurité d'un système cryptographique ne doit jamais être basé sur le secret de l'algorithme utilisé, et doit rester indéchiffrable si un attaquant a la connaissance de tout le système à l'exception de la clé.*

Ce principe est aujourd'hui largement accepté par les cryptographes et les cryptanalystes du monde entier. En effet, il est au final plus *sûr* de faire reposer la sécurité du système sur la clé de chiffrement, et rendre les systèmes cryptographiques publics permet à toute la communauté scientifique d'étudier leur fonctionnement. Comme il est difficile de prouver mathématiquement qu'un système cryptographique est résistant aux attaques, ouvrir les systèmes à un maximum de chercheurs permet d'évaluer dans la pratique leur robustesse !

En gardant le principe du chiffre de César, il est possible de définir une forme plus sûre de chiffrement mono-alphabétique, en procédant à une permutation aléatoire de l'alphabet. Comme précédemment, on chiffre un message en *substituant* la lettre rencontrée par l'équivalent dans l'alphabet permuté. Ce type de chiffrement s'appelle donc un chiffrement par *substitution*.

| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Y | M | I | H | B | A | W | C | X | V | D | N | O | J | K | U | Q | P | R | T | F | E | L | G | Z | S |

Le fonctionnement ressemble donc au chiffre de César, mais le nombre de clés possibles augmente énormément. En pratique – et je vous encourage à comprendre d'où vient ce résultat, il s'agit d'un simple raisonnement combinatoire – ce nombre est de $26! = 400 \times 10^{24}$. Il y a donc peu de chance de pouvoir casser ce code en procédant de manière brutale et en testant chaque clé. Pourtant, en pratique, il est très simple de cryptanalyser ces codes. Nous sommes en fait trahis par la langue, mais prenons un exemple :

```
PHTZRUS H DWMU M'UJHCUX YU YUCHOX SRA MU VRAUHR YR
EAWBUSSURA. EWRA UDOQUA MUS UXXROS OM XU M'UXDWOQ EHS
EHA CHOM CHOS OM USQ EHAQO EWRA BHOAU YUS EKWQWTWEOUS.
AUXYUI-DWRS TU SWOA YUDHXQ MU AUSQHRAHXQ.
```

Tout d'abord, dans ce texte, on voit apparaître des artefacts aidant au déchiffrement. Les espaces en premier lieu nous aide à découper les mots, et vont faciliter la découverte de l'alphabet de substitution.

Nous voyons que `H` est une lettre seule, ce qui veut dire que *si le texte est en français* elle correspond à `a` ou `y`. La lettre `M` précèdre une apostrophe dans deux cas, et correspondrait donc à `l`, `d`, `n`, `m` ou `j`. En procédant ainsi, le cryptographe peut obtenir une partie de la clé et jouer au pendu avec le reste en émettant des hypothèses.

Une bonne pratique est donc de supprimer la ponctuation dans un premier temps, et de remplacer les espaces par une lettre rare dans la langue du message, comme le `w` en français. Le message chiffré aura déjà l'air beaucoup plus difficile à craquer, mais pas impossible. Les chiffrements mono-alphabétiques laissent apparaître, quoiqu'il arrive, les *statistiques* de la langue, c'est à dire la fréquence d'apparition des lettres, des digrammes, des trigrammes ou même des mots.

```
SCSQC OXCCJ SLJWX JYCXQ CGGCA CXJEC NWYXQ XQSPL YJTLX
JQXLT WQGJS CSQCO XCCJS LJWXJ YCCGA WGJYG JCRWQ IGLIC
ILICW GAEQJ NCSSC NQHYC OWYAG LJJCQ GEXCM TWQGJ AQJWJ
HYCRW QNCBY JAQJW JCJCA OWYAA LICXC TLXJQ JSLGQ RLSSC
ICXRL NWRRC XCQSO WYAPL YJTYX ICXLO CNHY JXCIX LQGAE
```

Dans le texte précédent, la lettre `C` est clairement la plus courante, avec 36 apparitions sur 225 lettres. On peut donc inférer qu'il s'agit probablement de la lettre `e`, la plus fréquente en français à hauteur de 16%, ou bien de l'espace si le cryptographe a choisi de remplacer les espaces par des lettres. On peut également s'intéresser à la distribution des digrammes, les groupes de deux lettres. Ainsi, dans cet exemple, les digramme `XC` et `CX` sont les plus courant, avec 6 occurrences chacun ! Si j'avais à craquer ce texte, j'infèrerai que ces deux digrammes sont `ET` et `TE`, deux des plus fréquents en français. Si le temps le permet, vous aurez l'occasion en TP de procéder à une cryptanalyse de chiffrement par substitution. Je vous encourage sinon à aller voir les références de ce cours, ou bien à lire *Le scarabée d'Or*, où l'auteur fait déchiffrer à son héros un message ancien indiquant l'emplacement d'un trésor, en notant les détails de son raisonnement.

**Fréquence des lettres et digrammes en français**

L'analyse fréquentielle a été décrite par Al-Kindi, au IX$^{e}$ siècle, dans son manuscrit *Sur le déchiffrement des messages cryptographiques*. Il a probablement utilisé cette méthode pour déchiffrer les documents administratifs de la dynastie abbasside ayant précédé son gouvernement. Partant d'un corpus de textes courant, on peut tirer la fréquence des lettres et digrammes en français :

| | | | |
|---|---|---|---|
| a | 9,42 | n | 7,15 |
| b | 1,02 | o | 5,14 |
| c | 2,64 | p | 2,86 |
| d | 3,39 | q | 1,06 |
| e | 15,87 | r | 6,46 |
| f | 0,95 | s | 7,90 |
| g | 1,04 | t | 7,26 |
| h | 0,77 | u | 6,24 |
| i | 8,41 | v | 2,15 |
| j | 0,89 | w | 0,00 |
| k | 0,00 | x | 0,30 |
| l | 5,34 | y | 0,24 |
| m | 3,24 | z | 0,3 |

| | |
|---|---|
| ES | 3,15 |
| LE | 2,46 |
| EN | 2,42 |
| DE | 2,15 |
| RE | 2,09 |
| NT | 1,97 |
| ON | 1,64 |
| TE | 1,63 |
| ER | 1,63 |
| SE | 1,55 |

Apprendre ces tables presque par coeur vous permet au passage d'être totalement imbattable au pendu !

## 1.2 Chiffrements poly-alphabétiques

L'analyse fréquentielle permet donc de craquer n'importe quel chiffrement par substitution, sous réserve d'avoir accès à suffisament de texte chiffré pour voir apparaître des distributions de lettres et de digrammes utilisables. Pour pallier à ce problème, les cryptographes ont développé à partir du XVI$^{e}$ siècle les

chiffrements dits *poly-alphabétiques*. Dans un chiffrement mono-alphabétique, les lettres du texte chiffré correspondent toujours à *une et une seule* lettre dans le texte clair. Ainsi, par une technique arbitraire, si nous décidions que la lettre `C` correspondait à la lettre `e`, il en allait de même pour toutes les lettres `C` du message. Les chiffrements poly-alphabétiques lèvent cette restriction, qui rend le système cryptographique sensible aux attaques fréquentielles. En pratique, chaque lettre du texte clair sera chiffré par un alphabet de substitution propre à cette lettre. L'alphabet sera lui déterminé pour chaque lettre par le système cryptographique, et l'ensemble des alphabets formeront donc la *clé* du message.

L'exemple le plus simple et le plus connu de chiffrements poly-alphabétiques est le chiffrement de Vigenère, inventé par le diplomate français Blaise de Vigenère en 1586 et basé sur des travaux antérieurs de mathématiciens italiens. Le fonctionnement est très simple, ce qui a contribué à sa popularité jusqu'au début du XX$^{e}$ siècle. Les interlocuteurs choisissent un mot qui servira de clé pour le message, par exemple `LION`. Cette clé est répétée au dessus du message clair, et fournit à chaque lettre un décalage correspondant à l'index de la lettre dans l'alphabet. Les lettres sont ensuite chiffrées par un code de César en utilisant ce décalage.

| **Clé** | L | I | O | N | L | I | O | N | ... |
|---|---|---|---|---|---|---|---|---|---|
| **Décalage** | 12 | 9 | 15 | 14 | 12 | 9 | 15 | 14 | ... |
| **Clair** | L | E | X | E | M | P | L | E | ... |
| **Chiffré** | W | M | L | R | X | X | Z | R | ... |

Bien que ce chiffre soit basé sur la substitution simple de César, sa cryptanalyse est nettement plus complexe ! On voit en effet que la lettre `e` du message clair a été associée dans le message chiffré aux lettres `M` et `R`, et que la lettre `X` correpond à la fois à `m` et `p`. Les chiffrements poly-alphabétiques masquent ainsi la distribution fréquentielle de la langue dans les messages chiffrés, et rendent plus difficile la cryptanalyse. L'aparté plus en avant dans cette section présente la méthode utilisée pour casser le chiffre de Vigenère, mais il existe des chiffrements poly-alphabétiques bien plus résistants encore. Ainsi, la machine Enigma utilisée pour les communications de l'Allemagne pendant la seconde guerre mondiale a tenu en échec la somme des cerveaux combinés de l'Angleterre et des États-Unis jusqu'en 1943, et elle était fondamentalement un chiffrement poly-alphabétique. Tout comme le chiffre de Vigenère, la faiblesse de ces méthodes tient dans le choix des alphabets de substitution. Là où Vigenère utilisait un César, Enigma utilisait un processus mécanique complexe qui présentait au final un biais statistique. A force de récupérer des messages chiffrés, les cryptanalystes des Alliés ont petit à petit réussi à reconstruire un équivalent de la machine et à déterminer les alphabets utilisés. Il est fort probable que cette avancée ait joué un grand rôle dans leur victoire, les Allemands ignorant que leurs communications étaient désormais interceptées par les États-Unis.

**Casser le chiffre de Vigenère**

Le chiffre de Vigenère a été réputé inviolable jusqu'en 1863 où Friedrich Kasiski a présenté une attaque désormais standard sur cette méthode de chiffrement. Il est fort probable que des méthodes de déchiffrement aient existé auparavant, mais sans être révélées. En effet, si les gouvernements aiment à garder secrets leurs algorithmes de chiffrement, il est encore plus bénéfique de pouvoir casser ceux des voisins sans que ceux-ci le sachent ! A priori, le chiffre de Vigenère cache la distribution statistique du message d'origine, mais en pratique si l'on connait la taille de la clé utilisée, il suffit de cryptanalyser un certain nombre de chiffres de César pour obtenir le message. On peut également se baser sur un mot connu dans le message d'origine, comme une signature ou une ponctuation (`STOP` dans les télégrammes). Si l'on ne connait pas la taille de la clé, il suffit de la deviner ! Prenons un exemple :

```
LRAFN SURTD ESNVV PFSPU CSXAC ZLHNF NVZJK SCLBJ IFQWC
HLSWI NDNFG MGIXF NVQTG BWZQX EBGCC COCKK NPLDC IKPBZ
ORGHB ZGJRE TFVZN DNFHR GGXRG JNWHH QTMEB SUICC WSBMM
JLQGJ MXKRG MYCMZ GICHM OWIYC M
```

Le sigle NDNF est répété dans le message, ainsi que les sigles GJ et YCM. Ces sigles sont respectivement séparés par 54, 24 et 12 lettres. Comme un chiffrement poly-alphabétique masquerait ce genre de biais statistiques avec une forte probabilité, des répétitions comme celles-ci sur un message aussi court sont dues au fait que les mêmes (di-tri-quadri-)grammes du texte clair ont été chiffrés avec la même partie de la clé ! Cette dernière est donc forcément un diviseur commun de 54, 24 et 12. On peut donc tester les clés de taille 3, 4, 6 (qui est la taille de la clé que j'ai utilisée, CRYPTO) jusqu'à obtenir une cryptanalyse sensée.

Il existe une méthode un peu moins artisanale de procéder à cette analyse. On utilise pour cela une métrique statistique appelée *indice de coïncidence*, qui est définie comme suit, avec $f_i$ la fréquence de chaque lettre dans le message chiffré :

$$\mathrm{IC} = n \sum_{i=1}^{n} f_i^2$$

Sans entrer dans les détails, cette métrique va nous indiquer le décalage entre notre texte et un texte où tous les symboles sont équiprobables, où dans ce cas on aura IC ≈ 1. Chaque langue aura un indice de coïncidence spécifique, reporté sur la table ci-contre. Cette métrique permet entre autres, pour un chiffrement mono-alphabétique, de connaître le langage utilisé dans le texte clair.

| **Langue** | **IC** |
|---|---|
| Anglais | 1,73 |
| Français | 2,02 |
| Allemand | 2,05 |
| Espagnol | 1,94 |

On utilise l'indice de coïncidence pour craquer le chiffre de Vigenère de la manière suivante. On suppose une taille de clé (2, 3, ...) et pour chaque taille on calcule l'indice de coïncidence du message à partir des lettres espacées de la taille supposée de la clé. Si l'on trouve un indice de coïncidence proche de 1, alors l'hypothèse de taille de clé n'est pas correcte. Dans le cas contraire, il est probable que l'on ait rencontré la bonne taille, ou un multiple de cette taille. La table suivante donne un exemple de cette utilisation, où la clé est de taille 5. Sur une taille de message courte, nous pouvons donc identifier la taille de la clé et procéder à une analyse fréquentielle qui donnera rapidement les décalages utilisés pour les codes de César.

```
QPWKAL VRXCQ ZIKGR BPFAE OMFLJ MSDZV
DHXCX JYEBI MTRQW NMEAI ZRVKC VKVLX
NEICF ZPZCZ ZHKML VZVZI ZRRQW DKECH
OSNYX XLSPM YKVQX JTDCI OMEEX DQVSR
            XLRLK ZHOV
```

| Hypothèse de taille | Indice de coïncidence |
|---|---|
| 1 | 1,12 |
| 2 | 1,19 |
| 3 | 1,05 |
| 4 | 1,17 |
| 5 | **1,82** |
| 6 | 0,99 |
| 7 | 1,00 |
| 8 | 1,05 |
| 9 | 1,16 |
| 10 | **2,07** |

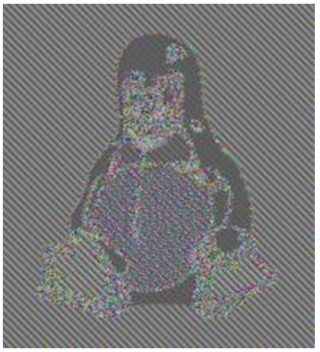
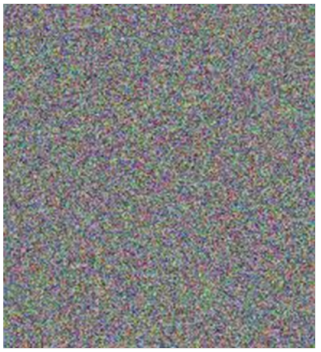
## 1.3 Chiffrements modernes et cryptographie symétrique

La seconde guerre mondiale a fait entrer la cryptographie dans le domaine industriel et scientifique. En particulier, tous les mathématiciens de l'époque étaient recrutés par les armées pour servir de cryptanalystes, et cela a généré un grand nombre d'avancées dans le domaine. En particulier, Claude Shannon a énoncé dans un papier de 1948 deux principes de base pour la conception de primitives cryptographiques :

– L'algorithme doit aider à la *confusion* entre la clé et le message. Cela revient, idéalement, à faire en sorte que chaque symbole du message chiffré dépendent du message clair et de tous les symboles de

la clé. Cette dépendance doit également apparaître aléatoire pour un attaquant
– L'algorithme doit *diffuser* les propriétés statistiques du message clair. Contrairement au principe de confusion, qui traitait de la relation entre le message chiffré et la clé, la diffusion implique que chaque lettre du message chiffré doit dépendre d'un maximum de lettres du message clair, afin de « cacher » sa distribution statistique.

Vous êtes maintenant capable de comprendre l'intérêt derrière ces deux maximes. La première porte sur des méthodes de chiffrement comme celle de Vigenère, ou la relation entre la clé et le message chiffré est trop déterministe et permet une cryptanalyse. La seconde vise à décourager les approches basées sur l'analyse fréquentielle du message chiffré. On peut illuster ces deux principes en image.

Dans ce cas, chaque bloc de pixel est chiffré indépendament en utilisant une substitution semblable à un chiffrement poly-alphabétique. On voit très clairement apparaitre l'image d'origine dans la seconde image, même si celle-ci est brouillée ! La confusion est bonne, mais la diffusion ne l'est clairement pas, et il faut développer encore l'algorithme pour retrouver une image proche d'un bruit aléatoire.

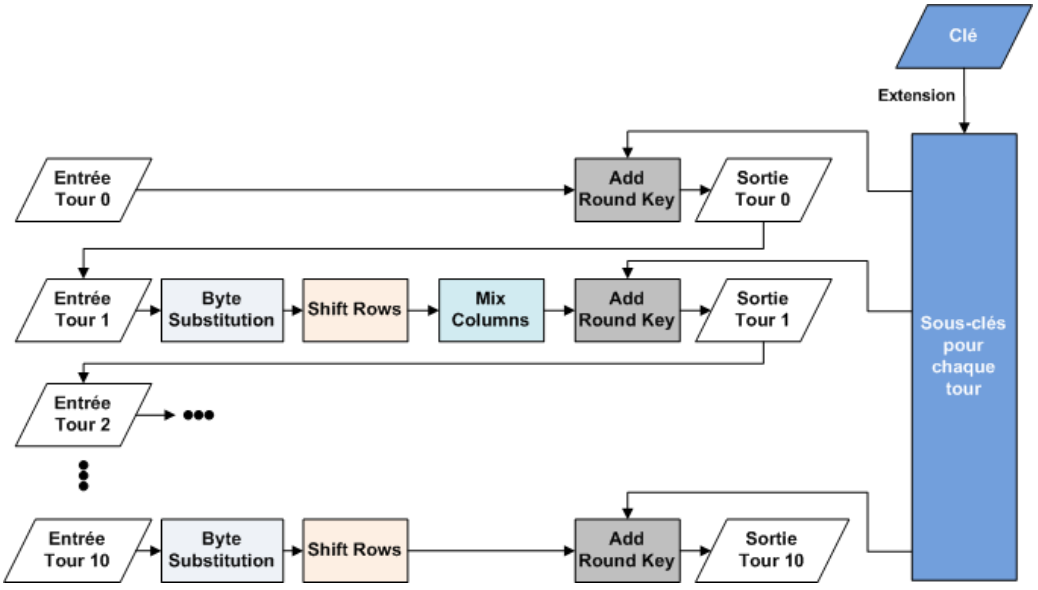
**Alan Turing et Claude Shannon, *Codebreakers* et pères de l'informatique**

La cryptanalyse durant la seconde guerre mondiale a coïncidé avec les premiers développement de l'informatique sous la forme que nous connaissons actuellement, et nombre de services de renseignements se sont attelés à la constructions de *bombes*, d'énormes circuits électro-mécaniques visant à analyser les statistiques des textes transmis chiffrés. Les premières bombes ont en fait été construites en Pologne, et transférées en Angleterre au prix de la vie de leurs concepteurs lors de l'invasion de la Pologne par Hitler au début de la guerre.

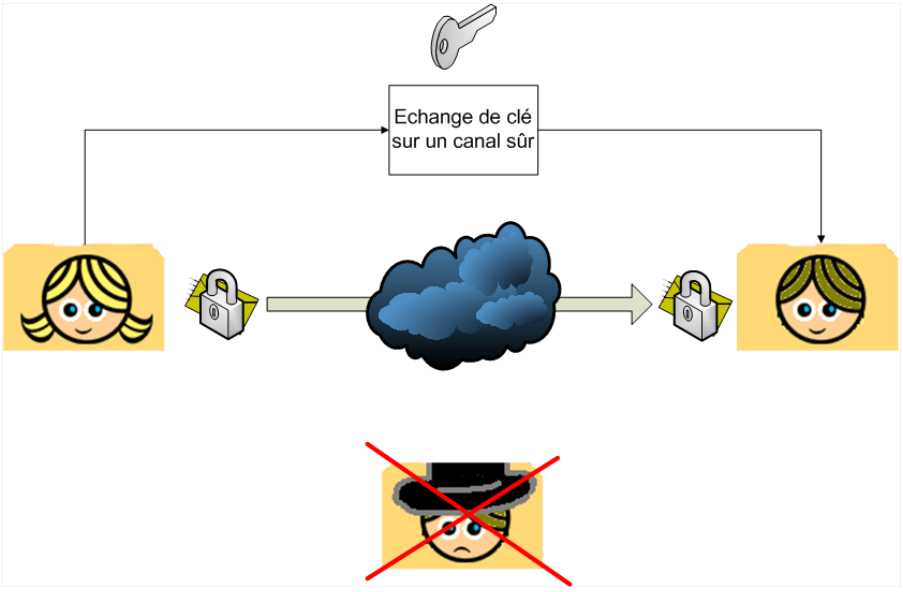
Parmi les mathématiciens en charge de la cryptanalyse en Angleterre, le plus influent était probablement Alan Turing. Turing est largement considéré comme étant le père de l'intelligence artificielle et de l'informatique en tant que discipline scientifique, et ses travaux sur la théorie de la *complexité* sont encore la référence aujourd'hui. Il a entre autre prouvé, d'une manière différente de Gödel, le problème d'indécidabilité posé par Hilbert au début du XX$^{e}$ siècle en inventant 30 ans avant l'heure un « ordinateur » basé sur un nombre restreint de commandes de base. Durant la guerre, la *bombe* qu'il a perfectionnée à partir du travail des cryptanalystes polonais a permis « de réduire la durée de la guerre de 2 à 4 ans », d'après le responsable des services de renseignements britannique.
Turing était homosexuel, un crime à l'époque en Angleterre, et a été condamné en 1952 pour indécence après avoir avoué une relation avec un autre homme. Sa condamnation lui ota son accès à son laboratoire et sa recherche, et le força à subir une castration chimique à base d'oestrogènes. Turing a été retrouvé mort 1 an plus tard, empoisonné au cyanure. Sa mort a été attribuée à un suicide mais des doutes subsistent encore.

Claude Shannon était un mathématicien américain. Il est principalement connu dans la communauté scientifique comme le fondateur de la *théorie de l'information*, sur laquelle se base la quasi-totalité de la théorie de télécommunications aujourd'hui. Également *codebreaker* pendant la guerre, il a travaillé avec Turing sur la cryptanalyse de l'Enigma navale. Ses contributions ont cependant été d'ordre théorique, avec la publication après la guerre des maximes présentées au début de cette section.
Ces travaux l'ont amené à formuler les bases de la théorie de l'information et des *codes correcteur d'erreur*. La correction d'erreur et la cryptanalyse sont en effet mathématiquement très liées ; dans un cas, on protège le message d'un attaquant cherchant à le déchiffrer, et dans l'autre on le protège des déformations dues à l'environnement. Shannon a également été actif en traitement du signal, ayant montré qu'il existait une manière optimale de passer d'un signal analogique à un signal numérique. Il a de plus été durant son master le premier a montrer qu'un circuit électronique pouvait résoudre n'importe quelle équation formulée selon l'algèbre de Boole, ouvrant ainsi la voie aux ordinateurs modernes.

Les méthodes de chiffrement modernes utilisent ces deux principes. En particulier, le standard actuel en la matière, l'AES (pour *Advanced Encryption System*) fonctionne de la manière suivante. Le message est découpé en blocs de 16 symboles, et chaque bloc passe à travers une transformation formée de 4 opérations simples.



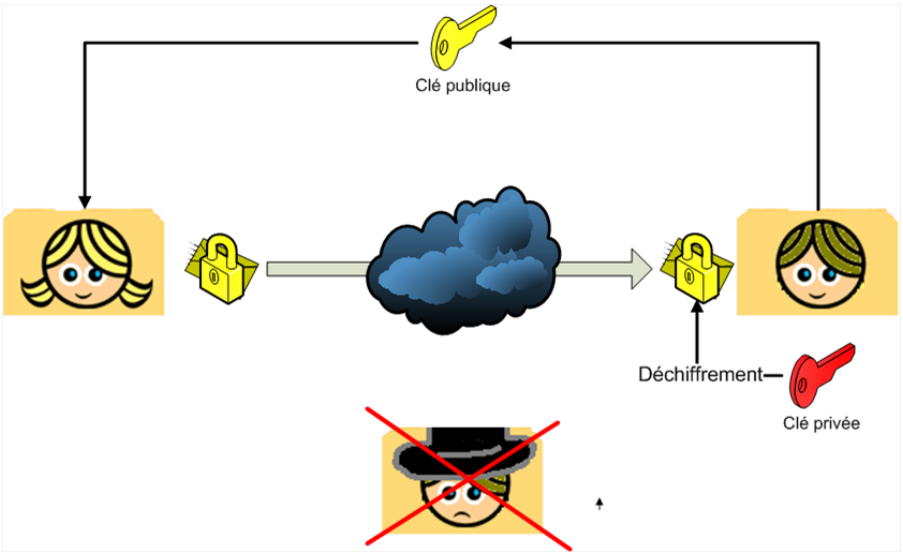

En amont du chiffrement, la clé est étendu en un certain nombre de sous-clés qui seront additionnées à chaque *tour* de l'algorithme. Durant un tour, on effectue une substitution déterministe à la manière d'un chiffrement mono-alphabétique, puis l'on *diffuse* les symboles à l'intérieur de la matrice formée par le bloc en inversant les lignes et en appliquant des opérations linéaires sur les colonnes. La première et la dernière étapes assurent donc la confusion, et les deux étapes intermédiaires la diffusion. Au bout d'une dizaine de tours, on obtient un bloc chiffré. Chacune de ces opérations étant inversible, il suffit d'effectuer le trajet inverse à travers l'algorithme pour déchiffrer le message. Un interlocuteur ayant connaissance de la clé pourra donc procéder à son extension en sous-clés et obtenir le texte clair.




On parle dans ce cas de chiffrement symétrique. L'émetteur et le destinataire utilisent un algorithme de chiffrement et de déchiffrement et une clé *identique et partagée*. La sécurité de la communication est donc assurée tant qu'un attaquant n'aura pas connaissance de la clé, selon le principe de Kerckhoffs. La difficulté provient du partage de la clé, qui doit être échangée de manière sécurisée entre les deux interlocuteurs. Si une rencontre physique peut avoir lieu, il n'y a pas de problème, mais il faut mettre en oeuvre d'autres techniques pour pouvoir communiquer la clé de manière sûre à travers Internet par exemple.

## 1.4 Cryptographie asymétrique

Ce problème d'échange de clé est fondamental en cryptographie, et est resté insoluble jusqu'en 1970 où Whitfield Diffie a présenté une idée révolutionnaire. Le principe était d'utiliser deux clés différentes ; une clé *publique* servant à chiffrer le message, et une clé *privée* servant à le déchiffrer. Tant qu'il n'est pas possible de deviner la clé privée à partir de la clé publique, le système est sécurisé. La clé publique est accessible librement et seul le détenteur de la clé privée peut déchiffrer le message.



**La bonne idée de Diffie et Hellman**

Bien que Diffie ait présenté son idée, il a fallu attendre 1977 pour voir apparaitre un système cryptographique asymétrique permettant l'envoi et la réception de message. Néanmoins, Diffie et Hellman ont proposé un algorithme très intelligent pour créer une clé partagée entre deux interlocuteurs, résolvant ainsi un des problèmes de la cryptographie symétrique. Comme tous les systèmes de cryptographie asymétrique, leur algorithme se base sur des fonctions *à sens unique*, qu'il est facile de calculer dans un sens mais difficile dans l'autre. Le plus connu est sans doute le problème de factorisation en nombres premiers ; à partir de 2 facteurs premiers $a$ et $b$, il est simple de calculer leur produit $c = a.b$, mais il est extrêmement difficile de retrouver $a$ et $b$ en ne connaissant que le résultat $c$ de l'opération, si $a$ et $b$ sont suffisament grands.

L'opération utilisée par Diffie et Hellman est proche. Il s'agit du problème du *logarithme discret*, qui est l'inverse de l'exponentiation discrète :

$$A = g^a \mod p$$

Comme pour la factorisation en nombre premier, il est extrêmement difficile pour de grandes valeurs de $p$ de découvrir l'exposant $a$ utilisé à partir de la valeur de $A$, $g$ et $p$. L'algorithme est le suivant. Alice et Bob cherchent à se créer une valeur secrète $s$ sans jamais l'échanger directement entre eux.

| Alice | | | | Bob | | |
|---|---|---|---|---|---|---|
| Secret | Public | Calcule | Envoie | Calcule | Public | Secret |
| $a$ | $p$ et $g$ | | $(p,g) \rightarrow$ | | | $b$ |
| $a$ | $p$, $g$ et $A$ | $A = g^a \mod p$ | $A \rightarrow$ | | $p$ et $g$ | $b$ |
| $a$ | $p$, $g$ et $A$ | | $\leftarrow B$ | $B = g^b \mod p$ | $p$, $g$, $A$ et $B$ | $b$ |
| $a$ et $s$ | $p$, $g$, $A$ et $D$ | $s = B^a \mod p$ | | $s = A^b \mod p$ | $p$, $g$, $A$ et $B$ | $b$ et $s$ |

En terme moins formels, on peut donner la description suivante :

1. Alice a un secret $a$ entier, quelconque. Elle choisit un grand nombre premier $p$ et une base $g$ qu'elle envoit à Bob, et elle envoit également la valeur $A = g^a \mod p$.
2. Bob a un secret $b$ entier, quelconque. Ayant reçu $g$ et $p$, il renvoit à Alice $B = g^b \mod p$.
3. Les deux parties calculent $s = B^a \mod p = A^b \mod p = g^{a.b} \mod p$ en utilisant leurs secrets respectifs, et sont donc en possession d'une valeur $s$ partagée sans jamais avoir envoyé $a$ et $b$ sur le réseau.

Un attaquant ne connaissant que $A$, $B$, $g$ et $p$ ne peut pas directement déduire la valeur de $s$. Pour cela, il aurait besoin de la valeur de $a$ *ou* de $b$, qu'il ne peut obtenir qu'en résolvant le problème du logarithme discret sur $A$ ou $B$ !

La référence actuelle des systèmes de cryptographie asymétrique s'appelle le RSA, du nom de ses inventeurs Ronald Rivest, Adi Shamir et Leonard Adleman, en 1977. Le fonctionnement est assez simple, mais requiert des notions d'arithmétique modulaire [1]. Prenez néanmoins le temps de lire cette page, qui vous donne une vision de la subtilité mathématique présente dans la cryptographie moderne. RSA doit générer 2 clés, selon l'algorithme suivant :

1. On choisit 2 nombres premiers $p$ et $q$.
2. La première partie de la clé publique est $n = pq$, qui devient le *modulo* dans les calculs arithmétiques qui s'ensuivent.

1. Au passage, il s'agit d'un sujet mathématique passionant, je vous encourage à vous y intéresser si vous avez un jour le temps ! En attendant, je suis à votre disposition si quelque chose vous échappe.

3. On calcule ensuite $A = (p-1)(q-1)$.
4. On choisit ensuite $e < A$ tel que $e$ soit premier avec $A$ (en pratique c'est simple, il suffit de choisir un nombre premier $e$ qui ne soit pas un diviseur de $A$). Le nombre $e$ forme la seconde partie de la clé publique. On peut donc publier $(n,e)$ aux yeux de tous.
5. Trouver $d = e^{-1} \mod A$. Il s'agit d'un inverse modulaire, donc on doit revenir à la définition d'inverse dans un anneau : $de = 1 \mod A$.
6. Le couple $(n,d)$ forme la clé privée du cryptosystème.

La sécurité vient du fait que pour former la clé privée, il nous faut calculer $A$. Pour un attaquant ne connaissant que $N$ le seul moyen d'y arriver est de factoriser $N$ pour découvrir $q$ et $p$, un problème très difficile si $p$ et $q$ sont grands (une clé de 2048 bits est composée de plus de 600 chiffres décimaux).

Pour chiffrer un message quelconque $M$, on le découpe en morceaux. Ces blocs deviennent les symboles, ou l'alphabet, du message, et on leur assigne un nombre $m$ tel que $m < n$. Pour chiffrer un bloc, on calcule :

$$c = m^e \mod n$$

Le déchiffrement s'effectue avec la clé privé de manière relativement simple :

$$m = c^d \mod n$$

Il est à noter que bien que le principe ne soit pas compliqué pour un ordinateur, quand les nombres sont si grands le temps de calcul est quand même élevé. En moyenne, un chiffrement à clé publique prendre 1000 fois plus de temps qu'un chiffrement à clé secrète. On utilisera plutôt les chiffrements à clé publique pour la signature de message, le chiffrement d'informations hautement sensibles (transactions bancaires) ou bien l'échange de clés secrètes. Par exemple, en vous connectant sur le site de votre banque, vous créez une clé secrète de chiffrement qui est envoyée à la banque, chiffrée à l'aide de la clé publique de la banque. Ensuite vous pouvez vous échanger des messages en chiffrant simplement à l'aide de la clé partagée, ce qui rend la communication plus fluide. Il n'est pas évident de voir immédiatement pourquoi RSA fonctionne. Pour cela, on a besoin de deux résultats d'arithmétiques :

1. *(Petit théorème de Fermat)* Soit $p$ un nombre premier et $a$ un nombre quelconque, premier avec $p$. On a alors $a^{(p-1)} = 1 \mod p$.
2. *(Théorème des restes chinois)* Soit $p$ et $q$ deux nombres premiers entre eux, et $(a,b)$ deux entiers quelconques inférieurs à $p$ et $q$. Si $a = b \mod p$ et $a = b \mod q$ alors $a = b \mod (pq)$.

On a :

$$ed = 1 \mod A \Leftrightarrow \exists h \in \mathbb{Z} \quad ed - 1 = h(p-1)(q-1)$$

On veut prouver que :

$$c^d = (m^e)^d = m \mod n$$

On peut vérifier que, par le petit théorème de Fermat :

$$(m^e)^d = m^{ed} = m \cdot m^{ed-1} = m \cdot m^{h(p-1)(q-1)} = m \cdot 1^{h(q-1)} = m \mod p$$

De même :

$$(m^e)^d = m^{ed} = m \cdot m^{ed-1} = m \cdot m^{h(p-1)(q-1)} = m \cdot 1^{h(p-1)} = m \mod q$$

On applique le théorème des restes chinois pour obtenir :

$$(m^e)^d = m \mod pq \quad \Leftrightarrow \quad (m^e)^d = m \mod n$$

**Exemple d'utilisation de RSA**

On prend des valeurs simples, par exemple $p = 2$ et $q = 5$. Dans ce cas, $A = 4$ et il nous faut un exposant $e$ qui soit premier avec 4 et inférieur à 4. Pas trop le choix, on prend $e = 3$. On cherche $d$ tel que $de = 1 \mod A$, ce qui nous donne par exemple $d = 7$. On aurait aussi 3 mais en pratique on a souvent $d \neq e$.

Comme on a $n = 5 \times 2 = 10$, donc notre alphabet doit contenir moins de 10 lettres. Choisissons par exemple $\{E,N,A,R,S,I,T,U,L\}$ et chiffrons le mot LUTIN (dont la représentation sera (9,8,7,6,2) [a] :

| Étape : | | | | | |
|---|---|---|---|---|---|
| Texte clair | L | U | T | I | N |
| Equivalent | 9 | 8 | 7 | 6 | 2 |
| $m^e$ | 729 | 512 | 343 | 216 | 8 |
| $m^e \mod n$ | 9 | 2 | 3 | 6 | 8 |

Pour le déchiffrement, on procède de manière similaire :

| Étape : | | | | | |
|---|---|---|---|---|---|
| Texte chiffré | 9 | 2 | 3 | 6 | 8 |
| $c^d$ | 4782969 | 128 | 2187 | 279936 | 2097152 |
| $c^d \mod n$ | 9 | 8 | 7 | 6 | 2 |
| Equivalent texte | L | U | T | I | N |

*a.* Sur des nombres si petits la calculette arrive à suivre. Pour trouver la valeur d'un grand nombre $a \mod 10$, rappelez-vous qu'il suffit de regarder la dernière décimale !

## 1.5 Fonctions de hachage et signature

Notre (courte) présentation des primitives cryptographiques se termine sur les *fonctions de hachage*. Dans sa forme la plus simple, une fonction de hachage est une fonction mathématique qui renvoie une valeur d'une taille fixe et prédéterminée. Par exemple, pour un message $x$ de $N$ lettres, où $N$ est un entier arbitraire, la fonction de hachage $h(x)$ vous renverra toujours une valeur représentée par $B$ bits de données, appelé un *haché*. La taille des hachés dépend de la fonction, mais est souvent de 128 ou 256 bits.

Cette fonction n'est cependant pas choisie complètement au hasard, et doit respecter quelques règles pour être utile dans un système cryptographique :

- Pour un une valeur de haché $y = h(x)$ donnée, il doit être difficile de retrouver $x$ (résistance aux *pré-images*).
- Il doit être difficile de trouver $x_1$ et $x_2$ tels que $h(x_1) = h(x_2)$ (résistance aux *collisions*).
- Pour un $x$ donné, il doit être difficile de trouver $x'$ tel que $h(x) = h(x')$ (résistance aux *secondes pré-images*).

Bien évidemment, une fonction de hachage doit être déterministe, et renvoyer la même valeur $h(x) = h(x') = y$ pour $x = x'$.

L'utilisation basique des fonctions de hachage se retrouve dans les algorithmes de détection d'erreur des protocoles réseau, comme le champ `CRC` du protocole Ethernet. En calculant le haché du paquet transmis, l'émetteur permet au destinataire de vérifier en calculant le haché à son tour que le message n'a pas été modifié. Cependant, une probabilité de non-détection reste, évidemment. Comme le domaine de définition de $h$ est plus grand que son domaine d'arrivée, la fonction n'est pas injective ! En conséquence, il est possible que le message ait été modifié de telle sorte que l'on obtienne une collision. Les propriété citées ci-avant

porte plutôt sur la résistance de ces fonctions à une attaque malicieuse. En les lisant, vous devriez intuiter que l'utilisation principale des fonctions de hachage va être sur la *signature* de documents numériques.

**Le paradoxe des anniversaires**

La résistance aux collisions des fonctions de hachage n'est pas une propriété facile à obtenir, et le « pourquoi » est en réalité simple. Les algorithmes de recherche de collisions utilisent tous le *paradoxe des anniversaires*, qui n'est un paradoxe en soi que parce qu'il va à l'encontre de notre intuition !

> *Combien de personnes doivent être présentes dans une salle pour que la probabilité que deux d'entre elles aient leur anniversaire le même jour soit de 50% ?*

Ce problème se résout de la manière suivante, en cherchant la probabilité $p(k)$ que $k$ personnes aient toutes un jour d'anniversaire différent. Pour $k = 1$, cette probabilité est de 1, bien évidemment, mais pour $k = 2$, on cherche le nombre de possibilité de jour anniversaire différents de la première personne. On a donc :

$$p(2) = \frac{365-1}{365}$$

En continuant ce raisonnement, on obtient l'expression suivante pour $p(k)$ :

$$p(k) = \prod_{i=0}^{k} \frac{365-i}{365}$$

Bien évidemment, la réponse qui nous intéresse est le cas inverse $1 - p(k)$. Une rapide évaluation de cette fonction sur des valeurs de $k$ incrémentales nous donne la réponse étonnament basse suivante ; à partir de $\mathbf{k = 23}$ personnes présentes simultanément dans une pièce, il y a 50% de chance que deux d'entre elles partagent leur jour anniversaire. Testez avec votre groupe de TP ! A l'échelle de votre promo, pour 90 personnes, cette probabilité est virtuellement de 100%.

Comment ce paradoxe se retrouve-t-il dans la recherche de collisions dans une fonction de hachage ? De la même manière que pour les personnes, le nombre de valeurs de $x$ à tester avant d'obtenir une collision est relativement faible face à l'intuition qu'on en aurait. En réalité on peut montrer que ce nombre est approximativement égal à la valeur $n$ suivante :

$$n = \sqrt{2\ln(2)d}$$

Où $d$ est la taille du domaine considéré. En binaire pour un haché de $B$ bits, il est donc fortement probable de trouver une collision en générant seulement $2^{B/2}$ valeurs.

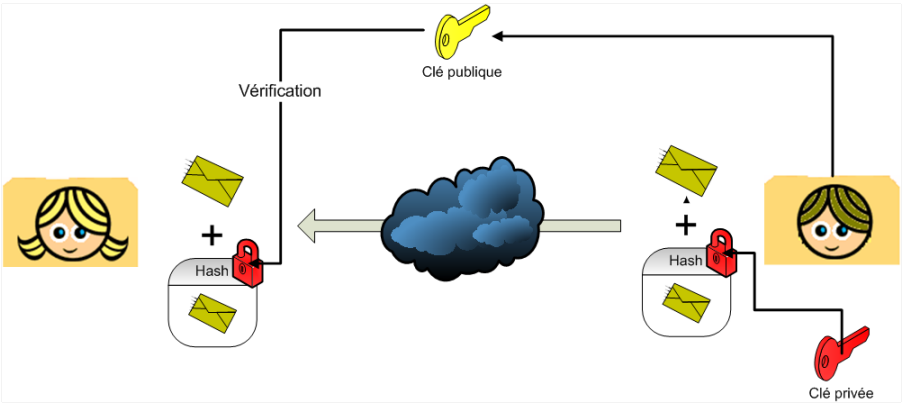
On pourrait donc penser qu'il suffit de calculer le haché d'un message et de transmettre celui-ci pour assurer *l'intégrité* du message, c'est à dire que le destinataire soit en mesure de s'assurer que le message n'a pas été modifié en cours de route. En réalité, un attaquant pourrait très bien intercepter le message, et recalculer le haché du message qu'il aurait entre temps modifié. Un bon schéma de signature doit donc à la fois assurer l'intégrité du message, mais également l'identité de l'émetteur ! Pour cela, on peut utiliser les primitives de cryptographie asymétrique, en se servant de la clé publique comme d'une preuve d'identité.

L'algorithme est assez simple si vous avez compris RSA. On suppose que l'émetteur dispose d'une clé publique $(n,e)$ et d'une clé privée $(n,d)$, et désire envoyer un message $m$ :

1. Il commence par calculer $H = h(m)$, puis il calcule une signature avec sa clé privée $\text{sig} = H^d \mod n$.
2. Le message avec sa signature est envoyé au destinataire, qui reçoit donc $(m, \text{sig})$ et connait $(n,e)$, la clé publique de l'émetteur.

3. Le destinataire calcule $H' = \text{sig}^e \mod n$, et vérifie que $h(m) = H'$ pour accepter le message.

L'émetteur utilise donc la symétrie [2] de l'algorithme de chiffrement RSA pour chiffrer le haché du message à l'aide de sa clé privée. En utilisant les propriétés de l'exponentiation discrète, le destinataire peut récupérer le haché original avec la clé publique. Si le message a été modifié, alors les hachés ne correspondront pas, et il en va de même si une nouvelle signature a été générée avec une clé publique différente de celle de l'émetteur. Ce fonctionnement est résumé dans le schéma ci-dessous.




Le principale problème de ce schéma, et de tous les schémas de signature et de chiffrement en réalité, vient donc de la *confiance* quant à l'identité de l'interlocuteur. Sur Internet, comme vous le verrez en TP, il est extrêmement simple d'usurper l'identité de quelqu'un. Récupérer une clé publique permet de chiffrer vos communications, mais que se passe-t-il si c'est en réalité la clé publique d'un attaquant qui s'est fait passer pour votre banque que vous êtes en train d'utiliser ? Cette problématique de confiance est centrale dans le monde informatique actuel, et des solutions existent. Elles sortent néanmoins du cadre d'une introduction à la cryptographie, et font l'objet du chapitre suivant !

2. Pun intended.

# Chapitre 2

# Sécurité des réseaux et des systèmes

La sécurisation des systèmes d'information est une discipline à part entière, qui requiert une connaissance étendue des vulnérabilités des systèmes et des mesures de protection associées. Il est bien évident qu'à la sortie de ce cours vous ne serez pas des experts en sécurité, mais cette introduction devrait vous donner les clés nécessaires pour comprendre les problèmes, les éviter à votre niveau et être capable de communiquer avec des « vrais » experts sur les projets que vous mènerez dans votre vie professionnelle. Il se trouve également que les utilisateurs sont la vraie grosse faille générale de tous les systèmes, et que l'éducation à la sécurité informatique est en passe de devenir centrale dans les formations supérieures.

Il y a plusieurs manières de traiter la sécurité des systèmes d'informations, et bien sûr un certain nombre de bonnes pratiques. Il est néanmoins essentiel de considérer la sécurité comme une approche globale ; il ne sert à rien d'avoir une porte blindée sur un entrepôt dont les fenêtres sont fragiles et accessibles, et vous n'êtes jamais plus sécurisés que le maillon le plus faible de votre chaîne de communication.

## 2.1 Cartographie des risques numériques

L'entreprise est la cible de nombreuses attaques, quelque soit sa taille et ses moyens informatiques. L'attrait de l'entreprise est avant tout en termes concurrentiels ; on parle souvent d'*intelligence économique*, qui est le terme un peu bisounours de l'espionnage industriel. Les entreprises de taille moyenne disposent aussi d'équipements informatiques puissants qui peuvent être réutilisés par des pirates pour lancer une attaque de manière masquée, à partir de l'entreprise. Une entreprise dispose également de plus d'angles d'attaque. Elle peut être étendue sur un grand espace avec des prises réseaux accessibles ou un réseau WiFi, et la présence d'un grand nombre d'employés ouvre la porte à plus d'erreurs de manipulation du systèmes et de manquements à la sécurité.

On peut citer 4 grands risques auxquels l'entreprise et les employés doivent se préparer :

- Le vol de supports et de données, par intrusion dans le système, les sauvegardes ou tout simplement le vol de machines. Il est par exemple extrêmement courant pour les cadres de grandes entreprises de se faire voler leur ordinateur portable dans un train ou un avion.
- L'intrusion sur le réseau non-autorisée à des fins malicieuses, que ce soit pour perturber le fonctionnement du réseau ou utiliser les ressources de l'entreprise pour attaquer une autre cible.
- L'interception des communications internes à l'entreprise, ou des employés à l'extérieur de l'entreprise.
- La manipulation des employés et les erreurs dans leur comportement. Cela va du simple post-it collé sur le clavier où est écrit le mot de passe d'un utilisateur, aux attaques dites de *social engineering* où l'attaquant va se faire passer pour un membre de l'équipe informatique pour obtenir des informations confidentielles de la part des employés. Il est également courant par exemple pour les encadrants de stage de laisser leur stagiaire utiliser leur accès sans quoi « ils sont trop bloqués pour travailler ».

Un cours portant sur les risques juridiques liés aux systèmes d'information est prévu plus tard dans l'année, mais il est intéressant de connaitre cet article du code civil, établissant ce qu'on appelle la *présomption de responsabilité du gardien* :

> *On est responsable non seulement du dommage que l'on cause par son propre fait, mais encore de celui qui est causé par des personnes dont on doit répondre ou des choses que l'on a sous sa garde.*

Cela signifie que l'entreprise (ou vous dans le cas de vos réseaux personnels) est présumée responsable en cas d'utilisation malicieuse de son réseau. Le manquement à cette règle peut porter gravement atteinte à l'image et à la crédibilité de l'entreprise. Prenons par exemple un cas où les employés utilisent le réseau de l'entreprise pour télécharger du contenu illicite protégé par droit d'auteur. Dans cette situation, l'entreprise est responsable de son réseau et du comportement de ses salariés, et peut être punie en civil en tant que personne morale pour ce délit. Cela peut aller beaucoup plus loin. Imaginez que votre voisin pirate votre réseau WiFi pour télécharger du contenu pédo-pornographique. Etant responsable des biens que vous avez sous votre garde, vous pourriez être inculpé pour ces faits – et c'est déjà arrivé !

En pratique, pour se dédouaner, il est nécessaire de :
- Prouver que la cause est exogène, c'est à dire hors de votre responsabilité ou celle de l'entreprise.
- Prouver que vous avez pris les précautions nécessaires et jugées suffisantes techniquement pour sécuriser l'utilisation de votre système d'information. Cela passe dans le cas de votre réseau WiFi personnelle par l'activation des méchanismes de sécurité les plus avancés disponibles sur votre équipement. Pour l'entreprise, il faudra prouver que la cause a délibérément contourné les procédures et systèmes de sécurité mis en place, et que ces derniers correspondaient à l'état de l'art en matière de sécurité à l'époque du délit.

## 2.2 L'antivirus

Un antivirus est un logiciel de protection dont le but est de détecter les virus ou logiciels malveillants. Son utilisation est *indispensable*, et nombre d'entre eux sont désormais accessibles gratuitement. Si vous utilisez un système d'exploitation Windows, je vous conseille l'antivirus édité par Microsoft s'appelant *Security Essentials*, qui est efficace et particulièrement discret. Le fonctionnement d'un antivirus est simple ; il inspecte la mémoire, les disques durs de l'ordinateur et les volumes amovibles (CD, DVD, clé USB, disque dur externe...) pour vérifier que les fichiers qui y sont présents ne contiennent pas de codes malveillants connus. Il permet aussi d'effectuer régulièrement des analyses planifiées.

Un antivirus peut fonctionner de deux manières différentes : tout d'abord à partir d'une base d'empreintes de virus puis, éventuellement en complément, d'un système d'intelligence artificielle pour détecter certains comportements suspects. A chaque fois qu'un nouveau logiciel malveillant est détecté, le fournisseur de votre antivirus alimente l'encyclopédie constituée de l'ensemble des empreintes, puis la diffuse par l'internet pour que son antivirus puisse le détecter.

Parmi les virus courants, les plus dangereux sont probablement à ce jour de deux sortes, car leur but porte vraiment sur l'utilisation malicieuse du réseau et le vol d'information et non simplement un *déni de service* détruisant les données ou empêchant l'utilisation de l'ordinateur :
- Les *keyloggers* sont des programmes cachés, résidant dans la mémoire de l'ordinateur et cherchant à récupérer des suites de caractères particulières. Ils visent avant tout la récupération de mots de passe et de codes de carte bancaire.
- Les *chevaux de Troie* sont eux aussi des programmes cachés difficilement détectables. Ils ne font globalement rien d'autre que d'offrir à un attaquant la possibilité de prendre le contrôle de la machine et d'effectuer des opérations à distance. Ces opérations peuvent aller de la récupération de données confidentielles au lancement d'attaque à partir de la machine infectée, le pirate se protégeant ainsi en usurpant l'identité de l'ordinateur qu'il utilise.

Les antivirus ne sont néanmoins pas infaillible. Tout d'abord, ils sont basés sur une large liste de *signatures* de virus connus, en cherchant à identifier directement leur code ou bien leurs effets et les symptomes de leurs attaques. De nouvelles *vulnérabilités* apparaissent tous les jours dans les programmes que vous utilisez, et en conséquence il est toujours possible que votre antivirus ait un léger décalage avec la réalité actuelle. Il est par conséquent indispensable de procéder à la mise à jour de cette base de connaissance de signatures le plus régulièrement possible. En pratique, la plupart des antivirus se mettent à jour de manière quotidienne sans intervention de l'utilisateur.

Votre comportement et votre connaissance du danger va aussi résorber le risque d'infection. En particulier, les recommendations usuelles s'appliquent :

- N'ouvrez jamais de fichiers qui peuvent vous sembler malicieux, même s'ils proviennent d'une personne de confiance. Une démonstration que j'ai toujours voulu faire en cours était l'envoi d'un fichier PDF de cours infecté, installant un cheval de Troie sur la machine d'un élève dont j'aurais pu prendre le contrôle en direct en cours. *Sans utiliser de méchanismes d'identification dans vos mails, vous n'êtes jamais sûrs de la provenance réelle des fichiers que vous recevez.* Les virus ne sont pas non plus cantonnés aux fichiers exécutables (`.exe`), même s'il s'agit du vecteur le plus évident. Les fichiers PDF par exemple peuvent contenir du code exécutable et de nombreuses vulnérabilités existent, certaines n'étant d'ailleurs toujours pas corrigée par les éditeurs de logiciel. Il en va de même pour les fichiers Word, Excel, Powerpoint, ...
- Évitez les sites un peu « louches ». L'avancée des technologies Internet relative aux développement de site web force les navigateurs à devenir de plus en plus complexes. En conséquence, les possibilités de failles dans ces navigateurs augmentent également. Un vecteur de virus à la mode porte en particulier sur la publicité en ligne – vous savez, ces trucs pénibles affichés au milieu des articles de votre blog préféré. Ces publicités sont fournies *hors du contrôle* du site web que vous visitez par des agences qui sont régulièrement piratées. Les attaquants injectent alors du code malicieux dans les annonces qui seront répercutées sur un très grand nombre de sites.

Sans entrer dans la paranoïa la plus totale, une démarche consciencieuse de votre part vous évitera tous ces ennuis. De plus, en tant que futurs cadres, vous serez peut être amenés à posséder des informations critiques pour votre entreprise, et la sécurité de votre poste de travail n'en devient que plus importante.

## 2.3 Le chiffrement des communications

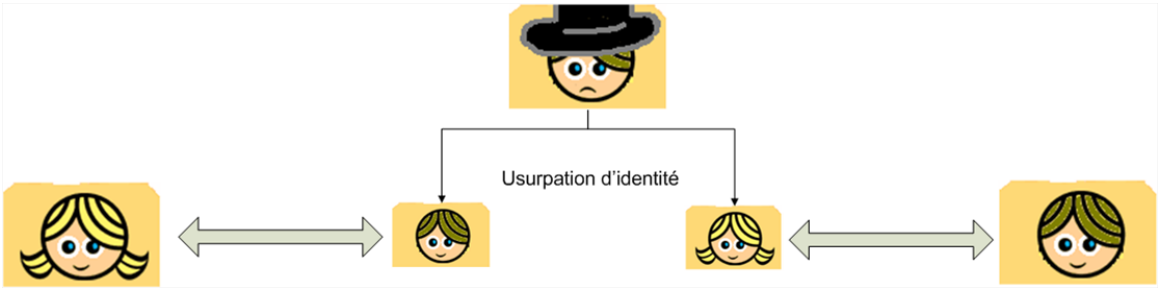
Vos connaissances toutes fraîches en cryptographie vous permettre de savoir que des solutions modernes existent afin de sécuriser vos communications. Comme nous l'avons vu en cours, la plupart des protocoles de l'Internet ne sont *pas sécurisés*. Il existe néanmoins un certains nombre d'améliorations de ces protocoles visant à intégrer la confidentialité et l'intégrité des messages transmis sur Internet. Une grande partie de la 3$^{e}$ séance de TP vise à vous familiariser avec l'utilisation de ces protocoles, le but n'étant pas de s'étendre sur leur fonctionnement mais plutôt d'être capable d'apprécier leur utilité et de savoir les mettre en oeuvre.

En tant qu'utilisateur des réseaux, vous êtes soumis à un risque permanent d'écoute de vos communications. Si cela parait évident sur les réseaux sans-fils comme le WiFi, vous verrez en TP que passer par un câble n'est pas une solution miracle pour éviter le *sniffing* de vos transmissions ! La sécurisation de votre réseau WiFi ne vous protège pas non plus de manière absolue, surtout quand elle est mal maitrisée. Les années précédentes, comme le temps le permettait, je mettais en oeuvre une des attaques classiques sur le chiffrement utilisé par la norme WEP [1] de sécurisation du WiFi. Cela prend environ 2 minutes, montre

1. Au passage, soyez gentils, allez immédiatement vérifier que votre réseau WiFi personnel utilise la norme WPA ou WPA2, et désactivez le WEP dans le cas contraire. La documentation de votre fournisseur d'accès Internet doit vous expliquer comment faire ces opérations.

en main, et les outils nécessaires pour procéder à cette attaque sont de plus en plus simples et accessibles. N'utilisez pas non plus de réseaux ouverts comme INSA-Invité quand cela est possible.

Le schéma de base d'une écoute de transmission est dit *man-in-the-middle* (MITM). L'attaquant va se placer entre vous et votre interlocuteur, de manière transparente si possible, comme le montre le schéma ci-après.

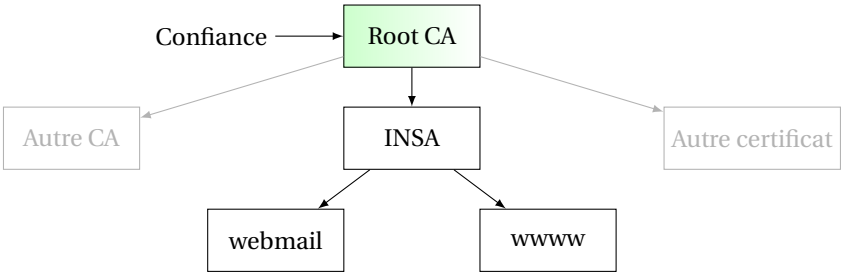



L'attaquant va donc chercher à usurper votre interlocuteur, et la première étape de sécurisation d'une transmission va être de valider son identité. Sur Internet, cette validation passe par un *certificat*, et utilise les principes de la cryptographie asymétrique. Nous avions vu que pour la signature de messages, il était possible d'utiliser une clé privée pour chiffrer la signature et ainsi valider, par l'utilisation de la clé publique, l'intégrité du message. Le problème reste néanmoins de s'assurer que *la clé publique utilisée correspond bien à l'interlocuteur*. Dans l'absolu, sur le schéma MITM ci-dessus, l'attaquent peut tout à fait utiliser sa propre clé, et vous faire croire qu'il s'agit de celle du tiers à qui vous vouliez parler.

Un certificat est donc tout simplement un conteneur de clé publique. Dans sa forme basique, le certificat contient un nom, une clé publique, et l'information de validation du certificat. Il peut contenir un grand nombre d'informations annexes, comme une durée de validité et des identités alternatives. La validation d'un certificat réside sur le principe d'une architecture de *confiance*, organisée hiérarchiquement à la manière du système DNS :

- A la base, vous placez votre confiance dans un certain nombre d'*Autorités de Certification*, appelées *Root CA*. Ces CAs sont enregistrés de base dans votre système d'exploitation. Sous Windows, vous pouvez accéder à ces certificats en utilisant la commande `certmgr.msc`, et dérouler la liste « Autorités racines de confiance ».
- Ces CAs peuvent signer les certificats d'autres autorités secondaires de certifications. Ainsi, l'INSA Lyon dispose d'une autorité de certification locale, ayant le droit d'émettre des certificats pour les sites de l'INSA de Lyon. L'autorité de l'INSA est signée, à ma connaissance, par le Root CA GlobalTrust. En faisant confiance à GlobalTrust, vous faites donc confiance à l'INSA Lyon.
- Au final, les certificats émis par une autorités sont signés à l'aide de la clé privée de l'autorité de certification. Cette dernière publie sa clé publique, et chacun peut vérifier l'identité du Root CA, et donc indirectement celle du certificat signé.

Il existe donc deux manières de vérifier l'identité d'un interlocuteur ; en acceptant directement son certificat, qu'il vous aurait remis en mains propres par exemple, ou par la hiérarchie de confiance en considérant que si vous vous fiez à l'autorité racine de certification, vous vous fiez aussi aux certificats qu'elle signe.



**Certificats et cartes bancaires**

La carte bancaire est un bon exemple de certificat « caché » dont vous faites une utilisation régulière. Votre carte bancaire contient un certificat qui vous identifie personnellement, et qui comprend :

– Votre nom
– Votre numéro de compte
– Votre clé privée
– La clé publique de votre banque

Le code PIN que vous entrez lorsque vous souhaitez payer par carte est en fait l'élément qui vient *libérer* votre clé privée. Cette dernière va ensuite servir à chiffrer la transaction. L'utilisation d'un code PIN en plus de la carte bancaire donne un système d'authentification dite *forte*. Ces systèmes doivent être basés sur au moins deux éléments identifiant une personne, appelés des *facteurs d'authentification* :

– Quelque chose que l'on connait (un mot de passe ou un code PIN)
– Quelque chose que l'on est (une empreinte digitale ou rétinienne)
– Quelque chose que l'on possède (une carte à puce)

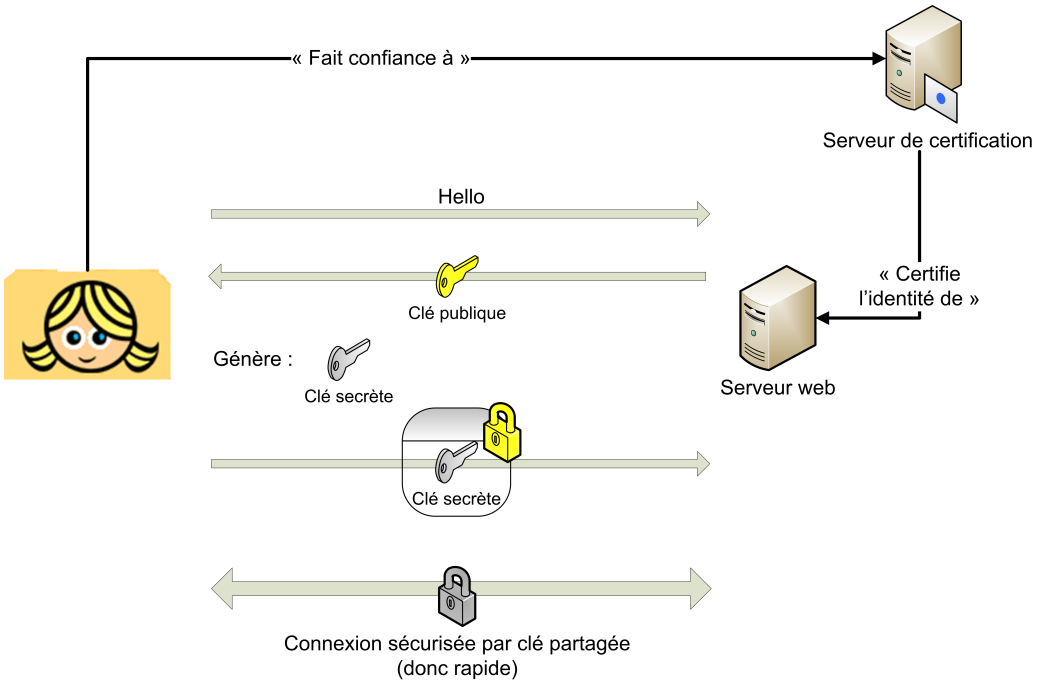
Par la suite, le fonctionnement de la transaction bancaire est assez simple :

1. Le commerçant émet une facture informatique à l'aide du terminal bancaire.
2. En entrant votre code PIN, vous libérez votre clé privée qui sert à signer cette facture.
3. Le terminal bancaire utilise ensuite la clé publique de votre banque pour chiffrer la facture et la signature, et envoie la transaction à votre banque.
4. En parallèle, le commerçant signe la facture avec sa propre clé privée, et envoit la facture et la signature à sa propre banque.
5. Si les deux factures correspondent, les banques transfèrent les fonds.

Cette architecture de confiance sert de base à tous les protocoles sécurisés d'Internet au niveau de la couche application, en particulier les protocoles liés à l'envoi et la réception de mails et à la lecture de pages web. On peut détailler le fonctionnement du protocole HTTPS (`S` pour *Secure*), afin de voir comment la communication chiffrée s'établit à partir du certificat. Comme on l'a vu, la cryptographie asymétrique est très lente. Le client va donc utiliser la clé publique du serveur pour lui transférer, *sur un canal sûr*, une clé de chiffrement symétrique partagée. La couche de sécurité ajoutée au protocole HTTP s'appelle `SSL/TLS`, ou *Secure Socket Layer*. Comme son nom l'indique, `SSL/TLS` sécurise la communication à travers les sockets de manière presque transparente pour le programmeur, et c'est donc un moyen simple d'ajouter de la sécurité à un protocole application. La couche `SSL/TLS` protège des attaques MITM en vérifiant l'identité du destinataire, et assure la confidentialité en chiffrant la communication.Pour un site web, le certificat contiendra la clé publique du serveur et le nom d'hôte du site web, qui lui sert d'identifiant. Comme le certificat est signé par l'autorité de certification, le client est capable de vérifier que le nom d'hôte présent sur le certificat correspond bien au nom d'hôte de la page web qu'il a demandée, et donc de valider l'identité de son interlocuteur.

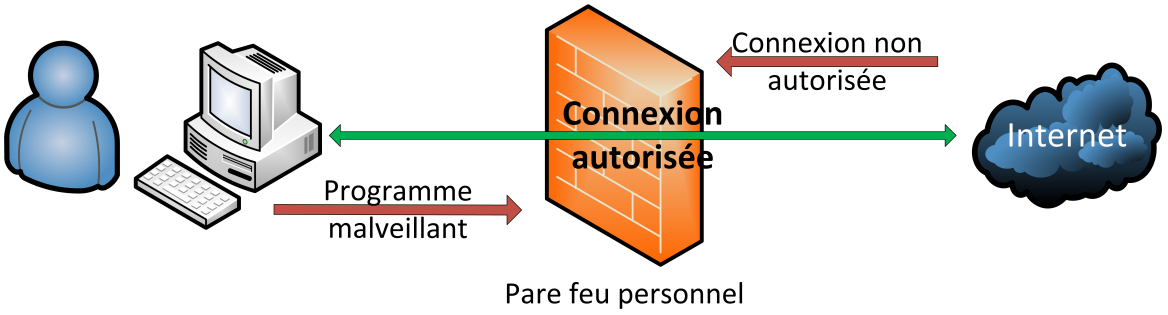
Tout n'est pas parfait hélas, et on verra en TP que la faiblesse de `SSL/TLS` réside dans le fait que l'utilisateur doit accepter le certificat du serveur. Par exemple, une attaque MITM pourra « changer » le certificat du serveur par son propre certificat. Dans ce cas, il est peu probable que l'attaquant dispose d'un certificat signé, et votre navigateur va vous indiquer un danger potentiel, en gros et en rouge. Le problème est que les utilisateurs ignorent parfois ces remarques, car elles arrivent parfois dans un cadre totalement sain – en particulier, l'année dernière, le certificat du webmail de l'INSA est resté invalide pendant plusieurs semaines. Une autre attaque que vous verrez porte sur la désécurisation de la connexion. Dans ce cas, l'attaquant « enlève » la couche `SSL/TLS` et vous transfère donc une page web non-sécurisée. Si vous ne vérifiez pas la

présence du certificat en vous assurant qu'un « cadenas » est présent à côté de l'adresse de la page, vous risquez de transmettre en clair vos informations personnelles ! Et dans ce cas le navigateur web ne vous préviendra pas ; pour lui, vous êtes juste en train de visiter un site web non-sécurisé.




## 2.4 Les firewalls

Un *firewall*, ou pare-feu, est un outil permettant de protéger un ordinateur connecté à un réseau ou à l'internet. Il protège d'attaques externes (filtrage entrant) et souvent de connexions illégitimes à destination de l'extérieur (filtrage sortant) initialisées par des programmes ou des personnes. Il existe deux types de firewalls : les firewalls personnels, installés sur les ordinateurs, et les firewalls d'entreprise, situés aux points de croisement du réseau.




De nos jours, les firewalls personnels peuvent filtrer les applications qui tentent de se connecter à l'internet ou qui attendent une connexion de l'extérieur. L'utilisateur est en général prévenu par un message d'alerte, via lequel il peut accepter ou refuser cette connexion au cas par cas ou définitivement. Ce mode d'utilisation est aussi appelé mode d'apprentissage. Certains firewalls personnels vérifient également que le logiciel qui tente de se connecter sur le réseau n'a pas été altéré par un cheval de Troie.

Les firewalls d'entreprise agisse à un niveau plus global dans le réseau. Leur but est dans ce cas de *contrôler les flux d'information* à travers le réseau. La définition d'un flux est donc importante pour la configuration des firewalls d'entreprise. On distingue plusieurs types de flux :

- Les flux entre différents VLANs, identifiés par le préfixe de sous-réseau qui leur est associé.
- Les flux entre machines, identifiées par leurs adresses IP.
- Les flux vers des applications spécifiques, identifiées par des adresses IP et des ports.
- Les flux spécifiques *à l'intérieur* des applications, comme par exemple le filtrage de certains sites web ou de certaines adresses e-mail.

La plupart des firewalls d'entreprise courants fonctionnent selon la 3e définition. Comme ils doivent avoir accès aux ports de source et de destination des paquets, ils travaillent au niveau de la couche transport, et sont appelés *packet filters*. Il existe un autre type de firewall en vogue, travaillant au niveau de la couche application, et pratiquant ce qu'on appelle du *deep packet inspection* – ou *inspection approfondie de paquets*. Plus d'information sur ces firewalls et leur fonctionnement est disponible en fin des transparents de cours.

## 2.5 Les mots de passe

Les mots de passe sont monnaie courante dans les systèmes d'information et les réseaux, et pourtant ils sont encore mal compris et mal utilisés. Tout d'abord, il faut bien comprendre qu'un mot de passe est un élément d'authentification qui est sensé identifier un utilisateur – on fait donc la supposition implicite que *seul* l'utilisateur connait son mot de passe, raison pour laquelle vous ne devez pas le partager. Il s'agit d'une authentification faible lorsque elle est utilisée seule, car elle ne fait appel qu'à un seul facteur d'authentification, contrairement à la carte bancaire comme on l'a vu dans la section sur les certificats. L'aparté vous renseigne sur quelques méthodes utilisées pour stocker et transmettre les mots de passe à travers le réseau. Ces techniques sont nombreuses et vastes, et leurs détails sortent du cadre de ce cours. En contrepartie, j'estime qu'il est intéressant de savoir comment choisir un mot de passe, et pourquoi on parle souvent de *mots de passe forts*.

**Stockage et transmission de mots de passe**

Vous vous en doutez, il est déconseillé de transmettre un mot de passe « en clair » sur le réseau. Les méthodes d'écoute étant nombreuses, il est relativement simple de voler beaucoup de mots de passe de cette manière. C'est pourtant ce que font un certain nombre de protocoles applicatifs par défaut, comme le protocole `IMAP` utilisé pour la lecture et la gestion de boites mails !

Si l'on veut sécuriser la transmission du mot de passe, on peut tout d'abord utiliser une fonction de hachage. Si vous retournez à la section correspondante dans le précédent chapitre, vous verrez que n'étant pas injective une fonction de hachage est difficile à « inverser ». On ne peut en général pas récupéré le mot de passe original en utilisant le haché du mot de passe. Néanmoins, il existe sur Internet des *rainbow tables*, qui sont en fait d'immenses dictionnaires de hachés de plusieurs giga-octets contenant tous les hachés de tous les mots de passe possibles inférieurs à une certaine taille (pour 8 caractères alphanumériques, ces tables font déjà 50 Go). Un mot de passe long est par conséquent indispensable pour éviter les attaques par *rainbow tables*. Si l'on observe l'authentification naïve par un haché du mot de passe, on voit également deux failles de sécurité importante :

| Utilisateur | | | Serveur | |
|---|---|---|---|---|
| Connait | Calcule | Envoie | Calcule | Connait |
| mp | $H = h(\text{mp})$ | $(\text{User}, H) \rightarrow$ | | $(\text{User}, \text{mp})$ |
| mp | | | Vérifie $H = h(\text{mp})$ | $(\text{User}, \text{mp})$ |
| mp | | $\leftarrow$ OK | | $(\text{User}, \text{mp})$ |

Tout d'abord, le serveur possède le mot de passe en clair, ce qui n'est pas idéal. En cas de compromission du serveur tous les mots de passes sont disponibles pour un attaquant ! En pratique, on commence tout d'abord pas stocker non pas le mot de passe directement, mais le haché du mot de passe, que le serveur compare au haché que le client lui envoie. La seconde faille de sécurité est que cette méthode ne protège pas contre le *rejeu*. Un attaquant ayant écouté cette communication pourrait tout à fait renvoyer $(\mathsf{User}, H)$ pour s'identifier auprès du serveur, sans connaitre le mot de passe original.

Il est difficile de se protéger à la fois de ces deux failles. Pour traiter la première, et éviter de stocker le mot de passe en clair en le rendant plus robuste, on utilise un *salage* du mot de passe à l'aide d'une petite quantité de données $s$. Le serveur stocke $(\mathsf{User}, s, h(\mathsf{mp}+s))$ et authentifie le client à l'aide de l'algorithme suivant :

| Utilisateur | | | Serveur | |
|---|---|---|---|---|
| Connait | Calcule | Envoie | Calcule | Connait |
| mp | | User → | | $(\mathsf{User}, s, h(\mathsf{mp}+s))$ |
| mp | | $\leftarrow s$ | | $(\mathsf{User}, s, h(\mathsf{mp}+s))$ |
| mp | $H = h(\mathsf{mp}+s)$ | $H \rightarrow$ | | $(\mathsf{User}, s, h(\mathsf{mp}+s))$ |
| mp | | | Vérifie $H = h(\mathsf{mp}+s)$ | $(\mathsf{User}, s, h(\mathsf{mp}+s))$ |
| mp | | ← OK | | $(\mathsf{User}, s, h(\mathsf{mp}+s))$ |

Le salage protège également d'une attaque par *rainbow table*, car ces dernières sont précalculées sans connaître le salage utilisé. Ce dernier étant choisi aléatoirement, il est impossible à priori de calculer une table pour tous les salages existant ! Certaines tables spécifiques existent pourtant. Par exemple, la sécurité WiFi en WPA est assurée par un mot de passe salé à l'aide de l'identifiant du réseau WiFi. Mais comme les opérateurs, par défaut, nomment leurs réseaux sous une forme standard – `Livebox-0f0e` – il devient possible de précalculer des *rainbow tables* spécifiques ! De telles tables existent et sont disponibles sur le web. Une bonne pratique pour protéger votre réseau WiFi est donc de changer l'identifiant de votre réseau !

Pour en revenir au sujet initial, pour le second cas, le serveur doit stocker le mot de passe en clair. Afin d'éviter le rejeu, le serveur va demander *à chaque connexion* un nouveau haché au client, en lui émettant un *challenge* – qui pourrait être traduit en français par *défi*. Ce challenge prend la forme d'un petit bloc de données similaire au salage. Le client va donc renvoyer le haché de son mot de passe et du défi et renvoyer cette valeur. Un attaquant qui écouterait la communication ne pourrait donc pas rejouer le haché renvoyé par le client, celui correspondant à un défi particulier ! Et il ne pourrait pas non plus découvrir le mot de passe original à partir du défi et du haché.

| Utilisateur | | | Serveur | |
|---|---|---|---|---|
| Connait | Calcule | Envoie | Calcule | Connait |
| mp | | User → | | $(\mathsf{User}, \mathsf{mp})$ |
| mp | | $\leftarrow d$ | Choisit $d$ | $(\mathsf{User}, \mathsf{mp})$ |
| $(\mathsf{mp}, d)$ | $H = h(\mathsf{mp}+d)$ | $H \rightarrow$ | | $(\mathsf{User}, \mathsf{mp}, d)$ |
| $(\mathsf{mp}, d)$ | | | Vérifie $H = h(\mathsf{mp}+d)$ | $(\mathsf{User}, \mathsf{mp}, d)$ |
| $(\mathsf{mp}, d)$ | | ← OK | | $(\mathsf{User}, \mathsf{mp}, d)$ |

Il existe deux types d'attaques sur les mots de passe, en plus de l'écoute et des rainbow tables présentées ci-dessus :

– Tout d'abord, on peut procéder à une *recherche exhaustive* en testant toutes les combinaisons de caractère possible. Cette recherche exhaustive est de nos jours de plus en plus puissante, même pour un

équipement informatique personnel. En particulier, il est désormais possible d'utiliser pour ces calculs lourds la carte graphique de votre ordinateur. Cette dernière comporte un processeur spécialisé dans le traitement d'opérations linéaires sur d'énormes matrices d'entier, afin de calculer l'affichage de polygones à l'écran. Comme les primitives de cryptographie sont basées sur des opérations linéaires, l'utilisation des cartes graphiques est rapidement devenue un standard dans les attaques de mots de passe.

– Les attaques par dictionnaire font aussi une recherche exhaustive, mais en limitant les possibilités à celles contenant des mots courant. Elles vont donc tester les mots du petit Robert, du Larousse, les prénoms et les noms propres, ainsi que les combinaisons de ces mots entre eux ou avec d'autres symboles. Au final, un mot de passe ressemblant à `moietmonchien007?!` n'est pas du tout sécurisé façe aux attaques par dictionnaire, malgré le fait qu'il contienne un nombre important de lettres.

La *force* d'un mot de passe s'évalue un peu à la manière de la résistance des chiffrements à clé secrètes présentés dans le chapitre précédent. On va disposer, pour créer le mot de passe, d'un alphabet de symboles et dans l'idéal, il faudra avoir choisi aléatoirement *et de manière équiprobable* dans tout l'alphabet un grand nombre de symboles. Ainsi, les mots du dictionnaire introduisent un fort biais statistique qui est utilisé dans les attaques sur les mots de passe. Si l'on choisit plus où moins aléatoirement son mot de passe, sa force est quantifiée à partir du nombre de tests nécessaires pour le retrouver à l'aide d'une recherche exhaustive.

Un mot de passe de $L$ lettres choisies dans un alphabet contenant $N$ symboles aura donc une *force* de $N^L$. On exprime généralement ce nombre à l'aide d'une puissance de 2, afin de le comparer aux clés utilisées dans les primitives cryptographiques. Un mot de passe ayant une force supérieure à $2^{100}$ est aujourd'hui considéré fort. Un mot de passe ayant une force de l'ordre de $2^{80}$ pourrait potentiellement être craqué en un temps raisonnable par un attaquant ayant des moyens techniques conséquent. Pour $2^{50}$ et en deça, n'importe ayant plus de 1000€ à dépenser dans une machine actuelle pourra pirater votre mot de passe.

| Type de mot de passe | Eq. $2^n$ | Force | Commentaire |
|---|---|---|---|
| Mot de passe de 8 caractères dans un alphabet de 70 symboles | 49 | Très faible | Taille usuelle |
| Mot de passe de 10 caractères dans un alphabet de 90 symboles | 65 | Faible | |
| Mot de passe de 12 caractères dans un alphabet de 90 symboles | 78 | Faible | Taille minimale recommandée par l'ANSSI pour des mots de passe ergonomiques ou utilisés de façon locale. |
| Mot de passe de 16 caractères dans un alphabet de 36 symboles | 82 | Moyen | Taille recommandée par l'ANSSI pour des mots de passe plus sûrs. |
| Mot de passe de 16 caractères dans un alphabet de 90 symboles | 104 | Fort | |
| Mot de passe de 20 caractères dans un alphabet de 90 symboles | 130 | Fort | Force équivalente à la plus petite taille de clé de l'algorithme de chiffrement standard AES (128 bits). |

Pour calculer la taille de l'alphabet utilisé pour votre mot de passe, il suffit d'additionner les groupes de symboles que vous utilisez. Il y a 26 minuscules, 26 majuscules, 10 chiffres et une dizaine de symboles courant (`!?:#$%*+=...`). Pour un alphabet à 90 symboles, vous devez utiliser des caractères moins courants, comme `<>§|...`. Vous pouvez utiliser Matlab et les fonctions `log2(nextpow2( ))` pour trouver la puissance de 2 proche du résultat du calcul de $N^L$ pour votre mot de passe, et ainsi évaluer sa force !

# Bibliographie et informations sur le contenu

Les ouvrages ci-dessous ont servi de sources pour certaines parties de ce cours :


– Kurose, James S. and Ross, Keith W., "Computer Networking : a Top Down Approach", *Pearson*, 6e édition, 2012
– Beissinger, Janet and Pless, Vera, "The Cryptoclub : Using Mathematics to Make and Break Secret Codes", *CRC Press*, 2006
– Articles Wikipédia (`www.wikipedia.org`) *en anglais* sur les fonctions de hachage, le système Diffie-Hellman, l'algorithme RSA, Claude Shannon, Alan Turing.


Sources des images :

– Le portrait de Claude Shannon a pour source `http://it-science.net/shannon.html` mais ne fait aucune mention d'un copyright.
– Le portrait d'Alan Turing est ©National Portrait Gallery, Londres, représentant Turing a l'instant de son élection à la Royal Society. Version basse résolution récupérée sur `http://www.npg.org.uk/collections/search/portrait/mw165875`.
– L'image de Tux chiffré est ©Larry Ewing (`lewing@isc.tamu.edu`).
Traitée à l'aide de The GIMP (`www.gimp.org`).