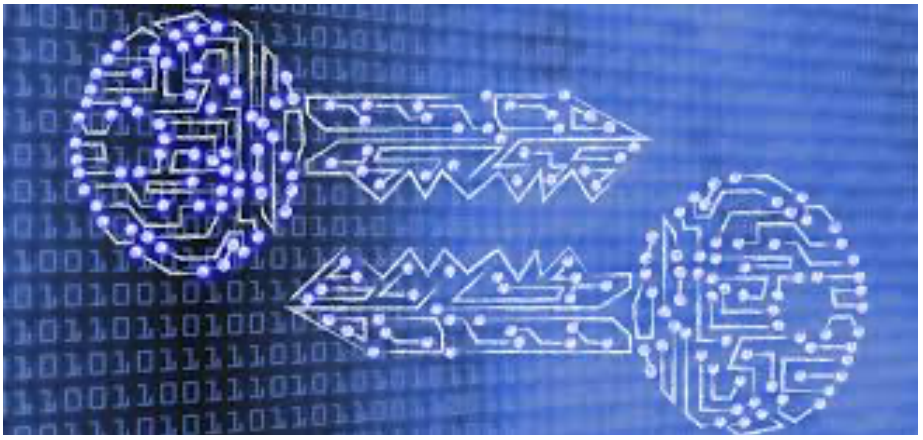
# Informatique 3OC

A. Ridard

## Cryptographie et sécurité



Gymnase Yverdon

canton de vaud

# Table des matières

# I. Définitions

La **cryptologie**, entre mathématiques et informatique, est traditionnellement définie comme la « science du secret ». Longtemps concentrée sur la problématique de la confidentialité, à des fins essentiellement militaires ou diplomatiques, la cryptologie a bénéficié d'un essor important suite au développement de la société de l'information.

La cryptologie est formée de deux branches :

- la **cryptographie** étudie la conception de mécanismes permettant d'assurer des propriétés de *sécurité* comme la *confidentialité*[1] et l'*intégrité*[2] des données, ou encore l'*authentification*[3] des personnes.
- La **cryptanalyse** s'intéresse à ces mêmes mécanismes en tentant d'analyser leurs failles, de les mettre en défaut par des *attaques*.

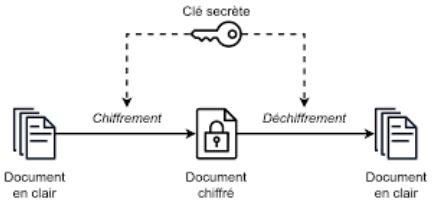
Sur un plan technique, on distingue :

- la cryptographie **symétrique** (à clé secrète)
- la cryptographie **asymétrique** (à clé publique)

# II. Cryptographie symétrique

## 1. Principe

En cryptographie symétrique, la clé secrète est commune et doit donc être partagée :




Dans ce cours, on désignera par :

- $m$ le message en clair
- $k$ la clé secrète
- $c$ le message chiffré
- $E_k$ la fonction de chiffrement : $c = E_k(m)$
- $D_k$ la fonction de déchiffrement : $m = D_k(c)$

Les fonctions de chiffrement et de déchiffrement sont réciproques l'une de l'autre :

$$D_k(E_k(m)) = m$$

## 2. Chiffrement (ancien) par substitution : César

Le chiffrement par décalage, appelé aussi chiffrement de César[4], est un exemple de chiffrement par substitution (mono-alphabétique).

Il consiste à choisir un entier naturel $k$ inférieur ou égal à 25 (la clé), et à décaler chaque lettre du message en clair de $k$ lettres dans l'alphabet (en recommençant bien entendu à la lettre « A » si l'on dépasse « Z »).

Chiffrer le message en clair **INFORMATIQUE** en utilisant la clé $k = 6$.

[1]. La confidentialité assure que les informations transmises ou stockées ne sont accessibles qu'aux personnes autorisées à en prendre connaissance.
[2]. L'intégrité consiste à empêcher, ou tout du moins à détecter, toute altération non autorisée de données, c'est à dire toute modification, suppression partielle ou insertion d'information.
[3]. L'authentification garantit qu'un correspondant est bien celui qu'il prétend être.
[4]. De l'empereur romain Jules César qui utilisait cette technique pour ses correspondances militaires

Pour formaliser un tel mécanisme, on remplace en général les lettres par leur rang dans l'alphabet **en commençant par 0**, de manière à pouvoir calculer modulo 26.

La fonction de chiffrement qui agit « lettre par lettre » est alors définie par :

$$\begin{array}{rccc} E_k : & \{0,1,\ldots,25\} & \longrightarrow & \{0,1,\ldots,25\} \\ & m_i & \longmapsto & c_i = m_i + k \ mod \ 26 \end{array}$$

où $m_i$ désigne la $i$-ième « lettre » du message en clair $m$.

1. Définir la fonction de déchiffrement $D_k$.
2. Déchiffrer le message **JGN** chiffré avec la clé $k = 13$.

En TP, on verra comment **décrypter**[a] c'est à dire **attaquer** ce type de chiffrés

- par force brute
- par analyse de fréquences

*a.* Retrouver le message en clair sans la clé

L'essor de l'informatique et la représentation de l'information en binaire conduit à la cryptographie « moderne » où les messages en clair $m$ sont des suites de 0 et de 1.

## 3. Chiffrement (moderne) par bloc

Une « primitive » de chiffrement par bloc est un algorithme traitant les données à chiffrer par blocs de taille $p$ fixe[5]. Un tel mécanisme permet donc uniquement de « combiner » une suite de $p$ bits de données avec une clé de $n$ bits afin d'obtenir un bloc de données chiffrées de même taille $p$ que le bloc de données claires.

L'une des principales propriétés attendues d'un mécanisme de chiffrement par bloc est d'être facilement inversible si l'on dispose de la clé secrète de chiffrement.

La « combinaison » la plus simple est le **ou exclusif**, appelé XOR, et noté $\oplus$.
En supposant que la taille $p$ d'un bloc coïncide avec la longueur $n$ de la clé, on peut définir la fonction de chiffrement suivante :

$$\begin{array}{rccc} E_k : & \{0,1\}^p & \longrightarrow & \{0,1\}^p \\ & m_i & \longmapsto & c_i = m_i \oplus k \end{array}$$

où $m_i$ désigne le $i$-ième bloc du message en clair $m$, et où $\oplus$ agit bit à bit.

La réciproque de $\oplus$ étant lui-même, définir la fonction de déchiffrement $D_k$.

Chiffrer le message **OC** (encodé en ASCII) avec la clé **1101**.

Le caractère déterministe[6] d'un tel mécanisme peut engendrer des failles de sécurité.

Afin de traiter des messages de taille quelconque[7] et d'assurer la confidentialité « globale » de ces messages, et pas uniquement une confidentialité « par bloc », il convient de définir un **mode opératoire** précisant :

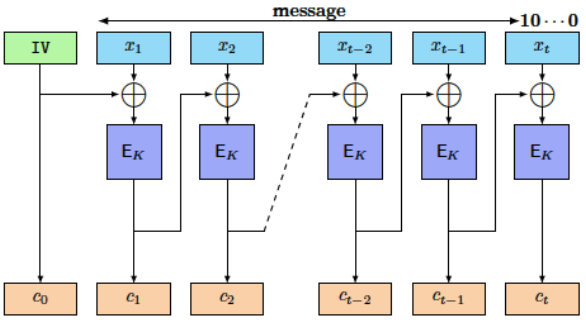
- comment convertir le message en une suite de blocs (« padding »)
- le mode de chiffrement de ces blocs

[5]. Typiquement, cette taille vaut 64 ou 128 bits en pratique.
[6]. A partir d'un bloc de données et d'une clé secrète, il produit toujours le même bloc de chiffré
[7]. Pas nécessairement un multiple de $p$

Notons enfin que :

- pour rompre le caractère déterministe du chiffrement, il est nécessaire de « randomiser » le processus, c'est-à-dire d'introduire une valeur aléatoire
- la définition d'un tel mode opératoire n'a nul besoin de préciser l'algorithme de chiffrement par bloc employé

Le mode opératoire le plus connu est le **CBC** :

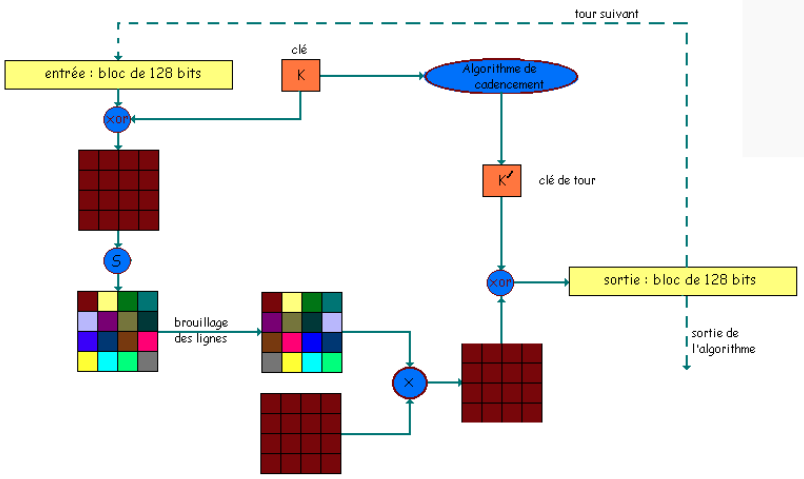



Chiffrer à l'aide de CBC le message **TGV** (encodé en ASCII) avec la clé **11010**.

1. Choisir une clé secrète de taille 6.
2. Choisir au hasard, en lançant une pièce, le bloc IV[a].
3. Chiffrer à l'aide de CBC[b] un message de 4 caractères ASCII.
4. Transmettre à votre voisin de droite :
   - la clé secrète **en main propre**
   - le chiffré par mail
5. Déchiffrer le message reçu de votre voisin de gauche.

*a.* Initial Vector
*b.* $E_K$ désignant la primitive XOR

Voici enfin la primitive **AES** utilisée aujourd'hui pour chiffrer les données contenues dans un fichier avant de les écrire sur un disque dur, ou encore les communications audio/vidéo en temps réel sans impacte perceptible sur les performances du système.



**AES** procède par blocs de 128 bits, avec une clé de 128 bits également.
Chaque bloc subit une séquence de 5 transformations répétées 10 fois :

1. **Addition de la clé secrète** (par un ou exclusif)
2. **Transformation non linéaire d'octets** : les 128 bits sont répartis en 16 octets disposés dans une matrice 4 x 4. Chaque octet est transformé par une fonction non linéaire $S$, assimilée à une substitution sur les entiers compris entre 0 et 255
3. **Brouillage des lignes** : les 3 dernières lignes sont décalées cycliquement vers la gauche (la 2ème est décalée d'une colonne, la 3ème de deux, et la 4ème de trois)
4. **Brouillage des colonnes** : chaque colonne est transformée par combinaison linéaire des différents éléments de la colonne (ce qui revient à multiplier la matrice $4 \times 4$ par une autre matrice $4 \times 4$ dans le corps fini à $2^8$ éléments)
5. **Addition de la clé de tour** (par un ou exclusif) : à chaque tour, une clé de tour est générée à partir de la clé secrète par un sous-algorithme (dit de cadencement)

## 4. Chiffrement (moderne) par flot

Commençons par le **masque jetable de Vernam** qui additionne (par un ou exclusif) le message en clair $m$ tout entier, et non plus par bloc, avec la clé [8] $s$ de même taille.

$$c = m \oplus s$$

Cet algorithme dispose d'une « sécurité parfaite » à condition que la clé ne soit jamais réutilisée pour un autre message. Ce mécanisme, en théorie parfait, est en pratique inutilisable puisqu'il nécessite une clé à usage unique aussi longue que le message à protéger, repoussant alors le problème au niveau de la **mise en accord de clé**...

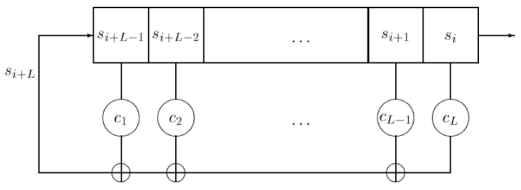
Pour que le principe du masque jetable de Vernam soit utilisable en pratique, nous avons recours à un **générateur pseudo-aléatoire** (GPA) qui construit une suite de 0 et de 1, en apparence aléatoire, à partir d'une simple graine « réellement aléatoire ». Ce mécanisme fournit une suite binaire $s$, dite chiffrante, de même taille que le message en clair, et peut donc être additionnée pour obtenir le message chiffré.

Un LFSR [9] (binaire) de longueur $L$ est un GPA composé :
- d'un registre à décalage contenant une suite de $L$ bits $\left(s_i, \ldots, , s_{i+L-1}\right)$
- et d'une fonction de rétroaction linéaire

A chaque top de l'horloge, le bit de poids faible $s_i$ constitue la sortie du registre, et les autres sont décalés vers la droite ; le nouveau bit $s_{i+L}$ placé dans la cellule de poids fort du registre est donné par une fonction linéaire :

$$s_{i+L} = c_1 s_{i+L-1} \oplus c_2 s_{i+L-2} \oplus \cdots \oplus c_{L-1} s_{i+1} \oplus c_L s_i$$

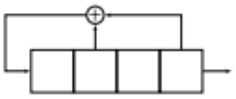
où les coefficients $c_i$ sont binaires.




Pour visualiser le fonctionnement, une animation est disponible ici

[8]. Dans cette section, la clé sera notée $s$ puisque l'on parelera de suite chiffrante
[9]. En français, on parle de registre à décalage à rétroaction linéaire

1. Chiffrer le message **TGV** avec la clé **1011** [a] et le LFSR suivant :



2. Comment le destinataire peut-il déchiffrer le message reçu ?

*a.* Qui correspond à l'état initial | 1 | 0 | 1 | 1 | du registre

Un simple LFSR ne garantit pas une sécurité suffisante, on utilise plutôt des registres combinés à décalage irrégulier comme le générateur A5/1 de GSM.

Le chiffrement symétrique est sécuritaire et performant, mais il souffre d'un défaut majeur. En effet, si deux personnes veulent établir un canal de communication sûr à l'aide d'un chiffrement symétrique, elles doivent d'abord **se mettre d'accord sur la clé secrète** (et évidemment sur l'algorithme utilisé). Or, les deux participants sont justement dans la situation où ils ne peuvent **pas encore** communiquer de façon sûre !

Deux possibilités s'offrent à eux :

- soit ils échangent la clé par un moyen de communication non sûr (email,courrier), mais un attaquant pourrait alors s'emparer de la clé et compromettre la sûreté des futures communications chiffrées avec cette clé.
- soit ils échangent la clé par un moyen sûr mais « non pratique », par exemple en se rencontrant physiquement [10].

Pour résoudre ce problème, diverses techniques ont été développées dès les années 1970, d'abord de manière privée, en particulier par les services secrets britanniques et américains, puis dans la recherche académique publique.

Concernant les deux méthodes de cryptographie asymétrique présentées ici, nous nous limiterons à une description de « haut niveau » puisque les détails mathématiques de ces techniques vont bien au-delà des attendus de l'école de maturité.

# III. Cryptographie asymétrique

## 1. Méthode de Diffie-Hellmann

Ce protocole de **mise en accord de clé** repose sur une fonction mathématique $M$ (pour « mélange ») à deux variables possédant les propriétés suivantes :

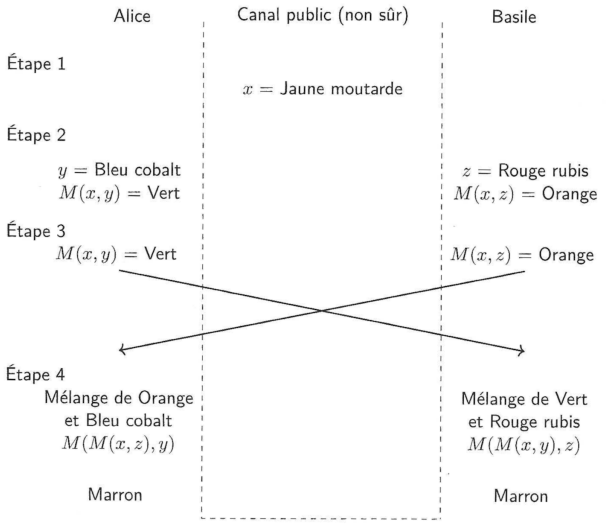
1. $M$ est connue, et on sait la calculer en un temps raisonnable
2. Connaissant $M(x,y)$ et $x$, on ne peut pas retrouver $y$ en un temps raisonnable [11]
3. $M(M(x,y),z) = M(M(x,z),y)$

Une analogie couramment utilisée est celle des pots de peinture. Supposons que l'on ait à notre disposition un très grand nombre de couleurs différentes. Les variables $x$, $y$ et $z$ correspondent à trois couleurs, et $M$ prend 10 cl de deux couleurs et les mélange (pour obtenir 20 cl de peinture d'une nouvelle couleur).

Dans cette analogie, $M$ vérifie-t-elle bien les 3 propriétés ?

[10]. Dans un parc, un soir de nuit sans lune, avec une mallette menottée au poignet ;)
[11]. Il s'agit en fait d'un problème *NP-complet* de complexité exponentielle

Voici le protocole de Diffie-Hellman :




Les étapes sont les suivantes :

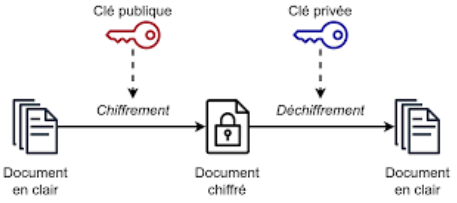
1. Alice et Basile se mettent d'accord sur une couleur de base qui est *publique*
2. ils choisissent chacun une couleur *privée*, ils la mélangent avec la couleur de base
3. ils envoient leur mélange respectif
4. ils ajoutent au mélange reçu leur couleur *privée*, et obtiennent ainsi la clé secrète

Ce protocole résout de manière élégante le problème de mise en accord de clé posé par l'utilisation d'un chiffrement symétrique, mais il laisse toutefois une dernière faiblesse : il ne permet pas l'*authentification* des participants. Autrement dit, Alice n'a pas la garantie que la personne avec qui elle communique soit bien Basile !

Le système RSA permet de résoudre ce problème, en attribuant à chacun une clé publique, en plus de sa clé privée.

## 2. Système RSA

Dans le système RSA, chaque participant possède une paire de clés (publique, privée) :




Dans ce cours, on désignera par :

- $m$ le message en clair
- $c$ le message chiffré
- $K^{pub}_{Basile}$ la fonction de chiffrement avec la clé publique de Basile : $c = K^{pub}_{Basile}(m)$
- $K^{priv}_{Basile}$ la fonction de déchiffrement avec la clé privée de Basile : $m = K^{priv}_{Basile}(c)$

La manière de créer ces paires de clés est complexe, mais on a toujours :

$$K^{priv}_{Basile}\left(K^{pub}_{Basile}(m)\right) = m$$

### RSA

Alice souhaite envoyer le message $m = 10$ à Basile qui a défini sa clé $\left(k_B^{pub}, k_B^{priv}\right)$ de la manière suivante :

- il a choisi deux nombres premiers[a] distincts : $p = 5$ et $q = 17$
- il a calculé $n = pq = 85$ et $\varphi(n) = (p-1)(q-1) = 64$
- il a choisi un entier $e = 5$ premier avec $\varphi(n)$
- il a calculé l'inverse $d$ de $e$ modulo $\varphi(n)$
- il a ainsi obtenu :

$$k_B^{pub} = (n, e) = (85, 5) \quad \text{et} \quad k_B^{priv} = (n, d) = (85, 13)$$

1. Alice utilise la fonction de chiffrement définie par :

$$K_{Basile}^{pub}(m) = m^e \mod n$$

   Calculer le chiffré $c$ reçu par Basile.

2. Basile utilise la fonction de déchiffrement définie par :

$$K_{Basile}^{priv}(c) = c^d \mod n$$

   Retrouver le message $m$ envoyé par Alice.

*a.* Dans la pratique, ce sont de très grands nombres d'une centaine de chiffres

Ce système permet de communiquer de manière sûre sans échange de clé, mais il est beaucoup plus coûteux en temps de calcul qu'un chiffrement symétrique. Il n'est donc pas utilisé pour envoyer des gros volumes de données ou des flux de communication (visioconférence avec vidéo et son), mais plutôt pour envoyer la clé secrète à partager dans le cadre d'un chiffrement symétrique efficace comme CBC-AES (par bloc) ou le générateur A5/1 de GSM (par flot).

### Partage de la clé secrète avec RSA

Considérons le couple[a] (clé publique, clé privée) suivant :

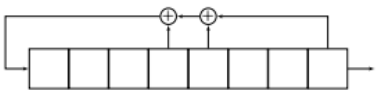
$$k_{Eleve}^{pub} = (n, e) = (10403, 7) \quad \text{et} \quad k_{Eleve}^{priv} = (n, d) = (10403, 8743)$$

1. *En tant qu'expéditeur*
   - Choisir un entier $m$ entre 1 et 255.
   - Chiffrer ce $m$ avec la clé publique : $c = K_{Eleve}^{pub}(m)$.
   - Envoyer par mail ce $c$ à votre voisin de droite.
2. *En tant que destinataire*
   - Déchiffrer le $c'$ que vous avez reçu avec la clé privée : $m' = K_{Eleve}^{priv}(c')$.

*a.* Pour faire bien, il faudrait que chaque élève ait son propre couple.

### Chiffrement symétrique (par flot) du message

Considérons le LFSR suivant :



1. *En tant qu'expéditeur*
   - Choisir un message de 2 caractères ASCII.
   - Encoder ce message sur 2 octets.
   - Initialiser le LFSR avec votre entier $m$ écrit sur un octet.
   - Avec ce LFSR, générer la suite binaire chiffrante de longueur 16.
   - Chiffrer votre message clair avec le principe du masque jetable de Vernam.
   - Envoyer par mail ce chiffré à votre voisin de droite.
2. *En tant que destinataire*
   - Déchiffrer, à l'aide de $c'$, le message chiffré reçu.

Un autre avantage de RSA, qui a fait sa popularité, est la possibilité de l'utiliser comme système d'authentification.

Comment Alice peut-elle être sûre que ce soit bien la clé publique de Basile?

# IV. Authentification des participants

## 1. Attaque de l'homme du milieu

Supposons qu'Alice soit cliente de l'établissement bancaire *BasileBanque*.
Les communications entre Alice et BasileBanque étant confidentielles, elles doivent évidemment être chiffrées, mais cela n'est pas suffisant.

En effet, Marion (la malveillante) peut procéder à l'attaque suivante. Elle envoie un email au format HTML [12] à Alice, lui demandant de se connecter d'urgence sur le site de la banque. Le contenu d'un tel email pourrait être :

```
<html>
...
    <body>
    Veuillez vous connecter d'urgence au site
    <a href="http://marion.com">BasileBanque.com</a>
    </body>
</html>
```

Ce message s'affichera de la façon suivante :

Veuillez vous connecter d'urgence au site **BasileBanque.com**

Si Alice n'est pas suffisamment méfiante ou tout simplement dans un état émotionnel qui diminuerait sa capacité à analyser la situation (par exemple, la peur de perdre beaucoup d'argent), si de plus Marion a pris le soin de développer une fraude de qualité [13] avec un mail très professionnel et en reproduisant parfaitement la page d'accueil du site BasileBanque.com, si enfin Marion a usé d'ingénierie sociale [14], alors Alice ne se rendra pas compte qu'elle est connectée au mauvais site!

Marion pourra ainsi recevoir tous les messages envoyés par Alice, elle pourra même les retransmettre au véritable site BasileBanque.com, en se faisant passer pour Alice, de manière passive ou en les modifiant (par exemple, en changeant le montant et le destinataire d'un virement bancaire).

Marion sera donc dans une position intermédiaire lui conférant un certain pouvoir, et malheureusement le chiffrement n'est d'aucune aide dans ce type d'attaque.

La faille fondamentale est ici l'**absence d'authentification**. Le serveur de Marion, qui imite le comportement du serveur de BasileBanque, n'a pas eu à prouver qu'il était bien celui qu'il prétendait être!

Voyons maintenant comment le système RSA peut résoudre ce problème...

[12]. Les applications de messagerie utilisent ce format pour représenter les emails avec « contenu riche » (images, polices différentes, couleurs, ...)
[13]. Cela est devenu particulièrement aisé avec l'essor de l'intelligence artificielle
[14]. Ensemble de techniques qui consistent pour un fraudeur à instaurer une relation de confiance avec une cible dans le but de la manipuler, pour lui soutirer des informations confidentielles ou lui faire exécuter des actions à des fins malveillantes

## 2. Certificats et tiers de confiance

Les communications en réseau sur Internet sont authentifiées de la manière suivante. Le participant qui veut prouver son identité présente un **certificat** qui doit être délivré par une entité, en laquelle les deux participants ont confiance, et appelée un **tiers de confiance**.

Si Alice souhaite correspondre avec BasileBanque, elle devra lui demander un certificat, délivré par un tiers de confiance, disons Theo. Plus précisément,

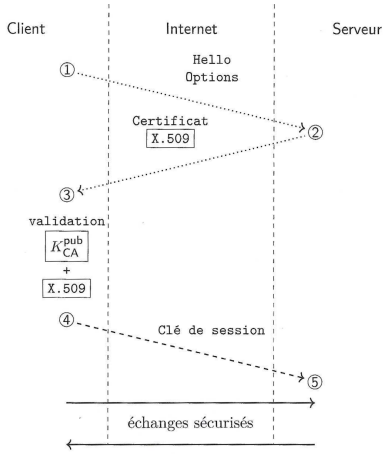
1. Basile fait vérifier son identité auprès de Theo qui va signer (avec sa clé privée) la clé publique de Basile, de manière à lui délivrer son certificat $s$ :
$$s = K_{Theo}^{priv}\left(K_{Basile}^{pub}\right)$$
2. Le site BasileBanque.com peut alors fournir à Alice qui souhaite s'y connecter ce certificat ainsi que la clé publique de Basile
3. En récupérant enfin la clé publique de Theo, Alice peut effectuer le calcul suivant qui est censé fournir la clé publique de Basile :
$$K_{Theo}^{pub}(s) = K_{Theo}^{pub}\left(K_{Theo}^{priv}\left(K_{Basile}^{pub}\right)\right) = K_{Basile}^{pub}$$
4. Si le résultat de ce calcul coïncide effectivement avec la clé publique de Basile, alors Alice aura bien authentifié le site et pourra initier avec lui une mise en accord de clé (avec Diffie-Hellman ou RSA) de manière totalement sécurisée !

# V. Protocole HTTPS

Le protocole HTTPS qui permet d'établir des communications sécurisées entre un client et un serveur web est en fait simplement le protocole HTTP, auquel a été ajouté une couche de cryptographie. Cette dernière est assurée par le protocole TLS qui permet l'authentification du serveur et la mise en place sécurisée d'une clé de chiffrement symétrique, appelée **clé de session**.

Une fois cette clé déterminée et connue du client et du serveur, les requêtes et les réponses HTTP peuvent être chiffrées avant l'envoi et déchiffrées lors de la réception. L'authentification empêche les *attaques de l'homme du milieu*, et le chiffrement empêche les routeurs et autres machines intermédiaires de lire les messages.

Voici la phase [15] de mise en place appelée « poignée de main TLS » :




[15]. N'étant pas compatible avec le protocole HTTP, un port différent a été choisi, à savoir le port 443

Les certificats sont délivrés par une **autorité de certification** (AC) et respectent le format X.509 présenté ci-dessous :

```
Certificate:
Data:
    Version: 3 (0x2)
    Issuer: C=US, O=Let's Encrypt, CN=Let's Encrypt Authority X3
    Validity
        Not Before: May  4 08:03:48 2020 GMT
        Not After : Aug  2 08:03:48 2020 GMT
    Subject: CN=nsi-terminale.fr
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
            Public-Key: (2048 bit)
            Modulus:
                00:ce:1d:f4:48:24:bd:0d:64:3a:74: ...
            Exponent: 65537 (0x10001)
Signature Algorithm: sha256WithRSAEncryption
     38:90:e9:9d:20:12:9d:35:b7:db:4f:64:14:fd:0e: ...
```

Pour terminer, on notera que l'AC ne chiffre pas la clé publique du serveur, mais un « résumé » de la partie grisée qui contient bien la clé publique du serveur.
C'est l'algorithme de **hachage**[16] *SHA256* qui produit ici ce « résumé » dont le chiffrement par l'AC fournit 38 :90 :e9 :...

[16]. Une fonction de hachage n'est pas inversible donc elle ne permet pas de retrouver toute la partie grisée, mais elle garantit que l'empreinte produite est unique.