Dans les cours et les tps précédents, nous avons vu comment les données circulées à travers internet avec 2 entités : source et destination. Il s'avère que des données peuvent être personnelles, confidentielles, vitales..., nous sômmes donc amenés à se poser un minium de questions :

- Comment chiffrer le contenu des communications pour que le contenu ne soit lisible que par la source et que par la destination ?
- Comment garantir que le serveur est bien la personne ou l'entité auquel on pense se connecter ?
- Comment garantir le chiffrement de la communication soit fiable sur tout internet ?

Pour votre culture globale, dans beaucoup de mes cours, je vais plus loin que le programme de terminale NSI . Heureusement, je précise toujours ce qui est à votre programme.

1 Un peu d'histoire de cryptologie :

Découvrons d'abord quelques terminologies :

— Cryptographie = chiffrement, messages secrets
— Cryptanalyse = déchiffrement
— Cryptologie= études de la cryptographie et de la cryptanalyse
— Protocoles de sécurité = protocoles de communication qui assurent la sécurité
— Autres fonctions utiles pour réaliser la sécurité :
  ☐ Fonctions de hachage = assurer l'intégrité
  ☐ Générateurs de nombres aléatoires = clés, nonce

Les propriétés de **sécurité** lors de la transmission de données (CIA) sont :

- Confidentialité = contrôle d'accès en lecture
- Intégrité = contrôle d'accès en écriture
- Authentification = vérification de l'identité pour prévenir toute usurpation
- Non-répudiation = ne pas pouvoir nier son envoi
- Contrôle d'accès = gestion des droits
- Audibilité = enregistrements des violations de la politique de sécurité

La **sureté** de fonctionnement est définie par :

- Disponibilité = rendre un service correct à un instant donné
- Fiabilité = capacité à rendre continuellement un service correct.

La taxinomie des **attaques** principales est :

- Attaques passives = analyse du trafic, interception des messages
- Attaques actives = modification, insertion,suppression, rejeu, déni de service

Important :

En informatique, lorsque l'on assure une bonne sécurité, cela se fait au détriment du temps { d'exécution et/ou de transmission} des échanges des données.

Le tableau ci-dessous pourrait servir à un support pour une frise chronologique sur la cryptologie, la liste est non exhaustive !

• la scytale le bâton du maréchal, spartiates grec • le carré de Polype • **le code de César** • la pierre de Rosette (Égypte) • Rongo Rongo de l'Ile de Pâques • le cadran d'Alberti	Antiquité
**********	**********
• **le carré de Vigenère:** Babbage et Vigenère • table d'Axel de Fersen (amant de Marie-Antoinette) • chiffres des hommes dansants (Sir Conan Doyle : Sherlock Holmes)	XVème - XVIIIème
**********	**********
• le télégraphe de Chappe • le cylindre de Jefferson • la réglette de St-Cyr	XVIIIème - XIXème
**********	**********
1ere guerre mondiale: • Allemand : code ADFFA 1917-Etats-unis entre en guerre contre l'Allemagne • Création d'Enigma en 1918 (Défaite) Télégramme Ziwermann Amélioration par Hitler – les Polonais se méfiaient d'Enigma(Rejewski) Enigma / « la bataille de l'atlantique » = milliers de clés possibles Bletchey Park = École du chiffre et du code - Alan Turing (Film «Imitation Game») • Code talker :Lenigre Navarro ou Basque. • Alliés: **Machine de Loren** – 1943 Colosses – Alain Turing • **Masque unique «One time Pad»** • les micro-points : texte ou image miniatures : sténographie • L'échange de clés Diffie-Hellman • **Le code RSA** • **Ips et https**	XXème siècle

L'histoire et le principe de Kerckchoffs (1883) ont démontré que la difficulté du déchiffrement ne doit pas dépendre du secret des algorithmes (des machines) mais du secret des clés. C'est-à-dire que **les messages en clair doivent comporter le moins de redondance possible → compression préalable ou extension en ajoutant du bruit (décupler la redondance).**

Pour information : les techniques pour gommer les redondances sont la confusion (=gomme les relations entre le texte en clair et le texte chiffré, ex substitution monoalphabétique) ; la diffusion (= disperse la redondance du texte en clair, ex : transposition) ; l'effet d'avalanche (=chaque bit de sortie dépend des bits du message en clair et des clés) ; la non-linéarité(=fonctions majorité sur 3bits, S-box de DES,...)

La cryptographie repose sur la **propriété substitution** (exemples : César, Vigenère, Playfair, WordPerfect, ...)

et **celle de transposition** (bâton de Plutarque, code ADFGVX, lorentz, la machine ENIGMA, ...)

Mais avant de rappeler les 2 types cryptographies, donnons une conclusion sur l'histoire des usages pour la cryptologie :

Les mots de passe et les clés sont les seules données numériques qui s'usent avec leurs utilisations. Les répétitions et l'information déduite sur un crytosystème sont exploitables par un attaquant $\Longrightarrow$ La discipline des utilisateurs est primordiale !

2 LA CRYPOGRAPHIE SYMÉTRIQUE :

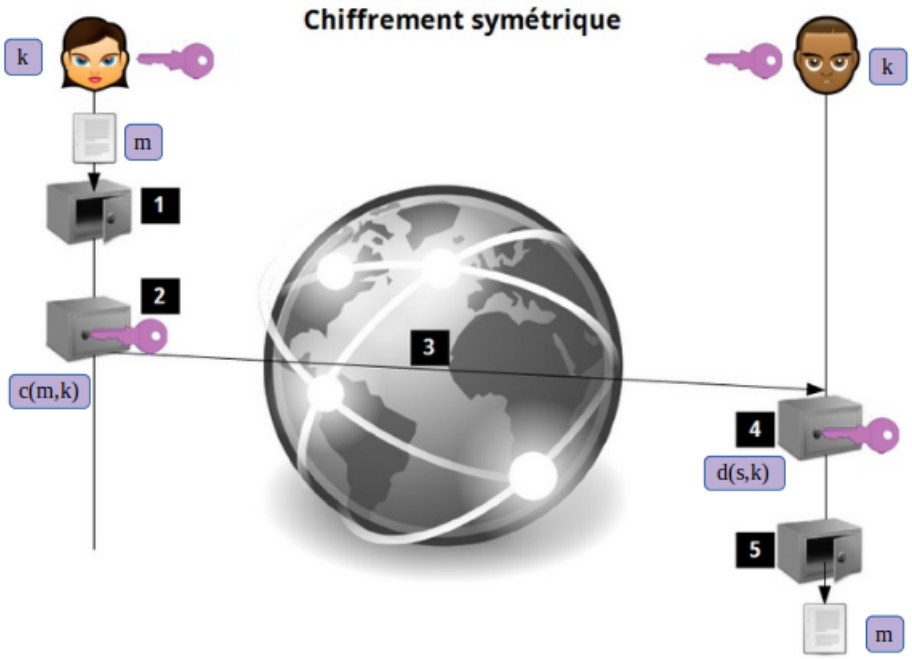
La même clé est utilisée pour chiffrer et déchiffrer, c'est pour cela qu'on l'appelle symétrique.

Formellement, on utilise 2 fonctions :

- **c(m,k)** est la fonction de chiffrement. Elle prend en arguments un message en clair m et une clé de chiffrement k et elle produit en sortie une chaîne de caractères chiffrée s.
- **d(s,k)** est la fonction de déchiffrement. Elle prend en arguments un message en chiffré s et une clé de déchiffrement k et elle produit en sortie une chaîne de caractères en clair m.

On pourra chiffrer en continu : STREAM CIPHERS la clé est étirée pour créer un flot de clés=keystream = masque sans jamais utiliser la même clé !

ou on chiffera par blocs : BLOCK CIPHERS principe du 'codebook' avec chacune des clés → confusion+diffusion.



Alice et Bob sont deux personnages récurrents lorsqu'on parle de cryptographie pour illustrer deux utilisateurs qui s'envoient un message chiffré. Alice est l'émettrice, Bob est le récepteur. Alice et Bob ont tous les deux la même clé qui permet d'ouvrir le coffre-fort.

— Alice met son message dans le coffre.

— Elle ferme le coffre à clé.

— Le coffre est transporté jusqu'à Bob .

— Bob ouvre le coffre avec son double de clé.

— Il peut lire le message.

EXERCICES :

— Écrire une fonction python **chiffrementCesar(message,decale)** qui ouvre dans le même répertoire de travail un fichier texte message en clair (message.txt) et qui enregistre un fichier sipher.txt avec un code de César qui décale les caractères de decale.

— Écrire une fonction python **dechiffrementCesar(sipher,decale)** qui récupère un fichier chiffré sipher.text et qui enregistre un fichier en clair messageD.txt avec la clé decale.

— le codage de Vigenère

"Demain, on se retrouve au musee du Louvre." avec le mot de passe : nsi et en utilisant le dictionnaire, alphaNum={'0' :0,'1' :1,'2' :2,"3" :3,"4" :4,"5" :5,"6" :6,"7" :7,"8" :8,'9' :9, 'a' :10,'b' :11,'c' :12,'d' :13,'e' :14,'f' :15,'g' :16,'h' :17,'i' :18, 'j' :19,'k' :20,'l' :21,'m' :22, 'n' :23,'o' :24,'p' :25,'q' :26,'r' :27,'s' :28, 't' :29,'u' :30,'v' :31,'w' :32,'x' :33,'y' :34,'z' :35} Écrire une fonction python **chiffrementVigenere(message,mdp)**

— A la maison, Écrire une fonction python **dechiffrementVigenere(message,mdp)**

— Écrire en Python une fonction **chiffreXor(msg,cle)** qui prend en arguments 2 chaînes d'octets (type bytes) et qui renvoie le chiffrement XOR du message avec la clé, sous forme d'une chaîne d'octets.

— Nous allons montrer que le chiffrement XOR n'apporte pas une grande sécurité. Soit la chaîne d'octets chiffrée :

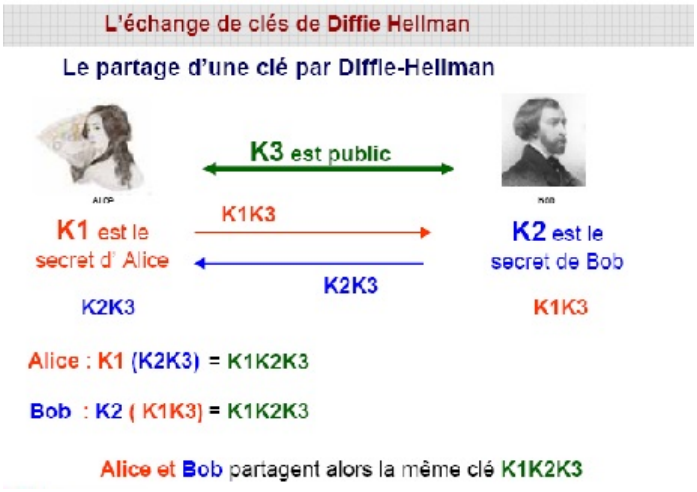
b'/0e6+y ;.< x-(7, ,/9b/0z48z :646<z*/9a/3(64+<'

On sait que les 4 derniers caractères du messsage en clair sont "nse !" et on sait aussi que la clé fait exactement 3 caractères en lettres majuscules et sans accents.

Écrire en Python une fonction **force()** qui utilise la fonction chiffreXor et qui essaye toutes les combinaisons de clé jusqu'à trouver la bonne. Mesurer le temps d'exécution avec la fonction time.time() du module time.

Comment fera-t-on l'échange des clés entre Alice et Bob ?

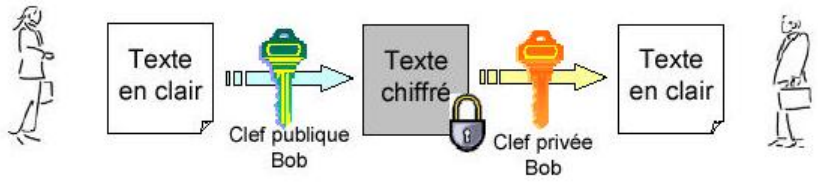
En 1976 est apparu le partage de clé par la méthode de DIFFIE-HELLMAN dont vous avez l'illustration ci-dessous :



3 LA CRYPOGRAPHIE ASYMÉTRIQUE :

Le principe d'un code asymétrique (aussi appelé à clé publique) est que, contrairement au code symétrique,les deux interlocuteurs ne partagent pas la même clé.

En effet, la personne qui veut envoyer un message utilise la clé publique de son correspondant. Celui-ci déchiffrera alors ce message à partir de sa clé privée que lui seul connait. On voit ici que contrairement à un codage symétrique, le chiffrement et le déchiffrement se font par des opérations complètement différentes.



Cette technique permet de répondre à la problématique du partage sécurisé des clés publiques entre des correspondants. En effet, en cas d'interception de la clé (publique) et d'un message codé par cette clé, l'intercepteur ne pourra pas retrouver le message d'origine, car il lui manque la clé privée possédée par le vrai destinataire du message codé.

L'exemple le plus connu de système cryptographique asymétrique est le **système RSA (1973 – Rivest / Shamir / Adleman)**.

Autre exemple : le protocole ssh2 utilise RSA avec Diffie-Hellman.

Mais la clé privée ne sert pas qu'à assurer la sécurité de la transmission des messages. En effet, si notre intercepteur, qui possède la clé publique, veut envoyer un message au vrai destinataire avec des intentions douteuses, comment le destinataire pourrait-il se rendre compte que l'expediteur n'est pas l'un de ses amis ?

La clé privée sert donc aussi à vérifier l'authenticité de l'identification de l'expéditeur d'un message codé par clé publique, c'est sa signature digitale .

- Soit A (Alice) l'expéditeur et B (Bob) le destinataire d'un message M.
- A possède sa clé privée prK(A) et une clé publique puK(A) qu'elle diffuse à B.
- B possède sa clé privée prK(B) et une clé publique puK(B) qu'il diffuse à A.
- M est codé par clé publique PuK(B), mais ce message est distribué avec un condensât du message, S, codé par la clé privée prK(A).
- M est déchiffré par la clé privée de B. Le résultat est lisible mais il manque l'authentification de l'expéditeur.
- S correspond donc en fait à la signature ou empreinte du message original, et donc de l'identifiant A.
- S doit être déchiffré par la clé publique de A. Si le résultat obtenu est le même que le condensât obtenu par la fonction de hachage sur le texte en clair, calculé par B, alors B est assuré de l'authenticité du message et de l'expéditeur.

Cette méthode d'authentification utilise donc la propriété des paires de clés asymétriques. Un message codé par clé publique peut être déchiffré par une clé privée et un message codé par une clé privée (ici le condensât) peut être décodé par une clé publique. Maintenant, on peut se demander ce qu'est un condensât, qui est le résultat d'une fonction de hachage sur un texte.

Retour sur RSA (1973 – Rivest / Shamir / Adleman) : cet algorithme est utilisé pour chiffrer et signer de façon sécuritaire lorsque on utilise une clé suffisamment grande soit 1024 bits au minimum, et actuellement : 4096bits.

Un peu de mathématiques expertes :

RSA utilise la théorie des nombres premiers et de la congruence pour de grands nombres, effectuons un exemple simple :

$n = p \cdot q$ en prenant p=61, q=71 nous avons n=4331 alors :

$\varphi(n) = (p-1) \cdot (q-1) = 60 * 70 = 4200$

prenons $\epsilon = 11$ (en général $\epsilon = 65537$),

ϵ doit être premier avec $\phi(n) = 2 \cdot 2 \cdot 2 \cdot 3 \cdot 5 \cdot 5 \cdot 7 = 4200$

on trouve la relation du théorème de Bézout suivante :

$-1909 \cdot 11 + 5 \cdot 4200 = 1 \rightarrow \epsilon^{-1} \equiv -1909 \equiv 2291[4200]$

On en déduit que la clé publique est le tuple (4331,11) et la clé privée est 2291 !

Il est alors facile de chiffrer ou déchiffrer un message :

Chiffrons le message M = 4301 < n = 4331

on a $4301 \cdot 11 \equiv 678[4331]$ DONC 678 envoyé sur le réseau et

il sera déchiffré par la clé privée : $678 \cdot 2291 \equiv 4301[4331]$

CQFD

NB : Astuces de calcul : Comment calculer assez facilement 678^{2291} ?

1 . $678^2 \equiv 45964 \equiv 598[4331]$ puis $678^3 \equiv 598 \cdot 678 \equiv 578678 \equiv 2661[4331]$ etc... Donc on ne manipule que n^2 au maximum !

2 . Nous avons en binaire : 2291 = 100011110011 et par ex $\epsilon = 65537 = 100000000000000001$

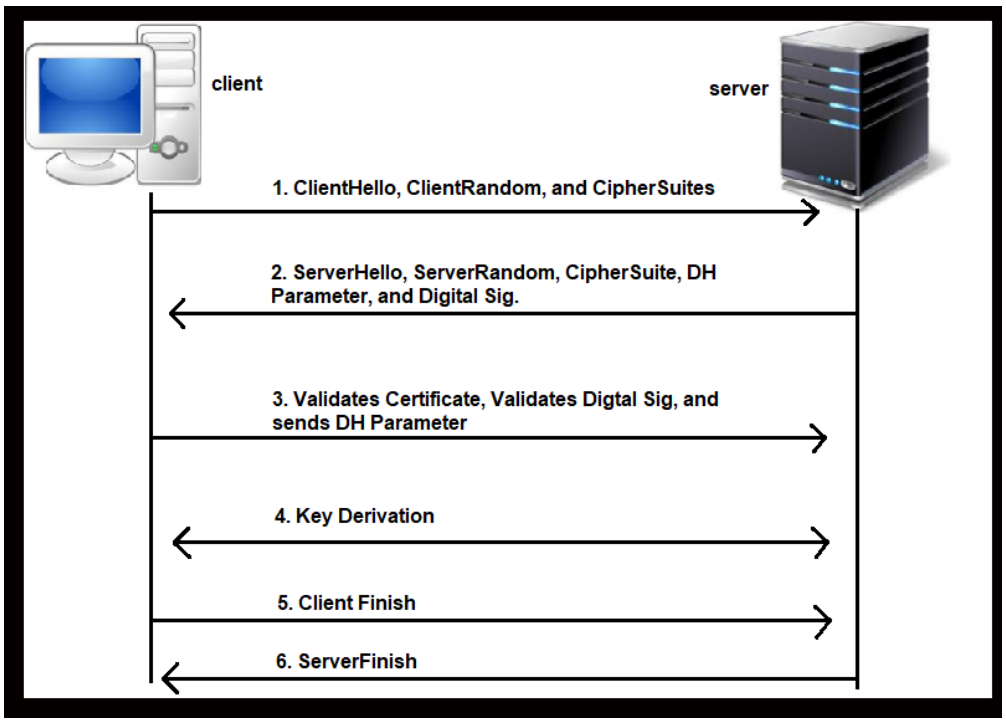
4 HTTPS ET LES PROTOCOLES DE SECURITE :

Sécurisation des communications.	Décrire les principes de chiffrement symétrique (clef partagée) et asymétrique (avec clef privée/clef publique). Décrire l'échange d'une clef symétrique en utilisant un protocole asymétrique pour sécuriser une communication HTTPS.	Les protocoles symétriques et asymétriques peuvent être illustrés en mode débranché, éventuellement avec description d'un chiffrement particulier. La négociation de la méthode chiffrement du protocole SSL (*Secure Sockets Layer*) n'est pas abordée.

Le protocole HTTPS : HyperTextTransfertPotocolSecure est un applicatif qui associe le protocole http avec le protocole TLS 1.3 TRANSPORT LAYER SECURITY 1.3 (ou SSL3.0 - SECURE SOCKETS LAYER 3.0).

L'IETF (Internet Engineering Task Force) qui publie les RFCs a affecté le port 443 au https.

Le principe est le suivant :



Pour visualiser ces échanges, nous allons faire l'exercice :

— lancer en tant qu'administrateur le logiciel Wireshark
— choisir la connexion internet
— Capturer / Démarrer
— lancer un navigateur et aller sur https ://monlycee.net
— retrouver les échanges du schéma ci-dessus

On pourra visualiser les échanges en sélectionnant le protocole TLS version 1.2 et 1.3 :

1946	340.774629	192.168.0.8	52.184.217.56	TLSv1.2	276	Client Hello
1948	340.858439	52.184.217.56	192.168.0.8	TLSv1.2	1026	Server Hello, Certificate, Server Key Exchange, Server Hello Done
1950	340.860853	192.168.0.8	52.184.217.56	TLSv1.2	212	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
1951	340.970077	52.184.217.56	192.168.0.8	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
1952	340.970077	52.184.217.56	192.168.0.8	TLSv1.2	123	Application Data
1954	340.972173	192.168.0.8	52.184.217.56	TLSv1.2	141	Application Data
1955	340.972521	192.168.0.8	52.184.217.56	TLSv1.2	92	Application Data
1956	340.972742	192.168.0.8	52.184.217.56	TLSv1.2	204	Application Data

899	307.354030	2a01:e34:ecb0:410:c…	2a01:4f8:192:209a::3	TLSv1.3	591	Client Hello
901	307.376414	2a01:4f8:192:209a::3	2a01:e34:ecb0:410:c…	TLSv1.3	1494	Server Hello, Change Cipher Spec, Application Data
903	307.376414	2a01:4f8:192:209a::3	2a01:e34:ecb0:410:c…	TLSv1.3	702	Application Data, Application Data, Application Data
919	307.450335	2a01:e34:ecb0:410:c…	2a01:4f8:192:209a::3	TLSv1.3	138	Change Cipher Spec, Application Data
920	307.452554	2a01:e34:ecb0:410:c…	2a01:4f8:192:209a::3	TLSv1.3	719	Application Data
921	307.468223	2a01:4f8:192:209a::3	2a01:e34:ecb0:410:c…	TLSv1.3	377	Application Data
922	307.468223	2a01:4f8:192:209a::3	2a01:e34:ecb0:410:c…	TLSv1.3	377	Application Data