Cours – Sécurisation des communications

Objectifs :

- ⇨ Savoir ce qu'est un chiffrement
- ⇨ Connaitre les deux types de chiffrements
- ⇨ Comprendre le principe d'une communication https
- ⇨

I – Un peu d'intimité

Lorsqu'on utilise internet, les informations que l'on échange transitent par tout un tas de routeurs dont on ne maîtrise pas le fonctionnement et qui peuvent, en plus d'acheminer l'information, la lire et même la copier.
Cela pose des problèmes à la fois en termes de vie privée (ais-je vraiment envie que des firmes américaines puissent connaître la totalité des forums que je consulte, ce que j'y lis et y écris ?), mais aussi en termes de sûreté.

Lorsqu'on se connecte à un site de banque en ligne par exemple, on voudrait :

- que la totalité des informations échangées (y compris le mot de passe de connexion) ne soient pas visibles par d'autres entités que la banque et nous-même ;
- être sûr que le site qu'on consulte est bien celui de notre banque ;
- être sûr que les informations transmises n'ont pas été altérées lors de leur transit su le réseau et sont bien complètes.

Pour répondre à ces problématiques (....., ..., ...), les informaticiens ont développé des systèmes basés sur le chiffrement.

Il faut noter que le chiffrement est très encadré en France sur le plan légal. Si l'utilisation de logiciels de chiffrement est maintenant autorisée, leur création ou leur distribution doit faire l'objet de déclaration ou d'autorisation préalables (sauf exceptions).

II – Qu'est-ce que le chiffrement ?

Le **chiffrement** est un procédé de cryptographie grâce auquel on souhaite rendre la compréhension d'un document impossible à toute personne qui n'a pas la clé de (dé)chiffrement.

Les militaires et les politiques ont de tout temps[1] utilisé des procédés de chiffrement pour rendre inintelligibles leurs messages internes à leurs adversaires.

Faisons tout de suite une clarification du vocabulaire :

- ➢ La ..... est la « science du secret ». Elle comprend la **cryptographie** (qui assure la protection de l'information en la rendant incompréhensible), mais aussi la **cryptoanalyse** (étude des messages crypté et des techniques de cryptographie pour en tirer des renseignements) et la stéganographie.
- ➢ Le ..... est le procédé visant à rendre l'information non compréhensible en utilisant un algorithme et une clé de chiffrement.

[1] La première utilisation attestée de techniques de chiffrement remonte aux égyptiens, vers 2000 avant JC

- Le ..... est l'opération permettant à partir de l'information chiffrée et de la clé de chiffrement de reconstituer l'information de départ.
- Le ..... consiste à retrouver l'information originale d'un message chiffré *sans en connaître la clé* de chiffrement.
- En français, ... un fichier est un terme impropre parce qu'il ne renvoie pas à la notion de clé de chiffrement. Il s'agit juste de rendre l'information incompréhensible. Concrètement, crypter un fichier est possible : cela signifie chiffrer un document sans connaître la clé de chiffrement.

Il existe deux catégories de système de chiffrement : les systèmes symétriques et les systèmes asymétriques.

III - Chiffrement symétrique

Le chiffrement symétrique ou chiffrement à clé secrète repose sur l'utilisation d'un algorithme réversible pour le chiffrement : ..... sert à chiffrer le message et à le déchiffrer.

Un tel système repose donc sur la communication sûre de la clé de chiffrement.
La sécurité offerte par le chiffrement est liée à la complexité de l'algorithme utilisé et à la longueur de la clé : plus l'algorithme est complexe, plus il est difficile à décrypter et plus la clé est longue, moins il est aisé de casser le chiffrement[2].

Par « difficile à décrypter », on entend « nécessite beaucoup de puissance de calcul (donc de temps) pour le décrypter ».

Les systèmes les plus utilisés reposent sur des algorithmes connus de tous et dont seule la clé secrète assure la sûreté. Il s'agit de DES (Data Encryption Standard), Blowfish, IDEA (International Data Encryption Algorithm) ou AES (Advanced Encryption Standard).

Ces algorithmes sont assez complexes et sont souvent basés sur l'opérateur logique OU EXCLUSIF (XOR) noté $\oplus$ qui est un opérateur par nature réversible.

E1	E2	S = E1 $\oplus$ E2
0	0	0
0	1	1
1	0	1
1	1	0

Table de vérité du OU exclusif

Le chiffrement symétrique permet de solutionner le problème de la ....., mais il est ne permet pas de traiter celui de ..... ou celui de .....

IV - Fonctions de hachage

Une **fonction de hachage** est une fonction qui prend en entré un message de taille quelconque et produit un haché (appelé aussi condensat, signature, empreinte ou *hash* en anglais) de taille fixe (par exemple 128 bits) correspondant à cette entrée.

La taille de l'entrée étant potentiellement très grande, il n'y a pas unicité du haché, c'est-à-dire qu'il peut exister plusieurs entrées distinctes qui produisent le même haché. On parle alors de **collision**. On écrit les fonctions de hachage de telle sorte que les cas de collision soient très rares.

Les fonctions de hachage sont très utiles en informatique, notamment pour les **tables de hachage** sur lesquelles sont basés les dictionnaires et qui permettent des recherches sur des tables en temps constant.

Une **fonction de hachage cryptographique** est une fonction de hachage pour laquelle le calcul du haché se fait dans un temps raisonnable (mais pas trop court) et pour laquelle l'opération inverse qui à partir du haché permettrait de retrouver l'entrée n'est pas possible.

[2] « Casser le chiffrement » est un synonyme de « décrypter » (déchiffrer sans connaître la clé).

Si on change ne serait-ce qu'un seul bit du message d'entrée, le condensat sera complètement différent.
De même, la taille de l'empreinte étant fixe, il est impossible de connaître la taille du message d'entrée en examinant le haché.

Entrée	Haché (md5)
"abc"	900150983cd24fb0d6963f7d28e17f72
"Abc"	35593b7ce5020eae3ca68fd5b6f3e031
Livre des fables de la fontaine (578 001 caractères)	0c1e2090b6bbaf0b72af07547648e5e9
Livre des fables de la fontaine en changeant 1 lettre	2b884392e83d1e179927a423c4faf2f7

Hash de quelques messages

Les algorithmes de hachage sont utilisés pour assurer ..... des données. Pour cela, il faut que celui qui reçoit les données connaisse également la signature des données. Une fois les données récupérées, il calcule le condensat de ce qu'il a reçu et le compare à la signature attendue. Si elle est identique c'est qu'il est bien en possession des données attendues. Si elle est différente il peut juste savoir que les données reçues ne correspondent pas exactement, mais il ne peut pas savoir à quel point elles ont été altérées ni à quel endroit.

Il ne faut pas confondre les méthodes de vérification d'intégrité basiques de la transmission comme le bit de parité ou le code de redondance cyclique (CRC) avec les fonctions de hachage : le but des première est de se prémunir contre des erreurs de transmissions « involontaires » (liées à la qualité du moyen de transmission) tandis que les secondes permettent également de se prémunir contre les altérations volontaires (et malicieuses) des données.

Il existe de nombreuses fonctions de hachage cryptographique comme md5 (Message Digest Algorithm 5), SHA-1 (Secure Hash Algorithm 1), SHA-2 ou SHA-3.

md5 et SHA-1 possèdent des failles de sécurité identifiées et ne sont plus considérées comme sûres pour la cryptographie.

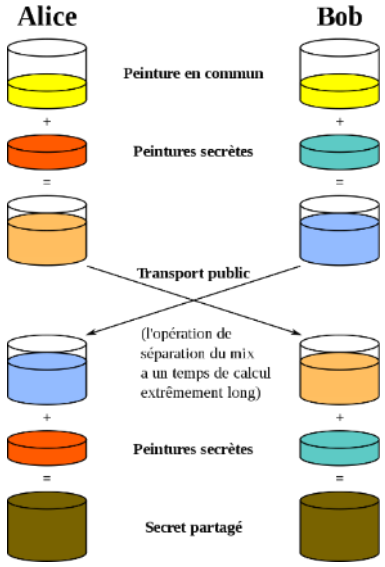
Application 1 :
Aller sur la page de téléchargement du logiciel avidemux : https://www.fosshub.com/Avidemux.html et télécharger la version 64 bits pour windows (« Avidemux 64-bits Windows Installer » - premier choix de la liste). Pour vérifier que vous avez téléchargé le bon fichier, utiliser l'application portable « hashtool.exe » qui se trouve dans le répertoire de l'activité. Comparer l'empreinte du fichier téléchargé avec celle indiquée sur le site (cliquez sur « signature » sur la ligne du fichier dans la page web). Est-ce le bon fichier ?

V - Chiffrement asymétrique

Le chiffrement asymétrique, ou chiffrement à clé publique permet de résoudre certains problèmes qu'il y avait avec les chiffrements symétriques, notamment la nécessité d'échanger par un moyen sûr la clé de chiffrement sur laquelle repose toute l'efficacité du cryptage.

Les cryptologues américains Whitfield Diffie et Martin Hellman imaginent en 1976 une méthode d'échange de clé par un canal peu sûr.
Le schéma ci-contre explique le principe en utilisant des couleurs, mais la méthode présentée par Diffie et Hellman est basée sur des nombres exponentiés.

C'est en 1977 que Ronald Rivest, Adi Shamir et Leonard Adleman découvrent le premier chiffrement asymétrique : RSA (d'après les initiales des 3 mathématiciens). Ce chiffrement est encore utilisé de nos jours et c'est lui dont nous allons exposer le principe dans les parties suivantes.



Source : Wikipédia

1) Principe

Le cryptage asymétrique repose sur l'utilisation de **deux clés qui sont liées entre elles** :

- une ..... que l'utilisateur doit donner à des tiers pour communiquer ensuite avec eux de manière sécurisée. Comme son nom l'indique elle peut (et doit) être rendue publique sans que cela compromette la sécurité du système.
- une ..... (ou clé secrète) qui elle doit rester secrète et qui servira à déchiffrer les messages qui ont été chiffrés avec la clé publique.

Les clés sont générées aléatoirement (et non choisies comme un mot de passe peut l'être) et conjointement par un programme. La longueur de la clé est ici un critère important de sûreté[3].

On note M_{clair} le message en clair (non chiffré), $C_K(M)$, le chiffrement par l'algorithme de chiffrement C en utilisant la clé K du message M et $M_{chiffré}$ le message chiffré.
On a donc $M_{chiffré} = C_{clé}(M_{clair})$

Notons également KP la clé publique et KS la clé secrète correspondante. Les propriétés mathématiques des clés font que :

$$C_{KS}(C_{KP}(M)) = M \quad \text{et} \quad C_{KP}(C_{KS}(M)) = M$$

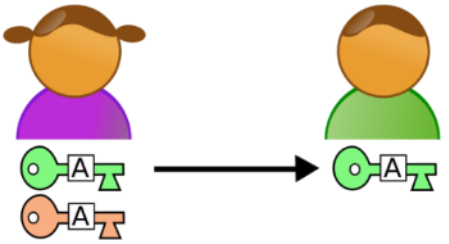
Autrement dit :

- ✓ ce que la clé publique chiffre, ... ;
- ✓ ce que la clé secrète chiffre, ...

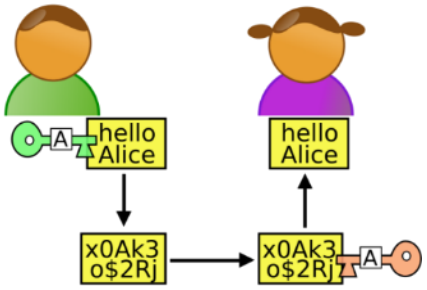
C'est cette propriété très particulière qui fait que le système permet ensuite de chiffrer un message à destination d'un utilisateur particulier mais également d'authentifier l'origine d'un message (signature numérique).

a. Chiffrement

Lorsque deux personnes (Alice et Bob[4]) veulent échanger des informations par un canal non sécurisé (comme le courriel), elles doivent tout d'abord chacune générer un couple clé publique/clé privée. Elles échangent ensuite leurs clés publiques (Bob communique sa clé publique à Alice et Alice communique sa clé publique à Bob).
Pour chiffrer un message à destination d'Alice, Bob utilisera alors la clé publique d'Alice (et un programme de cryptographie asymétrique). Par la suite ce message ne pourra être déchiffré qu'avec la clé privée d'Alice (et donc par Alice seule). De même si Alice veut répondre à Bob, elle chiffrera le message avec la clé publique de Bob. Bob pourra alors déchiffrer le message avec sa propre clé privée.



1re étape : Alice génère deux clefs. La clef publique (verte) qu'elle envoie à Bob et la clef privée (rouge) qu'elle conserve précieusement sans la divulguer à quiconque.



2ème et 3ème étapes : Bob chiffre le message avec la clef publique d'Alice et envoie le texte chiffré. Alice déchiffre le message grâce à sa clef privée.

Source : Wikipédia

b. Authentification

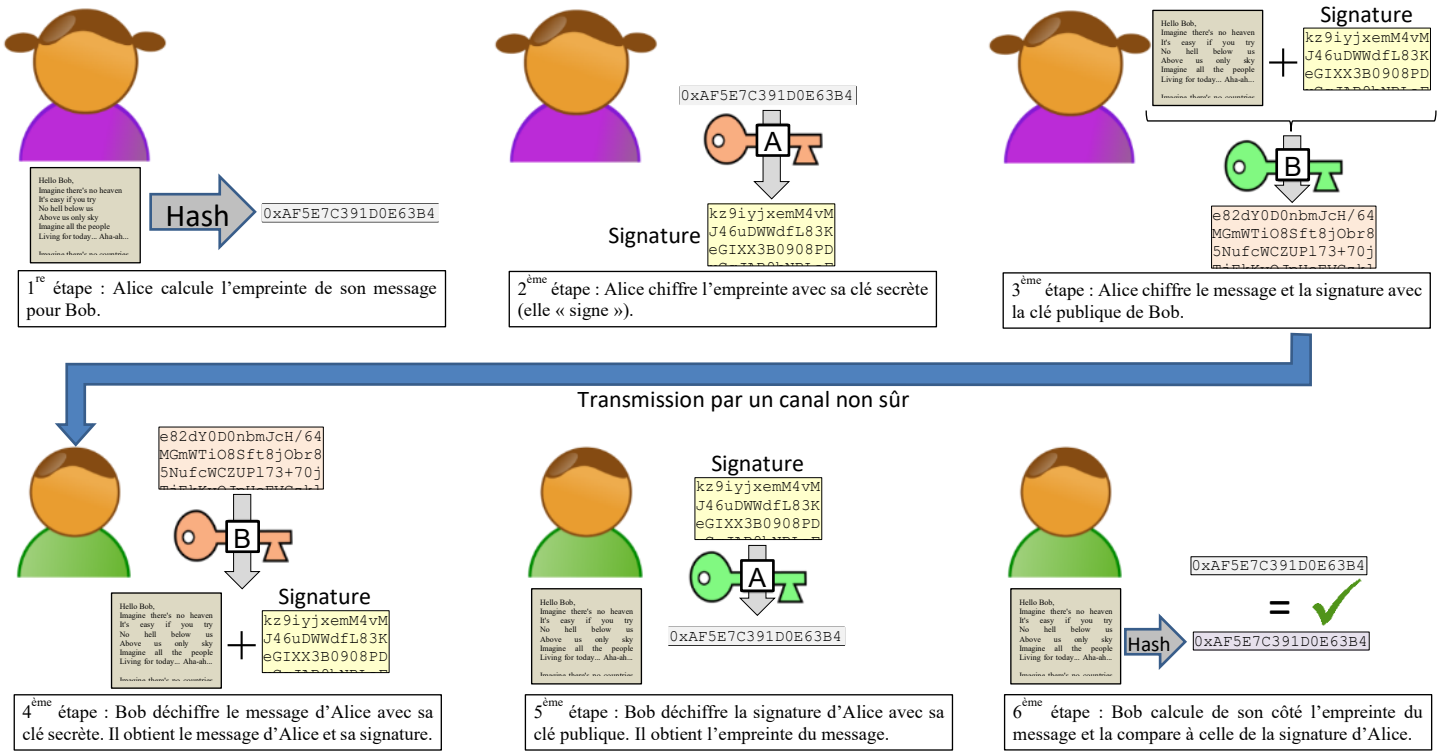
Le chiffrement asymétrique est ....., ce qui veut dire que si comme on vient de le voir ce qui est chiffré avec la clé publique peut être déchiffré uniquement avec la clé privée correspondante, l'inverse est également vrai : on peut chiffrer un message avec une clé privée qui ne sera déchiffrable qu'avec la clé publique correspondante.

Ainsi lorsqu'Alice envoie un message à Bob, elle peut calculer l'empreinte du message et chiffrer celle-ci avec sa clé privée (on appelle cela « signer numériquement ») avant de chiffrer le tout (message + signature) avec la clé publique de Bob.

Lorsque Bob reçoit le message, il commence par le déchiffrer avec sa clé privée, ce qui lui permet de voir le message d'Alice et sa signature. Puis il déchiffre la signature avec la clé publique d'Alice ce qui lui donne l'empreinte du message. Il n'a plus alors qu'à comparer l'empreinte du message reçu (qu'il calcule lui-même)

[3] En 2022, des clés de moins de 2048 bits pour RSA sont considérées comme non sûres. Les transactions bancaires utilisent des clés de 4096 bits.
[4] Plutôt que A et B, les ouvrages sur le chiffrement utilisent généralement ces deux prénoms pour représenter les partenaires d'une communication numérique chiffrée.

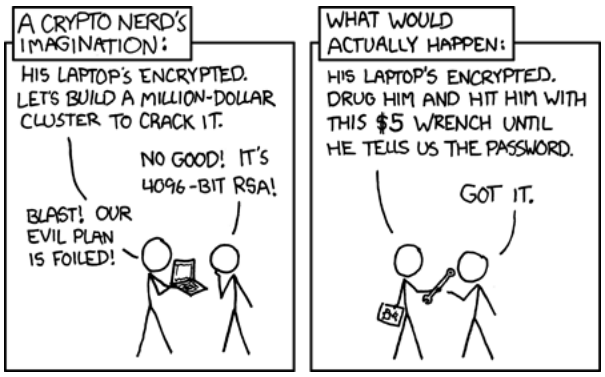
avec celle contenue dans la signature d'Alice. Si elles correspondent, c'est qu'Alice est bien l'auteur de ce message et que le message n'a pas été altéré.



Ce procédé de calcul d'empreinte est aussi utilisé pour vérifier que l'on a bien téléchargé le bon programme par exemple.

Le chiffrement asymétrique demande beaucoup de puissance de calcul et il est donc généralement utilisé pour chiffrer une petite quantité de données (un courriel, un certificat, une clé de chiffrement, …). Pour chiffrer de gros volumes de données, on utilisera de préférence un algorithme symétrique.

VI - Vulnérabilités



D'après le site xkcd.com

Aucun système de chiffrement n'est infaillible, et avec l'évolution de la technologie et de la cryptologie, de nouvelles failles sont régulièrement trouvées dans des méthodes de chiffrement, tandis que des clés de plus en plus grandes sont « cassées » par les programmes. La cryptologie est donc un domaine en évolution constante et une méthode de chiffrement réputée « sûre » un jour peut être considérée comme « compromise » (et donc non fiable) le lendemain.
Nous allons dans cette partie lister quelques façons classiques d'attaquer des systèmes de chiffrement.

1) La force brute

On peut essayer de déchiffrer le message en essayant toutes les combinaisons de clé possibles.
Par exemple si la clé est un nombre entier positif, on essayera successivement 1, 2, 3, … jusqu'à ce que le résultat du déchiffrement donne un texte cohérent.

Cette technique est très rudimentaire :

- Elle nécessite de connaître l'algorithme de chiffrement (ce sera souvent le cas)

- Il faut également disposer d'un bout d'information chiffrée et si possible de connaître l'information en clair correspondante
- Elle peut nécessiter un temps de calcul rédhibitoire si la clé est trop grande ou trop complexe
- Elle fonctionne sur tous les types de chiffrement

Application 2 :

1) Combien y a-t-il de mot de passe de 7 lettres utilisant uniquement des lettres minuscules ?

2) Même question si on utilise maintenant les majuscules également puis les chiffres.

3) Si on considère qu'un ordinateur puissant peut essayer 100 millions de combinaisons par secondes, combien de temps lui faudra-t-il dans le pire des cas pour casser le mot de passe dans chaque cas ?

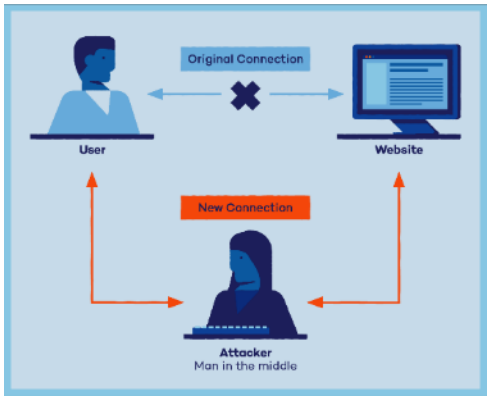
3) Avec le même ordinateur, on veut casser un mot de passe de 10 lettres minuscules. Combien de temps cela prendra-t-il pour essayer toutes les combinaisons ? Finalement est-il vraiment important d'augmenter la taille des mots de passe ?

2) L'attaque par dictionnaire

Comme pour l'attaque par force brute, on va essayer plusieur clé au hasard en espérant trouver la bonne. Ici on parie sur la faiblesse humaine et au lieu de tester tous les possible, on va juste essayer une série de mot de passe « classiques » repertoriés dans une base de donnée de mots de passe fréquents.

Cette méthode est bien plus rapide que la force brute (il suffit généralement d'une seconde pour tester tous les mots de passe du dictionnaire) mais inefficace si le mot de passe a été généré aléatoirement ou s'il s'agit d'une phrase complète par exemple.

3) L'attaque de l'homme du milieu (Man in the middle)

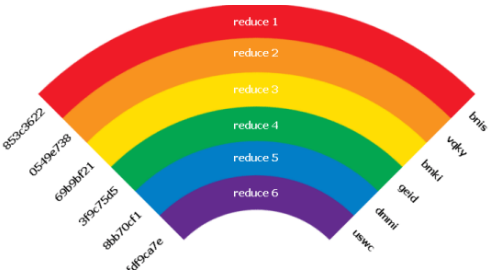


Dans ce type d'attaque, le pirate se place au milieu d'une communication entre deux postes A et B (par exemple un utilisateur et un serveur web). Toutes les informations vont alors transiter par lui, notamment les échanges de clés. Il peut ainsi récupérer la clé de A et fournir sa propre clé à B. qui croira que c'est celle de B. De même il reçoit la clé de B et transmet une autre clé à A qui croira qu'elle vient de B.

Ainsi A croit que le pirate est B et B croit que le pirate est A. Le pirate transférera toutes les requêtes de A vers B et inversement de manière à ce qu'auncun des deux ne se doute que la communication est écoutée.

Il doit pour cela être capable d'intercepter la requête originale (par exemple la demande de page web du client), ce qui peut se faire avec différentes techniques comme le DNS Spoofing.

4) Les tables arc-en-ciel



Les tables arc-en-ciel (*rainbow tables*) permettent de stocker des tableaux de différentes valeurs de haché et les mots de passe en clair correspondant. Connaissant l'empreinte d'un mot de passe, on peut ainsi retrouver le mot de passe en clair, contournant l'irréversibilité de la fonction de hachage.
La réalisation pratique d'une telle table pose plusieurs problèmes évidemment : celui du temps de calcul extrêmement long (mais qui peut être réparti sur plusieurs ordinateurs assez simplement et qui n'a à être accompli qu'une seule fois) et du stockage d'une telle table.

Application 3 :

1) Combien d'entrées contiendrait une table listant tous les hachés possibles avec l'algorithme de hachage md5 (qui produit une sortie sur 128 bits) ?

2) En déduire la taille de la table en mémoire si chaque mot de passe en clair comporte 20 caractères (20 octets).

Dans la pratique pour limiter la taille de ces tables, on utilise des fonctions de réduction qui en échange d'un temps de calcul permettent de réduire considérablement la taille de la table (c'est ces différentes fonctions de réductions successivement appliquées qui donnent son nom à la table arc-en-ciel). De même on ne va pas mettre tous les hachés possibles dans la table mais seulement une sélection.
Dans la pratique de telles tables font des tailles de l'ordre de la centaine de gigaoctet.

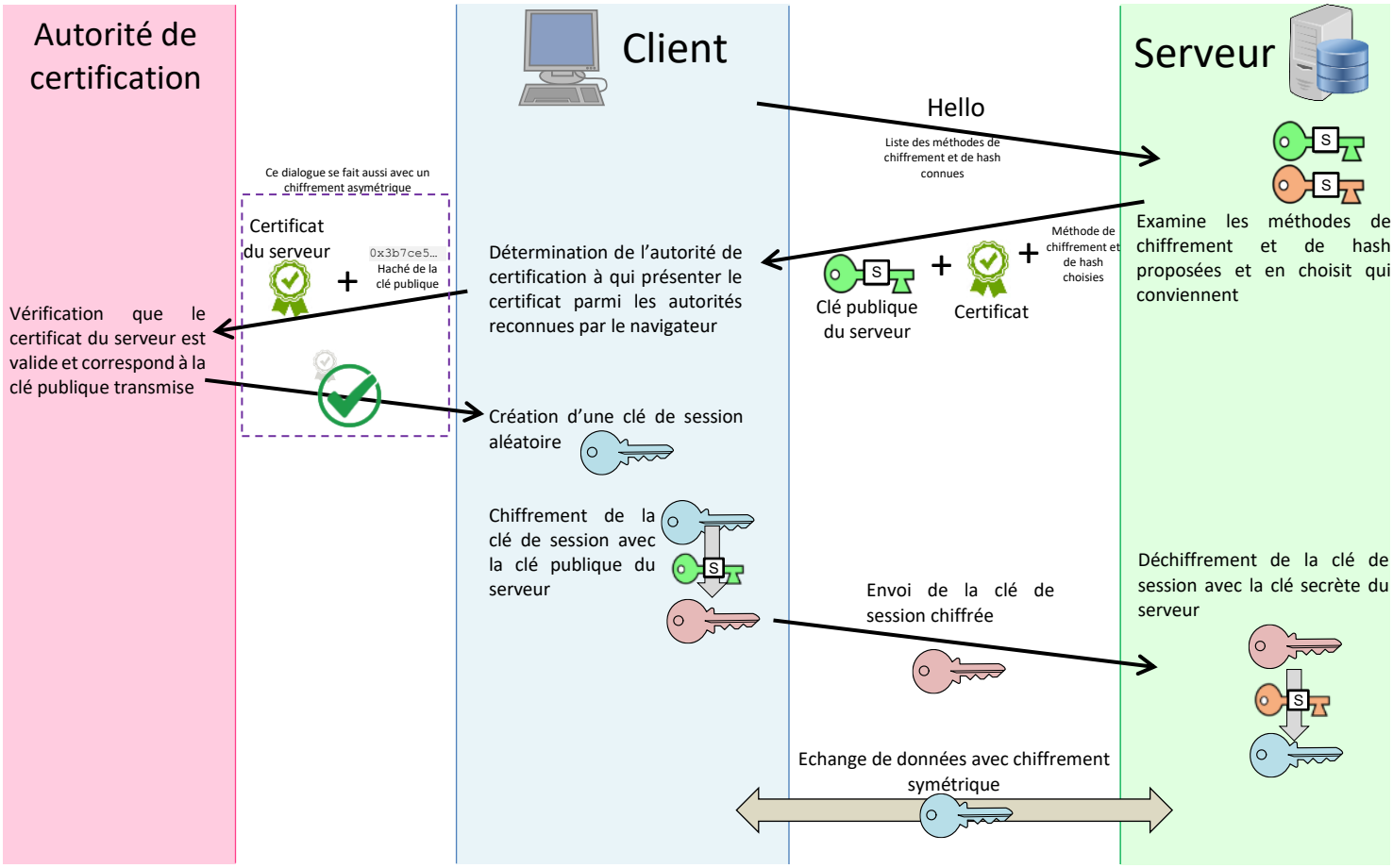
Chaque table arc-en-ciel ne correspond qu'à un seul algorithme de hachage (on trouve sur internet des tables arc-en-ciel pour md5 et SHA-1 sans difficulté) mais se montre très efficace si on récupère la base de données contenant les hash des mots de passe.

Il existe cependant une contre-mesure assez efficace aux tables arc-en-ciel : l'utilisation de sel[5] (et éventuellement de poivre).

VII - Le protocole https

Lorsqu'un navigateur demande une page à un serveur web, il utilise le protocole http (**H**yper**T**ext **T**ransport **P**rotocol). Celui-ci permet d'échanger facilement des données entre le serveur et le client mais il fait transiter les données en clair sur le réseau ce qui permet à n'importe quelle machine sur le parcours de prendre connaissance des informations échangées (notamment les mots de passe de connexion).
Pour pallier à ce problème, le protocole **https** (**H**yper**T**ext **T**ransfer **P**rotocol **S**ecure) a été développé afin d'assurer la confidentialité et l'authentification.
Ceci est obtenu en chiffrant toutes les communications avec un chiffrement symétrique utilisant une clé aléatoire (valable uniquement le temps de la session) sur laquelle le client et le serveur se seront mis d'accord en utilisant un chiffrement asymétrique.

[5] Cela consiste à ajouter une valeur aléatoire (le « sel » ou *salt* en anglais) au mot de passe avant de le hacher. Cette valeur est stockée avec le mot de passe et oblige un attaquant à recalculer toute la table avec cette valeur de salaison. La valeur du sel étant différente pour chaque mot de passe, il devient impossible d'utiliser les tables arc-en-ciel. Le poivre (*pepper*) quant à lui est une valeur secrète ajoutée à tous les mots de passe (la même pour tous).



Le client initie une connexion avec le serveur et les deux se mettent d'accord sur une méthode de chiffrement à utiliser.
Puis le serveur envoi au client sa clé publique ainsi que son certificat.
Le client vérifie le certificat en utilisant la clé publique de l'autorité qui l'a certifié (les navigateurs possèdent les clés publiques des principales autorités de certification). Il vérifie aussi que la clé publique correspond bien à celle décrite dans le certificat et que le certificat est encore valide et n'a pas été révoqué.
Le client est alors certain de s'adresser véritablement au serveur (et pas à un pirate).
Le client génère alors une clé de session aléatoire et suffisamment forte qu'il chiffre avec la clé publique du serveur. Il peut ensuite transmettre la clé chiffrée par le canal de transmission car seul la clé secrète du serveur pourra déchiffrer son message.
Le serveur récupère donc la clé de session en déchiffrant l'envoi. La phase de concertation (Handshake) est terminée et les deux pourront désormais communiquer en chiffrant leurs communications avec la clé de session en utilisant l'algorithme de chiffrage symétrique dont ils étaient convenu au départ.

Application 4 :
Aller sur la page du site web du lycée et récupérer les informations sur le certificat utilisé (il faut cliquer sur le cadenas à côté de l'url). Quelles informations importantes peut-on y trouver ?

Références :

Histoire du chiffrement : https://repo.zenk-security.com/Cryptographie%20.%20Algorithmes%20.%20Steganographie/RSA.pdf
Animation sur le chiffrement AES : https://formaestudio.com/portfolio/aes-animation/
RSA : https://interstices.info/nombres-premiers-et-cryptologie-lalgorithme-rsa/ et https://samuelgallay.github.io/CryptoTPE/rsa/
Chiffrement asymétrique : https://technique-et-droit-du-numerique.fr/chiffrement-asymetrique-a-cle-privee-et-cle-publique/
Fonctions de hachage : https://www.dcode.fr/fonction-hash
Tables arc-en-ciel : https://www.ionos.fr/digitalguide/serveur/securite/rainbow-tables
Hachage et salaison des mots de passe : https://crackstation.net/hashing-security.htm
https : https://www.ionos.fr/digitalguide/hebergement/aspects-techniques/le-https-cest-quoi/ et https://www.digitalberry.fr/handshake-protocole-tls-authentification/ et en vidéo : https://yewtu.be/watch?v=_UpuZ0Y3k-c&t=70s
Certificats : https://www.ssl.com/fr/article/navigateurs-et-validation-de-certificat/
Vidéo chiffrement et https : https://yewtu.be/watch?v=7W7WPMX7arI
Diaporama assez complet : http://perso.inforoutes.fr/grozanne/cryptographie/sld001.htm

Exercices – Sécurisation des communications

Exercice 1 : Les différents problèmes de sécurité

1) Associer à chaque objectif de sécurisation la bonne définition.

Confidentialité •	• S'assurer que les données transmises sont complètes et n'ont pas été altérées.
Intégrité •	• Vérifier l'identité de l'interlocuteur avec qui on échange.
Authentification •	• Faire en sorte que seuls les participants officiels d'une communication puissent en connaître le contenu.

2) Cocher les cases correspondant aux problèmes que permet de régler chaque outil.

Chiffrement symétrique	Algorithmes de hachage	Chiffrement asymétrique	
			Confidentialité
			Intégrité
			Authentification

Exercice 2 : HTTPS dans le désordre

Classer les propositions suivantes par ordre chronologique :

1. Le navigateur génère aléatoirement une clé de session
2. Le serveur déchiffre la clé de session en utilisant sa clé privée
3. Le navigateur du client demande le certificat du serveur et sa clé publique
4. Le serveur établit une session de communication par chiffrement symétrique avec le client
5. Le navigateur envoie la clé de session chiffrée au serveur
6. Le navigateur chiffre la clé de session à l'aide de la clé publique
7. Le navigateur reçoit le certificat et la clé publique du serveur
8. Le navigateur vérifie l'authenticité du certificat
9. Le serveur envoie le certificat d'authenticité et la clef publique au navigateur du client

Exercice 3 : Algorithmes de hash

Ecrire un programme python utilisant la bibliothèque hashlib pour calculer les signatures suivantes :

1) `"Premier test"` avec la fonction md5.
2) `"Deuxième test"` avec la fonction sha-256.
3) Ecrire un autre programme pour retrouver le mot de passe dont la signature md5 est : `031112dea62ade96196888e460609fb0`. Indice : le mot de passe est une chaîne de caractères constituée uniquement de chiffres.