Module Cyber Sécurité

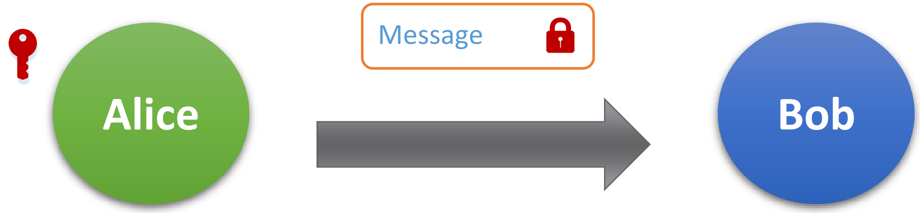
Chiffrement asymétrique
Certificat et TLS

Jean-marc MULLER
Sébastien SCHMITT

Chiffrement asymétrique

- Problématique majeure du chiffrement symétrique

  ➡ Alice veut envoyer un message chiffré avec une clé à Bob
  ➡ Bob ne dispose pas de la clé

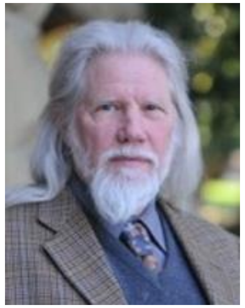


Comment transmettre la clé à Bob de manière sécurisée

Chiffrement asymétrique

- Système échange de clé DIFFIE-HELLMAN

  - ➡ Créé par Whitfield Diffie et Martin Hellman
  - ➡ Objectif : transmettre une clé symétrique en secret
  - ➡ création : clé symétrique (secrète) par échange de valeurs "publiques"
  - ➡ Se base sur la notion de nombres premiers et d'inverses modulaires


Whitfield DIFFIE

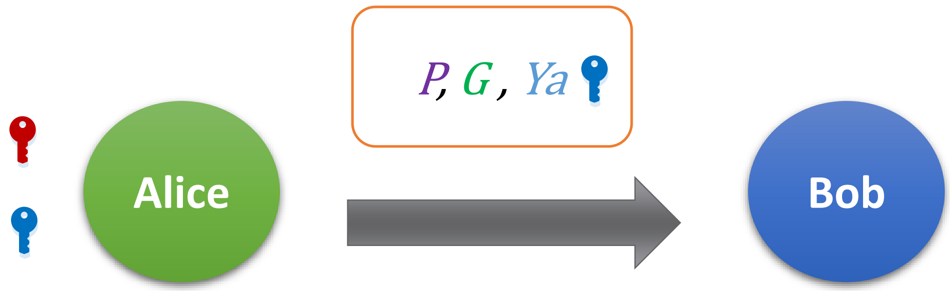

Martin HELLMAN

Chiffrement asymétrique

- Système échange de clés DIFFIE-HELLMAN

  ➡ Alice génère

    - **Module** : Grand nombre **premier** noté P
    - **Base** : nombre G qui doit être inférieur à P
    - **Valeur privée** : nombre Xa aléatoire
    - **Valeur publique** : nombre $Ya = G^{Xa}\ mod\ P$



Chiffrement asymétrique

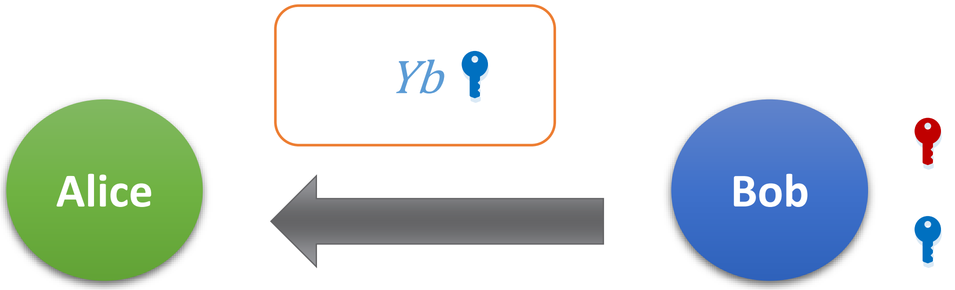
- Système échange de clés DIFFIE-HELLMAN

  ➡ Bob dispose :

    ➡ **Module** : noté P
    ➡ **Base** : nombre G

  ➡ Bob génère :

    ➡ **Valeur privée** : nombre Xb aléatoire
    ➡ **Valeur publique** : nombre $Yb = G^{Xb}\ mod\ P$



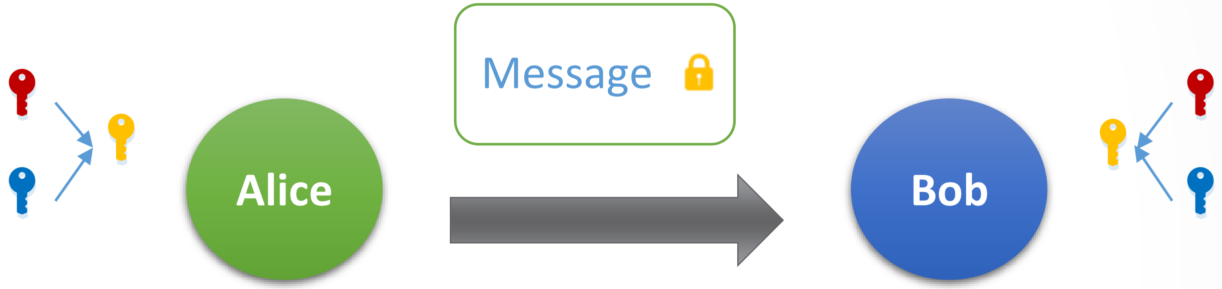
Chiffrement asymétrique

- Système échange de clés DIFFIE-HELLMAN

  - Valeurs
    - **Alice** : dispose de P, G, Xa, Ya, Yb
    - **Bob** : dispose de P, G, Xb et Yb, Ya

  - Calcul de la clé de session secrète partagée
    - **Alice** : $s = Yb^{Xa} \ mod \ P$
    - **Bob** : $s = Ya^{Xb} \ mod \ P$

  - Vérification
    - On sait que $Ya = G^{Xa} \ mod \ P$ et $Yb = G^{Xb} \ mod \ P$
    - Si on remplace Ya Alors $s = (G^{Xa} \ mod \ P)^{Xb} \ mod \ P = G^{XaXb} mod \ P$
    - Si on remplace Yb Alors $s = (G^{Xb} \ mod \ P)^{Xa} \ mod \ P = G^{XbXa} mod \ P$



Chiffrement asymétrique

- Chiffrement asymétrique
  - ➡ Cryptographie à clé publique
    - ➡ Avoir un grande difficulté au décryptage d'un message chiffré
    - ➡ Création d'une paire de clés liées mathématiquement
  - ➡ Algorithmes de chiffrement asymétrique

Benaloh · Blum–Goldwasser · Cayley–Purser · CEILIDH · Cramer–Shoup · Damgård–Jurik · DH · DSA · EPOC · ECDH · ECDSA · EKE · ElGamal (encryption · signature scheme) · GMR · Goldwasser–Micali · HFE · IES · Lamport · McEliece · Merkle–Hellman · MQV · Naccache–Stern · NTRUEncrypt · NTRUSign · Paillier · Rabin · RSA · Okamoto–Uchiyama · Schnorr · Schmidt– Samoa · SPEKE · SRP · STS · Three-pass protocol · XTR

Chiffrement asymétrique

- Généralité chiffrement asymétrique

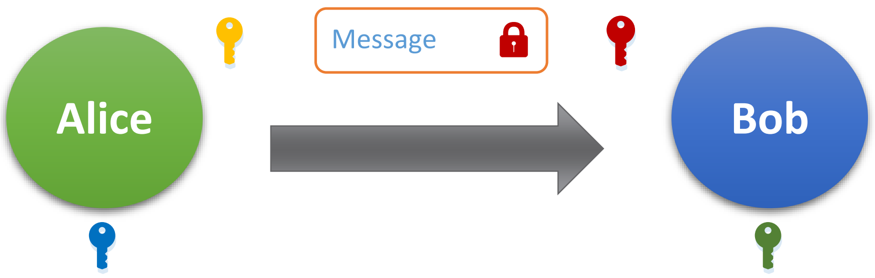
  - ➡ La clé publique est dite "publique" donc lisible par tout le monde
  - ➡ La clé privée est "privée" doit être impérativement protégée
  - ➡ Les algorithmes peuvent être connus (à minima par les deux interlocuteurs)
  - ➡ Déchiffrer le message doit être

    - ➡ Très (très) difficile sans clé (temps très long)
    - ➡ Très facile en utilisant la clé privée (temps très court)

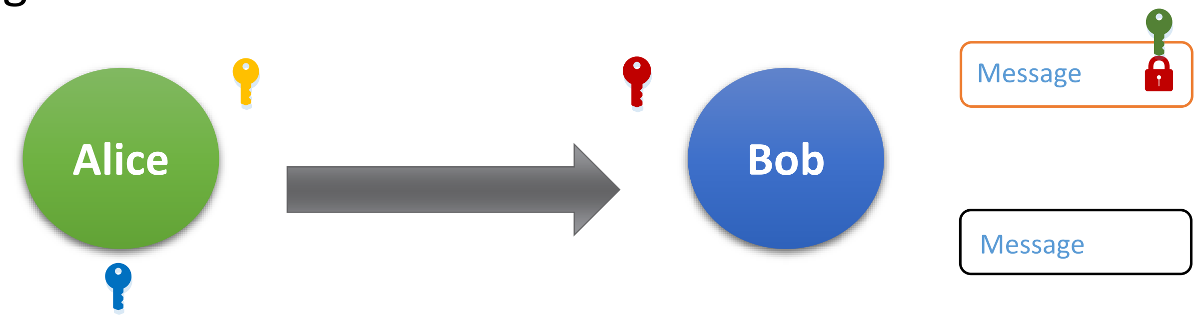
Chiffrement asymétrique

- Généralité chiffrement asymétrique

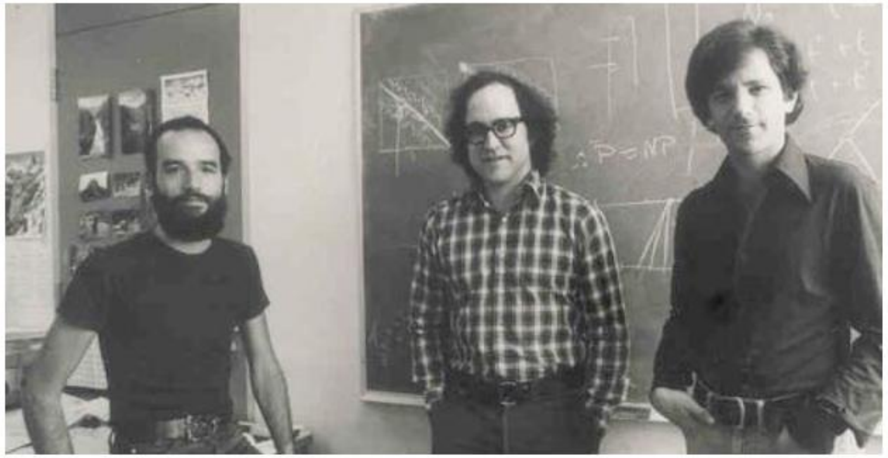
➡ Chiffrement d'un message et transfert



➡ Déchiffrement du message



- Chiffrement asymétrique RSA
  - ➡ **R**on Rivest, Adi **S**hamir et Len **A**dleman, 1978
  - ➡ Inventeur du "système à clé publique"
  - ➡ Résoudre le problème nombre de clés nécessaires par interlocuteurs
  - ➡ Ne plus avoir besoin de transmettre un "secret" (Diffie-Hellman)



Rivest, R.; A. Shamir; L. Adleman (1978). "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". Communications of the ACM 21 (2): 120–126.

Chiffrement asymétrique

- Chiffrement asymétrique RSA

➡ Concept mathématique

➡ Multiplier deux très grands entiers naturels (>100 chiffres)

Facile → $462^{126} \times 24^{262}$

➡ Factoriser les entiers en nombres premiers (Un entier naturel est produit de nombres premiers)

$$125 = 5 \times 5 \times 5 = 5^3$$

$$1\,010\,021 = 17 \times 19 \times 53 \times 59$$

Très ...très difficile → $462^{126} \times 24^{262} = ? \times ? \times \ \ldots.$

Chiffrement asymétrique

- Création des clés RSA

  - Calcul de la valeur ***n*** produits de 2 entiers premiers

    - Choix de nombres premiers ***p, q*** très grands et différents
    - Calcul d'une valeur ***n* = *p* x *q***

  - Calcul de la fonction d'Euler $\Phi(n) = n - 1$

    - Déterminer la quantité de nombres inférieurs à ***n*** et premiers avec ***n***
    - Soit $\Phi(n) = (p - 1)(q - 1)$

Chiffrement asymétrique

- Création des clés RSA

  - Création de la clé publique e
    - Choix d'une valeur de e tel que le Plus Grand Commun Diviseur avec $\Phi(n)$ soit égale à 1
      - PGCD $(e; \Phi(n))=1$
      - Reviens à choisir un entier $e < \Phi(n)$ au hasard **premier** avec $\Phi(n)$

  - Calcul de la clé privée d
    - Utilisation de la fonction d'inversion modulaire tel que $d \times e$ soit **congrue** avec $1 \bmod \Phi(n)$
    - C'est-à-dire que le reste du produit de la division euclidienne $d \times e$ par $\Phi(n)$ soit égal à 1
    - Peut –être résolu à l'aide des coefficient de Bézout et de l'algorithme d'Euclide étendue

$$d \times e \equiv 1 \bmod \Phi(n) \quad \text{ou} \quad d \times e = k\,\Phi(n) + 1$$

Chiffrement asymétrique

- Chiffrement asymétrique RSA

  ➡ Concept cryptographique

Message
Md
(Entier)

Clé publique
e et n

Clé privée
d et n

$$Mc = Md^{e} \bmod n$$

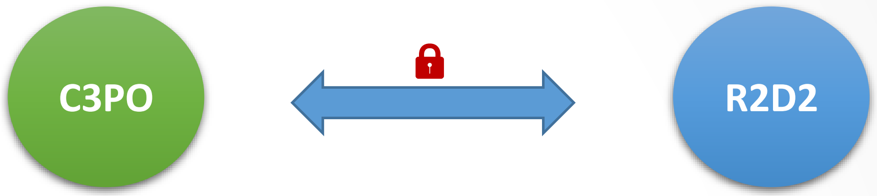
Chiffrement

Mc

$$Md = Mc^{d} \bmod n$$

Déchiffrement

EXERCICE



- Chiffrement RSA

  - C3PO veut discuter avec R2D2 en utilisant RSA

    - C3PO choisit 2 entiers premiers $\boldsymbol{p_3 = 11}$ et $\boldsymbol{q_3 = 19}$
    - R2D2 choisit 2 entiers premiers $\boldsymbol{p_2 = 5}$ et $\boldsymbol{q_2 = 11}$
      - Calculer la fonction d'Euler $\Phi_3(n_3)$ et $\Phi_2(n_2)$ pour C3PO **et** R2D2
    - C3PO choisit $\mathbf{e_3=7}$ et R2D2 choisit $\mathbf{e_2=7}$ comme clé publique
      - Vérifier que les clés publiques ($\mathbf{e_3}$ et $\mathbf{e_2}$) sont bien première avec $\Phi_3(n_3)$ et $\Phi_2(n_2)$
    - C3PO détermine $\mathbf{d_3=63}$ et R2D2 détermine $\mathbf{d_2=23}$ comme clé privée
    - C3PO veut envoyer le message $\mathbf{Md_3 = 2}$ à R2D2
      - Quel est le message chiffré $\mathbf{Mc_3}$ réceptionné par R2D2 ?
      - Quel est le message déchiffré calculé par R2D2 ?
    - R2D2 veut répondre avec le message $\mathbf{Md_2 = 4}$ à C3PO
      - Quel est le message chiffré réceptionné par C3PO ?

CORRECTION

- Chiffrement RSA

  - Calcul de la fonction d'Euler
    - $\Phi_3(n_3) = (p_3 \ - 1)(q_3 - 1) = (11 - 1)(19 - 1) = 180$
    - $\Phi_2(n_2) = (p_2 \ - 1)(q_2 - 1) = (5 - 1)(11 - 1) = 40$

  - Vérification des clés publiques
    - Calcul du PGCD(e_3 ; $\Phi_3(n_3)$) et PGCD(e_2 ; $\Phi_2(n_2)$)
      - Factorisation en nombres premiers

$$e_3 = 7 = 7^1 \times 1^1$$
$$\Phi_3(n_3) = 180 = 2^2 \times 3^3 \times 5^1 \times 1^1$$

$$e_2 = 7 = 7^1 \times 1^1$$
$$\Phi_2(n_2) = 40 = 2^3 \times 5^1 \times 1^1$$

      - Les facteurs communs sont bien égaux à 1, donc les clés publiques sont premières avec $\Phi(n)$

CORRECTION

- Chiffrement RSA
  - Calcul du message chiffré $\mathbf{Mc_3}$ envoyé à R2D2
    - Utilisation de la clé publique de R2D2 soit les valeurs e_2 et $n_2 = (p_2 \times q_2)$
      - $\mathrm{Mc}_3 = (Md_3)^{e_2} \, mod \, (n_2) = 2^7 \, mod \, (5 \times 11) = 2^7 \, mod \, (55) = 18$
  - Calcul du message déchiffré envoyé à R2D2
    - Utilisation de la clé privé de R2D2 soit les valeurs d_2 et $n_2 = (p_2 \times q_2)$
      - $M = (Mc_3)^{d_2} \, mod \, (n_2) = 18^{23} \, mod \, (55) = 2$
  - Calcul du message chiffré $\mathbf{Mc_2}$ envoyé à C3PO
    - Utilisation de la clé publique de C3PO soit les valeurs e_3 et $n_3 = (p_3 \times q_3)$
      - $\mathrm{Mc}_2 = (Md_2)^{e_3} \, mod \, (n_3) = 4^7 \, mod \, (11 \times 19) = 4^7 \, mod \, (209) = 82$

Chiffrement asymétrique

- Chiffrement asymétrique RSA

  - Tout est basé sur le nombre $\boldsymbol{n}$ $(\boldsymbol{p} \times \boldsymbol{q})$
  - La difficulté repose sur le temps pour trouver $\boldsymbol{p}$ et $\boldsymbol{q}$
  - La complexité augmente de manière exponentielle avec la tailles de $\boldsymbol{n}$
  - Méthode d'attaque :
    - La factorisation (Calcul)
      - Objectif est de trouver $\boldsymbol{n}$
        - 1978 : Méthode de Schroeppel avec 1 µs /opération

Table I.

Digits	Number of operations	Time
50	1.4×10^{10}	3.9 hours
75	9.0×10^{12}	104 days
100	2.3×10^{15}	74 years
200	1.2×10^{23}	3.8×10^{9} years
300	1.5×10^{29}	4.9×10^{15} years
500	1.3×10^{39}	4.2×10^{25} years

Chiffrement asymétrique

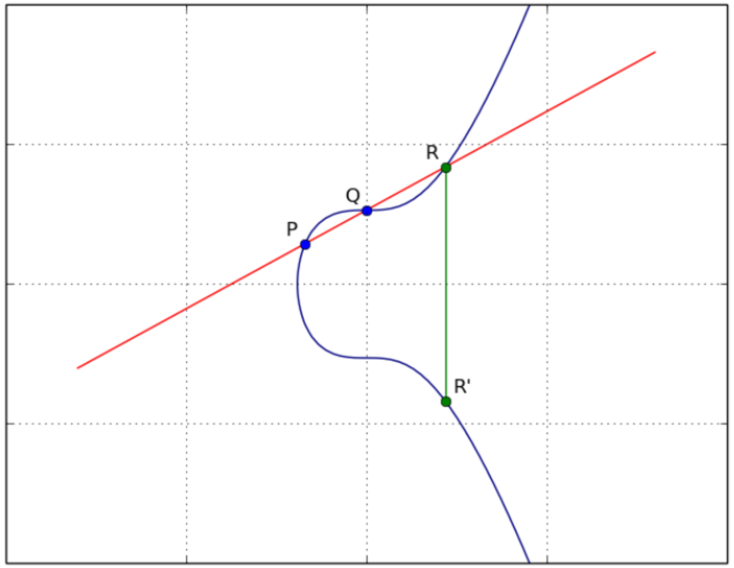
- Cryptographie à courbe elliptique (ECC)

➡ Principe général

➡ Basé sur la résolution de logarithme (≠ des entiers pour RSA)

➡ Utilise une courbe elliptique

$$exemple: y^2 = x^3 - x + 1$$



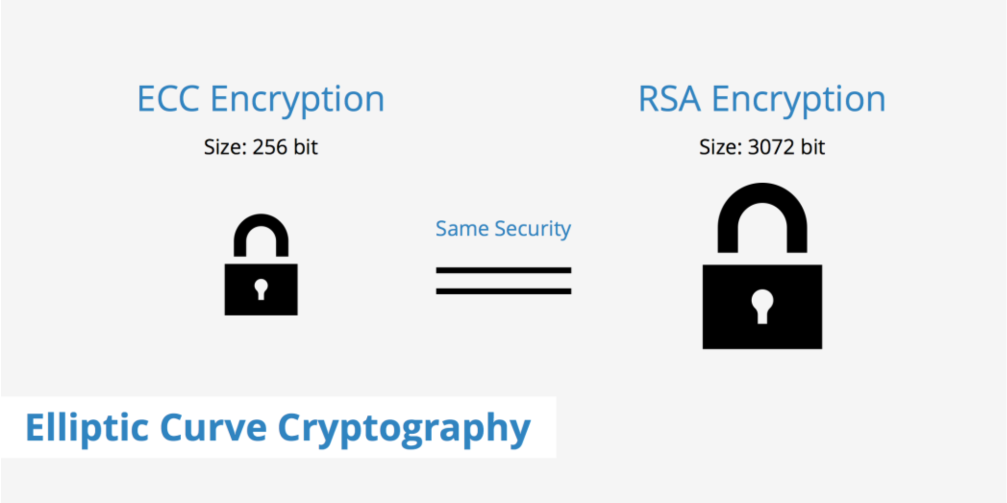
- Clé privée : un entier kn choisit au hasard
- Clé publique : $kn \times P$ (où P est un point sur la courbe)

Chiffrement asymétrique

- Cryptographie à courbe elliptique (ECC)

  - ➡ Avantages

    - ➡ Résolution elliptique plus difficile que RSA
    - ➡ Clé de taille inférieure
    - ➡ Calcul nécessite moins de puissance
    - ➡ Peut-être utilisée en chiffrement ou échange de clé (ECDH)



Chiffrement asymétrique

- Cryptographie à courbe elliptique (ECC)

  - Inconvénients

    - Théorie complexe
    - Implémentation technique beaucoup plus difficile que RSA
    - Dissymétrie de calcul entre clé publique et privée
    - Utilisation très récente (effets de bords ?)
    - Beaucoup de brevets qui verrouillent la technologie
    - Il existe probablement des solutions de contournement

Conclusion

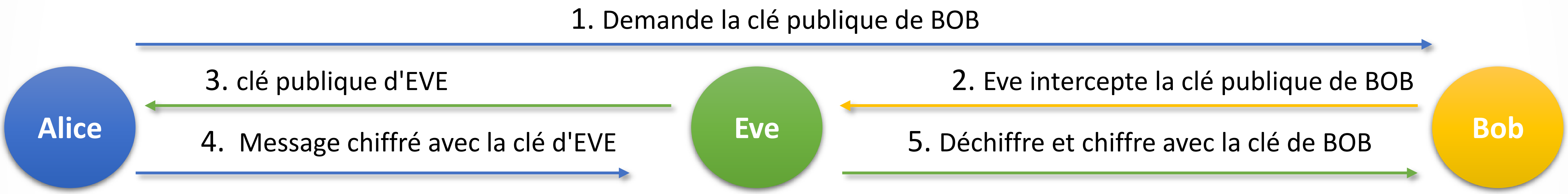
- Comparatif chiffrements

SYMETRIQUE	ASYMETRIQUE
Avantages	
➡Adapté au trafic en temps réel ➡Opérations rapides ➡Clés courtes (mini 256 Bits)	➡Facilité d'échange de clés (uniquement clés publiques) ➡Clé plus petite (Courbe elliptique)
Inconvénients	
➡Difficulté d'échange de secret partagé ➡Multiplication des clés selon le nombre d'interlocuteur	➡Lenteur des opérations (mathématique) ➡Grande taille de clés (2048 Bits)

Certificats

- Implémentation du chiffrement asymétrique

  ➡ Failles connues



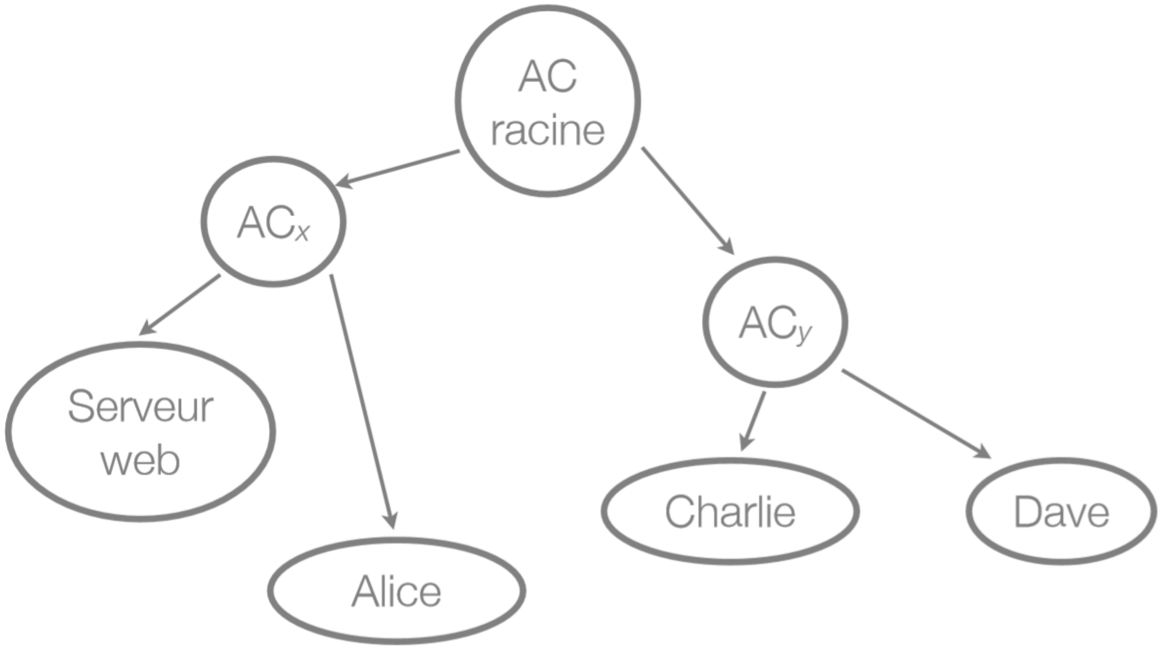
Comment Alice pourrait être certaines de l’origine du message ?

Certificats

- Certificats électroniques
  - Objectif
    - Garantir l'association entité-clé (Lien identité et clé)
    - Chiffrement
      - Utilisation de la partie publique comme clé de chiffrement
    - Signature
      - Non-répudiation
      - Authentification
  - Attribution des certificats
    - Infrastructure à clés publiques (X.509)
    - Réseaux de confiances (PGP)

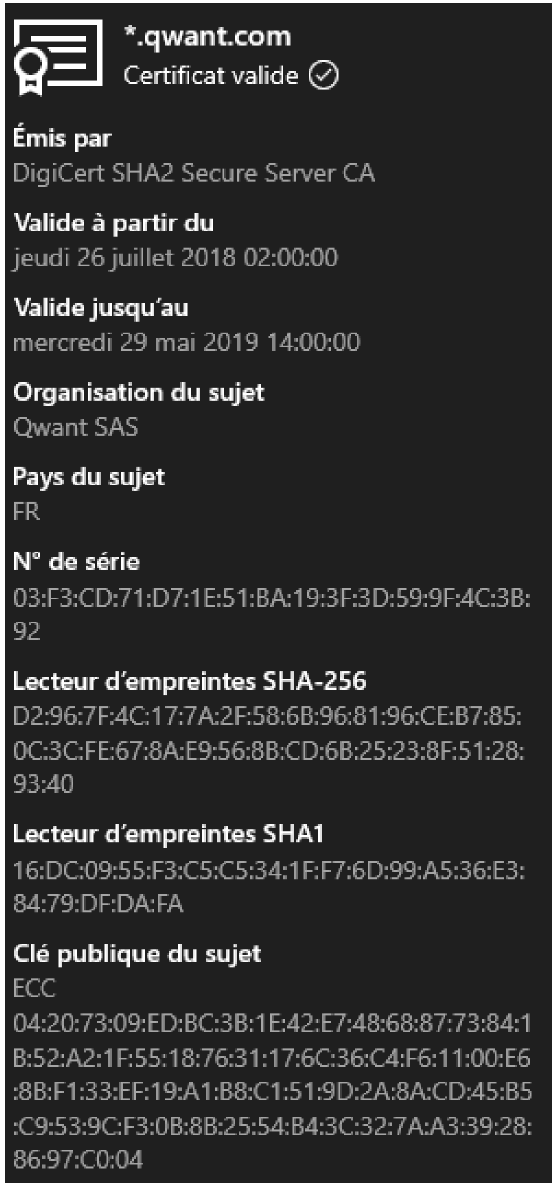
Certificats

- Infrastructure à clés publiques
  - ➡ Cryptographie à clef publique
    - ➡ Repose sur la confiance accordée à la clef publique de l'interlocuteur
  - ➡ ICP ou Infrastructure de Gestion de Clefs (IGC) ou Public Key Infrastructure (PKI)
    - ➡ Machines, procédures, humains, logiciels
    - ➡ Organisation hiérarchique
  - ➡ Objectif des Autorités de certification (AC)
    - ➡ Garantir le lien clé-entité
    - ➡ Signature des demandes de certificats (CSR)
    - ➡ Publication des procédures
    - ➡ Sécuriser les clés sous séquestre



Certificats

- Certificat X.509
  - ➡ Structure normée selon la RFC 5280
  - ➡ Champs obligatoires
    - ➡ Version
    - ➡ Numéro de série
    - ➡ Id de l’algorithme (signature)
    - ➡ Nom du signataire du certificat
    - ➡ Date de validité
    - ➡ Sujet (Distinguished Name ou DN)
    - ➡ Infos sur la clef publique
    - ➡ Algorithme utilisé pour la signature
    - ➡ La clef publique du sujet



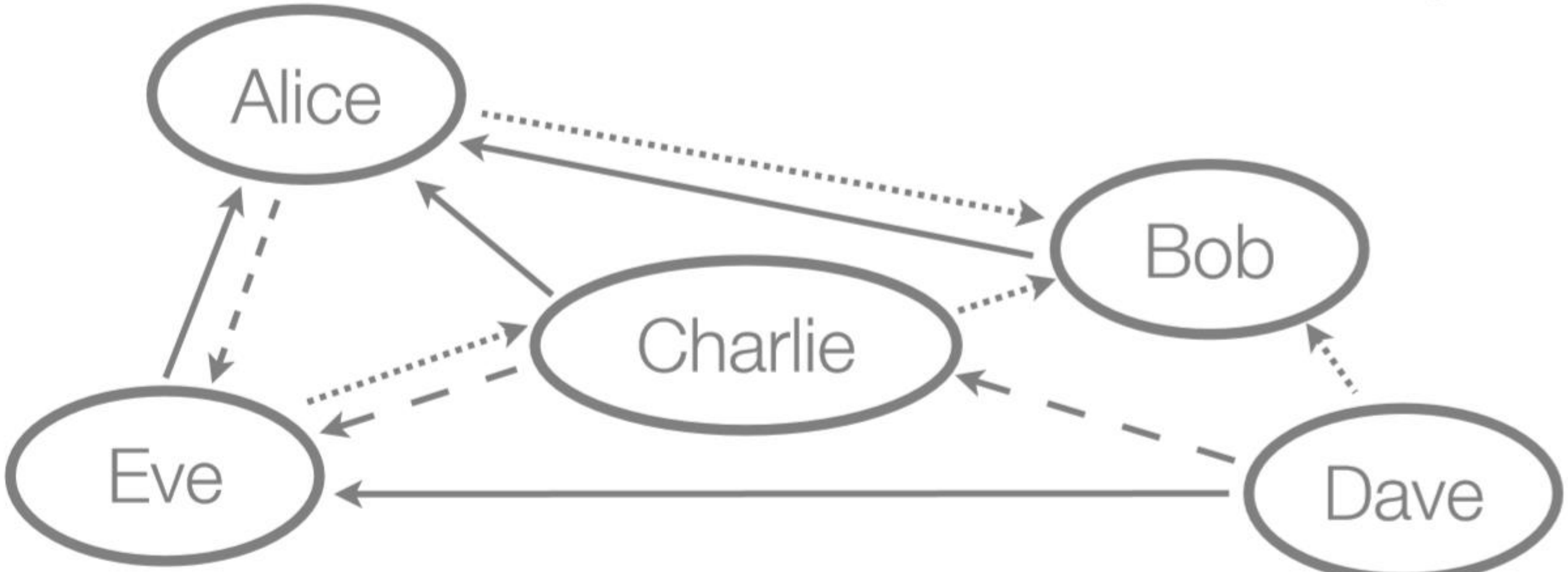
Certificats

- Pretty Good Privacy (PGP)

  - Date de 1991
  - Logiciel propriétaire de Symantec depuis 2010
  - Remplacé par GnuPG (open source de 2007)
    - RFC4880
  - Permet de garantir
    - La non-répudiation
      - Création d'un condensat à l'aide d'un algorithme de HASH (SHA-1)
    - Confidentialité
      - Chiffrement asymétrique (RSA, DSS, Diffie-Hellman..)
      - Chiffrement symétrique (3DES, IDEA..)
    - Compression
      - Utilisation de la fonction ZIP

Certificats

- GnuPGP
  - Serveur en ligne permettant de déposer sa clé publique (http://pgp.mit.edu/)
  - N'est pas basé sur une infrastructure hiérarchique
  - Basé sur une toile de confiance (Web of trust)
    - Niveau de confiance
      - Complète
      - Marginale
      - Non sûre
      - Inconnue



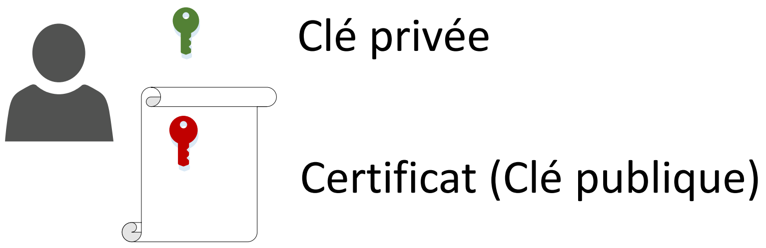
Certificats

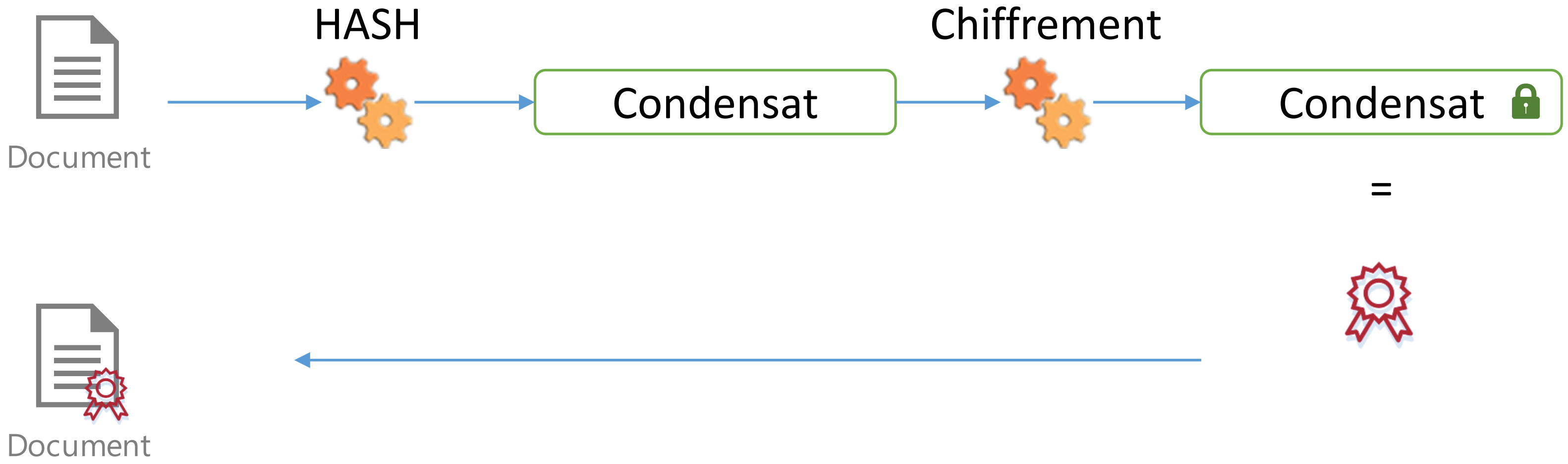
- Signature électronique
  - Objectif
    - S’assurer de la non-modification d’une donnée (Intégrité)
    - S'assurer de l'identité de son auteur
  - N'assure pas
    - La confidentialité des données (sauf si en complément du chiffrement)
  - Conditions
    - Disposer d'un certificat électronique (X509)
    - Disposer d'une clé privée
    - Disposer de la clé publique associée

Certificats

- Signature électronique

➡ Principe de base

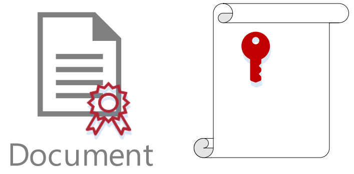


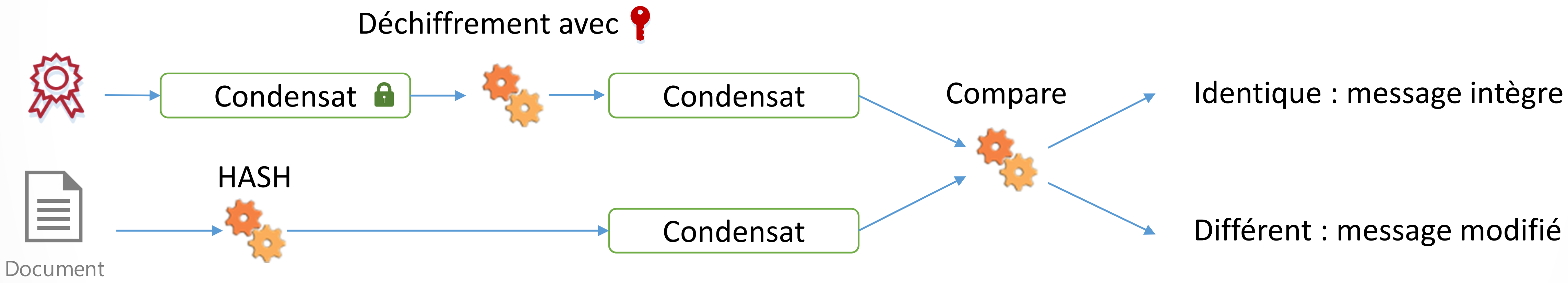


Certificats

- Signature électronique

  ➡ Vérification de la signature





Certificats

- Utilisation de la signature électronique
  - Document
  - E-mail
  - Transaction
  - Certificat
    - Certificat X509

Clé privée de l'autorité de certification CA

Application RSA : protocoleTLS

- Historique TLS (Transport Layer Security)

  - TLS v1.0 (1999)
    - Reprise du protocole SSL v3 par IETF
  - TLS v1.2 (2008)
    - Refonte totale du standard
    - Uniformisation de RFC
    - Ajout de HMAC et SHA256
  - TLS v1.3 (2018)
    - Suppression des rétrocompatibilités avec MD5, RC4, DSA, SHA-224 et autres
    - ECC : Elliptic Curve Cryptography
    - Uniquement les navigateurs récents sont compatibles

Application RSA : protocoleTLS

- Suite cryptographique

  - Description des algorithmes
    - Authentification (côté serveur) (AU)
    - Échange de clé (Kx)
    - Chiffrement des données (ENC)
    - Intégrité des données (MAC)

TLS_DHE_RSA_WITH_AES_128_CBC_SHA

DHE : échange de clé Diffie-Hellmann
RSA : signature des échanges
AES_128_CBC : chiffrement des données
SHA : contrôle de l'intégrité des données

Application RSA : protocoleTLS

- Suite cryptographique

  ➡ Exemple de suite du logiciel "openssl"

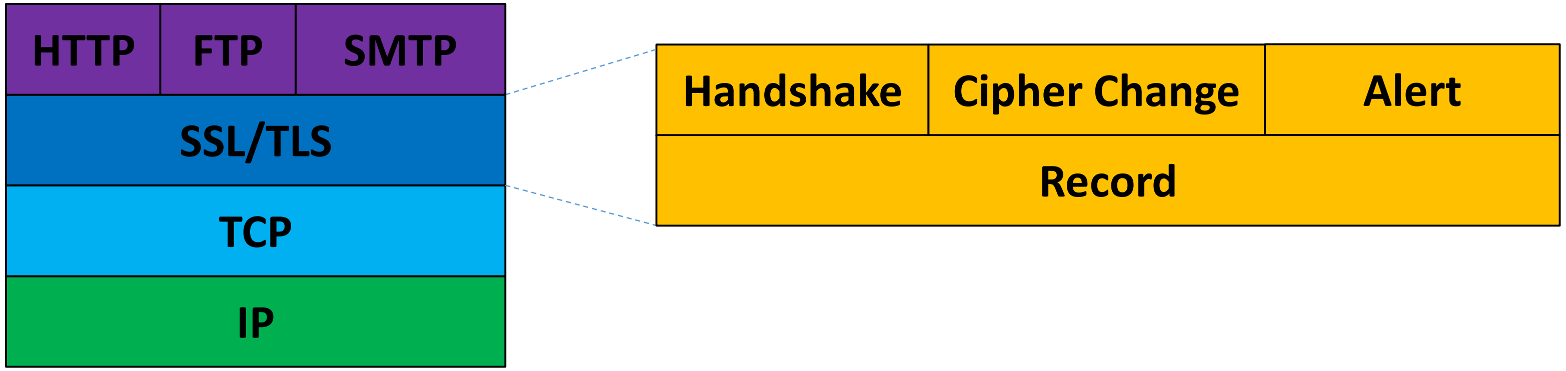
```
ECDH-ECDSA-AES256-SHA   SSLv3 Kx=ECDH/ECDSA Au=ECDH Enc=AES(256)  Mac=SHA1
AES256-GCM-SHA384       TLSv1.2 Kx=RSA      Au=RSA  Enc=AESGCM(256) Mac=AEAD
AES256-SHA256           TLSv1.2 Kx=RSA      Au=RSA  Enc=AES(256)  Mac=SHA256
AES256-SHA              SSLv3 Kx=RSA      Au=RSA  Enc=AES(256)  Mac=SHA1
```

  ➡ Exemple de suite d'un site web

TLS_RSA_WITH_AES_128_CBC_SHA256 (0x3c)	
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 (0x67)	DH 2048 bits FS
TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c)	
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x9e)	DH 2048 bits FS
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027)	ECDH secp256r1 (eq. 3072 bits RSA) FS

Application RSA : protocoleTLS

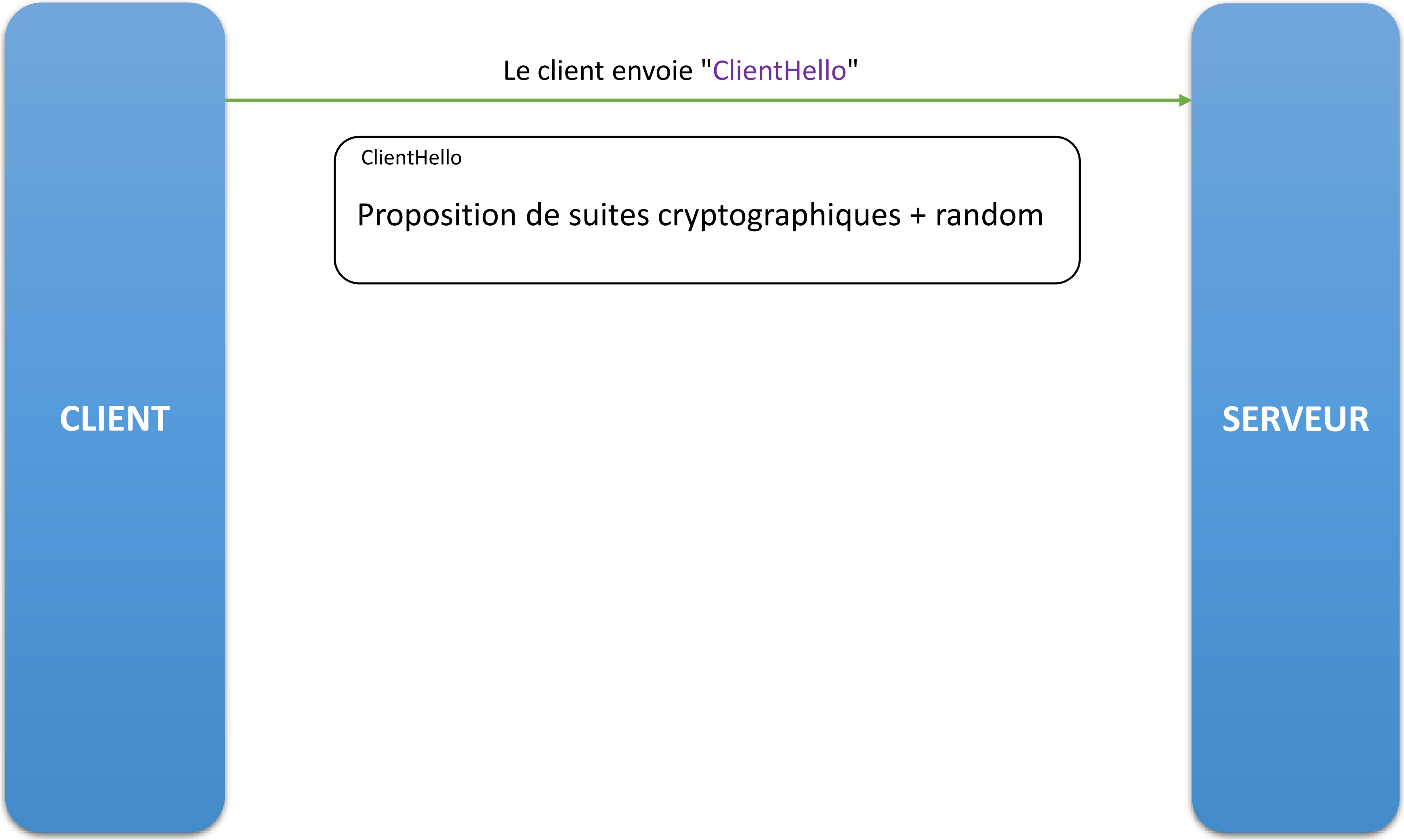
- Positionnement dans la pile TCP/IP



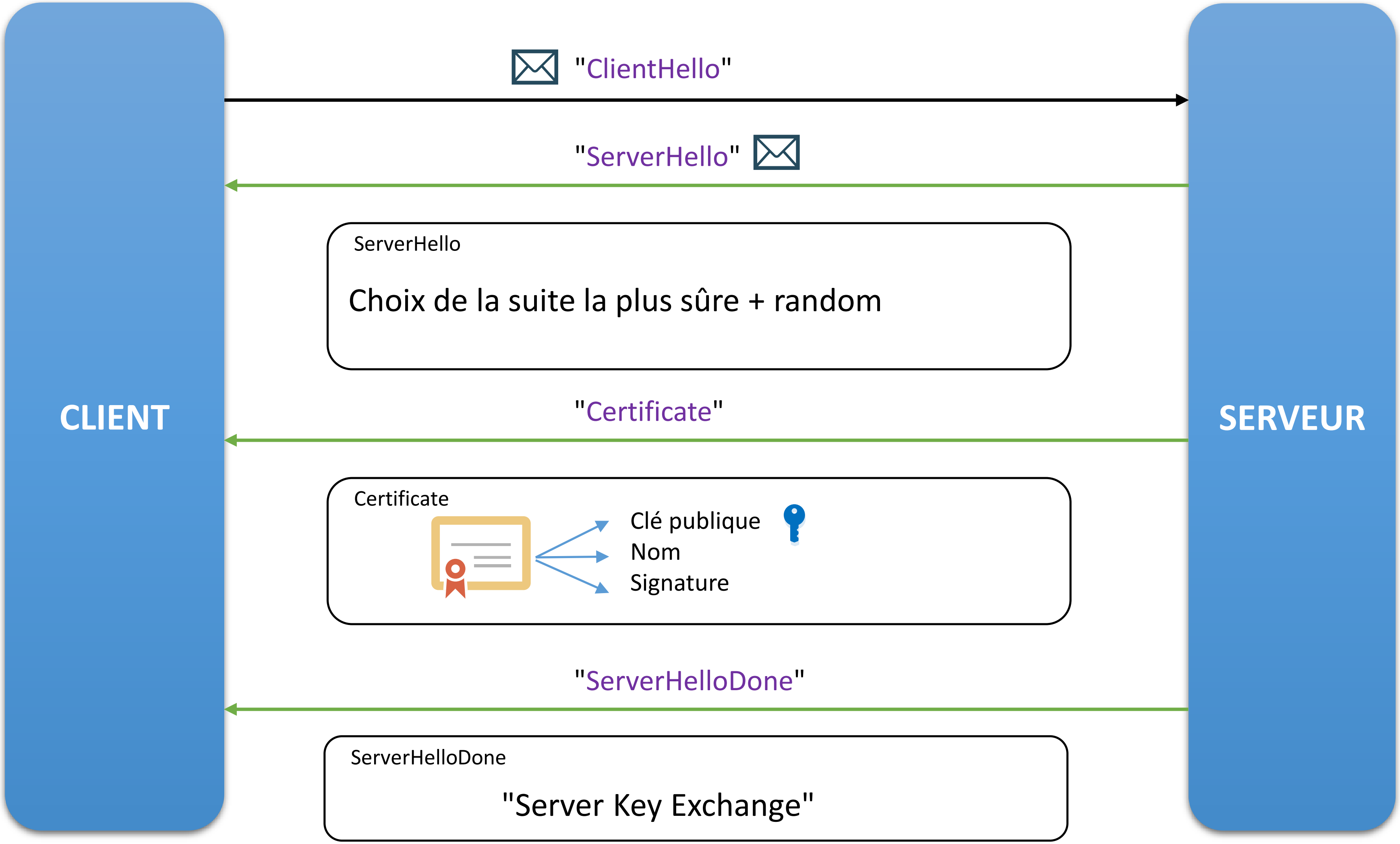
Application RSA : protocoleTLS

- Principe de fonctionnement (4 sous-protocoles)
  - ➡ TLS Handshake Protocol
    - ➡ Authentification
    - ➡ Échange de clés
  - ➡ TLS Record Protocol
    - ➡ Séparation et réassemblage en blocs
    - ➡ Compression
    - ➡ Chiffrements et contrôle d'intégrité
  - ➡ TLS Alarm Protocol
    - ➡ Messages d'erreur de type "fatale" ou "warning"
  - ➡ TLS Cipher Change Protocol
    - ➡ Négociation de la suite cryptographique

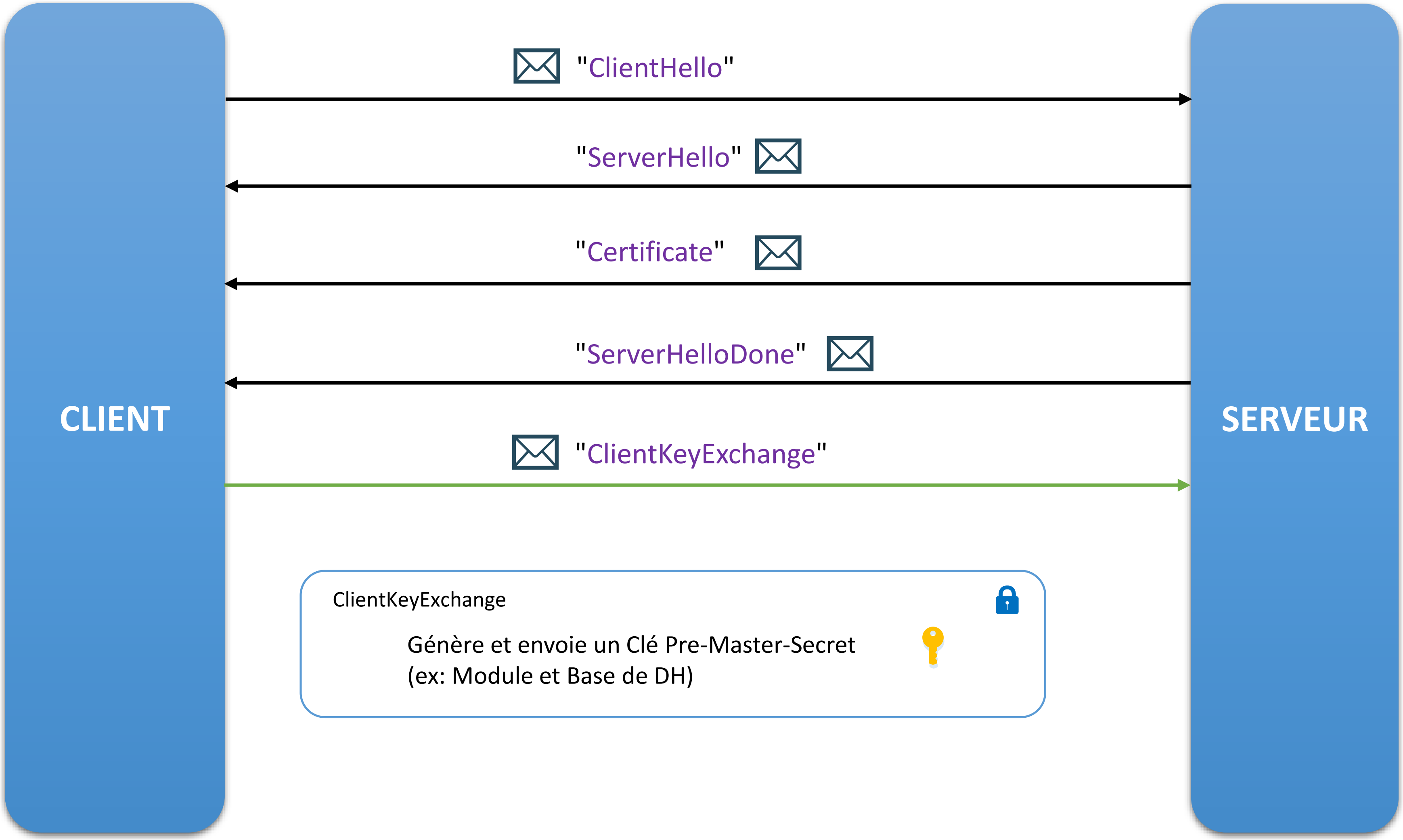
Échanges TLS 1.2 (Handshake Protocol)



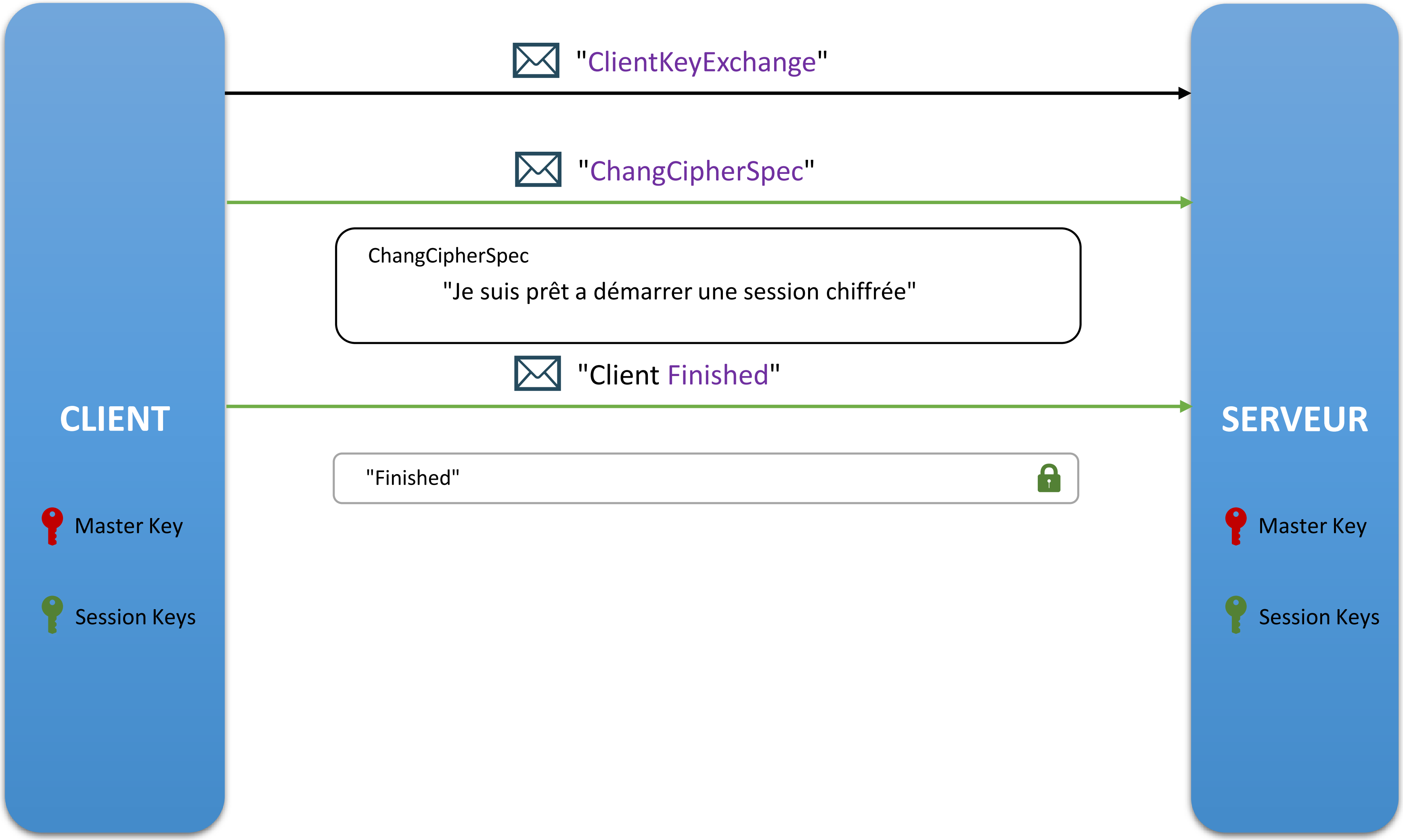
Échanges TLS 1.2



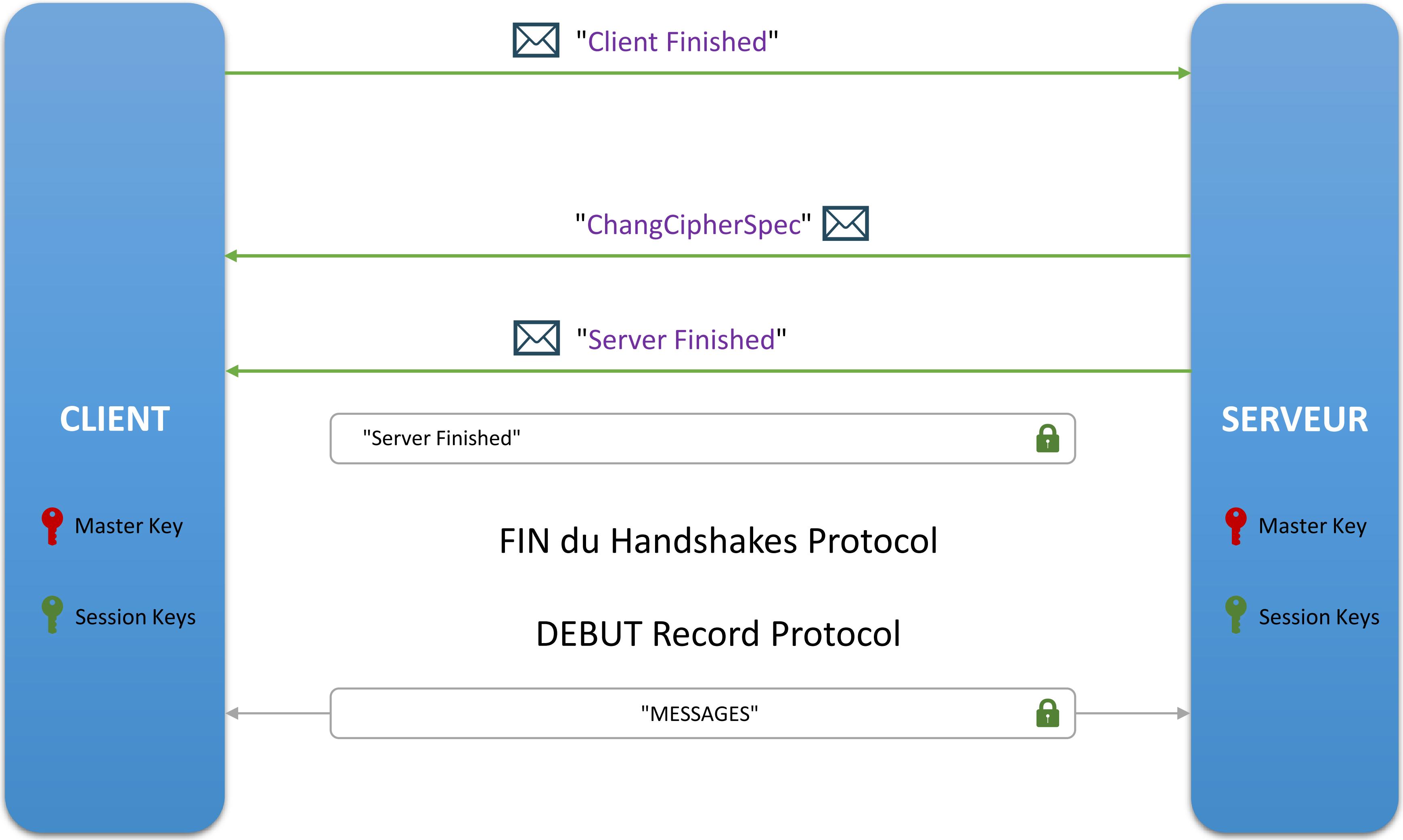
Échanges TLS 1.2



Échanges TLS 1.2



Échanges TLS 1.2



Échanges TLS 1.2

- Génération des clés
  - ➡ Après échange de la clé Pre-Master-Key
  - ➡ Calcul de la Master-Key (48 Octets soit 384 Bits) client et serveur
    - ➡ Rajout de plusieurs aléas pour rendre la clé unique
    - ➡ PFR (fonction Pseudo-aléatoire) dépendant du chiffrement choisi

Master Key

Master_Key = PFR(Pre_master_key, ClientHello.random + ServerHello.random)[0..47]

Échanges TLS 1.2

- Génération des clés de sessions

  - Dérivation des clés de sessions à partir de la Master-Key
    - Clé symétrique pour le chiffrement des données
    - Clé pour authentification MAC (Message Authentication Code)
    - Vecteur d'initialisation si besoin selon mode d'opération

CLIENT	SERVEUR
Client_write_Mac_Key	Server_write_Mac_Key
Client_write_key	Server_write_key
Client_write_IV	Server_write_IV

Échanges TLS 1.2

- Protocole Record

  - Encapsulation
    - Préparation des données
    - Processus de segmentation
    - Processus de signature

  - Confidentialité
    - Chiffrement du message (Client_write_key)

  - Intégrité
    - Vérification de la validité du message
    - non-répudiation par signature (Client_write_Mac)

Échanges TLS 1.2

- Protocole Record

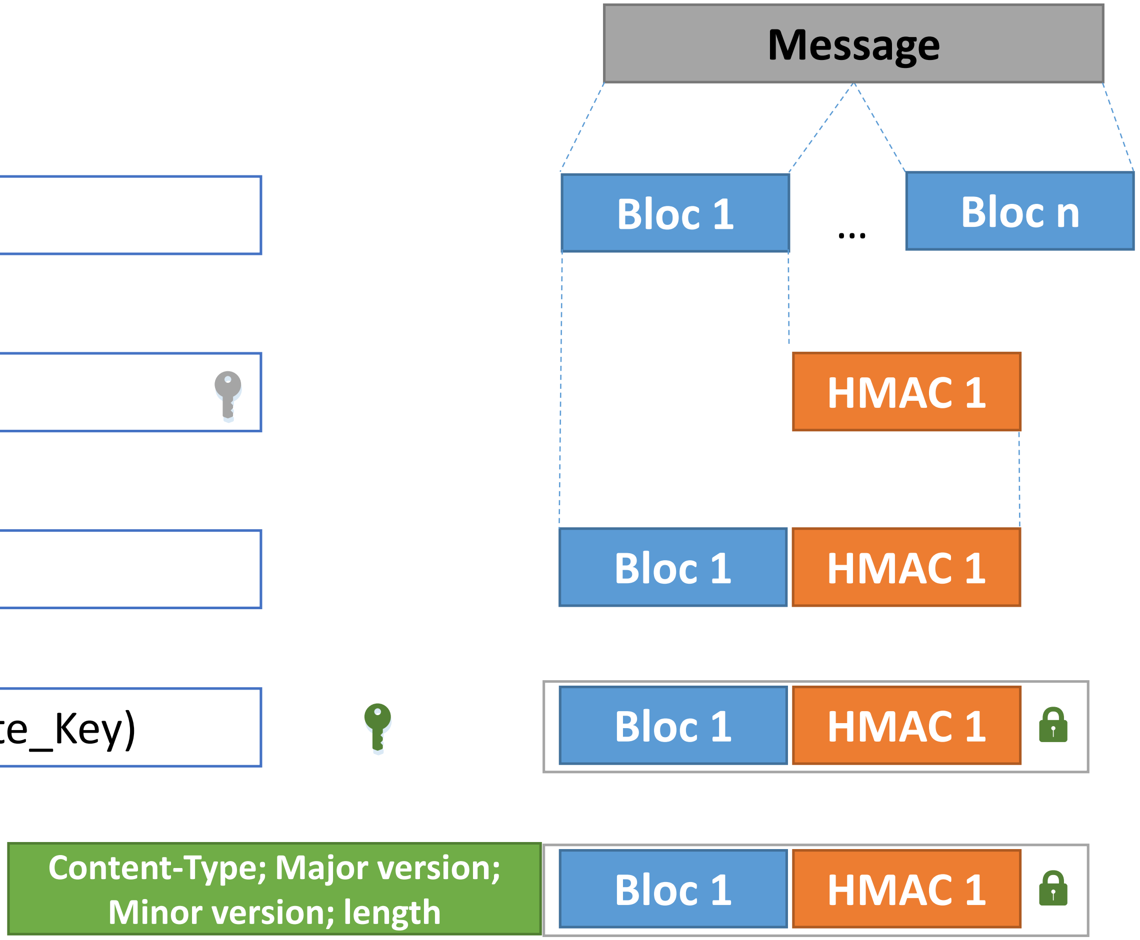
Fragmentation (<16 Ko)

Génération HMAC (Mac_Key)

Concaténation (||)

Chiffrement symétrique (Write_Key)

Ajout "Record Header"



Échanges TLS 1.2

- Signature HMAC

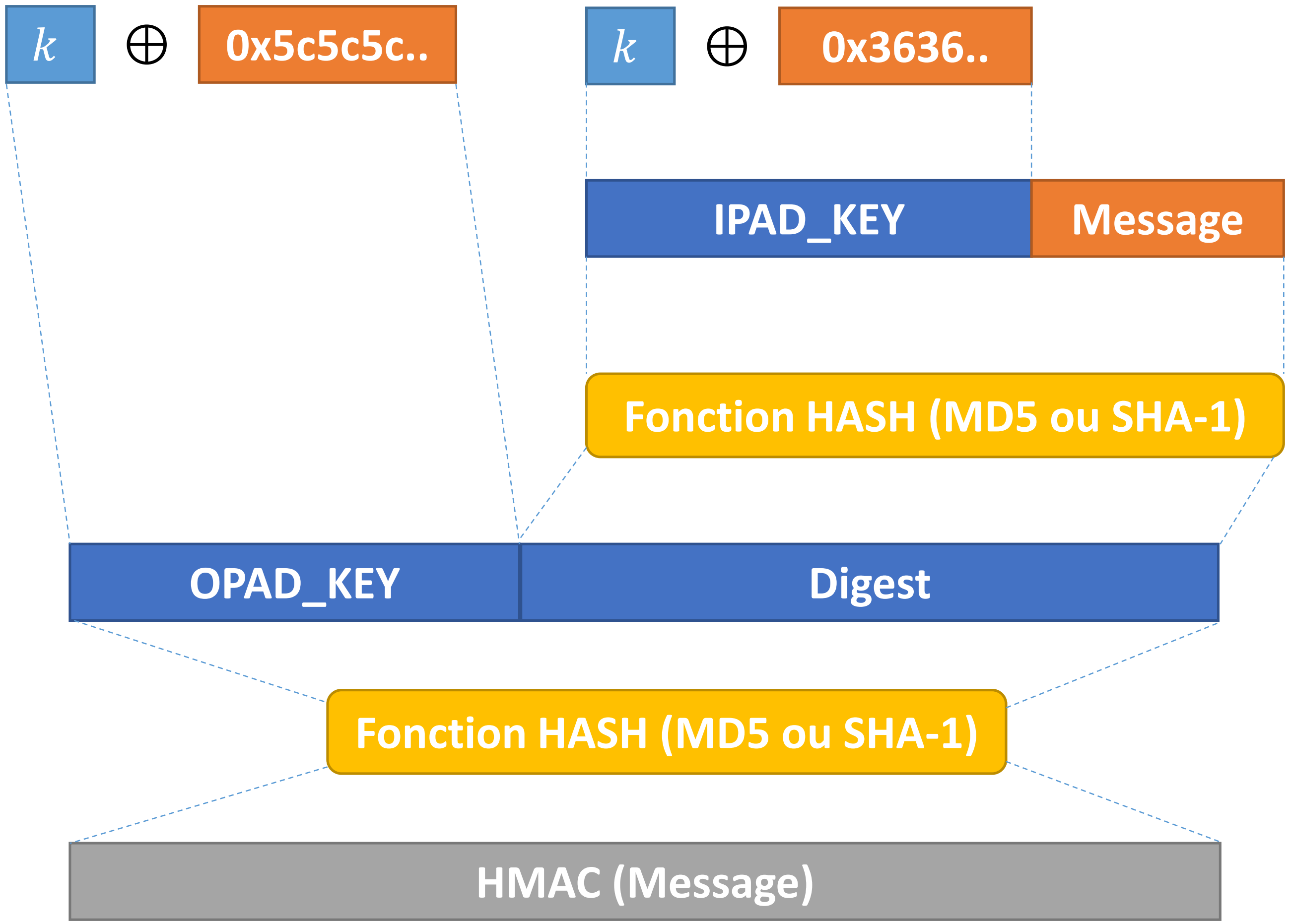
  - Utilise une fonction de hash cumulative
  - Génère une signature 512 Bits

$$\boxed{HMAC(m) = h\left(\left(k \oplus opad\right) \mid\mid h\left(\left(k \oplus ipad\right) \mid\mid m\right)\right)}$$

  - h : fonction de Hash MD5 ou SHA-1
  - k : Clé Client_write_Mac_Key et Server_Write_Mac_Key
  - $opad$: Padding 0x36
  - $ipad$: Padding 0x5c
  - m : message

Échanges TLS 1.2

- Signature HMAC



Échanges TLS 1.2

- Réception du message par le serveur

➡ Déchiffrement du message

Déchiffrement (Server_Write_Key)

Bloc 1 | HMAC 1

Contrôle de l'intégrité et non-répudiation

Bloc 1 | HMAC 1

➡ Contrôle par le serveur avec k=Server_Write_Mac_Key

Bloc 1

HMAC =? HMAC 1 ➡ True

➡ Le message n'est modifié

➡ Le message provient bien de "Client"

Vulnérabilité SSL/TLS

- Vulnérabilités depuis 2011

  - 2011 : BEAST (IV prédictif dans le mode CBC)
  - 2012 : CRIME (utilisation de la compression comme canal auxiliaire)
  - 2013 : Lucky 13 (Faille TLS Record - utilisation oracle de padding CBC)
  - 2014 : goto fail Apple (Faux certificats – Attaque M-in-M)
  - 2014 : goto fail GnuTLS (Faux certificats – Attaque M-in M)
  - 2014 : Heartbleed (débordement de tampon en lecture)
  - 2014 : Triple Handshake (Mauvais implémentation Osx et iOS)
  - 2014 : EarlyCCS (Erreur de contrôle temporel dans le Change_Cipher)
  - 2014 : ShellShock (vulnérabilité BASH -> prise de contrôle apache)
  - 2014 : POODLE (exploitation d'un oracle de padding CBC avec SSLv3)
  - 2015 : SMACK/FREAK (automates d'état déficients)
  - 2015 : LogJam (Ciphers Obsolètes + mauvaise négociation DH)
  - 2016 : DROWN (Attaque via SSL V2)

FIN