Introduction à la cryptographie

Dr. Y. Challal

Définitions

- **Le mot « Cryptographie » est composé des mots grecques :**
  - CRYPTO = caché
  - GRAPHY = écrire

- **C'est l'art de l'écriture secrète.**

- **Science permettant de préserver la confidentialité des échanges.**

- **Cryptanalyse : l'art de décrypter des messages chiffrés.**

Objectifs

- **Garantir la confidentialité**
- **Vérifier l'intégrité des données**
- **Gérer l'authentification**
- **Assurer la non-répudiation**

Confidentialité et chiffrement

- **La confidentialité est la propriété qui assure que l'information est rendu inintelligible aux individus, entités, et processus non autorisés.**

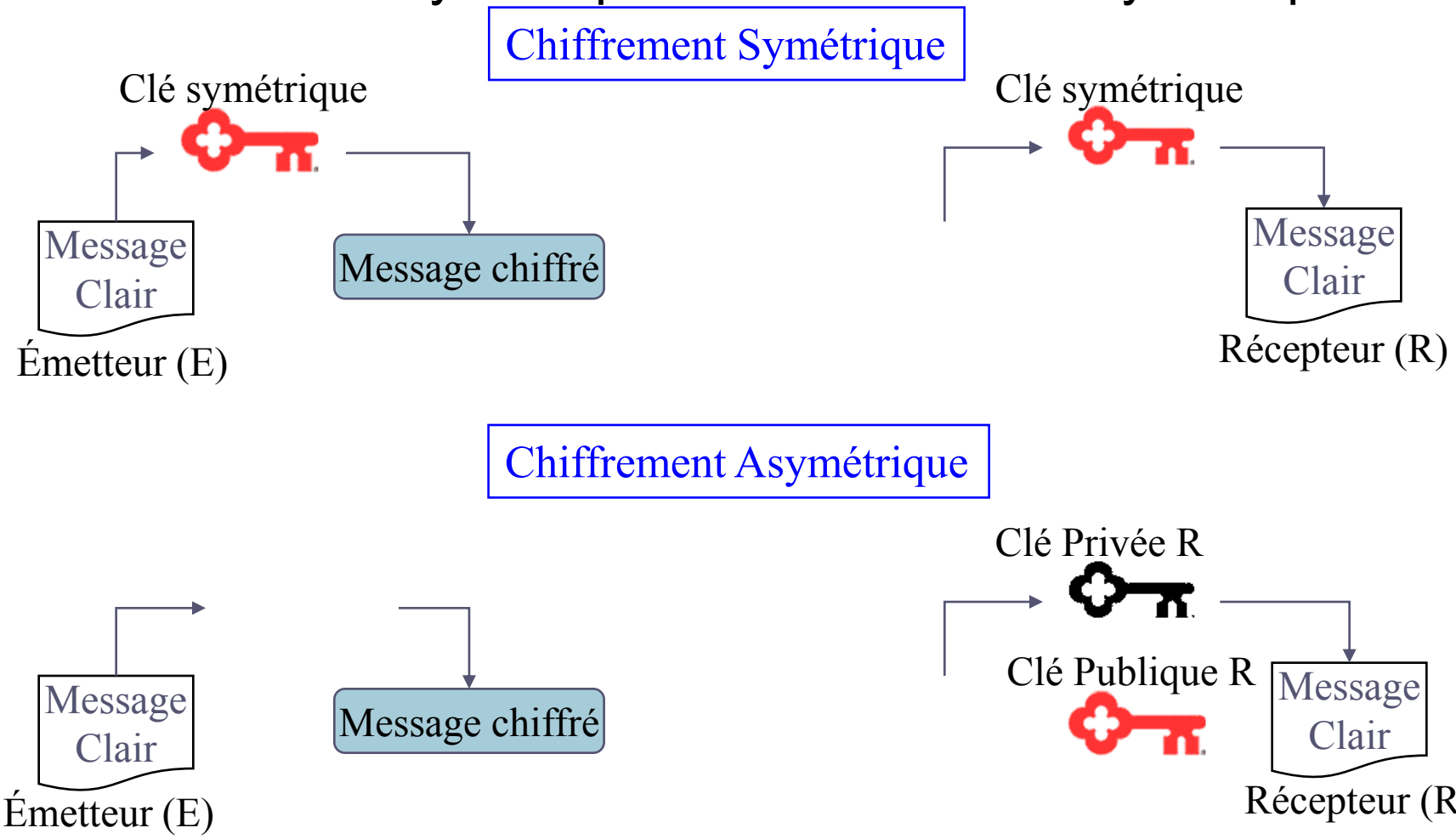
- **Le chiffrement est une transformation cryptographique qui transforme un message clair en un message inintelligible (dit message chiffré), afin de cacher la signification du message original aux tierces entités non autorisées à l'utiliser ou le lire. Le déchiffrement est l'opération qui permet de restaurer le message original à partir du message chiffré.**

Clé de chiffrement

- **Dans la cryptographie moderne, l'habilité de maintenir un message chiffré secret, repose non pas sur l'algorithme de chiffrement (qui est largement connu), mais sur une information secrète dite CLE qui doit être utilisée avec l'algorithme pour produire le message chiffré.**

- **Selon que la clé utilisée pour le chiffrement et le déchiffrement est la même ou pas, on parle de système cryptographique symétrique ou asymétrique.**

Chiffrement symétrique vs. Chiffrement asymétrique



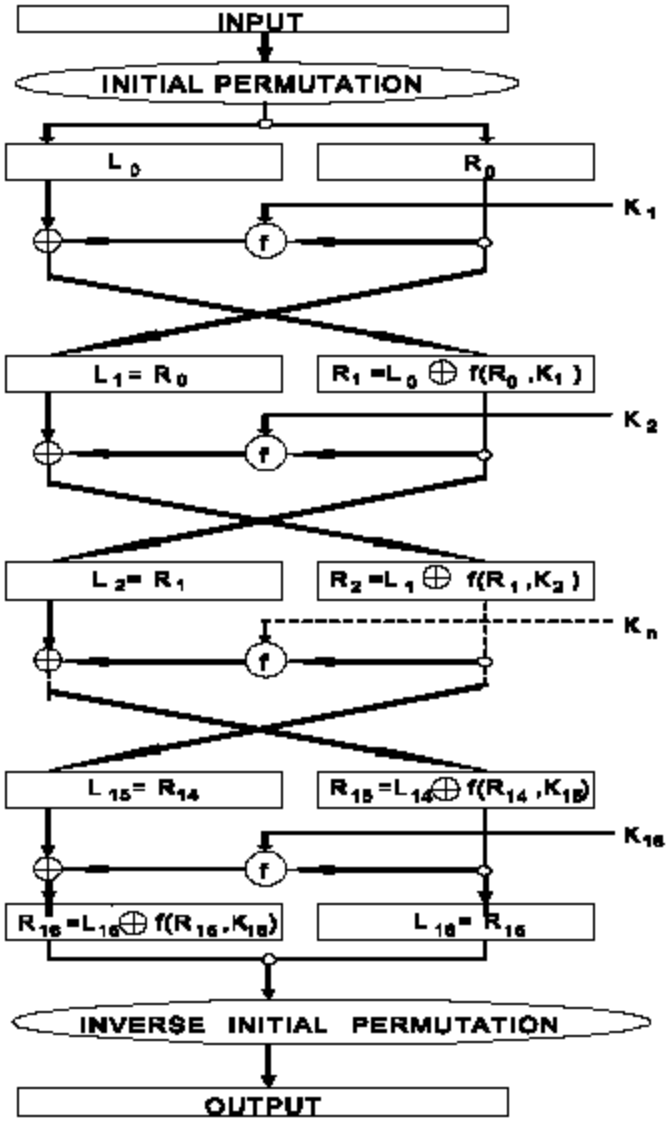
Algorithmes de chiffrement symétrique

- **Chiffrement par bloc : division du texte clair en blocs fixe, puis chiffrement bloc par bloc**
  - DES: IBM, Standard NIST 1976
  - 3DES: W. Diffie, M. Hellman, W. Tuchmann 1978.
  - IDEA: Xuejia Lai et James Massey en 1992
  - Blowfish: Bruce Schneier en 1993
  - AES (Rijndael): Joan Daemen et Vincent Rijmen 2000
- **Chiffrement par flux : le bloc a une dimension unitaire (1 bit, 1 octet, …), ou une taille relativement petite.**
  - RC4: Ron Rivest 1987
  - SEAL: Don Coppersmith et Phillip Rogaway pour IBM 1993.

DES

- **Blocs 64 bits**
- **Clé 56 bits : suffisante à l'époque**
- **Raisonnablement sûr**
- **Définis officiellement dans FIPS46-3**
- **Permutation initiale + calcul médian en fonction de la clé + permutation finale.**

DES

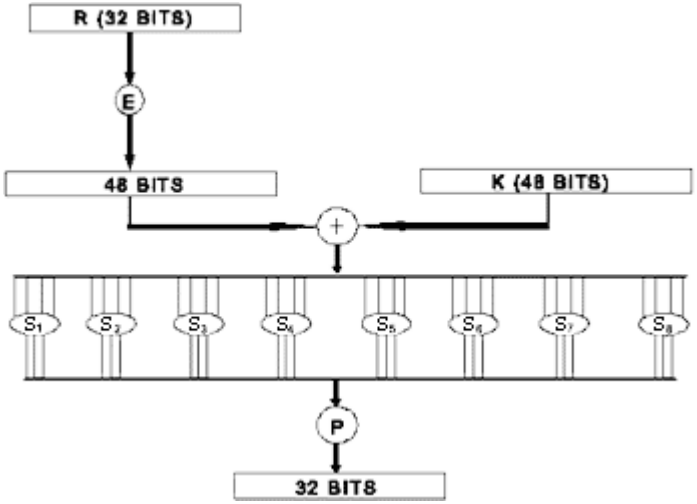


Permutation initiale

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

DES

f



E

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

P

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

S_1

Ligne No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Sécurité de DES

- **RSA Security a lancé le DES Challenge**
  - DES Challenge I 1997: DESCHALL en 96 j
  - DES Challenge II-1 1998: Distributed.net en 41j
  - DES Challenge II-2 1998: EFF Deep Crack 56h
  - DES Challenge III 1999: Deep Crack et Distributed.net 22h15
  - En 2000 AES deviens le standard à la place de DES

Chiffrement symétrique: Modes d'opération

- **4 modes sont définis dans FIPS 81 (1980)**
  - **Electronic Code Book** (ECB),
  - **Cipher Block Chaining** (CBC),
  - **Cipher FeedBack** (CFB) et
  - **Output FeedBack** (OFB).

- **Notation:**
  - T[n]: n-ième bloc du texte clair
  - C[n]: n-ième bloc du texte chiffré
  - E(m): fonction de chiffrement
  - D(m): fonction de déchiffrement
  - IV : Initialization Vector
  - ^ : XOR

Electronic Code Book (ECB)

- **Chiffrement :C[n] = E(T[n])**
- **Déchiffrement :T[n] = D(C[n])**
- **Le même texte clair et clé de chiffrement donnent le même texte chiffré.**

CBC : Cipher Block Chaining

- **Chiffrement :**
  - C[0] = E(T[0] ^ IV)
  - C[n] = E(T[n] ^ C[n-1]) , si (n > 0)
- **Déchiffrement :**
  - T[0] = D(C[n]) ^ IV
  - T[n] = D(C[n]) ^ C[n-1] , si (n > 0)

- **IV est envoyé en clair avec le message chiffré**

CFB : Cipher Feedback

- **I[n]: bloc temporaire**
- **Chiffrement :**
  - I[0] = VI
  - I[n] = C[n-1] , si (n > 0)
  - C[n] = T[n] ^ E(I[n])
- **Déchiffrement :**
  - I[0] = VI
  - I[n] = C[n-1], si (n > 0)
  - T[n] = C[n] ^ E(I[n])

- **Offre une sécurité plus élevée**

OFB : Output Feedback

- **I[n]=nième bloc temporaire**
- **R[n]=nième bloc temporaire second**

- **Chiffrement :**
  - I[0] = VI
  - I[n] = R[n-1] , si (n > 0)
  - R[n] = E(I[n])
  - C[n] = T[n] ^ R[n]
- **Déchiffrement :**
  - I[0] = VI
  - I[n] = R[n-1] , si (n > 0)
  - R[n] = E(I[n])
  - T[n] = C[n] ^ R[n]

Algorithmes de chiffrement asymétrique

- **Implique une paire de clé (SK,PK):**
  - Une clé publique (PK), publiée dans les annuaires,
  - Une clé privé (SK) maintenue secrète chez son propriétaire

- **Un message chiffré avec la clé publique PK, ne peut être déchiffré qu'avec la clé secrète SK.**

- **RSA: Rivest, Shamir et Adleman 1978**

RSA

- **Fondé sur la difficulté de factoriser des grands nombres qui sont le produit de deux grands nombres premiers.**

- **Multiplier deux grands nombres premiers est une fonction à sens unique:**
  - Il est facile de multiplier deux nombres pour obtenir un produit, mais difficile de factoriser ce produit et de retrouver les deux grands nombres premiers.

RSA

- **Initialisation**
  - Choisir deux nombres premiers, p et q, les deux étant plus grands que 10^{100}.
  - Calculer n = p · q (n est le *modulus*)
  - Choisir e aléatoire tel que e et ((p - 1) · (q - 1)) n'aient aucun facteur commun excepté 1
  - Trouver d tel que : ed = 1 mod((p - 1)(q - 1)).
  - Clé publique : (n,e).
    Clé privée : (n,d) ou (p,q,d) si on désire garder p et q.
- **Chiffrement/Déchiffrement**
  - L'expéditeur crée le texte chiffré c à partir du message m :
    $c = m^e \bmod(n)$, où (n,e) est la clé publique du destinataire
  - Le destinataire reçoit c et effectue le déchiffrement :
    $m = c^d \bmod(n)$, où (n,d) est la clé privée du destinataire.

Sécurité de RSA

- **Clé publique (e,n) connue**
- **Pour déchiffrer un message m, il faut connaître d**
- **ed=1 mod(p-1)(q-1)**
- **Pour calculer d il faut donc connaître p et q**
- **n=pq**
- **Il faut factoriser n en ses facteurs premiers p et q**
- **Or personne n'a pus le faire en un temps raisonnable.**

Échange de clé Diffie-Hellman

- **Diffie et Hellman 1976**

- **Objectif:**
  - Deux entités voudraient se mettre d'accord sur un secret afin de s'échanger des messages confidentiels,

- **Principe:**
  - Fondé sur la difficulté du calcul du logarithme discret

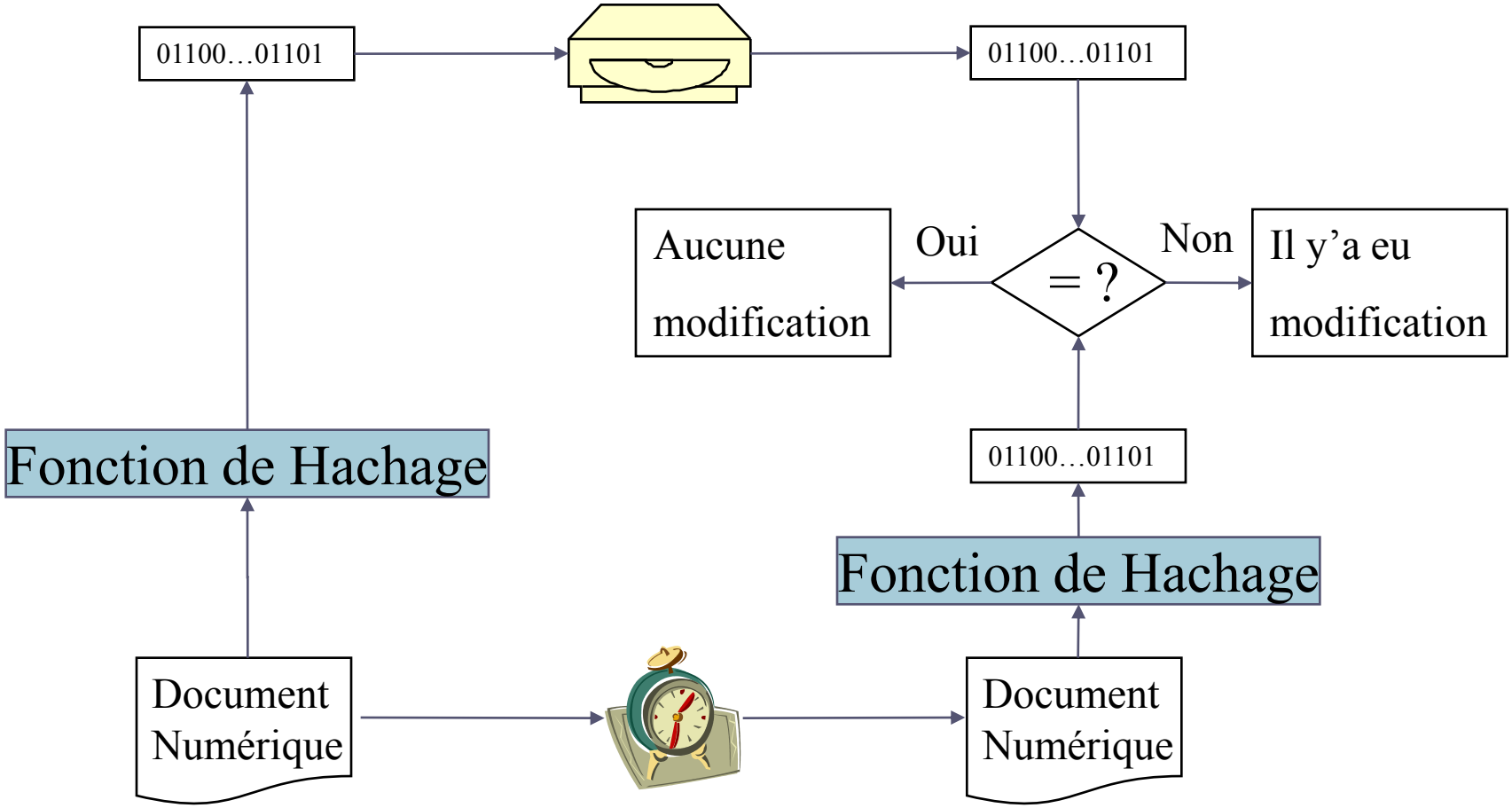
Échange de clé Diffie-Hellman

Alice				Bob		
Sec		Calc		Calc		Sec
	p, g				**p, g**	
a						**b**
		$\mathbf{g}^{\mathbf{a}} \bmod p$	$\longrightarrow$		…	
	…		$\longleftarrow$	$\mathbf{g}^{\mathbf{b}} \bmod p$		
	$(\mathbf{g}^{\mathbf{b}} \bmod p)^{\mathbf{a}} \bmod p$		=		$(\mathbf{g}^{\mathbf{a}} \bmod p)^{\mathbf{b}} \bmod p$	

Intégrité et fonction de hachage

- **L'intégrité : permet de vérifier qu'une données n'a pas été modifiée par une entité tierce (accidentellement ou intentionnellement).**
- **Une fonction de hachage est typiquement utilisée pour vérifier l'intégrité de données.**
- **« A hash function is a computationally efficient function mapping binary strings of arbitrary length to binary strings of some fixed length, called hash values » [Menezes et al.]**
- **Cryptographic hash functions have the following properties**
  - Given m, it is easy to compute h(m)
  - Given h, it is hard to compute m such that h(m)=h
  - Given m, it is hard to find another message, m', such that h(m)=h(m').

Intégrité et fonction de hachage



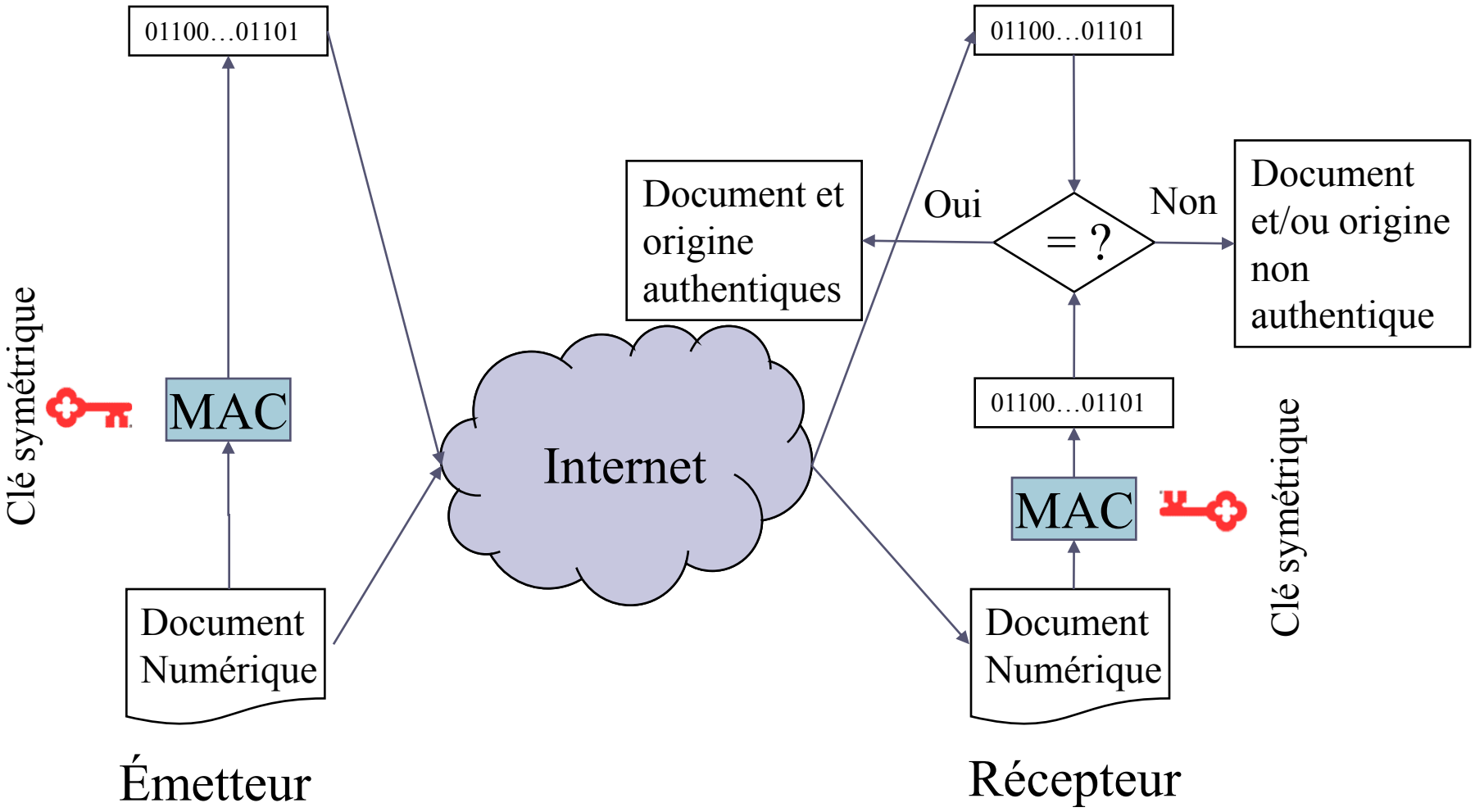
Exemples de fonctions de hachage

- **MD2 (Message Digest 2) :**
  - Opère sur des bloques de 16 octets, manipule des mots de 8 bits
  - Output 128 bits.
- **MD4 (Message Digest 4) :**
  - Manipule des mots de 32 bits, plus performant sur des processeurs 32 bits.
- **MD5 (Message Digest 5) :**
  - Une passe de plus / MD4, plus sûre
- **SHA-1 (Secure Hash Algorithm) :**
  - Proposé par le NIST
  - Input message 2^{64} octets (au max), output 160 bits.

Authentification de l'origine de données et MAC

- **L'authentification de l'origine de données permet de vérifier que la source de données est bien l'identité prétendue.**
- **Message Authentication Code (MAC) est un mécanisme cryptographique qui permet de vérifier l'authenticité de l'origine des données et leur intégrité en même temps.**
- **A MAC algorithm is a family of functions h_k parameterized by a secret key k with the following properties :**
  - Given a key k and an input m, $h_k(m)$ is easy to compute,
  - Given zero or more pairs $(m_j, h_k(m_j))$, it is computationally infeasible to compute any pair $(m, h_k(m))$ for any new input m.

Authentification de l'origine de données et MAC



Exemples de MAC

- **HMAC : Mihir Bellare, Ran Canetti, et Hugo Krawczyk 1996**
  - FIPS PUB 198, RFC 2104
- **HMAC-MD5**
- **HMAC-SHA-1**

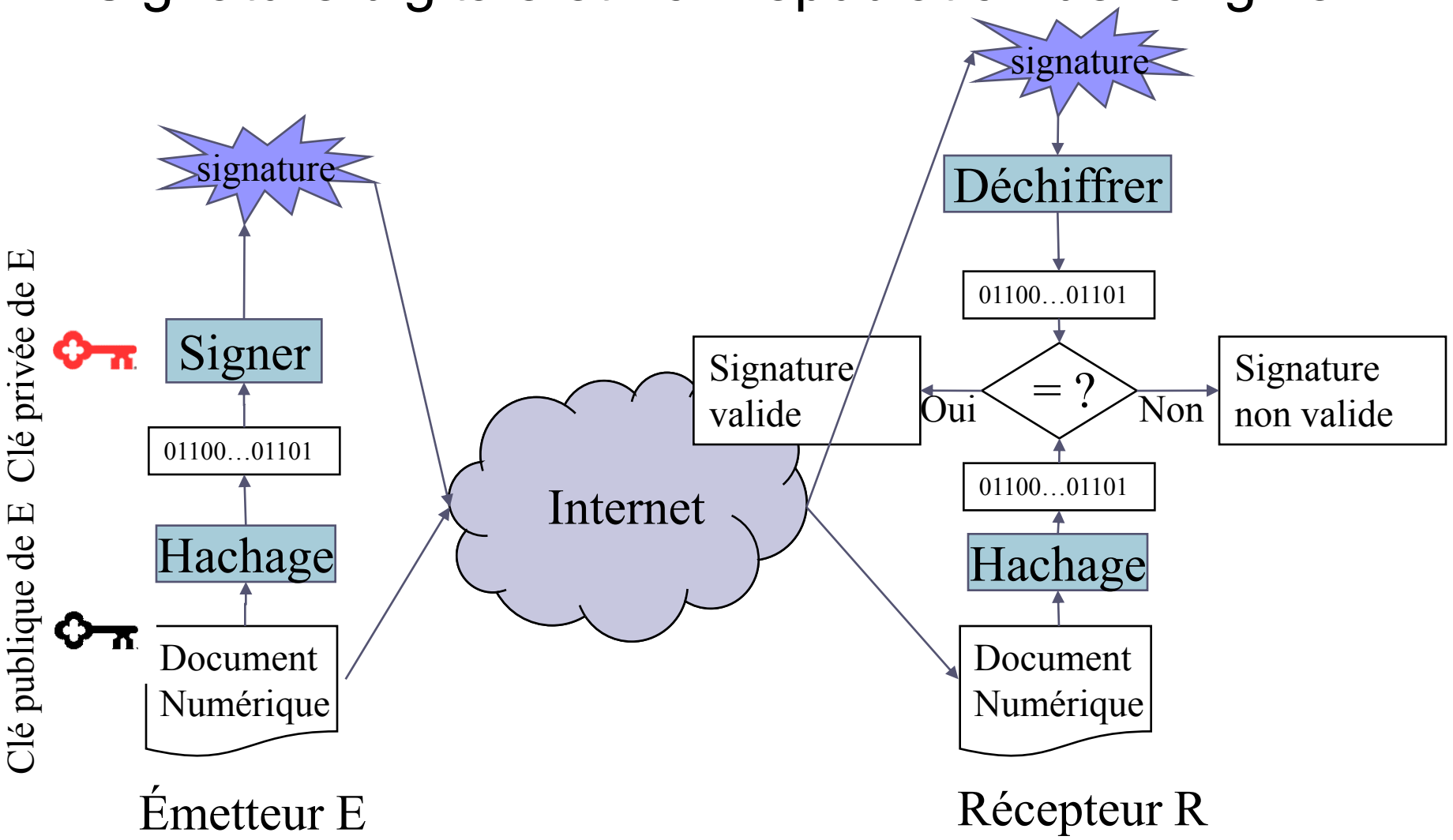
$$HMAC_K(m) = h\Big((K \oplus opad) \,||\, h\big((K \oplus ipad) \,||\, m\big)\Big),$$

- **opad= 0x5c5c5c...5c5c**
- **Ipad= 0x363636...3636**

Non répudiation de l'origine

- **La non répudiation de l'origine assure que l'émetteur du message ne pourra pas nier avoir émis le message dans le futur.**
- **La signature digitale est un mécanisme cryptographique qui permet d'assurer la non répudiation de l'origine.**
- **Repose sur un système cryptographique asymétrique**
- **La signature est calculée en utilisant la clé privé**
- **La signature est vérifiée en utilisant la clé publique**

Signature digitale et non répudiation de l'origine



Signature digitale avec RSA

- **Initialisation**
  - Choisir deux nombres premiers, p et q, les deux étant plus grands que 10^{100}
  - Calculer n = p · q (n est le *modulus*)
  - Choisir e aléatoire tel que e et ((p - 1) · (q - 1)) n'aient aucun facteur commun excepté 1
  - Trouver d tel que : ed = 1 mod((p - 1)(q - 1)).
  - Clé publique : (n,e).
    Clé privée : (n,d) ou (p,q,d) si on désire garder p et q.
- **Signature digitale**
  - L'expéditeur crée la signature s à partir du message m :
    $s = m^d \bmod(n)$, où (n,d) est la clé privée de l'expéditeur.
  - Le destinataire reçoit s et m et effectue la vérification de m :
    $m = s^e \bmod(n)$, où (n,e) est la clé publique de l'expéditeur.
- **Très lent, et clé très longue.**

Historique

- **50 av. JC. : Julius Cesar utilise une simple substitution de l'alphabet pour les communications gouvernementales**
- **1918 : Gilbert Vernam, mathématicien américain, inventa le one-time pad, l'algorithme de chiffrement le plus sûr jusqu'à aujoud'hui, mais impratiquable**
- **1923 : Dr. Albert Scherbius, hollondais résidant en Allemagne, met au point la machine Enigma qui sert à encoder des messages. Le prix très cher en fait un échec.**
- **1925 : La marine de guerre allemande reprend le projet Enigma en le confiant au Chiffrierstelle, le service de chiffrement**

Historique

- **1937 : Enigma M3 est adoptée par le Wehrmacht, l'armée allemande**

- **1939 : début de la seconde guerre mondiale, où des milliers de scientifiques britanniques, polonais et français travaillaient pour solutionner Enigma, et les milliers de messages chiffrés.**
  - L'équipe de Alan Turing trouva la solution

- **1976 : IBM publie un algorithme de chiffrement basé sur Lucifer. Il devient le DES (Data Encryption Standard)**

- **1976 : Whitfield Diffie et Martin Hellman introduisent l'idée d'un système à clé publique**

Historique

- **1978 : l'algorithme de chiffrement à clé publique RSA est publié par Rivest, Shamir et Adleman**
- **1978 : Le RC4 est développé par Ronald Rivest pour la RSA Security et sera gardé secret jusqu'en 1994, où l'algorithme est rendu public anonymement dans une liste de distribution de Cypherpunks**
- **1991 : Phil Zimmermann rend disponible sa première versio de PGP**
- **1992 : IDEA est inventé en Suisse par Xuejia Lai et James Massey**

Historique

- **1992 : MD5 est développé par Ronald L. Rivest**
- **1994 : Ron Rivest, déjà auteur de RC2 et RC4, publie RC5**
- **2000 : Rijndael devient l'AES, le standard du chiffrement avancé**

Atelier: OpenSSL

OpenSSL (1)

- **Projet OpenSSL**
  - http://www.openssl.org
  - 60.000 lignes de code (langage C)
  - Utilisées par de nombreuses applications; openssh, apache+mod_ssl,…
  - Fondé sur la bibliothèque cryptographique SSLeay d'Eric Toung et Tim Hudson
- **Objectifs**
  - Mise en œuvre des protocoles SSL
  - Bibliothèque cryptographique

OpenSSL (2)

- **Interface de programmation en C**
  - Bibliothèque SSL/TLS (libssl.a)
    - ✓ Mise en œuvre des protocoles SSLv2, SSLv3, TLSv1
  - Bibliothèque cryptographique
    - ✓ Cryptographie clé publique et certificats X509: RSA, DSA, DH
    - ✓ Chiffrement: DES, 3DES, Blowfish, RC2, IDEA, RC4, + modes ECB, CBC,CFB,OFb pour les algorithmes par blocs
    - ✓ Hachage: MD2, MD4, MD5, SHA1, MDC2, RIPEMD160

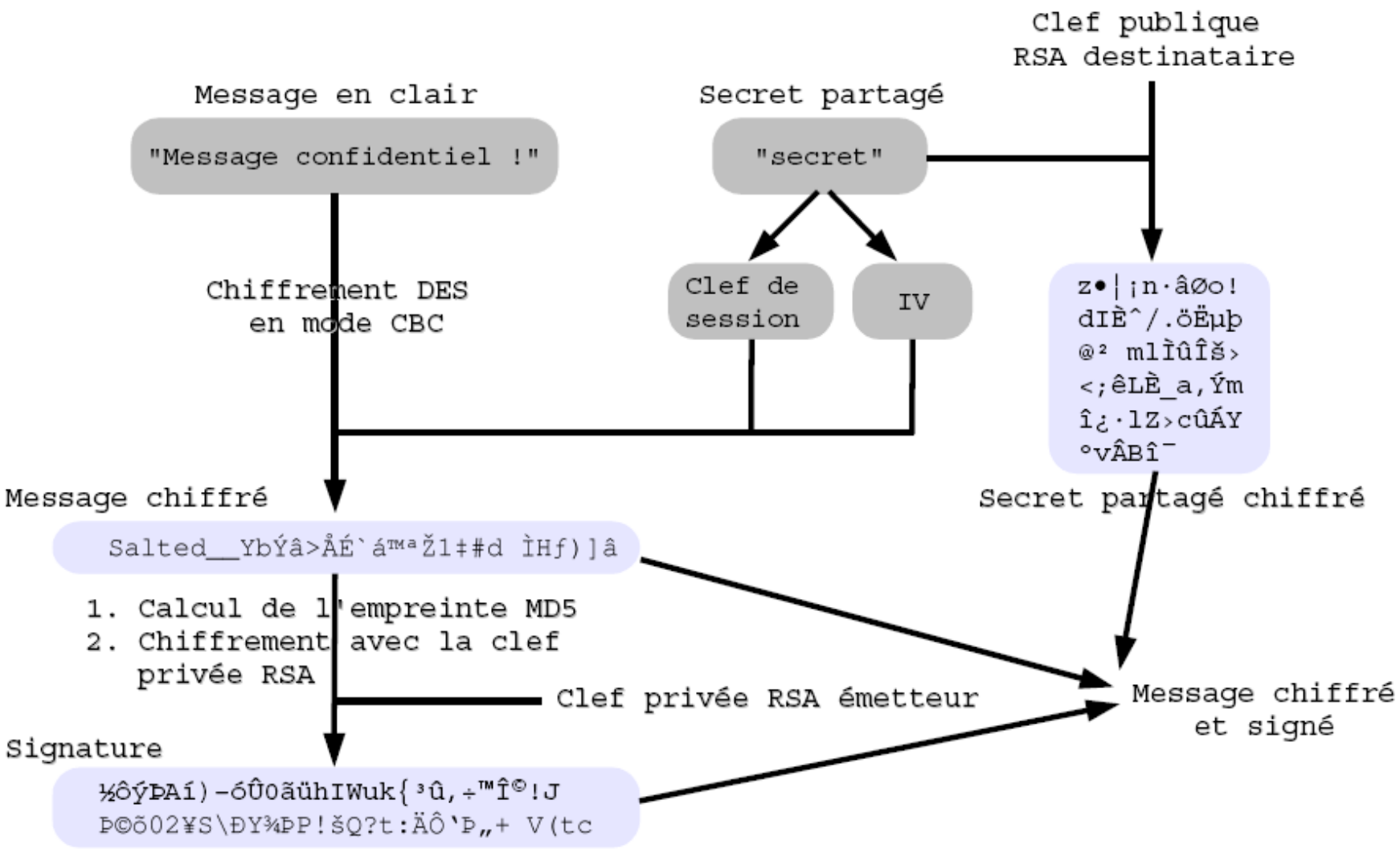
- **Suite d'applications en ligne de commande**
  - openssl(1)
- **Intégration à multiple langages**
  - PHP,
  - Perl
  - …

Atelier OpenSSL

- **SRC voudrait émettre un message M chiffré et signé à DST**

- **SRC génère sa paire de clé privé / publique**
- **DST génère sa paire de clé privé / publique**

- **SRC génère un secret KAB, et le transmet à DST chiffré avec sa clé publique**
- **DST déchiffre KAB avec sa clé privée**
- **SRC chiffre M avec KAB**
- **SRC signe M avec sa clé privé**
- **SRC envoie le résultat à DST**
- **DST vérifie la signature en utilisant la clé publique de SRC**
- **DST déchiffre M en utilisant KAB**


Clef publique
RSA destinataire
Message en clair
"Message confidentiel !"
Secret partagé
"secret"
Chiffrement DES
en mode CBC
Clef de
session
IV
z•|¡n·âØo!
dIÈ^/.öËµþ
@² mlÌûÎš›
<;êLÈ_a,Ým
î¿·lZ›cûÁY
°vÂBî¯
Message chiffré
Secret partagé chiffré
Salted__YbÝâ>ÅÉ`á™ªŽ1‡#d ÌHƒ)]â
1. Calcul de l'empreinte MD5
2. Chiffrement avec la clef
privée RSA
Clef privée RSA émetteur
Message chiffré
et signé
Signature
½ôÝÞAí)-óÛ0ãühIWuk{³û,÷™Î©!J
Þ©õ02¥S\ÐY¾ÞP!šQ?t:ÄÔ'Þ„+ V(tc

Au travail !

Utilisez OpenSSL pour réaliser ce scénario

Atelier OpenSSL

- **Génération des clés privées:**
  - src_rsa.pem, et dst_rsa.pem au format PEM avec mot de passe.
- **PEM: Privacy Enhanced Mail**
  - est un format pour le stockage de clés RSA.
  - Des informations concernant le chiffrement des clés et IV sont

```
$ openssl genrsa -des 512 | tee src_rsa.pem
Generating RSA private key, 512 bit long modulus
  ..............++++++++++++
  .....++++++++++++
  e is 65537 (0x10001)
  Enter PEM pass phrase:
  Verifying password - Enter PEM pass phrase:

  -----BEGIN RSA PRIVATE KEY-----
  Proc-Type: 4,ENCRYPTED
  DEK-Info: DES-CBC,F797B90DFD6AE2BB
  RujFw/mVRdBjfFI8xzrHCd+AvOpD3JfoD5XMEbXDlmprBR9nOdUc/wvjHUYNO4z6
  fq3OGPpb8IGS9IilZDllS3RDui60m3UhcGMH1hd2p+E=
  -----END RSA PRIVATE KEY-----
```

Atelier OpenSSL

- **Extraction des clés publiques src_rsa_pub.pem, et dst_rsa_pub.pem**

```
$ openssl rsa -in src_rsa.pem -pubout | tee src_rsa_pub.pem
read RSA key
Enter PEM pass phrase:
writing RSA key
-----BEGIN PUBLIC KEY-----
MfwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAL5ApUqqRf86ZklRhstCbGSP9REkOo2R
U3E/q1ZRkGxrxKszYv3PEYatDmwZbhtofbndqVfHRUGULSHFsYP8azcCAwEAAQ==
-----END PUBLIC KEY-----
```

Atelier OpenSSL

- **Transport de la clé partagée KAB+IV**
  - Chiffrement du mot de passe avec la clé publique de DST

```
$ echo -n "secret" | openssl rsautl -encrypt -pubin -inkey dst_rsa_pub.pem \
                                                        | tee secret.crypt
z•|¡n·âØo!dIÈ^/.öËµþ@² mlÌûÎš›<;êLÈ_a,Ýmî¿·lZ›cûÁY°vÂBî¯
```

- **Chiffrement du message en DES**

```
$ echo -n "Message confidentiel \!" | openssl enc -des-cbc | tee message.crypt
enter des-cbc encryption password:
Verifying password - enter des-cbc encryption password:
Salted__YbÝâ>ÅÉ`á™ªŽ1‡#d ÌHƒ)]â™
```

- **Calcul de l'empreinte MD5 du message chiffré**

```
$ openssl dgst -md5 -binary message.crypt | tee message.crypt.dgst
```

Atelier OpenSSL

- **Signature du message chiffré avec la clé privée de l'émetteur**

```
openssl rsautl -in message.crypt.dgst -sign -inkey src_rsa.pem \
                                                    | tee message.crypt.sign
 Enter PEM pass phrase:
½ôýÞAí)-óÛ0ãühIWuk{³û,÷™Î©!J
Þ©õ02¥S\ÐY¾ÞP!šQ?t:ÄÔ'Þ„+ V(tc
```

- **Constitution du message à transmettre**
  - Message chiffré: message.crypt
  - Signature du message chiffré: message.crypt.sign
  - Clé partagée: secret.crypt
  - Paramètres administratifs
    - ✓ Identifiant de la clé RSA de l'émetteur
    - ✓ Algorithme utilisé: DES en mode CBC
    - ✓ Nature des différents attachements

Atelier OpenSSL

- **Réception du message chiffré et signé**
- **Vérification de la signature en utilisant la clé publique de SRC**

```
$ cat message.crypt.sign | openssl rsautl -verify -pubin -inkey src_rsa_pub.pem \
                                                              -out dgst1
```

- Re-calcul de l'empreinte du message chiffré

```
$ cat message.crypt | openssl dgst -md5 -binary > dgst2
```

- Comparaison des deux empreintes

```
$ diff -s dgst1 dgst2
  Les fichiers dgst1 et dgst2 sont identiques
```

Atelier OpenSSL

- **Déchiffrement du message reçus**
  - Déchiffrement de la clé de session KAB en utilisant la clé publique de DST

```
$ cat secret.crypt | openssl rsautl -decrypt -inkey dst_rsa.pem
  Enter PEM pass phrase:
  secret
```

  - Déchiffrement du message chiffré avec la clé de session KAB

```
$ cat message.crypt | openssl enc -d -des-cbc
  enter des-cbc decryption password:
  Message confidentiel \!
```

  - Services assurés
    - ✓ Authentification des parties SRC et DST: reste à régler le problème du transport sécurisé des clés publiques des deux parties
    - ✓ Intégrité et Confidentialité du message chiffré