Chapitre 8

Crypto systèmes aymétriques et signatures

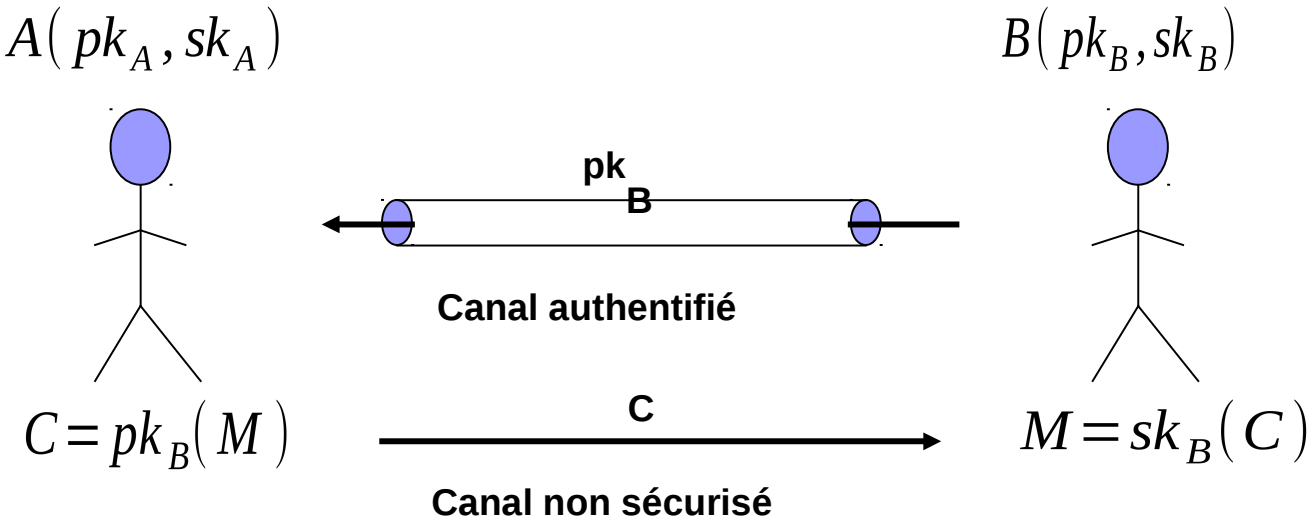
Remarques:

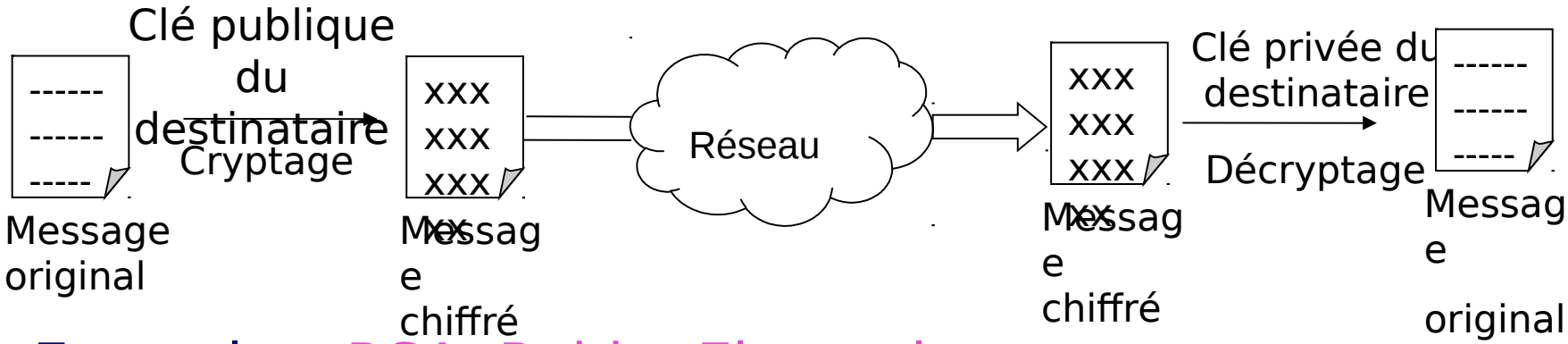
1) Ce document doit être complété par les notes de cours

2) Les transparents sont basés sur le livre « handbook of applied cryptography »

Chiffrement

- Chiffrement asymétrique





- Exemples: RSA, Rabin, Elgamal…

Cryptosystèmes asymétriques

public-key encryption scheme	computational problem
RSA	integer factorization problem (§3.2) RSA problem (§3.3)
Rabin	integer factorization problem (§3.2) square roots modulo composite n (§3.5.2)
ElGamal	discrete logarithm problem (§3.6) Diffie-Hellman problem (§3.7)
generalized ElGamal	generalized discrete logarithm problem (§3.6) generalized Diffie-Hellman problem (§3.7)
McEliece	linear code decoding problem
Merkle-Hellman knapsack	subset sum problem (§3.10)
Chor-Rivest knapsack	subset sum problem (§3.10)
Goldwasser-Micali probabilistic	quadratic residuosity problem (§3.4)
Blum-Goldwasser probabilistic	integer factorization problem (§3.2) Rabin problem (§3.9.3)

Cryptosystème Rabin

8.10 Algorithm Key generation for Rabin public-key encryption

SUMMARY: each entity creates a public key and a corresponding private key.

Each entity A should do the following:

1. Generate two large random (and distinct) primes p and q, each roughly the same size.
2. Compute $n = pq$.
3. A's public key is n; A's private key is (p, q).

Cryptosystème Rabin

8.11 Algorithm Rabin public-key encryption

SUMMARY: B encrypts a message m for A, which A decrypts.

1. *Encryption.* B should do the following:
   (a) Obtain A's authentic public key n.
   (b) Represent the message as an integer m in the range $\{0, 1, \ldots, n-1\}$.
   (c) Compute $c = m^2 \bmod n$.
   (d) Send the ciphertext c to A.
2. *Decryption.* To recover plaintext m from c, A should do the following:
   (a) Use Algorithm 3.44 to find the four square roots m_1, m_2, m_3, and m_4 of c modulo n.[2] (See also Note 8.12.)
   (b) The message sent was either m_1, m_2, m_3, or m_4. A somehow (cf. Note 8.14) decides which of these is m.

Cryptosystème Rabin

8.12 Note (*finding square roots of c modulo $n = pq$ when $p \equiv q \equiv 3 \pmod 4$*) If p and q are both chosen to be $\equiv 3 \pmod 4$, then Algorithm 3.44 for computing the four square roots of c modulo n simplifies as follows:

1. Use the extended Euclidean algorithm (Algorithm 2.107) to find integers a and b satisfying $ap + bq = 1$. Note that a and b can be computed once and for all during the key generation stage (Algorithm 8.10).
2. Compute $r = c^{(p+1)/4} \bmod p$.
3. Compute $s = c^{(q+1)/4} \bmod q$.
4. Compute $x = (aps + bqr) \bmod n$.
5. Compute $y = (aps - bqr) \bmod n$.
6. The four square roots of c modulo n are x, $-x \bmod n$, y, and $-y \bmod n$.

ElGamel

8.17 Algorithm Key generation for ElGamal public-key encryption

SUMMARY: each entity creates a public key and a corresponding private key.
Each entity A should do the following:

1. Generate a large random prime p and a generator α of the multiplicative group $\mathbb{Z}_p^*$ of the integers modulo p (using Algorithm 4.84).
2. Select a random integer a, $1 \leq a \leq p-2$, and compute $\alpha^a \bmod p$ (using Algorithm 2.143).
3. A's public key is (p, α, α^a); A's private key is a.

ElGamel

8.18 Algorithm ElGamal public-key encryption

SUMMARY: B encrypts a message m for A, which A decrypts.

1. *Encryption.* B should do the following:
   (a) Obtain A's authentic public key (p, α, α^a).
   (b) Represent the message as an integer m in the range $\{0, 1, \ldots, p-1\}$.
   (c) Select a random integer k, $1 \leq k \leq p-2$.
   (d) Compute $\gamma = \alpha^k \bmod p$ and $\delta = m \cdot (\alpha^a)^k \bmod p$.
   (e) Send the ciphertext $c = (\gamma, \delta)$ to A.
2. *Decryption.* To recover plaintext m from c, A should do the following:
   (a) Use the private key a to compute $\gamma^{p-1-a} \bmod p$ (note: $\gamma^{p-1-a} = \gamma^{-a} = \alpha^{-ak}$).
   (b) Recover m by computing $(\gamma^{-a}) \cdot \delta \bmod p$.

Generalized ElGamel

8.25 Algorithm Key generation for generalized ElGamal public-key encryption

SUMMARY: each entity creates a public key and a corresponding private key.
Each entity A should do the following:

1. Select an appropriate cyclic group G of order n, with generator α. (It is assumed here that G is written multiplicatively.)
2. Select a random integer a, $1 \leq a \leq n-1$, and compute the group element α^a.
3. A's public key is (α, α^a), together with a description of how to multiply elements in G; A's private key is a.

RSA : key generation

Algorithm Key generation for RSA public-key encryption

SUMMARY: each entity creates an RSA public key and a corresponding private key. Each entity A should do the following:

1. Generate two large random (and distinct) primes p and q, each roughly the same size.
2. Compute $n = pq$ and $\phi = (p-1)(q-1)$. (See Note 8.5.)
3. Select a random integer e, $1 < e < \phi$, such that $\gcd(e, \phi) = 1$.
4. Use the extended Euclidean algorithm (Algorithm 2.107) to compute the unique integer d, $1 < d < \phi$, such that $ed \equiv 1 \pmod{\phi}$.
5. A's public key is (n, e); A's private key is d.

2.107 Algorithm Extended Euclidean algorithm

INPUT: two non-negative integers a and b with $a \geq b$.
OUTPUT: $d = \gcd(a, b)$ and integers x, y satisfying $ax + by = d$.

1. If $b = 0$ then set $d \leftarrow a$, $x \leftarrow 1$, $y \leftarrow 0$, and return(d,x,y).
2. Set $x_2 \leftarrow 1$, $x_1 \leftarrow 0$, $y_2 \leftarrow 0$, $y_1 \leftarrow 1$.
3. While $b > 0$ do the following:
   3.1 $q \leftarrow \lfloor a/b \rfloor$, $r \leftarrow a - qb$, $x \leftarrow x_2 - qx_1$, $y \leftarrow y_2 - qy_1$.
   3.2 $a \leftarrow b$, $b \leftarrow r$, $x_2 \leftarrow x_1$, $x_1 \leftarrow x$, $y_2 \leftarrow y_1$, and $y_1 \leftarrow y$.
4. Set $d \leftarrow a$, $x \leftarrow x_2$, $y \leftarrow y_2$, and return(d,x,y).

RSA : encryption & decryption

Algorithm RSA public-key encryption

SUMMARY: B encrypts a message m for A, which A decrypts.

1. *Encryption.* B should do the following:
   (a) Obtain A's authentic public key (n, e).
   (b) Represent the message as an integer m in the interval $[0, n-1]$.
   (c) Compute $c = m^e \bmod n$ (e.g., using Algorithm 2.143).
   (d) Send the ciphertext c to A.
2. *Decryption.* To recover plaintext m from c, A should do the following:
   (a) Use the private key d to recover $m = c^d \bmod n$.

Signatures digitales

Signature RSA

11.19 Algorithm RSA signature generation and verification

SUMMARY: entity A signs a message $m \in \mathcal{M}$. Any entity B can verify A's signature and recover the message m from the signature.

1. *Signature generation.* Entity A should do the following:
   (a) Compute $\bar{m} = R(m)$, an integer in the range $[0, n-1]$.
   (b) Compute $s = \bar{m}^d \bmod n$.
   (c) A's signature for m is s.
2. *Verification.* To verify A's signature s and recover the message m, B should:
   (a) Obtain A's authentic public key (n, e).
   (b) Compute $\bar{m} = s^e \bmod n$.
   (c) Verify that $\bar{m} \in \mathcal{M}_R$; if not, reject the signature.
   (d) Recover $m = R^{-1}(\bar{m})$.

Signature RSA

11.19 Algorithm RSA signature generation and verification

SUMMARY: entity A signs a message $m \in \mathcal{M}$. Any entity B can verify A's signature and recover the message m from the signature.

1. *Signature generation.* Entity A should do the following:
   (a) Compute $\bar{m} = R(m)$, an integer in the range $[0, n-1]$.
   (b) Compute $s = \bar{m}^d \bmod n$.
   (c) A's signature for m is s.
2. *Verification.* To verify A's signature s and recover the message m, B should:
   (a) Obtain A's authentic public key (n, e).
   (b) Compute $\bar{m} = s^e \bmod n$.
   (c) Verify that $\bar{m} \in \mathcal{M}_R$; if not, reject the signature.
   (d) Recover $m = R^{-1}(\bar{m})$.

Signature Rabin

11.25 Algorithm Rabin signature generation and verification

SUMMARY: entity A signs a message $m \in \mathcal{M}$. Any entity B can verify A's signature and recover the message m from the signature.

1. *Signature generation.* Entity A should do the following:
   (a) Compute $\bar{m} = R(m)$.
   (b) Compute a square root s of $\bar{m} \bmod n$ (using Algorithm 3.44).
   (c) A's signature for m is s.
2. *Verification.* To verify A's signature s and recover the message m, B should:
   (a) Obtain A's authentic public key n.
   (b) Compute $\bar{m} = s^2 \bmod n$.
   (c) Verify that $\bar{m} \in \mathcal{M}_R$; if not, reject the signature.
   (d) Recover $m = R^{-1}(\bar{m})$.

Signature Rabin

Example (*Rabin signature generation with artificially small parameters*)

Key generation. Entity A selects primes $p = 7$, $q = 11$, and computes $n = 77$. A's public key is $n = 77$; A's private key is $(p = 7, q = 11)$. The signing space is $\mathcal{M}_S = Q_{77} = \{1, 4, 9, 15, 16, 23, 25, 36, 37, 53, 58, 60, 64, 67, 71\}$. For the sake of simplicity (but see Note 11.27), take $\mathcal{M} = \mathcal{M}_S$ and the redundancy function R to be the identity map (i.e., $\bar{m} = R(m) = m$).

Signature generation. To sign a message $m = 23$, A computes $R(m) = \bar{m} = 23$, and then finds a square root of $\bar{m}$ modulo 77. If s denotes such a square root, then $s \equiv \pm 3 \pmod 7$ and $s \equiv \pm 1 \pmod{11}$, implying $s = 10, 32, 45$, or 67. The signature for m is chosen to be $s = 45$. (The signature could be any one of the four square roots.)

Signature verification. B computes $\bar{m} = s^2 \bmod 77 = 23$. Since $\bar{m} = 23 \in \mathcal{M}_R$, B accepts the signature and recovers $m = R^{-1}(\bar{m}) = 23$. □