Security Models TD

# Session 1

## Exercice 1 (Fonction négligeable)

f est négligeable :

Il existe N appartient à N, pour tout q un polynôme.

Pour tout n > N, f(n) <= 1/q(n)

1/ f ¤ g est négligeable :

Soit p(x) un polynôme, p = somme de i=x à d (ai \* x)

On définit que q = somme de i = 0 à d (bi \* x^i) avec bi = max(i, ai) donc pour n>1, p(n) <= q(n) <= q^x(n)

* F est négligeable, il existe un Ni tel que pour tout n>N, f(n) < q(n)
* G est négligeable, il existe Nx tel que pour n>N, g(x) <q(n)

N = max(Ni, Nx), pour tout n > N, f(n) ¤ g(n) < 1/q²(n)

🡺Pour tout n > N, f(n) ¤ g(n) < 1/p(n) donc f ¤ g est négligeable.

2/

Pour tout k, f^k est négligeabe.

Soit p(.) et q(.) comme dans 1/

Pour tout n>I, p(n) <) q(n) <= q^k(n)

F est négligeable, il existe un N tel que pour tout n > N, f(n) < 1/q(n)

Donc pour tout n > N, f^k(n) < 1/q^k(n)

Donc pour tout n > N, f^k(n) < 1/p(n)

3/

Pour tout u, l appartenant à R, lf + ug est négligeable

Soit p(.) un polynome et q(.) tel que q(x) = p(x)/l+u

F est négligable, il existe N, tel que pour tout n> N, f(n) < 1/q(n)

G est négligable, il existe N, tel que pour tout n> N, g(n) < 1/q(n)

Soit N = max(Ni, Nx) alors pour tout n > N, lf(n) + ug(n) < l/q(n) + u/q(n)

⬄ lf(n) + u g(n) <= 1/p(n)

## Exercice 2 (DL, CDH, DDH assumption)

DL : Discret Logarithm

Adv^DL(A) = Pr[A(g^x) -> x]

CDH : Computationnal Diffrie-Hellman

Adv^CDH(A) = Pr[A(g^x, g^y) -> g^xy]

DDH : Decisionnal DH

Adv^DDU(A) = Pr[A(g^x, g^y, g^xy) -> 1]

1/ Solving DL 🡺 Solving CDH

Preuve par réduction : Soit A contre DL. On désigne B contre CDH.

Soit X = g^x et Y = g^y

B(X, Y) : {

run A(X) -> x

retourner Y^x g^xy

Adv^CDH(A) = Pr[A(g^x, g^y) -> g^xy)]

= Pr[A^x(g^x) -> v : v = x]

= Pr[A(g^x) -> x]

= Adv^DL(A)

2/ Solving CDH 🡺Solving DDH

Soit A contre CDH. On construit B contre DDH de la manière qui suit :

B(X, Y, Z) :

Si A(X, Y) = Z

Retourner 1

Retourner 0

Adv^DDH (A) = Pr[A(g^x, g^y, g^xy) -> 1

-Pr[A(g^x, g^y, g^r) -> 1]

= Pr[A(g^x, g^y) -> g^xy]

-Pr[A(g^x, g^y) -> g^r)]

= Pr[A(g^x, g^y) -> g^xy]

-1/q

=Adv^CDH(A) – 1/q

## Exercice 3

Adversary Challenger

🡨 pk 🡨 sk,pk 🡨 Genere

m0, m1 🡪m0, m1🡪 b 🡨$ {0,1}

b’ 🡨 Epk(mb) 🡨

Adversary gagne ssi b’ = b

Comme l’adversaire connait m0 et m1, et il a accès au schéma de chiffrement, il a aussi Encpk(m0) et Encpk(m1). Une fois qu’il a reçu le challenge Epk(mb), il peut le comparer a Enc(m0) et à Enc(m1). Et avec le schéma de chiffrement qui est deteriniste, on a Enc(m0) = Enc(m1) ou Enc(m0) = Enc(mb). Sachant que l’on peut décidersi b’ = 1 ou b’ = 0

* A gagne IND-CPA avec des probabilité non négligeable.

## Exercice 4 :

E(x) = E(x) || H(x)

n l

IND-CPA secure

A C

m0, m1 🡪 b 🡨 {0,1}

E{mb,mb} = E(mb) || H(mb)

E(mb) E(mb)

H(mb) 🡨

H(m0) = H(mb)

E(mb) = E(mb) || H(mb)

O^n || H(m0) == E(mb) || H(mb)

Si les l derniers bits sont les mêmes :

A return b’ = 0

Sinon

B’ = 1

## Exercice 5 (One way security)

Soit f une fonction de hash.

E : x aléatoire

Epk(m) = f(x) || x xor m

If f is OW then E is OW-CPA ?

On suppose qu’il existe A contre OW-CPA et soit B contre OW de f :

B(y = f(x)) :

Prendre r aléatoire

A(f, y||r) -> m

X <- m xor r

Retourner x

# Session 2

## Exercice 1 :

CFB : cipher feedback

A IV

Sk 🡪 Enc sk 🡪 Enc sk 🡪 Enc

m0 🡪 xor m1 🡪 xor m2 🡪 xor

C0 C1 C2

C0 = IV

Ci = Esk(IV) xor Pi

xor Esk (Ci-1)

k xor k = 0

Pi = Ci xor E(Ci-1)

E(m1 xor 0^n/2 \* 1^n/2)

C = IV || C1 || C2

CFB n’est pas IND-CCAE securise

m0 = 0^n

m1 = 1^n

ci = E(IV) xor mb xor 1^n

🡪CFB(mb) = IV ||G

Dec(IV || Ci xor 1^n) = IV || D1

Si D1 == 0^n

A retourne 1

Sinon 0

Dans un premier temps Ci = E(IV) xor mb

* B = 0 : mb = O^n donc Ci = E(IV) xor O^n = E(IV)

Donc D1 = C1 xor 1^n xor E(IV) = 1^n

Alors D1 = 0 est faux

* B = 1 : mi = 1^n donc Ci = E(IV) xor 1^n

Donc D1 = C1 xor 1^n xor E(IV) = O^n

Alors D1 = O^n est vrai

CTR : counter

…

A passes

C = IV || B

A compute : m = B xor Dctr(IV || O^n)

Si m = O^n retourner 0

Sinon retourner 1

OFB : output feedback

IV

Sk 🡪 E sk 🡪 E

Sk 🡪 E

P2 🡪 xor p3🡪 xor

P1 xor

C2 C3

C1

Oo = IV

Oi = Esk(Oi – 1)

Ci = Oi xor Pi

Pi = Ci xor Oi

m0 = O^n

mi = 1^n

A reçoit C= Eofb(mb) = IV || C1

Defb (IV || C1 xor 1^n) = IV || D1

b = 0 : C1 = IV xor m0 = IV

donc C1 xor 1^n = IV xor 1^n

donc D1 = C1 xor 1^n xor IV = IV xor 1^n xor IV = 1^n = m1

b = 1 : C1 = IV xor m1 = IV xor 1^n

donc C1 xor 1^n = IV

donc D1 = C1 xor 1^n xor IV = O^n = m0

si D1 = m1 retourner 0

sinon retourner 1

## Exercice 2 :

Zheng & Seberry crypto system

f : one-way trapdoor function

H, G public hash function

E(x) = f(r) || (G(r) xor (x || H(x)))

OW (one-way) : f(x) il est difficile de trouver x

Avec une trapdoor 🡪 facile

Dechiffrement : inverser f pour avoir r

Compute G(r)

Xor G(r) avec G(r) xor (x || H(x))

Compute H sur le premier block et check si il est egal au second block

A choisie m0 et m1

A reçoit c = f(r) || G(r) xor C\*mb || H(mb))

Dec(c’) avec

b = 1 mbbarre

c’ = f(r) || G(r) xor (mb || H(mb)) xor m0 || H(m0) xor m1 || H(m1)

b = 0 b = 0 b = 1

m0 = mbbarre

Dec( f(r) || G(r) xor (mbbarre || H(mbbarre))

Si dec(c’) = m0 retourner 1

Sinon retourner 0

## Exercice 3

E1, E2 : deux schema de chiffrement symétrique

E’((k1, k2), m) = E2(k2, E1(k1, m))

Montrez que si E1 or E2 est IND-CPA secure alors E’ est IND-CPA secure alors E’ est IND-CPA secure

On suppose que A casse E’ donc on construit B tel que B casse E1, et C casse E2

Adv B : choisit aléatoirement k2 pour E2

Run algo A, quand A fait des requêtes (m0, m1), B demande à son orale avec (m0 , m1) pour avoir c’ = E2(k2, c) et retourne c’ à 1

Quand A retourne b’, B retourne b’

Adv C : sample k1 pour E1 :

Quand A queries (m0, m1), l’adversaire C compute c0 = E1 (k1, m0), c1 = E1(k1 , m1) and queries (c0, c1) pour obtenir c’ et retourne c’ à A.

Quand A retourne b’, C retourne b’.

N = pq

Pgcd(phi(n), n) = 1

Pk = n

Sk = phi(n)

E = Zn \* Zn 🡪 Z²n\*

(m, r) 🡪 (1 + n)^m \* r^n

Not IND-CCA2 secure : Dec((1 + n)^m \* r^n) = m

1. Game
2. Cb = E(mb)

C\* = C x r^n\* = (1 + n)^mb x (r x r\*)^n = (1 + n)^mb (r)^n

Adversaire doit être définis par son comportement :

* Honnête
* Semi-honnête (garde des enregistrements de tous les messages qu’il a vu)
* Active

# Session 3

## Exercice 1 :

1. <m1, m2> € T /

(UL) T |- <m1, m2>/ T |- m1

(C) T |- m1 | T |- m1 / T |- {m1} m1

Voir feuille papier

Le seul termes où m3 apparait est {<m1, m4>}m3 mais il n’y a pas de règles pour avoir m3

T1 = {{m1}m2, m2, {m3}<m1, m4>, {<m1, m4>}<m1, m2>} >= <m3, m4>

Subproof : T |- m4

{m1}m2 € T m2 € T

1. (A)

T |- {m1}m2 T |- m2 m2 € T

(D) (A)

{m1, m4}<m1, m4> T |- m1 T |- m2

(A) (P)

T |- {m1, m4}<m1, m4> T |- <m1, m2>

<m1, m4>

(UR)

T |- m4

m2 € T

1. (A)

{m3}<> € T T |- m2 T |- m4

(P)

T |- {m3}<m2, m> T |- <m2, m4>

(D)

T |- m4 T |- m3

(P)

<m3, m4>

## Exercice 2 :

<a, {z}b> s = <x, y>

Sigma = {x 🡨 a, y 🡨 {z}b}

Sigma = {x 🡨 a, z 🡨 {y}b}

NOPE

T = {<a, z>}b s = {<y, {x}b>}b

Sigma = {y 🡨 a, y 🡨 {x}b, x 🡨 b}

## Exercice 3 :

Montrez que P est une preuve minimal de T |- u 🡺 P une preuve simple de T |- u

On suppose que P est une preuve non simple alors il y a une branche de P dans laquelle T |- w 2 fois.

On peut couper la dérivation entre ces 2 branches et avoir une preuves P’ plus petite

Contradiction avec P est minimale.

A 🡪 B : <{k1}k2, m>

B 🡪 : {m}<k1, k2>

Resultat local de McAllister

Compute S(T U {s}) =

T U {k1, k2, <k1, k2>, {k1}k2}

On peut seulement appliquer UL et UR sur le premier élément

Pour le chiffrement / déchiffrement on ne doit pas tomber dans S(T U {s})

A partir de UL on a {k1}k2

A partir d’ici on ne peut pas appliquer les règles :

* UL et UR déà faite
* P et C on ne tombe pas dans S
* D on ne connait pas la clefs

S(t) tel que :

* T € S(t)
* <u,v> € S(t) => u, v € S(t)
* {u}v € S(t) => u, v € S(t)

# Session 4

## Exercice 1 :

C’est le protocole de NeedHam Schroeder

Rôle : sequence finis d’actions

(u -> v) … (un -> vn) 🡪 (send -> rcv)

R1 : initiator R1 = (init, Xb 🡪 {A, Na}pk(xb))

({Na, xNa}pk(B) 🡪{xNb}pk(xB)

R2 : responder R2 = ({Xa, XNa}pk(b) 🡪 ({XNa, Nb}pk(a)}

({Nb}pk(b) 🡪 stop

## Exercice 2 :

3 sessions entre A et B

Attaque : 1.1 A🡪B : A

2 .1 A 🡪 B : A

3.1 A 🡪 B : A

1.2 B 🡪 I(A) : B, N1, M1, O1

2.2 B 🡪 I(A) : B, N2, M2, O2

3.2 B 🡪 I(A) : B, N3, M3, O3

1.2 I(B) 🡪 A : B, N1, N2, N3

1.3 A 🡪 B : N1, {N1, N2, N3, S}pkB

1.4 B 🡪 A : N1, N2, {N2, N3, S, N1}pkB

2.3