Criptografia FIB

Criptografia de clau pública

Anna Rio

Departament de Matemàtica Aplicada II • Universitat Politècnica de Catalunya

Tres inconvenients dels criptosistemes de clau secreta

Distribució de claus (Key distribution)

Una clau per a cada parell d'usuaris

Distribució prèvia a través d'un canal insegur

Gestió de claus (Key Management)

Xarxa de n usuaris $\Longrightarrow$ $\binom{n}{2} = \mathcal{O}(n^2)$ claus

Cada usuari ha d'emmagatzemar $n-1$ claus

Sense signatura digital (Without digital signature)

No donen mecanismes d'autenticació

(no repudi, identificació, data/hora, etc.)

Però tenen avantatges...

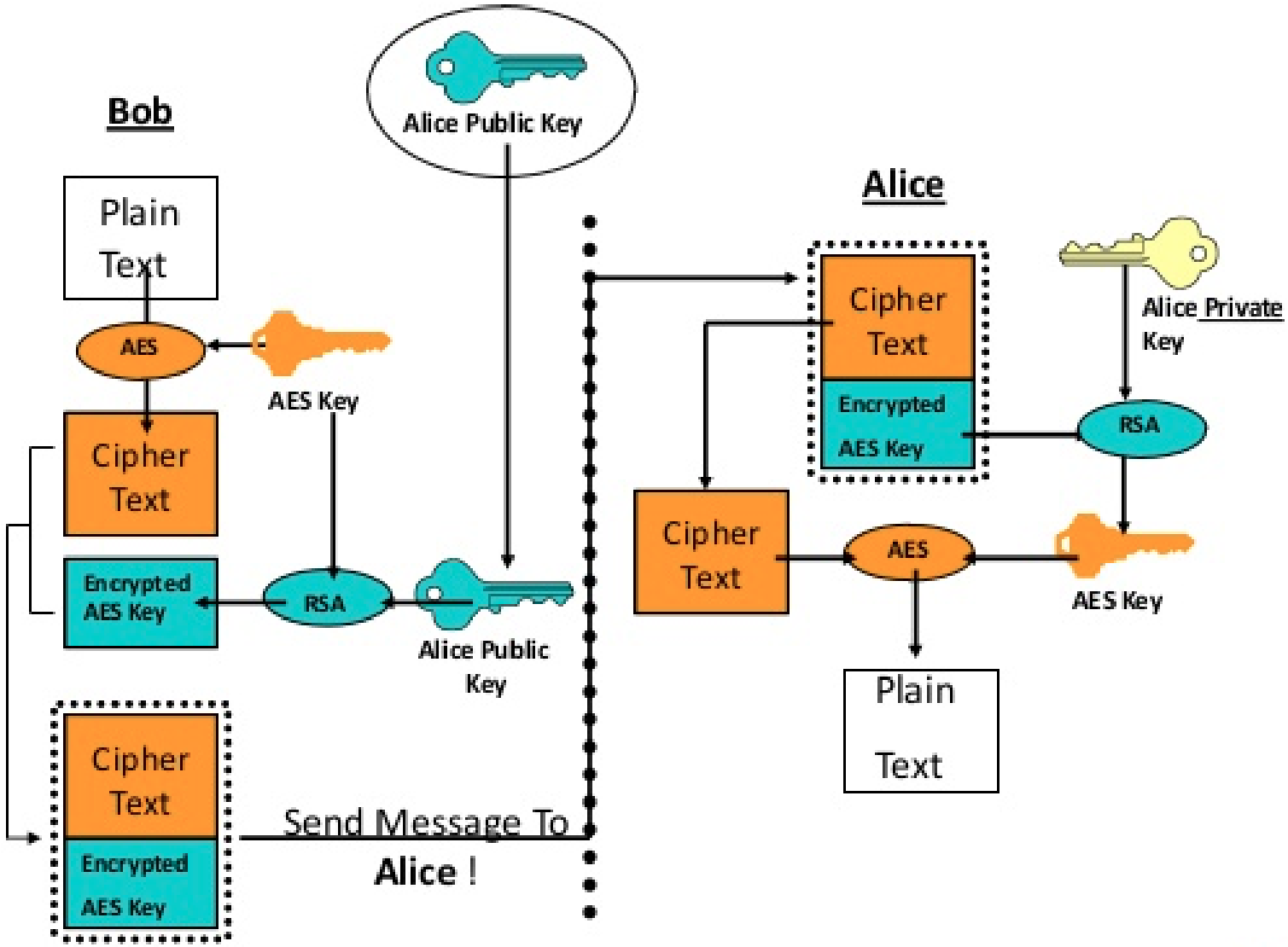
Són més ràpids

En software, DES és com a mínim 100 vegades més ràpid que RSA

En hardware, DES és entre 1000 i 10000 vegades més ràpid que RSA

Sistemes **híbrids**: incorporen algoritmes de clau secreta i algoritmes de clau pública

Components of Hybrid Cryptography



Criptografia de clau pública

Whitfield Diffie

Martin Hellman

New Directions in Cryptography, IEEE Transactions on Information Theory 22 (6): 644–654, 1976

Diffie-Hellman (1976)

Intercanvi de claus a través d'un canal insegur (canal autenticat però no confidencial)

- A i B acorden (públicament) un primer p i un element

$$x \in \mathbb{Z}/p\mathbb{Z} = \{0, 1, 2, \ldots, p-1\}$$

(operació: producte mòdul p)

- A genera un nombre enter a, envia $x^a \bmod p \longrightarrow$ B
- B genera un nombre enter b, envia $x^b \bmod p \longrightarrow$ A
- A calcula $(x^b)^a \bmod p$, B calcula $(x^a)^b \bmod p$

CLAU = aquest element comú $x^{ab} \in \mathbb{Z}/p\mathbb{Z}$

Intercanvi de claus Diffie-Hellman

Eficiència: Bon algoritme d'exponenciació a $\mathbb{Z}/p\mathbb{Z}$?

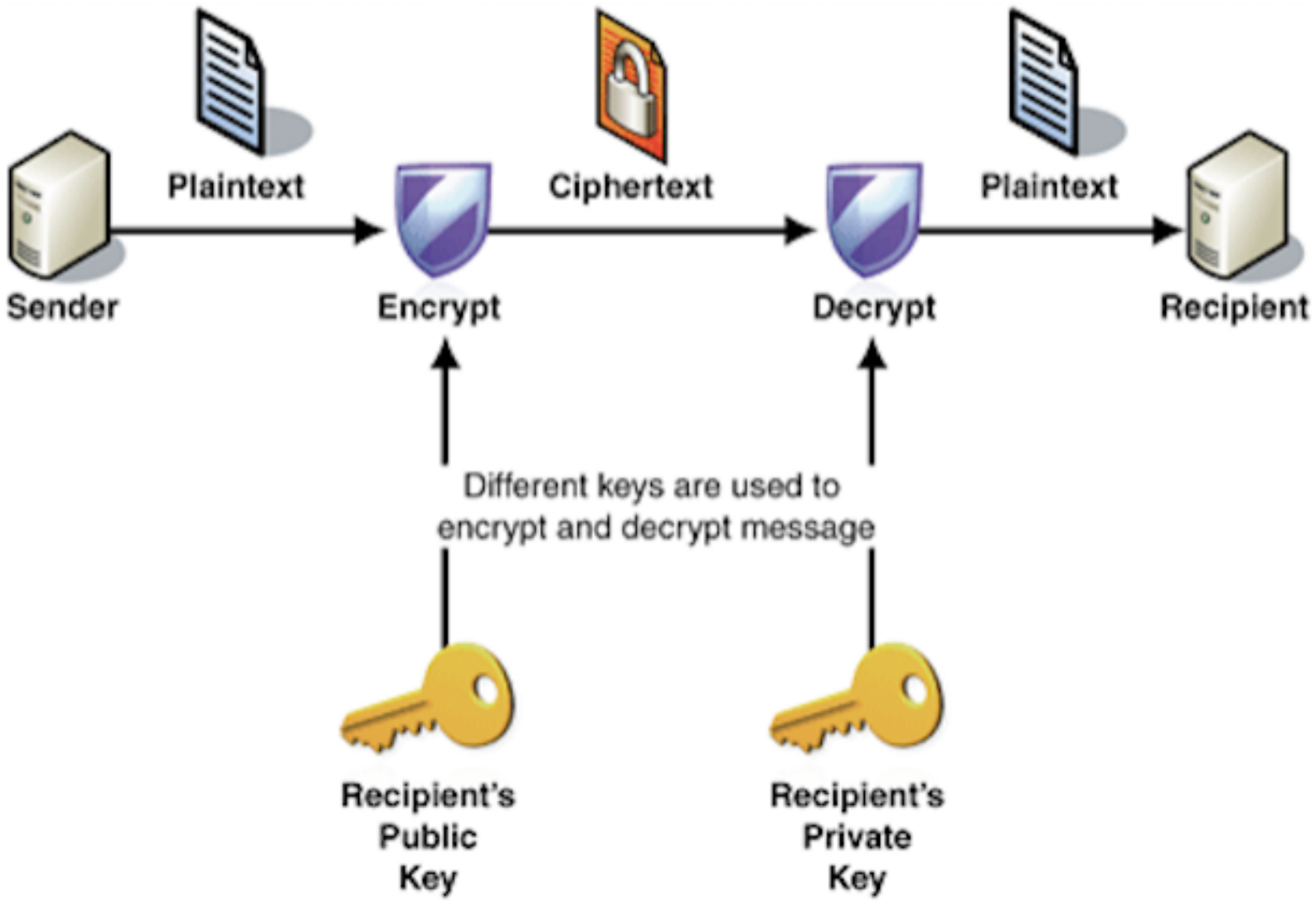
(També es necessita un bon generador de nombres aleatoris)

Secret: Problema de Diffie i Hellman

Coneguts $x^a,\ x^b \in \mathbb{Z}/p\mathbb{Z}$ (elements intercanviats)

es pot calcular fàcilment $x^{ab} \in \mathbb{Z}/p\mathbb{Z}$? (clau)

Sistemes de clau pública



Sistemes de clau pública

- **La regla per xifrar es pot fer pública sense comprometre la seguretat del sistema**
- Es pot establir comunicació xifrada sense intercanvi previ de claus
- Permeten la signatura de documents i l'autenticació de signatures
- Cada usuari té dues claus
  - una clau pública k que s'usa per xifrar/autenticar
  - una clau privada T_k que s'usa per desxifrar/signar

  (criptosistemes asimètrics)
- La seguretat depèn del secret de la clau privada. Ha d'ésser impossible obtenir-la a partir de la pública

Seguretat computacional

Impossible $\approx$ Computacionalment intractable

Sabem quina és la fórmula matemàtica per obtenir la clau privada a partir de la pública però no existeix (o no es coneix?) un bon algoritme per calcular-la ràpidament.

Funcions unidireccionals

Una funció unidireccional és $F : \mathcal{M} \longrightarrow \mathcal{C}$ invertible tal que

- és fàcil (tractable) calcular $F(x)$
- és difícil (intractable) calcular $F^{-1}(y)$

Els conceptes fàcil i difícil els defineix la teoria de la complexitat algorítmica:
Fàcil $\approx$ existeix un algoritme de complexitat polinòmica per resoldre'l

Una porta trampa per a F és una informació addicional que permet invertir-la fàcilment

$$\begin{aligned} a &\longrightarrow x^a \bmod p \\ (p, q) &\longrightarrow pq \end{aligned}$$

Criptosistema de clau pública

Família $\{F_k\}$ de funcions unidireccionals amb porta trampa

$$\begin{aligned} \mathit{Xifrar}(\, x, (k, T_k)\,) &= F_k\,(x) \\ \mathit{Desxifrar}(\, y, (k, T_k)\,) &= F_k^{-1}\,(y) \end{aligned}$$

Matemàticament: *Desxifrar* només depèn de k
Computacionalment: depèn de T_k (clau privada, porta trampa)

La seguretat només depèn de què cada usuari mantingui en secret la seva clau privada. (Ha d'ésser impossible obtenir-la a partir de la pública)

Existeixen funcions unidireccionals?
Només conjecturalment

El criptosistema RSA (Rivest-Shamir-Adleman, 1978)

Generació de claus

Cada usuari

1. tria dos nombres primers p i q
2. calcula $n = pq$
3. calcula $\varphi(n) = (p-1)(q-1)$
4. tria e tal que $\gcd(e, \varphi(n)) = 1$
5. calcula $d = e^{-1} \bmod \varphi(n)$
6. dóna a conèixer la clau pública $\{n, e\}$
7. guarda la clau privada d (o bé $\{d, p, q\}$)

Claus RSA

Mòdul n de 1024 bits (309 xifres decimals)

p
84997172489332578490384442745470933566648021786255802843114129
10633887577008418131538304526889595141427192053810212528423419
13527539164815056302021637877 3

q
12498707513769968201013034478471988740117176620918442925481400
72671578056682003180611331386571634174065507550509272324872789
84454650243654196318594606447 67

n
10623547984416231311262362237670606283574838895956233591624382
11426385162644599890062869006179036928029108937300098743285623
57789700487670277809790099753313862825239708084105047965267520
60747600512765302119035357533637842950893724249852886076997137
38635598015442404762112284450806251799198351285190964723308 91

e
65537

d
17594334471039836222659212311774533561487602633124640951445907
42149012703563557563932187953837872776725811130423313816565017
97679083432743823389453551844338427178964159288543712996418572
64697286846478411297010522536915508500419178724741886020587392
57329691571217940699653246780450866451583806390361244939905 7

Xifratge RSA

Missatge per a l'usuari **(n,e)**

1. Bloc de missatge $\rightsquigarrow m \in \mathbf{Z}/n\mathbf{Z}$
2. Criptograma $c = m^e \bmod n$

A la pràctica els criptosistemes de clau pública no s'utilitzen per xifrar missatges sinó per xifrar claus temporals utilitzades per sistemes simètrics. És amb aquests darrers que es xifren els missatges, perquè són molt més eficients.

Criptograma rebut per l'usuari (n,e,**d**)

1. Missatge $m = c^d \bmod n$ (Teorema d'Euler)
2. $m \in \mathbf{Z}/n\mathbf{Z} \rightsquigarrow$ Bloc de missatge

Teorema d'Euler (1707-1783)

Si x és un enter tal que $\gcd(x, n) = 1$, aleshores $x^{\varphi(n)} \equiv 1 \bmod n$

Suposem que $ed \equiv 1 \bmod \varphi(n)$. Això vol dir que

$$ed = 1 + \lambda\varphi(n) \qquad \text{(per a algun } \lambda\text{)}$$

Llavors, si $c \equiv m^e \bmod n$,

$$c^d \equiv (m^e)^d = m^{ed} = m^{1+\lambda\varphi(n)} = m \cdot (m^{\varphi(n)})^\lambda \equiv m \cdot 1^\lambda = m$$

Per tant, si calculem $c^d \bmod n$ recuperarem m

Teorema d'Euler (1707-1783)

Si x és un enter tal que $\gcd(x, n) = 1$, aleshores $x^{\varphi(n)} \equiv 1 \bmod n$

Suposem que $ed \equiv 1 \bmod \varphi(n)$. Això vol dir que

$$ed = 1 + \lambda\varphi(n) \qquad \text{(per a algun } \lambda\text{)}$$

Llavors, si $c \equiv m^e \bmod n$,

$$c^d \equiv (m^e)^d = m^{ed} = m^{1+\lambda\varphi(n)} = m \cdot (m^{\varphi(n)})^\lambda \equiv m \cdot 1^\lambda = m$$

Per tant, si calculem $c^d \bmod n$ recuperarem m

Teorema d'Euler (1707-1783)

Si x és un enter tal que $\gcd(x, n) = 1$, aleshores $x^{\varphi(n)} \equiv 1 \bmod n$

Suposem que $ed \equiv 1 \bmod \varphi(n)$. Això vol dir que

$$ed = 1 + \lambda\varphi(n) \qquad \text{(per a algun } \lambda\text{)}$$

Llavors, si $c \equiv m^e \bmod n$,

$$c^d \equiv (m^e)^d = m^{ed} = m^{1+\lambda\varphi(n)} = m \cdot (m^{\varphi(n)})^\lambda \equiv m \cdot 1^\lambda = m$$

Per tant, si calculem $c^d \bmod n$ recuperarem m

Signatura RSA

L'usuari A **signa** un missatge m destinat a l'usuari B

1. Signatura (rúbrica) $r = h(m)^{d_A} \bmod n_A$
   $h(m)$ és un hash del missatge. Per exemple, SHA-256
2. $s = r^{e_B} \bmod n_B$

L'usuari B **autentica** la signatura del missatge m per part de A

1. $s^{d_B} \bmod n_B = r$
2. $r^{e_A} \bmod n_A = h(m)$

Eficiència Claus RSA

MCD i inversos modulars d'enters $< n$

Algoritme d'Euclides (estès) $\mathcal{O}(\ell(n)^2)$

$\gcd(\varphi(n), e) = 1$	$\varphi(n)\,x + ed = 1$	
$\text{mcd}(19872,\ 343)$	1	0
$\text{mcd}(343,\ 19872 \bmod 343)$	0	1
$\text{mcd}(321,\ 343 \bmod 321)$	1	-57
$\text{mcd}(22,\ 321 \bmod 22)$	-1	58
$\text{mcd}(13,\ 22 \bmod 13)$	15	-869
$\text{mcd}(9,\ 13 \bmod 9)$	-16	927
$\text{mcd}(4,\ 9 \bmod 4)$	31	-1796
$\text{mcd}(1,\ 4 \bmod 1) = \text{mcd}(1,\ 0) = \mathbf{1}$	$x = -78$	$\mathbf{d = 4519}$

Eficiència RSA

Exponenciació modular d'enters $< n$

$$m^e \bmod n$$

Algoritme de quadrats successius $\mathcal{O}(\ell(n)^3)$

$$\begin{aligned} 23 = 10111 &= 2 \cdot 11 + 1 = 2(2 \cdot 5 + 1) + 1 = \cdots = \\ &= 2(2(2(2(0+1)+0)+1)+1)+1 \\ a^{23} &= ((((1 * a)^2 * 1)^2 * a)^2 * a)^2 * a \qquad \text{(8 productes)} \end{aligned}$$

$$65537 = 2^{16} + 1 = 1000000000000001$$

$$a^{65537} = ((((((((((((((((1 * a)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2 * a$$

Desxifratge/Signatura RSA

Per calcular $m = c^d \bmod n$

1. Calcular

$$\begin{array}{ll} d_1 = d \mod (p-1) & p_1 = p^{-1} \mod q \\ d_2 = d \mod (q-1) & q_1 = q^{-1} \mod p \end{array}$$

clau privada $(d_1, d_2, p, q, p_1, q_1)$

2. Calcular $c_1 = c^{d_1} \pmod p$ i $c_2 = c^{d_2} \pmod q$
3. Resoldre el sistema xinès

$$\left.\begin{array}{ll} m \equiv c_1 \mod p \\ m \equiv c_2 \mod q \end{array}\right\}$$

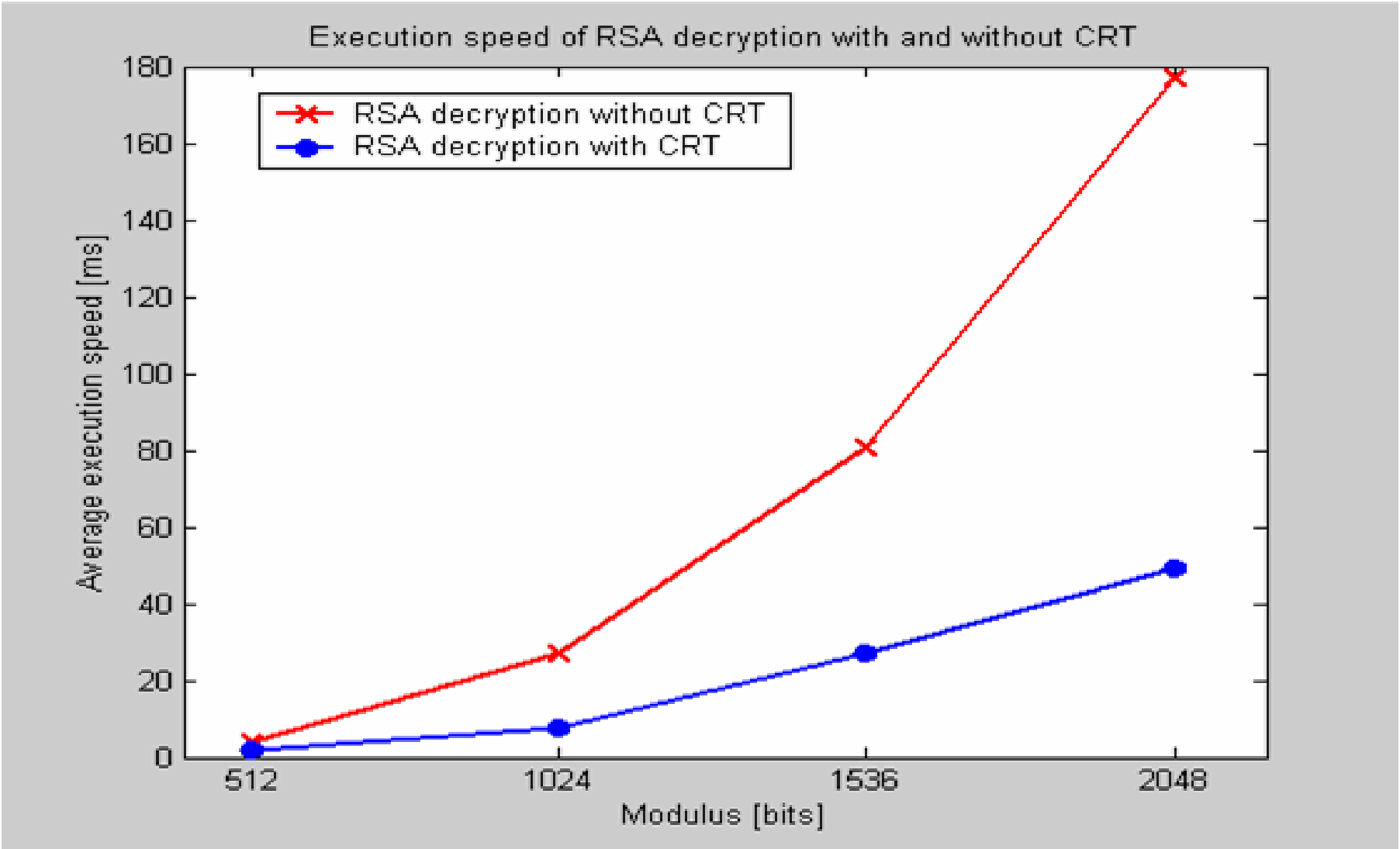
és a dir, calcular $m = c_1 q_1 q + c_2 p_1 p \mod n$

SLE 66CX322P

136-Kbytes ROM, 5052 bytes RAM, 32-Kbytes EEPROM

Operation	**Modulus**	**Exponent**	**Calculation Time**		
			5 MHz	10 MHz	15 MHz
RSA Encrypt / Signature Verify	1024 bit 2048 bit	17 bit 17 bit	20 ms 630 ms	11 ms 315 ms	7 ms 210 ms
RSA Decrypt / Signature Generate	1024 bit	1024 bit	820 ms	410 ms	273 ms
RSA Decrypt / Signature Generate using CRT	eq.1024 bit eq.2048 bit	eq.1024 bit eq.2048 bit	250 ms 1840 ms	125 ms 920 ms	83 ms 614 ms
DSA Signature Generate	512 bit	160 bit	97 ms	49 ms	32 ms
DSA Signature Verify	512 bit	160 bit	117 ms	59 ms	39 ms
DSA Signature Generate	1024 bit	160 bit	438 ms	219 ms	146 ms
DSA Signature Verify	1024 bit	160 bit	711 ms	356 ms	237 ms

Operation	**Operand Length**	**Calculation Time**		
		5 MHz	10 MHz	15 MHz
EC-DSA GF(2^n) Signature Generate	192 bit	285 ms	142 ms	95 ms
EC-DSA GF(2^n) Signature Verify	192 bit	540 ms	270 ms	180 ms


Execution speed of RSA decryption with and without CRT
RSA decryption without CRT
RSA decryption with CRT
Average execution speed [ms]
Modulus [bits]
0
20
40
60
80
100
120
140
160
180
512
1024
1536
2048