NORTHERN ARIZONA UNIVERSITY

INF 638
Cryptography & Cryptosystems

Section 8: RSA Asymmetrical Cryptography

Dr. Bertrand Cambou
Professor of Practice NAU, Cybersecurity
School of Informatics, Computing, and Cyber-Systems
Bertrand.cambou@nau.edu

INF 633: Cryptography & Cryptosystems

- 1- Motivation & Definitions
- 2- Elements of Number theory
- 3- Early Cryptographic methods
- 4- Symmetrical Cryptography: DES
- 5- Symmetrical Cryptography: AES
- 6- Quantum Cryptography: Key distribution
- 7- Elements of Asymmetrical Cryptography
- 8- Asymmetrical Cryptography: RSA
- 9- ECC Key Distribution
- 10- PKI & Digital Signatures
- 11- Hash Functions
- 12- Smartcards

RSA Cryptography

- Pick two prime numbers p, and q
- N = p x q → Φ (N)
- Find e, a coprime of Φ (N)
- Find d, the inverse of e
- Encrypt/decrypt

Mathematics

Euler

Euclidian

Extended Euclidian

Fermat-Euler

Fast multiply modulo

8-RSA Cryptography

- 8-1 Number theory
  - Euclidian algorithms
  - Extended Euclidian algorithms
  - Euler-Fermat theorems
- 8-2 RSA protocol
- 8-3 Fast multiply modulo

Reminder

Construction of the finite field $\mathbb{Z}_m$, with modulo m

1- The set $\mathbb{Z}_m = \{0, 1, 2, \ldots, m\text{-}1\}$

2- The two operations "+" and "x" for all of a and b $\in \mathbb{Z}_m$ are "closed":

$a + b \equiv c \bmod m$; then $c \in \mathbb{Z}_m$

$a \times b \equiv d \bmod m$; then $d \in \mathbb{Z}_m$

3- Associativity for all of a, b, c $\in \mathbb{Z}_m$:

$a + (b + c) \equiv (a + b) + c \bmod m$

$a \times (b \times c) \equiv (a \times b) \times c \bmod m$

Reminder

Construction of the finite field $\mathbb{Z}_m$, with modulo m

4- There is a "*0*" for "*+*", and "*1*" for "*x*" for all elements a of $\mathbb{Z}_m$:

$$a + 0 \equiv a \bmod m$$

$$a \times 1 \equiv a \bmod m$$

5- The negative inverse exist for the addition (not in $\mathbb{Z}_m$) such as:

$$a + (-a) \equiv 0 \bmod m$$

6- The negative inverse exist for the multiplication within $\mathbb{Z}_m$:

$$a \times a^{-1} \equiv 1 \bmod m$$

Only if $gcd(a, m) = 1$

Find gcd(m,n) greater common denominator

Prime factorization does not work for large numbers

Find $a_1, a_2, \ldots, a_f$ prime numbers of $m = a_1^{e_1}\, a_2^{e_2} \ldots a_f^{e_f}$

Find $b_1, b_2, \ldots, b_k$ prime numbers of $n = b_1^{g_1}\, b_2^{g_2} \ldots b_f^{g_k}$

$\downarrow$

gcd(m,n)

Example find

gcd(5040,2100) ; gcd(2366,1456) ; gcd(11319,7623)

Euclidian Algorithm (EA) - Method to find gcd

How to simplify prime factorization

The gcd of integers $\boldsymbol{r_0}$ and $\boldsymbol{r_1}$ is: **g= gcd ($\mathbf{r_0}$, $\mathbf{r_1}$)**

if $\mathbf{r_0 > r_1}$ $\boldsymbol{r_0 \equiv r_2 \, mod \, r_1}$ ➔ **g = gcd ($\mathbf{r_1}$, $\mathbf{r_2}$)**

Euclidian Algorithm (EA) - Method to find gcd

i	r_i	r_{i+1}	$r_{i+2} = r_i - k\,(r_{i+1})$
0	$r_0 =$	$r_1 =$	$r_2 =$
1	$r_1 =$	$r_2 =$	$r_3 =$
2	$r_2 =$	$r_3 =$	$r_4 =$
3	$r_3 =$	$r_4 =$	$r_5 =$
i	$r_i =$	$r_{i+1} = g\ r_i$	$\boldsymbol{r_{i+2} = 0}$
i+1	$\boldsymbol{r_{i+1} = g}$		
i+2	$\boldsymbol{r_{i+2} = 0}$		

When $r_{i+2} = 0$ ➔ g = r_{i+1}

Example#1 of the use of EA:
Find g = gcd (973, 301)

i	r_i	r_{i+1}	$r_{i+2} = r_i - k\,(r_{i+1})$
0	r_0 = **973**	r_1 = **301**	r_2 =
1	r_1 = **301**	r_2 =	r_3 =
2	r_2 =	r_3 =	r_4 =
3	r_3 =	r_4 =	r_5 =
4	r_4 =		
5	r_5 =		

Example#2 of the use of EA:
Find g = gcd (1131, 481)

i	r_i	r_{i+1}	$r_{i+2} = r_i - k\,(r_{i+1})$
0	r_0 = **1131**	r_1 = **481**	r_2 =
1	r_1 = **481**	r_2 =	r_3 =
2	r_2 =	r_3 =	r_4 =
3	r_3 =	r_4 =	r_5 =
4	r_4 =	r_5 =	r_6 =
5	r_5 =		
6	r_6 =		

Homework – 6: Use EA to find:

gcd(5040,2100)

gcd(2366,1456)

gcd(11319,7623)

i	r_i	r_{i+1}	$r_{i+2} = r_i - k\,(r_{i+1})$
0	$r_0 =$	$r_1 =$	$r_2 =$
1	$r_1 =$	$r_2 =$	$r_3 =$
2	$r_2 =$	$r_3 =$	$r_4 =$
3	$r_3 =$	$r_4 =$	$r_5 =$
4	$r_4 =$	$r_5 =$	$r_6 =$
5	$r_5 =$	$r_6 =$	$r_7 =$
6	$r_6 =$	$r_7 =$	$r_8 =$

RSA Cryptography

- 1- Number theory
  - Euclidian algorithms
  - Extended Euclidian algorithms
  - Euler-Fermat theorems
- 2- RSA protocol
- 3- Fast multiply modulo

Extended Euclidian Algorithm (EEA)
Diophantine equation

Assuming $g = gcd(r_0, r_1)$ with $r_0 > r_1$

it exist s and t such as

$$g = sr_0 + tr_1$$

Extended Euclidian Algorithm (EEA)
Method to find an inverse

Diophantine equation

Assuming $g = gcd(r_0, r_1)$ with $r_0 > r_1$ => it exist s and t such as $g = s\,r_0 + t\,r_1$

$$r_0 = s_0\, r_0 + t_0\, r_1 \rightarrow s_0 = 1 \quad ; t_0 = 0$$

$$r_1 = s_1\, r_0 + t_1\, r_1 \rightarrow s_1 = 0 \quad ; t_1 = 1$$

$$r_0 = q_1 r_1 + r_2 \qquad r_2 = s_2\, r_0 + t_2\, r_1 \rightarrow s_2 = 1 \quad ; t_2 = -q_1$$

$$r_1 = q_2 r_2 + r_3 \qquad r_3 = s_3\, r_0 + t_3\, r_1 \rightarrow s_3 = -q_2 \quad ; t_3 = 1 + q_1\, q_2$$

$$\boxed{r_{i-2} = q_{i-1} r_{i-1} + r_i} \quad \boxed{r_i = s_i\, r_0 + t_i\, r_1} \rightarrow \boxed{s_i = s_{i-2} - q_{i-1}\, s_{i-1}} ; \boxed{t_i = t_{i-2} - q_{i-1}\, t_{i-1}}$$

$$r_{i-1} = q_i r_i + \cancel{r_{i+1}}$$

The iteration stops when $r_{i+1} = 0 \rightarrow r_{i-1} = q_i r_i$ $\quad$ r_i divide all terms $r_0, r_1, \ldots, r_{i-1}$

$$\rightarrow r_i = g \; ; s = s_i \; ; t = t_i$$

Extended Euclidian Algorithm (EEA): how to find s and t ?

Serie i	R R_i	Q Q_i	S S_i	T T_i	Euclidian Algorithm (EA) $R_i = Q_{i+1}\, R_{i+1} + R_{i+2}$	EEA: S $S_i = S_{i-2} - Q_{i-1}\, S_{i-1}$	EEA: T $T_i = T_{i-2} - Q_{i-1}\, T_{i-1}$
0	R_0 =	-	S_0 = 1	T_0 = 0	R0 = Q1 R1 + R2		
1	R_1 =	Q_1 =	S_1 = 0	T_1 = 1	R1 = Q2 R2 + R3		
2	R_2 =	Q_2 =	S_2 =	T_2 =	R2 = Q3 R3 + R4	S2 = S0 – Q1 S1	T2 = T0 – Q1 T1
3	R_3 =	Q_3 =	S_3 =	T_3 =	R3 = Q4 R4 + R5	S3 = S1 – Q2 S2	T3 = T1 – Q2 T2
4	R_4 =	Q_4 =	S_4 =	T_4 =	R4 = Q5 R5 + R6	S4 = S2 – Q3 S3	T4 = T2 – Q3 T3
5	R_5 =	Q_5 =	S_5 =	T_5 =	R5 = Q6 R6 + R7	S5 = S3 – Q4 S4	T5 = T3 – Q4 T4
i-1	R_{i-1} =	Q_{i-1} =	S_{i-1} =	T_{i-1} =	Ri-1 = Q7 Ri + 0	Si-1 = Si-3 – Qi-2 Si-2	Ti-1 = Ti-3 – Qi-2 T1-2
i	R_i = g	Q_i =	S_i = s	$T_i = t$	**Stop**	Si = Si-2 – Qi-1 Si-1	Ti = Ti-2 – Qi-1 T1-1
i+1	ri-1 = 0						

Example: s and t for gcd(973, 301) = s(973) + t(301)

	R R_i	Q Q_i	S S_i	T T_i	Euclidian Algorithm (EA) $R_i = Q_{i+1}\, R_{i+1} + R_{i+2}$	EEA: S $S_i = S_{i-2} - Q_{i-1}\, S_{i-1}$	EEA: T $T_i = T_{i-2} - Q_{i-1}\, T_{i-1}$
0	R_0 = **973**	-	S_0 = **1**	T_0 = **0**	R0 = Q1 R1 + R2		
1	R_1 = **301**	Q_1 =	S_1 = **0**	T_1 = **1**	R1 = Q2 R2 + R3		
2	R_2 =	Q_2 =	S_2 =	T_2 =	R2 = Q3 R3 + R4	S2 = S0 – Q1 S1	T2 = T0 – Q1 T1
3	R_3 =	Q_3 =	S_3 =	T_3 =	R3 = Q4 R4 + R5	S3 = S1 – Q2 S2 **-4 = 0 – 4 x 1**	T3 = T1 – Q2 T2
4	R_4 =	Q_4 =	S_4 =	T_4 =		S4 = S2 – Q3 S3	T4 = T2 – Q3 T3
5	R_5 =						

Example: s and t for gcd(11200, 3533)= s (11200) + t (3533)

Serie i	**R** R_i	**Q** Q_i	**S** S_i	**T** T_i	**Euclidian Algorithm (EA)** $R_i = Q_{i+1} R_{i+1} + R_{i+2}$	**EEA: S** $S_i = S_{i-2} - Q_{i-1} S_{i-1}$	**EEA: T** $T_i = T_{i-2} - Q_{i-1} T_{i-1}$
0	R_0 = **11200**	-	S_0 = **1**	T_0 = **0**	R0 = Q1 R1 + R2		
1	R_1 = **3533**	Q_1 =	S_1 = **0**	t_1 = **1**	R1 = Q2 R2 + R3		
2	R_2 =	Q_2 =	S_2 =	T_2 =	R2 = Q3 R3 + R4	S2 = S0 – Q1 S1	T2 = T0 – Q1 T1
3	R_3 =	Q_3 =	S_3 =	T_3 =	R3 = Q4 R4 + R5	S3 = S1 – Q2 S2	T3 = T1 – Q2 T2
4	R_4 =	Q_4 =	S_4 =	T_4 =	R4 = Q5 R5 + R6	S4 = S2 – Q3 S3	T4 = T2 – Q3 T3
5	R_5 =	Q_5 =	S_5 =	T_5 =	R5 = Q6 R6 + R7	S5 = S3 – Q4 S4	T5 = T3 – Q4 T4
6	R_6 =	Q_6 =	S_6 =	T_6 =	R6 = Q7 R7 + R8	S6 = S4 – Q5 S5	T6 = T4 – Q5 T5
7	R_7 =	Q_7 =	S_7 =	T_7 =	R7 = Q8 R8 + 0	S7 = S5 – Q6 S6	T7 = T5 – Q6 T6
8	$\boldsymbol{r}$ **=**	Q_8 =	$\boldsymbol{s}$ =	$\boldsymbol{t}$ =		S8 = S6 – Q7 S7	T8 = T6 – Q7 T7

Homework 7: EEA: how to find s and t for gcd(12345, 3473)

Serie i	R R_i	Q Q_i	S S_i	T T_i	Euclidian Algorithm (EA) $R_i = Q_{i+1}\, R_{i+1} + R_{i+2}$	EEA: S $S_i = S_{i-2} - Q_{i-1}\, S_{i-1}$	EEA: T $T_i = T_{i-2} - Q_{i-1}\, T_{i-1}$
0	R_0 =	-	S_0 = 1	T_0 = 0	R0 = Q1 R1 + R2		
1	R_1 =	Q_1 =	S_1 = 0	T_1 = 1	R1 = Q2 R2 + R3		
2	R_2 =	Q_2 =	S_2 =	T_2 =	R2 = Q3 R3 + R4	S2 = S0 – Q1 S1	T2 = T0 – Q1 T1
3	R_3 =	Q_3 =	S_3 =	T_3 =	R3 = Q4 R4 + R5	S3 = S1 – Q2 S2	T3 = T1 – Q2 T2
4	R_4 =	Q_4 =	S_4 =	T_4 =	R4 = Q5 R5 + R6	S4 = S2 – Q3 S3	T4 = T2 – Q3 T3
5	R_5 =	Q_5 =	S_5 =	T_5 =	R5 = Q6 R6 + R7	S5 = S3 – Q4 S4	T5 = T3 – Q4 T4
6	R_6 =	Q_6 =	S_6 =	T_6 =	R6 = Q7 R7 + R8	S6 = S4 – Q5 S5	T6 = T4 – Q5 T5
7	R_7 =	Q_7 =	S_7 =	T_7 =	R7 = Q8 R8 + R9	S7 = S5 – Q6 S6	T7 = T5 – Q6 T6
8	R_8 =	Q_8 =	S_8 =	T_8 =	R8 = Q9 R9 + R10	S8 = S6 – Q7 S7	T8 = T6 – Q7 T7

RSA Cryptography

- 1- Number theory
  - Eucledian algorithms
  - Extented Eucledian algorithms
  - Euler-Fermat theorems
- 2- RSA protocol
- 3- Fast multiply modulo

Reminder

Little Fermat theorem

Theorem (little Fermat Theorem)

Let n be prime, $a < n$, a and n relatively primes, then: $\boldsymbol{a^{n-1} mod\ (n) = 1}$

Other form: $\boldsymbol{a^{n}\ \ mod\ (n)} = a$

General form: $\boldsymbol{a^{k(n-1)}\ \ mod\ (n)} = 1$

Little Fermat theorem

Example #1 :

$$8^7 \bmod 7 = ?$$

Example #2 :

$$2^{43} \bmod 43 = ?$$

Euler's phi function

For integer $n>0 \in \mathbb{Z}_n = \{0, 1, 2, \ldots, n\}$ we define:

$\phi(n)$ =number of positive integer lower than n relatively prime to n.

[Ex: n=15, $\phi(15)=8$ $(\underline{1}, \underline{2}, 3, \underline{4}, 5, 6, \underline{7}, \underline{8}, 9, 10, \underline{11}, 12, \underline{13}, \underline{14})$]

➔ If n is prime: $\phi(n)=(n-1)$

Theorem

If p, q are primes and $n=p \times q$ then: $\phi(n)=\phi(p) \times \phi(q)=(p-1) \times (q-1)$

Generalization

Assuming that n is the multiplication of m prime numbers: $n=\boldsymbol{p_1^{e_1} p_2^{e_2} \ldots p_i^{e_i} \ldots p_m^{e_m}}$

$$\Phi(n) = \prod_{i=1}^{m} (p_i^{e_i} - p_i^{e_{i-1}})$$

Euler – Fermat theorem

Theorem

Let *a<n*, a and n relatively primes, then: $\boldsymbol{a^{\phi(n)} mod(n) = 1}$

General form: $\boldsymbol{a^{k\phi(n)} mod(n) = 1}$

Euler – Fermat theorem

Example

n = 12 and a = 5 => ϕ (12) = 4

5^5 mod 12 = ?

5^{17} mod 12 = ?

RSA Cryptography

- 1- Number theory
  - Eucledian algorithms
  - Extented Eucledian algorithms
  - Euler-Fermat theorems
- 2- RSA protocol
- 3- Fast multiply modulo

RSA: Key generation & encryption

1- Prepare *n, e, d*

1- Select p, q (large numbers, typ. = 1000bits) p and q are prime, $p \neq q$

2- Calculate n: $n = p \times q$

3- Calculate $\phi(n)$: $\phi(n) = (p-1)\,(q-1)$

4- Select integer e: $gcd\,(\phi(n), e) = 1$; $e \in \{1, 2, \ldots, \phi(n) - 1\}$ Use EA

5- Calculate d : $d \times e \equiv 1 \bmod \phi(n)$ Use EEA

Public Key ➔ $Kpub = \{e, n\}$

Private Key ➔ $Kpriv = \{d, n\}$

$\phi(n)$ is kept secret

RSA: Key generation & encryption

2- Encryption & decryption

6- **Encryption** $P \in \mathbb{Z}_n$ *{0, 1, …, n-1}* - Two methods for C and C':

Plaintext: $P < n$

Cypher text C: $C = \mathbf{P^e \ mod \ n}$

Cypher text C': $C' = \mathbf{P^d \ mod \ n}$

7- **Decryption** C *or* $C' \in \mathbb{Z}_n$ *{0, 1, …, n-1}*

Cypher text: C or C'

Plaintext P: $P = \mathbf{C^d \ mod \ n}$

$P = \mathbf{C'^e \ mod \ n}$

First example of RSA :

1) $p = 3$ and $q = 11$
2) $n = 33$
3) $\phi(33) = 2 \times 10 = 20$
4) Let us pick $e = 3$
5) $d = 3^{-1} \bmod 20 = ?$

if P=4

6) $C = ?$
7) $P = ?$

Second example of RSA :

1)p = 101 and q = 113

2)n = 101 x 113 = 11413

3)ϕ (11413) = 100 x 112 = 11200

4)Let us pick e = 3533

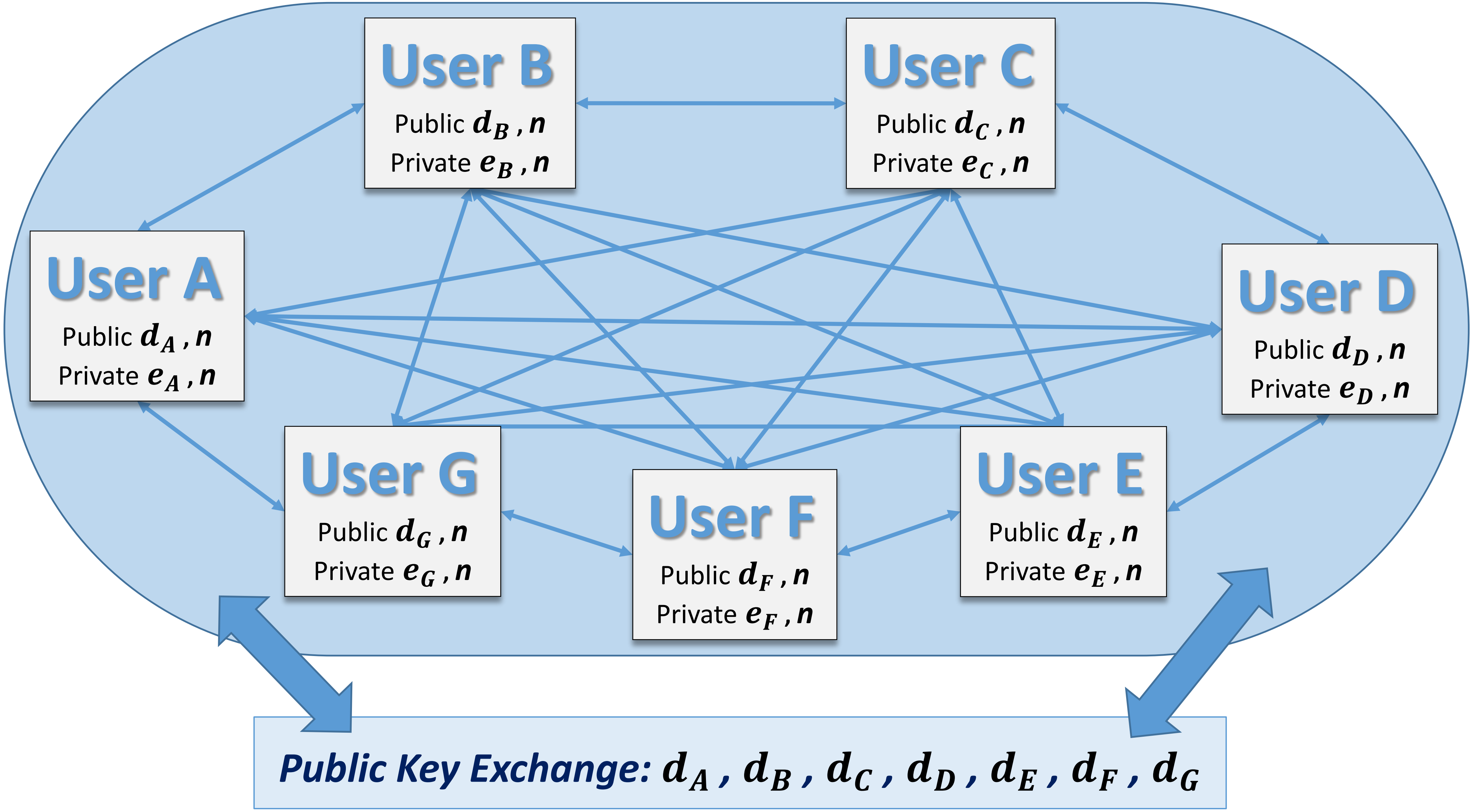
5)d = 3533^{-1} mod 11200 = ?

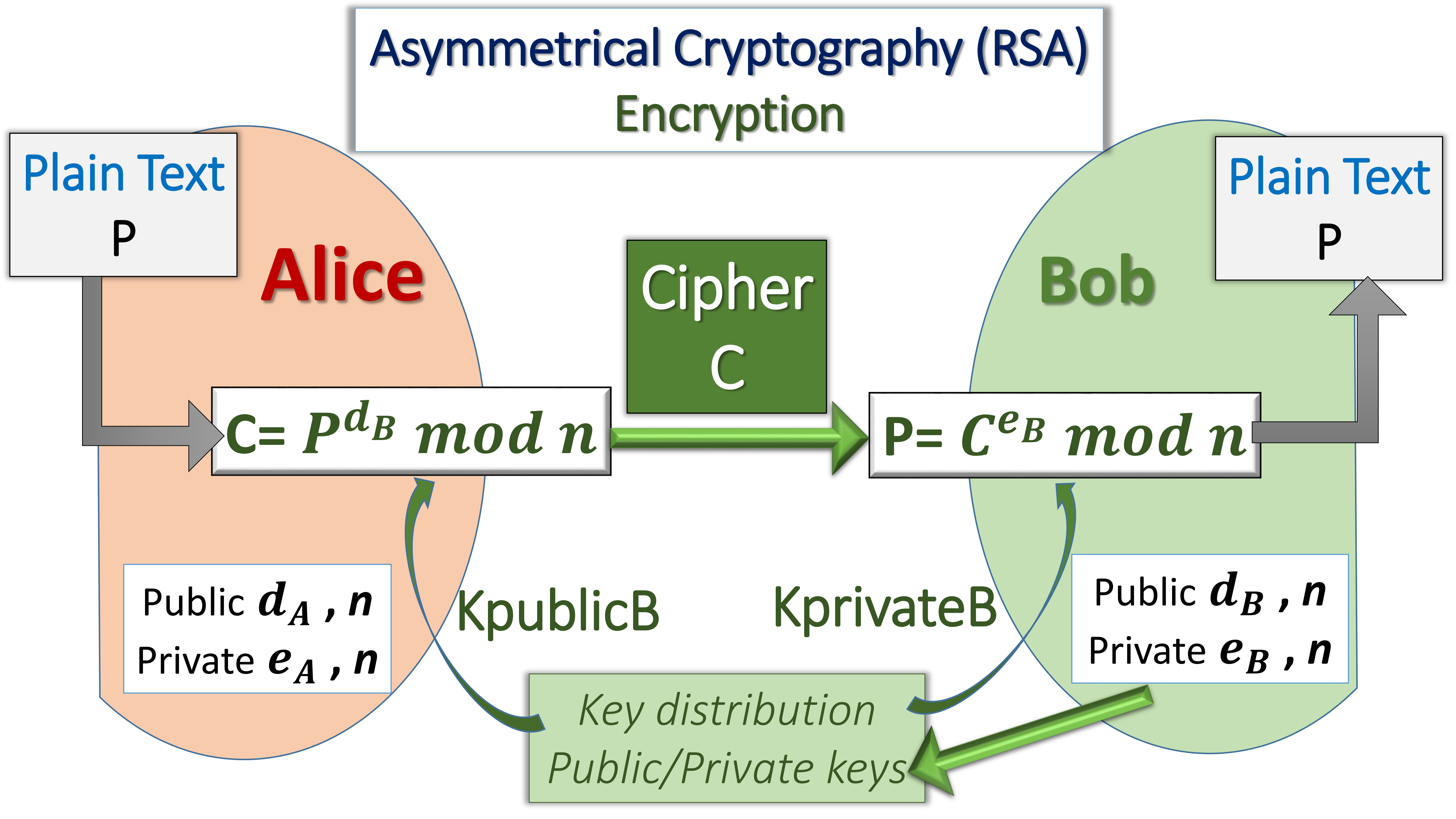
if P=9726 (use exponent modulo calculator)

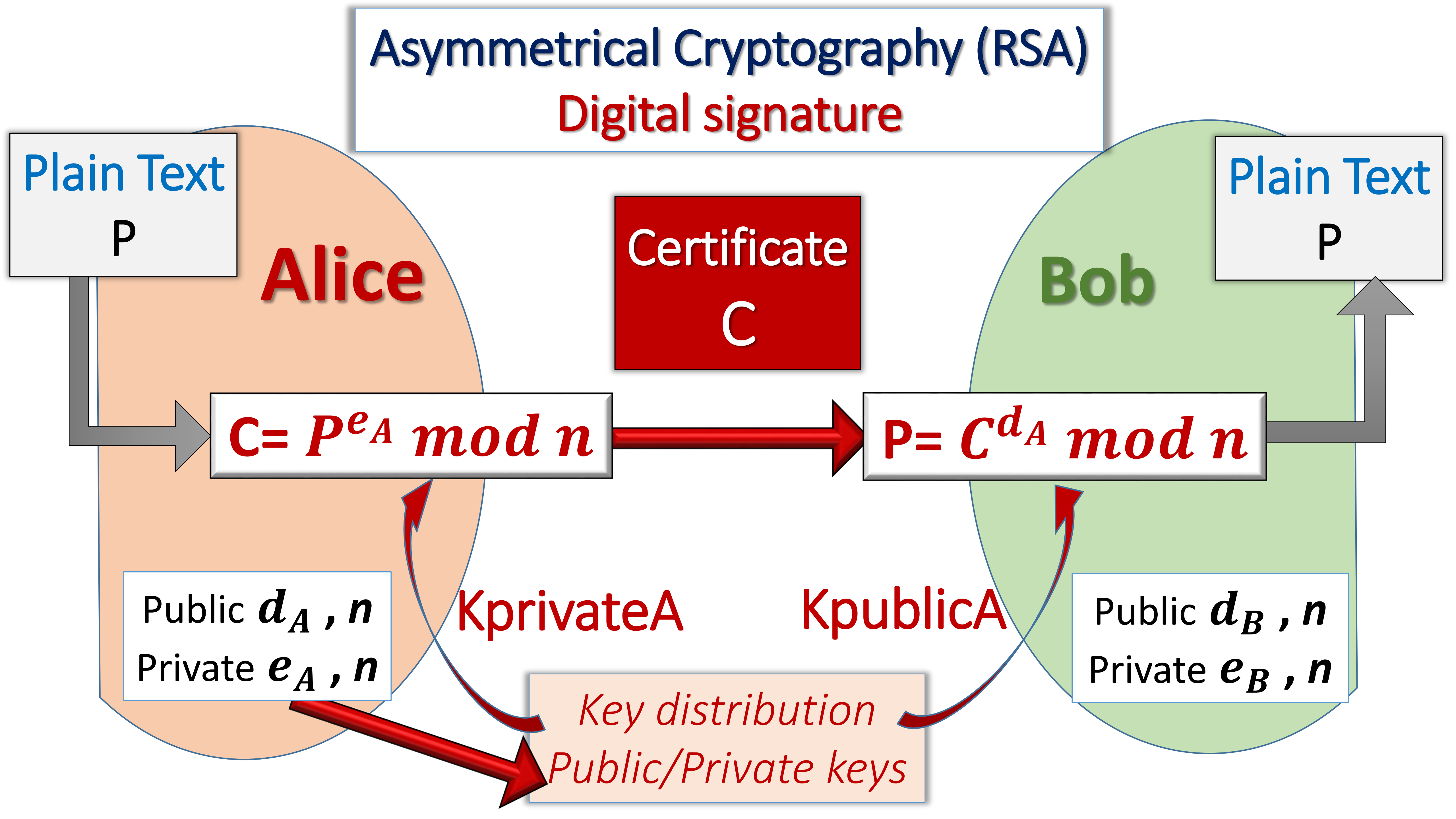
6)C = 9726^{e} mod 11413 = ?

7)P = C^{d} mod 11413 = ?

PKI with RSA Key exchange




Asymmetrical Cryptography (RSA)
Encryption
Plain Text
P
Alice
Cipher
C
Bob
Plain Text
P
C= $P^{d_B} \; mod \; n$
P= $C^{e_B} \; mod \; n$
Public d_A , n
Private e_A , n
KpublicB
KprivateB
Public d_B , n
Private e_B , n
Key distribution
Public/Private keys


Asymmetrical Cryptography (RSA)
Digital signature
Plain Text
P
Alice
Certificate
C
Bob
Plain Text
P
$C = P^{e_A} \bmod n$
$P = C^{d_A} \bmod n$
Public d_A , n
Private e_A , n
KprivateA
KpublicA
Public d_B , n
Private e_B , n
Key distribution
Public/Private keys

NORTHERN ARIZONA UNIVERSITY

RSA Cryptography

- 1- Number theory
  - Eucledian algorithms
  - Extented Eucledian algorithms
  - Euler-Fermat theorems
- 2- RSA protocol
- 3- Fast multiply modulo

Fast Exponentiation algorithm: square-multiply

X^8 : 7 multiplications vs 3

X^1x X = X^2

X^2x X = X^3

X^3x X = X^4

X^4x X = X^5

X^5x X = X^6

X^6 x X = X^7

X^7 x X = X^8

X^1x X^1 = X^{10}

X^{10}x X^{10} = X^{100}

X^{100}x X^{100}= X^{1000}

8 = 1000

X^{26} : 25 multiplications vs 6

X^1x X^1 = X^{10}

X^{10}x X^1 = X^{11}

X^{11}x X^{11} = X^{110}

X^{110}x X^{110} = X^{1100}

X^{1100}x X^1 = X^{1101}

X^{1101}x X^{1101}= X^{11010}

1

1 **Square:** add 0 on the right

Mult : add 1 on the right

0 **Square:** add 0 on the right

1 **Square:** add 0 on the right

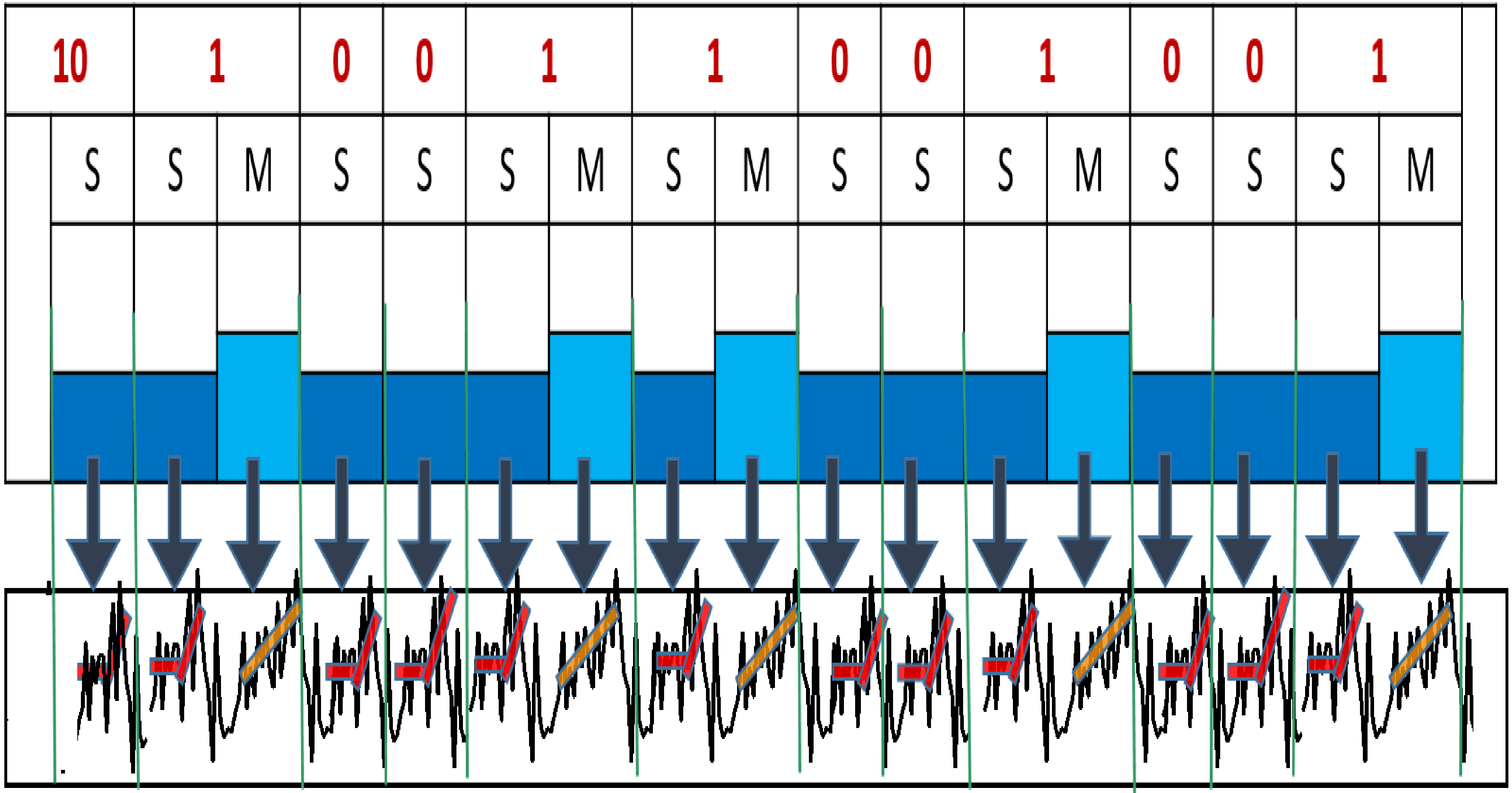
Mult : add 1 on the right

0 **Square:** add 0 0n the right

26 = 11010

Fast Exponentiation – power analysis

5321 = 1010011001001



Optional Homework 7A: Fast Exponentiation 3473

QUESTIONS ?

Dr. Bertrand Cambou
Professor of Practice NAU, Cybersecurity
School of Informatics, Computing, and Cyber-Systems
Bertrand.cambou@nau.edu