RSA

Presentation by:
V. Balasubramanian
SSN College of Engineering

Objectives

1. How RSA works
2. Practical aspects of RSA, such as computation of the parameters, and fast encryption and decryption
3. Security estimations
4. Implementational aspects

Rivest Shamir Adelman

Introduction

- After Whitfield Diffie and Martin Hellman introduced public-key cryptography in their landmark 1976 paper, a new branch of cryptography suddenly opened up.
- As a consequence, cryptologists started looking for methods with which public key encryption could be realized.
- In 1977, Ronald Rivest, Adi Shamir and Leonard Adleman proposed a scheme which became the most widely used asymmetric cryptographic scheme, RSA

Algebraic Structures

Two Algebraic Structures

RSA uses two algebraic structures: a ring and a group.

Encryption/Decryption Ring Encryption and decryption are done using the commutative ring $\mathbf{R} = \langle \mathbf{Z}_n, +, \times \rangle$ with two arithmetic operations: addition and multiplication. In RSA, this ring is public because the modulus n is public. Anyone can send a message to Bob using this ring to do encryption.

Key-Generation Group RSA uses a multiplicative group $\mathbf{G} = \langle \mathbf{Z}_{\phi(n)}^*, \times \rangle$ for key generation. This group supports only multiplication and division (using multiplicative inverses), which are needed for generating public and private keys. This group is hidden from the public because its modulus, $\phi(n)$, is hidden from the public. We will see shortly that if Eve can find this modulus, she can easily attack the cryptosystem.

RSA uses two algebraic structures:
a public ring $\mathbf{R} = \langle \mathbf{Z}_n, +, \times \rangle$ and a private group $\mathbf{G} = \langle \mathbf{Z}_{\phi(n)}^*, \times \rangle$.

Key Generation

RSA_Key_Generation

{

Select two large primes p and q such that $p \neq q$.

$n \leftarrow p \times q$

$\phi(n) \leftarrow (p-1) \times (q-1)$

Select e such that $1 < e < \phi(n)$ and e is coprime to $\phi(n)$

$d \leftarrow e^{-1} \bmod \phi(n)$ // d is inverse of e modulo $\phi(n)$

Public_key $\leftarrow$ (e, n) // To be announced publicly

Private_key $\leftarrow d$ // To be kept secret

return Public_key and Private_key

}

Encryption

```
RSA_Encryption (P, e, n)                    // P is the plaintext in Z_n and P < n
{
   C ← Fast_Exponentiation (P, e, n)        // Calculation of (P^e mod n)
   return C
}
```

RSA Encryption Given the public key $(n, e) = k_{pub}$ and the plaintext x, the encryption function is:

$$y = e_{k_{pub}}(x) \equiv x^e \bmod n \tag{7.1}$$

where $x, y \in \mathbb{Z}_n$.

Decryption

```
RSA_Decryption (C, d, n)                    //C is the ciphertext in Z_n
{
    P ← Fast_Exponentiation (C, d, n)       // Calculation of (C^d mod n)
    return P
}
```

In RSA, p and q must be at least 512 bits; n must be at least 1024 bits.

RSA Decryption Given the private key $d = k_{pr}$ and the ciphertext y, the decryption function is:

$$x = d_{k_{pr}}(y) \equiv y^d \bmod n \qquad (7.2)$$

where $x, y \in \mathbb{Z}_n$.

Proof

Proof of RSA

We can prove that encryption and decryption are inverses of each other using the second version of Euler's theorem discussed in Chapter 9:

If $n = p \times q$, $a < n$, and k is an integer, then $a^{k \times \phi(n)+1} \equiv a \pmod{n}$.

Assume that the plaintext retrieved by Bob is P_1 and prove that it is equal to P.

$$P_1 = C^d \bmod n = (P^e \bmod n)^d \bmod n = P^{ed} \bmod n$$
$$ed = k\phi(n) + 1 \qquad \text{// } d \text{ and } e \text{ are inverses modulo } \phi(n)$$
$$P_1 = P^{ed} \bmod n \rightarrow P_1 = P^{k\phi(n)+1} \bmod n$$
$$P_1 = P^{k\phi(n)+1} \bmod n = P \bmod n \qquad \text{// Euler's theorem (second version)}$$

Example

Example 10.5

Bob chooses 7 and 11 as p and q and calculates $n = 7 \times 11 = 77$. The value of $\phi(n) = (7 - 1)(11 - 1)$ or 60. Now he chooses two exponents, e and d, from $\mathbf{Z}_{60}^*$. If he chooses e to be 13, then d is 37. Note that $e \times d \bmod 60 = 1$ (they are inverses of each other). Now imagine that Alice wants to send the plaintext 5 to Bob. She uses the public exponent 13 to encrypt 5.

Plaintext: 5	$C = 5^{13} = 26 \bmod 77$	Ciphertext: 26

Bob receives the ciphertext 26 and uses the private key 37 to decipher the ciphertext:

Ciphertext: 26	$P = 26^{37} = 5 \bmod 77$	Plaintext: 5

The plaintext 5 sent by Alice is received as plaintext 5 by Bob.

Illustration

Alice

message $x = 4$

Bob

1. choose $p = 3$ and $q = 11$
2. $n = p \cdot q = 33$
3. $\Phi(n) = (3-1)(11-1) = 20$
4. choose $e = 3$
5. $d \equiv e^{-1} \equiv 7 \bmod 20$

$k_{pub}=(33,3)$ ←

$y = x^e \equiv 4^3 \equiv 31 \bmod 33$

$y=31$ →

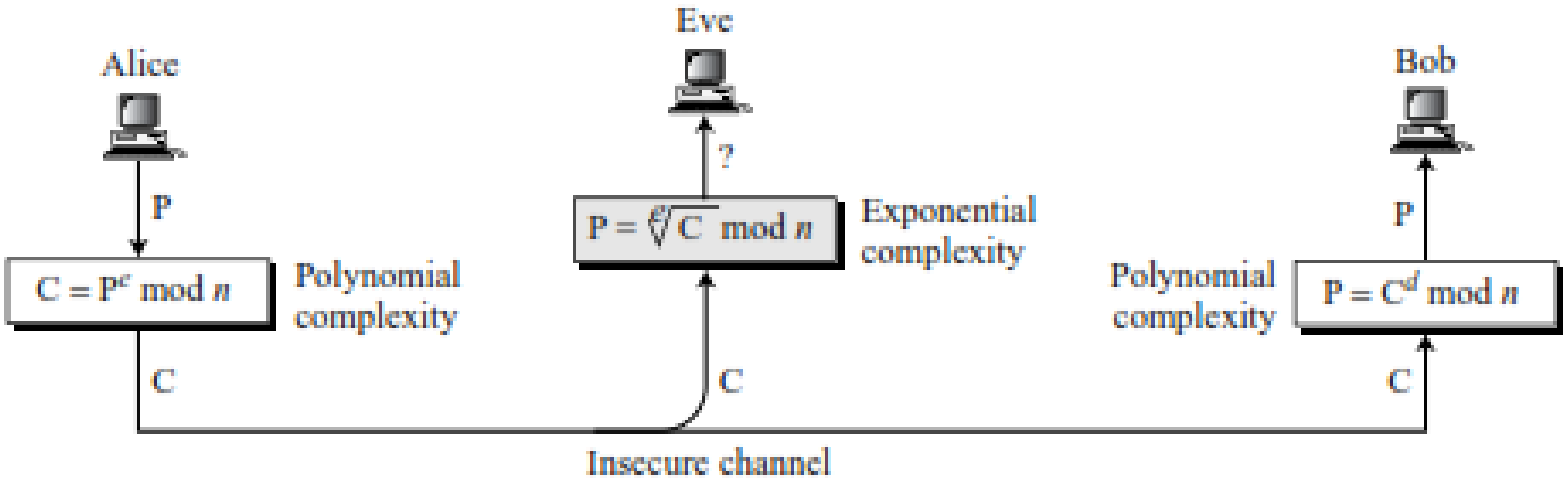
$y^d = 31^7 \equiv 4 = x \bmod 33$

Fast Exponentiation

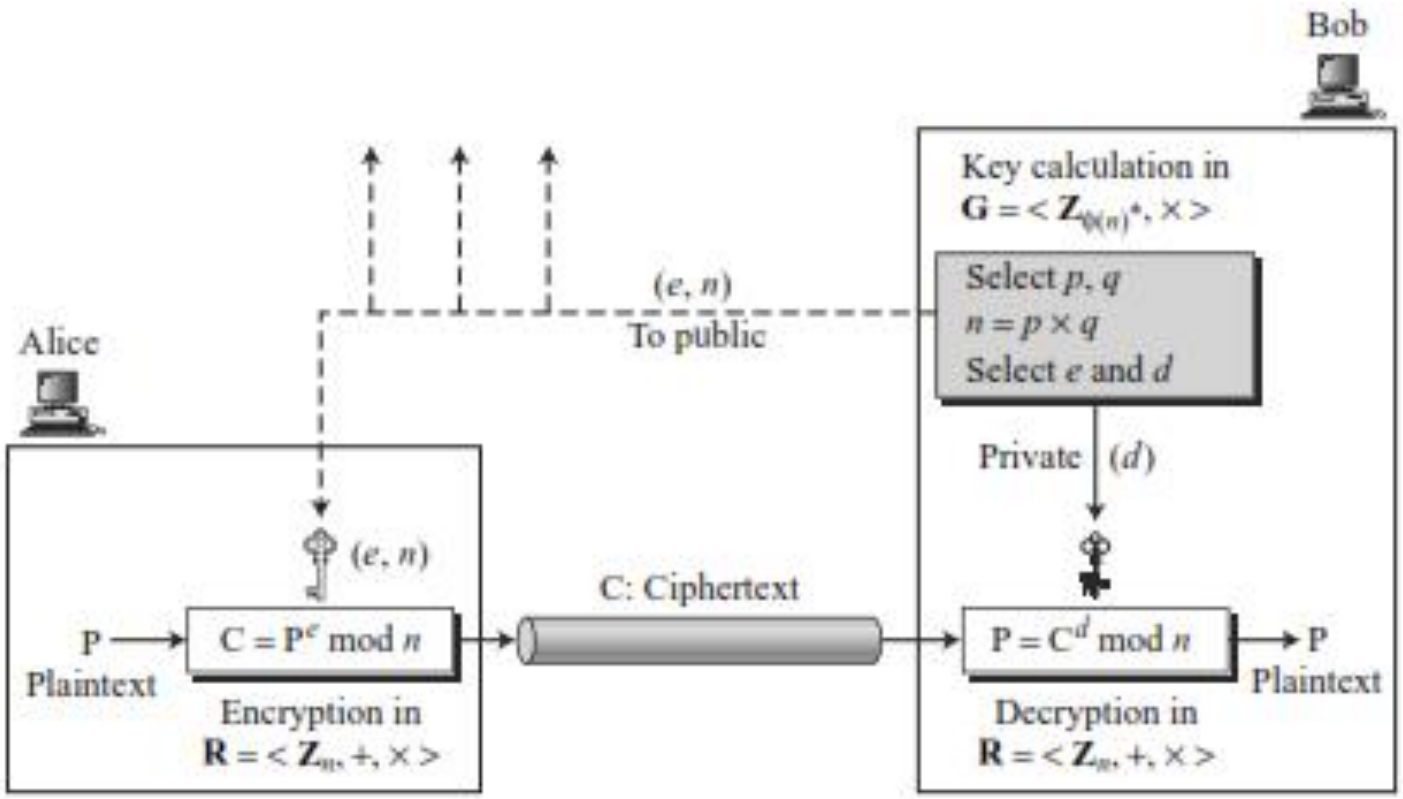
Square-and-Multiply for Modular Exponentiation
Input:
base element x
exponent $H = \sum_{i=0}^{t} h_i 2^i$ with $h_i \in 0, 1$ and $h_t = 1$
and modulus n
Output: $x^H \bmod n$
Initialization: $r = x$
Algorithm:

```
1    FOR i = t − 1 DOWNTO 0
1.1      r = r^2 mod n
         IF h_i = 1
1.2          r = r · x mod n
2    RETURN (r)
```

Complexity


Alice
Eve
Bob
P
?
P
$C = P^e \bmod n$
Polynomial complexity
$P = \sqrt[e]{C} \bmod n$
Exponential complexity
Polynomial complexity
$P = C^d \bmod n$
C
C
C
Insecure channel

RSA Encryption, Decryption, Key Generation



Example

Now assume that another person, John, wants to send a message to Bob. John can use the same public key announced by Bob (probably on his website), 13; John's plaintext is 63. John calculates the following:

Plaintext: 63	$C = 63^{13} = 28 \bmod 77$	Ciphertext: 28

Bob receives the ciphertext 28 and uses his private key 37 to decipher the ciphertext:

Ciphertext: 28	$P = 28^{37} = 63 \bmod 77$	Plaintext: 63

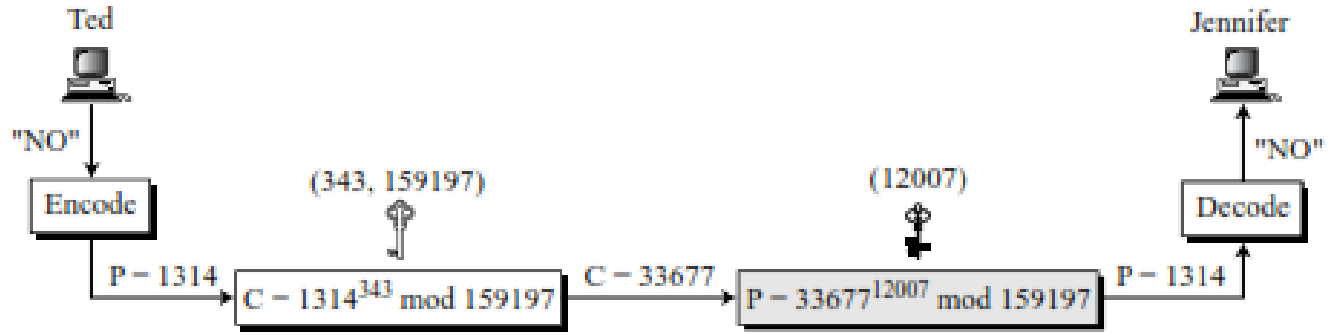
Example

Jennifer creates a pair of keys for herself. She chooses $p = 397$ and $q = 401$. She calculates $n = 397 \times 401 = 159197$. She then calculates $\phi(n) = 396 \times 400 = 158400$. She then chooses $e = 343$ and $d = 12007$. Show how Ted can send a message to Jennifer if he knows e and n.

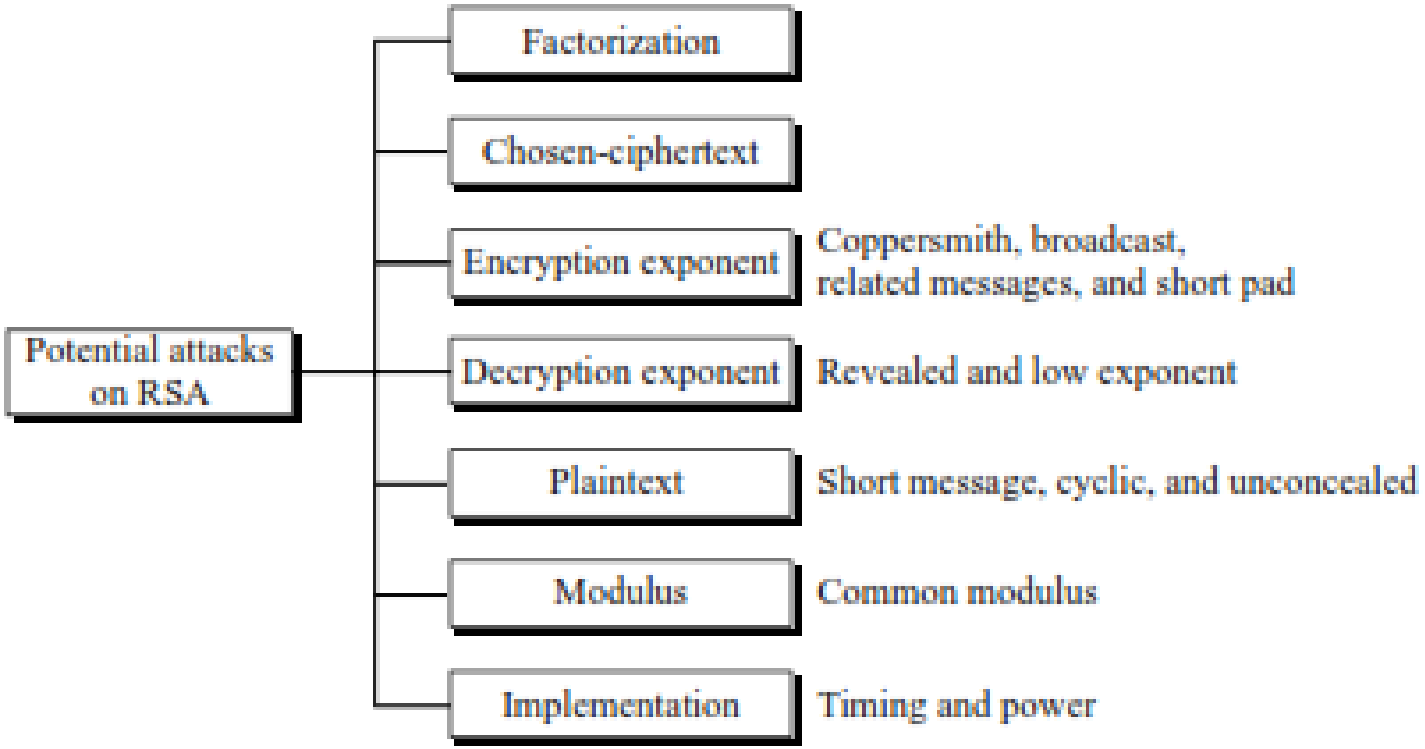
Solution

Suppose Ted wants to send the message "NO" to Jennifer. He changes each character to a number (from 00 to 25), with each character coded as two digits. He then concatenates the two coded characters and gets a four-digit number. The plaintext is 1314. Ted then uses e and n to encrypt the message. The ciphertext is $1314^{343} = 33677 \bmod 159197$. Jennifer receives the message 33677 and uses the decryption key d to decipher it as $33677^{12007} = 1314 \bmod 159197$. Jennifer then decodes 1314 as the message "NO". Figure 10.7 shows the process.

Figure 10.7 *Encryption and decryption in Example 10.7*



RSA Attacks


Potential attacks on RSA
Factorization
Chosen-ciphertext
Encryption exponent
Coppersmith, broadcast, related messages, and short pad
Decryption exponent
Revealed and low exponent
Plaintext
Short message, cyclic, and unconcealed
Modulus
Common modulus
Implementation
Timing and power

Factorization Attack

Factorization Attack

The security of RSA is based on the idea that the modulus is so large that it is infeasible to factor it in a reasonable time. Bob selects p and q and calculates $n = p \times q$. Although n is public, p and q are secret. If Eve can factor n and obtain p and q, she can calculate $\phi(n) = (p - 1)(q - 1)$. Eve then can calculate $d = e^{-1} \bmod \phi(n)$ because e is public. The private exponent d is the trapdoor that Eve can use to decrypt any encrypted message.

As we learned in Chapter 9, there are many factorization algorithms, but none of them can factor a large integer with polynomial time complexity. To be secure, RSA presently requires that n should be more than 300 decimal digits, which means that the modulus must be at least 1024 bits. Even using the largest and fastest computer available today, factoring an integer of this size would take an infeasibly long period of time. This means that RSA is secure as long as an efficient algorithm for factorization has not been found.

Chosen Ciphertext attack

Chosen-Ciphertext Attack

A potential attack on RSA is based on the multiplicative property of RSA. Assume that Alice creates the ciphertext $C = P^e \bmod n$ and sends C to Bob. Also assume that Bob will decrypt an arbitrary ciphertext for Eve, other than C. Eve intercepts C and uses the following steps to find P:

a. Eve chooses a random integer X in Z_n^*.
b. Eve calculates $Y = C \times X^e \bmod n$.
c. Eve sends Y to Bob for decryption and get $Z = Y^d \bmod n$; This step is an instance of a chosen-ciphertext attack.
d. Eve can easily find P because

$$Z = Y^d \bmod n = (C \times X^e)^d \bmod n = (C^d \times X^{ed}) \bmod n = (C^d \times X) \bmod n = (P \times X) \bmod n$$
$$Z = (P \times X) \bmod n \quad \rightarrow \quad P = Z \times X^{-1} \bmod n$$

Attacks on e (Cppersmith Theorem attack)

Attacks on the Encryption Exponent

To reduce the encryption time, it is tempting to use a small encryption exponent e. The common value for e is $e = 3$ (the second prime). However, there are some potential attacks on low encryption exponent that we briefly discuss here. These attacks do not generally result in a breakdown of the system, but they still need to be prevented. To thwart these kinds of attacks, the recommendation is to use $e = 2^{16} + 1 = 65537$ (or a prime close to this value).

Attacks on d

Revealed Decryption Exponent Attack It is obvious that if Eve can find the decryption exponent, d, she can decrypt the current encrypted message. However, the attack does not stop here. If Eve knows the value of d, she can use a probabilistic algorithm (not discussed here) to factor n and find the value of p and q. Consequently, if Bob changes only the compromised decryption exponent but keeps the same modulus, n, Eve will be able to decrypt future messages because she has the factorization of n. This means that if Bob finds out that the decryption exponent is compromised, he needs to choose new value for p and q, calculate n, and create totally new private and public keys.

In RSA, if d is comprised, then p, q, n, e, and d must be regenerated.

Small d Values

Low Decryption Exponent Attack Bob may think that using a small private-key d, would make the decryption process faster for him. Wiener showed that if $d < 1/3\ n^{1/4}$, a special type of attack based on *continuous fraction*, a topic discussed in number theory, can jeopardize the security of RSA. For this to happen, it must be the case that $q < p < 2q$. If these two conditions exist, Eve can factor n in polynomial time.

In RSA, the recommendation is to have $d \geq 1/3\ n^{1/4}$ to prevent low decryption exponent attack.

Short Message attack

Short Message Attack In the **short message attack,** if Eve knows the set of possible plaintexts, she then knows one more piece of information in addition to the fact that the ciphertext is the permutation of plaintext. Eve can encrypt all of the possible messages until the result is the same as the ciphertext intercepted. For example, if it is known that Alice is sending a four-digit number to Bob, Eve can easily try plaintext numbers from 0000 to 9999 to find the plaintext. For this reason, short messages must be padded with random bits at the front and the end to thwart this type of attack. It is strongly recommended that messages be padded with random bits before encryption using a method called OAEP, which is discussed later in this chapter.

Timing Attack

Power Attack The **Power attack** is similar to the timing attack. Kocher showed that if Eve can precisely measure the power consumed during decryption, she can launch a power attack based on the principle discussed for timing attack. An iteration involving multiplication and squaring consumes more power than an iteration that uses only squaring. The same kind of techniques used to prevent timing attacks can be used to thwart power attacks.

Fast Decryption

Example 7.6. Let the RSA parameters be given by:

$$\begin{aligned} p &= 11 & e &= 7 \\ q &= 13 & d &\equiv e^{-1} \equiv 103 \bmod 120 \\ n &= p \cdot q = 143 \end{aligned}$$

We now compute an RSA decryption for the ciphertext $y = 15$ using the CRT, i.e., the value $y^d = 15^{103} \bmod 143$. In the first step, we compute the modular representation of y:

$$\begin{aligned} y_p &\equiv 15 \equiv 4 \bmod 11 \\ y_p &\equiv 15 \equiv 2 \bmod 13 \end{aligned}$$

In the second step, we perform the exponentiation in the transform domain with the short exponents. These are:

$$\begin{aligned} d_p &\equiv 103 \equiv 3 \bmod 10 \\ d_q &\equiv 103 \equiv 7 \bmod 12 \end{aligned}$$

Here are the exponentiations:

$$\begin{aligned} x_p &\equiv y_p^{d_p} = 4^3 = 64 \equiv 9 \bmod 11 \\ x_q &\equiv y_q^{d_q} = 2^7 = 128 \equiv 11 \bmod 13 \end{aligned}$$

Fast Decryption

$$c_p = 13^{-1} \equiv 2^{-1} \equiv 6 \bmod 11 \qquad c_q = 11^{-1} \equiv 6 \bmod 13$$

The plaintext x follows now as:

$$x \equiv [qc_p]x_p + [pc_q]x_q \bmod n$$
$$x \equiv [13 \cdot 6]9 + [11 \cdot 6]11 \bmod 143$$
$$x \equiv 702 + 726 = 1428 \equiv 141 \bmod 143$$

RSA Factoring Attacks

Decimal digits	Bit length	Date
100	330	April 1991
110	364	April 1992
120	397	June 1993
129	426	April 1994
140	463	February 1999
155	512	August 1999
200	664	May 2005

RSA algorithm summary

Key Generation by Alice	
Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calcuate $\phi(n) = (p - 1)(q - 1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Public Key	
Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

Openssl

==> openssl genrsa

```
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
..........+++++
e is 65537 (0x010001)
-----BEGIN RSA PRIVATE KEY-----
MIIEpgIBAAKCAQEA7Nk5LpuPA1ZiWqCHfwVilholwLTkb2hTwX4iOarC8fEfWMLk
heqfvi7Aik7PZOKtLVl4q8jebjqxBm2hnow+TcBfPCGiJGlzFn9ABGO122qtJ20e
YsjhB7kronG/DG82b/pc7ostT3ZBpjEykt8ytTPBxAt5+pFclXzj65eP4ERki25w
RMqCzFKWzc/q++fWfun053m+T26HeP4zCHMkPV9vTIJuI+dtuJ0Xal9k8kf61KBU
7wFDuDbo3rfs7dk3Nvlb68/IxSo2aLl8vMhJ5SlbM6F4ifUcZqfh9cDpqfw0Lljl
xaI9nJkbQzJ5jnqDncSAVtERHNSu4zOxsXxLkQIDAQABAoIBAQCMkLL2LU5oLal7
ndAsm5a3+JalUuZMFD+5E1HepbDDBFaVSD12GPQrW/XbX8CIYFtlEbejRDh2dRDA
/umvfg4v+N7Mgi2HrKmoDWeB82dnzFztjD4/ZHbhLYlvAFDhYVOOi6kBzcnbYhS+
PG6Gaj7e+dOitSj0g7bN5WwjUzUJZIlG/VKKWnRGZtstXX5jm0wQew6PW561tv8Y
s/J2AAab/2THcHQQ5tqsaCwyCLimSBvE7DoYxcf9LKIpTEuZOpA5Bz66PJdgnYzY
EXutIw8DU1iA6Uw5KZHaAl0ByAitpCezfThLjJkSkVP5fKRrsek9sDRIygtVa3Qa
nEEVwMjdAoGBAP+sKcVtLOqtRMcEzPqksdlgNX1Dxy99Hb+3VmVrJuFukssEDGKM
BPaaE5cCzSDRZJ4p7Bt+Xpp4gAMDnA2a4VWEMTCbVdvEgOTxSGBPvLpYqcpt3cQA
gLC2TLWlVgWDi08BJdLiwBBnBSajT72RveUeMyFyh3QLgWrvRvIHHiDXAoGBAO0m
4z8NymYt6+KYN0yBWYfBjuXlOZVjVAURcKcrnQrQ6zB/KT7IzR+NkSA/JwmCv454
pIbPLyAYtvqdi8wEIwClROvB0dra6kVuUTlKLSjy6wvKqlNEibFptJf42JtvsqHZ
y3NzH/Zw25ha7jGilwfgPR3vjl+cMlA6Ph19RSHXAoGBAPxrPiZDnmfnPxL6I2GF
vnDgbo7Intu8u+Uunaatfopsf5Ld4VheAvxwq8yYoGq5MIySuR9/yOjbHI0lQBmS
gsvKIkJp6f96ZwMhUCJ+NscHiEJp69t535QXt25S2LXC5IPQj0ZARf0rqMM30x9G
x2NwSGzKRP8F6PTpXXLQIierAoGBANtoBAy9HImmaLtuqpK8hXGFEUj+52SfzgcW
WIxBXHy6Ry3KKAWvT5+moSMdamdxMPqGAWm721StqPR6t/DbzuqDyqz318jMirwL
0VfYmalt/SoeqplSJrYeHvgPQY+lKrZlQragR20AgxoU3pTLYUHnI4RDs/j4ENCK
4hb0Y/ZnAoGBAIoAitJHM6tCypiQB+wMvBjmTeV90smglGoq20a7MiwKMHlidxxI
YcY5AVMCeahOxpIA7cNMWTMvgZru+rQblLXCWD/tZ3Gd9MQzf0sunxWKJhUV/Mk0
mPX/HkBJaXu2kN5gkFXAHCs9E8SJLWy6FG4tKox+Uw9wo0GlBr+WVAHQ
-----END RSA PRIVATE KEY-----
```

Private Key

```
==> openssl genrsa -out key.pem 2048

Generating RSA private key, 2048 bit long modulus (2 primes)
.....................................................................+++++
...+++++
e is 65537 (0x010001)
```

Internals

```
==> openssl rsa -in key.pem -inform pem -noout -text

RSA Private-Key: (2048 bit, 2 primes)
modulus:
    00:ab:af:a4:ed:1f:82:f6:99:f0:93:cb:57:bb:cd:
    85:08:6d:da:30:4c:3c:a6:25:94:4b:ea:2e:71:45:
    0c:ca:62:8e:a4:d1:fe:cc:4a:e2:c8:18:fd:ad:23:
    5f:e6:a3:f0:b4:57:a9:6f:e9:38:ca:e4:42:af:ed:
    73:b2:15:3a:04:1a:40:f6:e7:40:4c:69:53:25:95:
    ca:9d:44:6e:05:56:cf:7b:b9:d6:57:9f:58:61:01:
    00:c1:fe:fe:7e:10:96:8f:2c:d8:16:33:39:9e:37:
    4f:c3:9b:77:01:b9:70:ea:10:c7:71:81:29:19:25:
    50:25:71:12:f5:2c:31:8d:61:26:39:09:6c:98:cd:
    78:00:aa:6a:cf:0c:5c:27:07:89:0c:d1:06:5a:91:
    92:6f:6f:2e:24:46:f0:99:a3:77:29:ec:d2:38:4d:
    91:aa:fb:b6:7f:5e:3d:b4:eb:3e:7b:96:0e:3a:a9:
    8f:48:5b:0b:e4:5b:61:ba:dd:65:a3:e9:33:14:9e:
    86:94:01:e3:69:83:f0:78:9c:3e:28:32:b8:58:19:
    ba:2c:f2:9b:2d:2a:7b:e7:ec:58:1e:b4:fc:5a:39:
    24:ae:0a:8e:26:15:ce:19:f0:b0:d7:b0:c3:90:03:
    38:66:4a:5b:ba:a8:f1:d6:d3:94:98:fd:bc:4d:67:
    c2:61
publicExponent: 65537 (0x10001)
privateExponent:
    55:f8:59:1c:c8:07:bb:56:70:6a:81:8b:48:26:6c:
    b4:40:d5:de:13:7e:d7:2f:c0:27:97:77:74:0e:c0:
    8d:e3:76:4c:40:3f:57:ab:34:0e:40:bd:5e:62:75:
    56:37:c7:83:76:d6:08:8c:ff:7c:51:7a:b7:3f:af:
    0c:80:a6:91:81:58:00:8a:e1:de:a1:6b:1a:49:fc:
    b0:6d:a0:ae:19:bf:41:d4:57:e9:7e:88:31:e2:df:
    af:44:f1:c8:cc:a3:a7:c4:2b:dc:4a:00:53:22:9d:
    55:74:d6:cd:cd:3f:26:66:0a:88:e2:c5:62:ab:15:
    8b:fa:28:25:0e:e1:2d:4a:a3:71:67:ae:d5:4c:b9:
    af:ee:da:65:fd:0c:44:18:f4:b1:e6:bc:b0:c5:17:
    a9:20:a9:52:35:a0:63:c4:77:3d:7b:f7:4d:9d:c9:
    1c:dc:d0:83:7a:44:aa:f9:d5:66:79:c4:e0:b4:57:
    3a:ba:9f:9a:eb:f4:04:5e:c5:d5:c7:3c:d7:11:a2:
    74:44:22:1f:b0:69:4f:8e:43:f2:e8:8d:e1:ec:c8:
    96:3a:8d:11:60:be:48:58:17:51:ca:b5:60:ae:f3:
    4d:f5:a3:1d:85:7a:f6:4b:f2:f5:ce:0a:76:a8:a4:
    99:f0:7e:06:cc:52:e5:f0:31:9f:45:98:7a:81:00:
    85
prime1:
    00:d3:e3:2f:f5:7b:13:51:1b:bc:c1:b2:a3:77:b0:
    0a:25:41:cf:64:89:15:22:b6:cf:9f:84:27:10:df:
    f3:c7:d1:be:9c:8e:42:df:fe:f1:b0:03:2e:d3:d1:
    99:b6:52:67:d7:ee:98:01:43:0d:99:0c:e2:76:a1:
    dc:f6:88:57:a6:28:c2:89:67:ea:d1:5a:28:75:70:
    96:8e:3a:0e:0a:33:b2:8e:53:c5:06:58:c1:18:20:
    81:99:54:d0:45:7a:21:8f:b8:10:c9:ce:72:82:91:
    56:45:ed:2c:e8:cc:0d:b3:6e:30:78:5d:35:fd:79:
    bd:68:99:2f:b0:08:93:df:cf
prime2:
    00:cf:6d:e0:04:21:c7:7b:d1:1a:2b:68:d8:bd:23:
    c0:cf:66:19:65:89:8c:68:be:da:87:27:3e:40:e5:
    b3:20:03:26:59:19:4f:eb:b6:86:28:36:b0:74:32:
    e2:12:13:c4:12:38:ee:71:98:0b:91:64:af:be:3b:
    87:a2:f0:9f:d8:36:6c:7e:ce:a9:a9:29:de:42:f7:
    8c:d7:2e:c4:ef:36:88:e5:e9:ec:e4:ff:dd:22:26:
    5d:a6:79:6f:75:6c:5d:3a:52:da:54:3a:3d:5b:96:
    e7:1f:0a:0d:c7:6d:f6:3a:2d:91:3b:bf:08:e5:92:
    39:a5:4c:70:46:aa:46:16:cf
exponent1:
    0f:b0:b2:1b:76:7a:ae:b5:e4:1b:5f:d4:15:07:d7:
    28:7d:20:13:6c:c7:40:e3:d2:aa:18:4a:20:48:c5:
    2f:95:cb:8c:a2:48:37:78:14:83:99:28:bd:8c:b6:
    da:36:6d:f4:22:79:e5:16:07:0a:bf:56:81:bc:68:
    b5:64:d1:40:bf:a1:f0:34:de:c1:93:f0:8d:09:c2:
    4c:53:e6:38:41:2d:c6:b6:53:4f:ae:00:d6:7d:89:
    bb:45:f9:8a:3b:8a:02:af:79:a6:c7:ff:d8:c5:54:
    63:27:35:fd:23:27:1c:93:5b:49:7e:75:82:08:a2:
    ca:fd:14:f7:ef:1a:ac:27
exponent2:
    00:93:39:f5:6a:79:5f:51:6e:95:18:82:8e:73:90:
    d0:e5:64:1e:5a:87:4a:75:7e:21:35:14:91:87:16:
    82:11:12:ab:41:4a:4a:03:8f:c5:a0:fd:50:38:e9:
    74:b4:47:fb:3e:c3:d1:da:26:84:ef:69:7b:a3:96:
    35:2b:5d:86:d6:bb:aa:3e:47:08:fc:dc:8e:b9:11:
    63:91:c7:cc:57:cd:69:55:66:b7:91:c2:59:7e:47:
    a4:e8:e8:00:48:63:e5:b7:e3:de:bb:31:ab:23:3a:
    f4:48:7f:a6:50:0a:a8:5d:9a:c2:1e:99:f5:02:9c:
    ca:f5:9c:4f:84:98:8e:ae:d9
coefficient:
    39:04:e4:6f:11:0e:b9:51:62:b1:d4:48:2c:93:56:
    67:fb:c6:78:24:dd:e4:e7:8c:32:15:7c:30:53:a5:
    82:7c:66:00:81:b4:1e:e6:a0:b1:33:45:a8:63:63:
    1c:b6:e9:cf:55:ec:b2:4e:7c:31:a0:3f:4e:cf:3c:
    57:df:9e:3e:31:a4:c3:e1:41:a4:00:f1:87:34:c1:
    17:84:71:56:c7:13:d1:e0:e2:c2:c9:da:ae:61:62:
    36:46:0c:c3:37:62:2b:58:7c:79:8b:b6:ad:d7:54:
    4e:8c:78:b5:fb:d8:71:38:83:65:5a:e4:d7:60:da:
    cf:a3:5a:fb:71:1c:46:aa
```

Public Key

```
==> openssl rsa -in key.pem -inform pem -pubout > key.pub

writing RSA key

==> cat key.pub

-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAq6+k7R+C9pnwk8tXu82F
CG3aMEw8piWUS+oucUUMymKOpNH+zEriyBj9rSNf5qPwtFepb+k4yuRCr+1zshU6
BBpA9udATGlTJZXKnURuBVbPe7nWV59YYQEAwf7+fhCWjyzYFjM5njdPw5t3Ablw
6hDHcYEpGSVQJXES9SwxjWEmOQ1smMl4AKpqzwxcJweJDNEGWpGSb28uJEbwmaN3
KezSOE2Rqvu2f149tOs+e5YOOqmPSFsL5Fthutllo+kzFJ6GlAHjaYPweJw+KDK4
WBm6LPKbLSp75+xYHrT8WjkkrgqOJhXOGfCwl7DDkAM4zkpbuqjx1tOUmP28TWfC
YQIDAQAB
-----END PUBLIC KEY-----
```

Answers

- What are the principal elements of a public-key cryptosystem?

Plaintext: This is the readable message or data that is fed into the algorithm as input. **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext. **Public and private keys:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the encryption algorithm depend on the public or private key that is provided as input. **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts. **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

Answers

- What are the roles of the public and private key?

A user's private key is kept private and known only to the user. The user's public key is made available to others to use. The private key can be used to encrypt a signature that can be verified by anyone with the public key. Or the public key can be used to encrypt information that can only be decrypted by the possessor of the private key.

Answers

- What are three broad categories of applications of public-key cryptosystems?

Encryption/decryption: The sender encrypts a message with the recipient's public key. **Digital signature:** The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message. **Key exchange:** Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

Answers

- What requirements must a public-key cryptosystems fulfill to be a secure algorithm?

1. It is computationally easy for a party B to generate a pair (public key PU_b, private key PR_b).
2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, M, to generate the corresponding ciphertext:

$$C = E(PU_b, M)$$

It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message:

$$M = D(PR_b, C) = D(PR_b, E(PU_b, M))$$

It is computationally infeasible for an opponent, knowing the public key, PU_b, to determine the private key, PR_b.

It is computationally infeasible for an opponent, knowing the public key, PU_b, and a ciphertext, C, to recover the original message, M.

Answers

- What is a trap-door one-way function?

A **one-way function** is one that maps a domain into a range such that every function value has a unique inverse, with the condition that the calculation of the function is easy whereas the calculation of the inverse is infeasible:

What is a trap-door one-way function?

A **trap-door one-way function** is easy to calculate in one direction and infeasible to calculate in the other direction unless certain additional information is known. With the additional information the inverse can be calculated in polynomial time.

Answers

- Describe in general terms an efficient procedure for picking a prime number.

1. Pick an odd integer n at random (e.g., using a pseudorandom number generator).
2. Pick an integer $a < n$ at random.
3. Perform the probabilistic primality test, such as Miller-Rabin. If n fails the test, reject the value n and go to step 1.
4. If n has passed a sufficient number of tests, accept n; otherwise, go to step 2.

Example

Perform encryption and decryption using the RSA algorithm, as in Figure 9.5, for the following:

a. $p = 3; q = 11, e = 7; M = 5$
b. $p = 5; q = 11, e = 3; M = 9$
c. $p = 7; q = 11, e = 17; M = 8$
d. $p = 11; q = 13, e = 11; M = 7$
e. $p = 17; q = 31, e = 7; M = 2$

Solution

a. $n = 33$; $\phi(n) = 20$; $d = 3$; C = 26.
b. $n = 55$; $\phi(n) = 40$; $d = 27$; C = 14.
c. $n = 77$; $\phi(n) = 60$; $d = 53$; C = 57.
d. $n = 143$; $\phi(n) = 120$; $d = 11$; C = 106.
e. $n = 527$; $\phi(n) = 480$; $d = 343$; C = 128. For decryption, we have

$$
\begin{aligned}
128^{343} \bmod 527 &= 128^{256} \times 128^{64} \times 128^{16} \times 128^{4} \times 128^{2} \times 128^{1} \bmod 527 \\
&= 35 \times 256 \times 35 \times 101 \times 47 \times 128 = 2 \bmod 527 \\
&= 2 \bmod 257
\end{aligned}
$$

Question

- In a public-key system using RSA, you intercept the ciphertext $C = 10$ sent to a user whose public key is $e = 5$, $n = 35$. *What is the plaintext M?*

Answer

- 5

Question

- In an RSA system, the public key of a given user is $e = 31, n = 3599$. *What is the pri-*vate key of this user? *Hint: First use trial-and-error to determine p and q; then use the* extended Euclidean algorithm to find the multiplicative inverse of 31 modulo f(*n)*.

Solution

By trail and error, we determine that $p = 59$ and $q = 61$. Hence $\phi(n) = 58 \times 60 = 3480$. Then, using the extended Euclidean algorithm, we find that the multiplicative inverse of 31 modulo $\phi(n)$ is 3031.

MCQ

T F 1. Asymmetric encryption utilizes only a public key for encryption and decryption.

T F 2. Asymmetric encryption can be used for confidentiality but not for authentication.

T F 3. Asymmetric encryption transforms plaintext into ciphertext.

T F 4. Plaintext is transformed into ciphertext using two keys and a decryption algorithm.

T F 5. A major advance in symmetric cryptography occurred with the development of the rotor encryption/decryption machine.

MCQ

T F 6. Public-key encryption is more secure from cryptanalysis than symmetric encryption.

T F 7. Much of the theory of public-key cryptosystems is based on number theory.

T F 8. Asymmetric algorithms rely on one key for encryption and a different but related key for decryption.

T F 9. The encryption algorithm performs various transformation on the ciphertext.

T F` 10. If the authenticator is encrypted with the sender's private key, it serves as a signature that verifies origin, content, and sequencing.

MCQ

T F 11. A trap-door one-way function is easy to calculate in one direction and infeasible to calculate in the other direction unless certain additional information is known.

T F 12. A public-key encryption scheme is not vulnerable to a brute-force attack.

T F 13. Before the application of the public-key cryptosystem each participant must generate a pair of keys.

T F 14. The defense against the brute-force approach for RSA is to use a large key space.

T F 15. Timing attacks are ciphertext attacks that are only applicable to RSA.

Solution

1. F
2. F
3. T
4. F
5. T
6. F
7. T
8. T
9. F
10. T
11. T
12. F
13. T
14. T
15. F

MCQ

1. Asymmetric encryption is also known as _____.

   A. public-key encryption
   B. private-key encryption
   C. optimal encryption
   D. digital-key encryption

2. Public-key encryption is also known as _____.

   A. digital-key encryption
   B. asymmetric encryption
   C. one way time exchange encryption
   D. optimal-key encryption

3. Asymmetric encryption can be used for _____.

   A. both confidentiality and authentication
   B. neither confidentiality nor authentication
   C. confidentiality
   D. authentication

MCQ

Plaintext is recovered from the ciphertext using the paired key and a _____.

A. digital signature
B. recovery encryption
C. decryption algorithm
D. encryption algorithm

5. The most widely used public-key cryptosystem is _____.

A. optimal asymmetric encryption
B. asymmetric encryption
C. RSA
D. DES

6. Public-key algorithms are based on _____.

A. permutation
B. mathematical functions
C. substitution
D. symmetry

MCQ

7. _____ are two related keys, a public key and a private key that are used to perform complementary operations, such as encryption and decryption or signature generation and signature verification.

   A. Asymmetric keys B. Key exchanges

   C. Symmetric keys D. Cipher keys

8. The _____ indicates that the subscriber identified in the certificate has sole control and access to the private key.

   A. OAEP B. Public Key Certificate

   C. Digital Signature D. PKI

MCQ

A _____ is a cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that deriving the private key from the public key is computationally infeasible.

A. Private Key (Symmetric) Cryptographic Algorithm

B. Key Exchange Cryptographic Algorithm

C. Public Key (Asymmetric) Cryptographic Algorithm

D. RSA Digital Cryptographic Algorithm

. A public-key encryption scheme has _____ ingredients.

A. six B. four

C. eight D. two

MCQ

11. The key used in symmetric encryption is referred to as a _____ key.

A. public B. secret

C. private D. decryption

12. The readable message or data that is fed into the algorithm as input is the _____.

A. ciphertext B. exchange

C. plaintext D. encryption

13. Two issues to consider with the computation required to use RSA are encryption/decryption and _____.

A. time complexity B. trap-door one-way functions

C. key generation D. asymmetric encryption padding

MCQ

14. _____ depend on the running time of the decryption algorithm.

A. Mathematical attacks
B. Timing attacks
C. Chosen ciphertext attacks
D. Brute-force attacks

15. We define the _____ of an algorithm to be f(n) if, for all n and all inputs of length n the execution of the algorithm takes at most f(n) steps. This is a common measure of the efficiency of an algorithm.

A. time complexity
B. one-way function
C. timing attack
D. OAEP