Unit- 3 Asymmetric Ciphers

- Also known as **public key cryptosystem**/ cryptography/ ciphers
- Different keys for encryption and decryption

- ✓ The development of public-key cryptography is the greatest and perhaps the only true revolution in the entire history of cryptography.
- ✓ It is **asymmetric, involving the use of two separate keys, in contrast to symmetric encryption, which uses only one key**.
- ✓ Anyone knowing the public key can encrypt messages or verify signatures, but **cannot** decrypt messages or create signatures, counter-intuitive though this may seem.
- ✓ It works by **the clever use of number theory problems that are easy one way but hard the other.**
- ✓ Note that **public key schemes are neither more nor less secure than private key** (security depends on the key size for both), **nor do they replace private key schemes** (they are too slow to do so), **rather they complement them**. Both also have issues with key distribution, requiring the use of some suitable protocol.

Why Public-Key Cryptography?

⇒ Developed to address two key issues**:**

1. **Key distribution** – how to have secure communications in general without having to trust a KDC (Key Distribution Center) with your key
2. **Digital signatures** – how to verify a message comes intact from the claimed sender

⇒ It was publicly introduced by Whitfield Diffie, Martin Hellman and Ralph Merkle in 1976

More Number Theory

A number of concepts from number theory are essential in the design of public-key cryptographic algorithms.

Prime Numbers:

- ➔ An integer **p > 1** is a **prime number** if and only if its only divisors are **± 1** and **±p**.
- ➔ Prime numbers play a critical role in number theory and in the techniques discussed in following chapters
- Any integer a $a > 1$ can be factored in a unique way as

$$a = p_1^{a_1} \times p_2^{a_2} \times \cdots \times p_t^{a_t}$$

Where,

$$p_1 < p_2 < \ldots < p_t$$

And each a_i is a positive integer.

- This is known as the **fundamental theorem of arithmetic**.
- Example:

$$91 = 7 \times 13$$
$$3600 = 2^4 \times 3^2 \times 5^2$$
$$11011 = 7 \times 11^2 \times 13$$

- If **P** is the set of all prime numbers, then any positive integer **a** can be written uniquely in the following form:

$$a = \prod_{p \in P} p^{a_p} \quad \text{where each } a_p \geq 0$$

- The right-hand side is **the product over all possible prime numbers p**; for any particular value of **a**, most of the exponents a_p will be 0.
- Multiplication of two numbers is equivalent to adding the corresponding exponents.

Given ,

$$a = \prod_{p \in P} p^{a_p}, \; b = \prod_{p \in P} p^{b_p}.$$

Define $\boldsymbol{k = ab}$. We know that the integer $\boldsymbol{k}$ can be expressed as the product of powers of primes:

$$k = \prod_{p \in P} p^{k_p}.$$

It follows that

$$k_p = a_p + b_p \text{ for all } p \in P.$$

$$k = 12 \times 18 = (2^2 \times 3) \times (2 \times 3^2) = 216$$
$$k_2 = 2 + 1 = 3; \; k_3 = 1 + 2 = 3$$
$$216 = 2^3 \times 3^3 = 8 \times 27$$

- Any integer of the form $\mathbf{p^n}$ can be divided only by an integer that is of a lesser or equal power of the same prime number, $\mathbf{p^j}$ with $\mathbf{j \leq n}$. Thus, we can say the following.

$$a = \prod_{p \in P} p^{a_p}, b = \prod_{p \in P} p^{b_p}$$

$$\text{If a} \mid \text{b, then } a_p \leq b_p \text{ for all } p.$$

$a = 12; b = 36; 12|36$
$12 = 2^2 \times 3; 36 = 2^2 \times 3^2$
$a_2 = 2 = b_2$
$a_3 = 1 \leq 2 = b_3$
Thus, the inequality $a_p \leq b_p$ is satisfied for all prime numbers.

- It is easy to determine the greatest common divisor of two positive integers if we express each integer as the product of primes.

$$300 = 2^2 \times 3^1 \times 5^2$$
$$18 = 2^1 \times 3^2$$
$$\gcd(18, 300) = 2^1 \times 3^1 \times 5^0 = 6$$

The following relationship always holds:

$$\text{If } k = \gcd(a, b), \text{ then } k_p = \min(a_p, b_p) \text{ for all } p.$$

Note: Determining the prime factors of a large number is no easy task, so the preceding relationship does not directly lead to a practical method of calculating the greatest common divisor

Fermat's Theorem

- Also known as **Fermat's little theorem**
- Given in 1640 by French mathematician *Pierre de Fermat*
- Fermat's theorem states the following:

  If **p** is prime and **a** is a positive integer not divisible by **p**, then

$$a^{p-1} \equiv 1 \pmod{p}$$

Here, **a** and **p** are relatively prime.

Example:

Let a = 2 and P = 17

According to Fermat's little theorem

$2^{17-1} \equiv 1 \pmod{17}$

[65536 % 17 ≡ 1 that means (65536-1) is divisible by 17]

- An alternative form of Fermat's theorem is also useful:
  If **p** is prime and a is **a** positive integer, then

$$a^p \equiv a(\mathrm{mod}\, p)$$

Here **a** and **p** need not to be relatively prime

$p = 5, a = 3 \qquad a^p = 3^5 = 243 \equiv 3(\mathrm{mod}\, 5) = a(\mathrm{mod}\, p)$

Euler's Totient Function

- By Leonhard Euler (pronounced "oiler") in 1763
- Also known as **Euler's phi function** or simply the **phi function**.
- Euler's totient function, written $\phi(n)$, and defined as the number of positive integers less than **n** and relatively prime to **n**.
- By convention, $\phi(1) = 1$

DETERMINE $\phi(37)$ AND $\phi(35)$.

Because 37 is prime, all of the positive integers from 1 through 36 are relatively prime to 37. Thus $\phi(37) = 36$.

To determine $\phi(35)$, we list all of the positive integers less than 35 that are relatively prime to it:

1, 2, 3, 4, 6, 8, 9, 11, 12, 13, 16, 17, 18
19, 22, 23, 24, 26, 27, 29, 31, 32, 33, 34

There are 24 numbers on the list, so $\phi(35) = 24$.

Try yourself:

Verify:

$\phi(4) = 2$ $\phi(12) = 4$ $\phi(15) = 8$ $\phi(32) = 16$

- The value ϕ**(1)** is without meaning but is defined to have the value 1.
- It should be clear that, for a prime number **p**,

$$\phi(p) = p - 1$$

- Now suppose that we have two prime numbers **p** and **q** with **p ≠q**. Then we can show that, for **n = pq**,

$$\phi(n) = \phi(pq) = \phi(p) \times \phi(q) = (p - 1) \times (q - 1)$$

Some Values of Euler's Totient Function ϕ(n)

n	$\phi(n)$
1	1
2	1
3	2
4	2
5	4
6	2
7	6
8	4
9	6
10	4

n	$\phi(n)$
11	10
12	4
13	12
14	6
15	8
16	8
17	16
18	6
19	18
20	8

n	$\phi(n)$
21	12
22	10
23	22
24	8
25	20
26	12
27	18
28	12
29	28
30	8

$\phi(21) = \phi(3) \times \phi(7) = (3 - 1) \times (7 - 1) = 2 \times 6 = 12$
where the 12 integers are {1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20}.

Euler's Theorem

- Also known as the Fermat–Euler theorem or Euler's totient theorem (Given in 1763 by Euler)
- Euler's theorem states that for every **a** and **n** that are relatively prime:

$$a^{\phi(n)} \equiv 1 (\text{mod } n)$$

(Integers **a** and **n** are coprime)

$$a = 3; n = 10; \phi(10) = 4 \quad a^{\phi(n)} = 3^4 = 81 = 1 (\text{mod } 10) = 1 (\text{mod } n)$$
$$a = 2; n = 11; \phi(11) = 10 \quad a^{\phi(n)} = 2^{10} = 1024 = 1 (\text{mod } 11) = 1 (\text{mod } n)$$

- As is the case for Fermat's theorem, an alternative form of the theorem is also useful:

$$a^{\phi(n)+1} \equiv a (\text{mod } n)$$

(Integers **a** and **p** need not to be relatively prime)

Testing for Primality

⇒ For many cryptographic algorithms, it is necessary to select one or more very large prime numbers at random.

⇒ Thus, we are faced with the task of determining **whether a given large number is prime**.

⇒ There is no simple yet efficient means of accomplishing this task.

Miller-Rabin Algorithm

⇒ Also referred to in the literature as the **Rabin-Miller algorithm**, or the **Rabin-Miller test**, or the **Miller- Rabin test**.

⇒ Due to Miller and Rabin

⇒ Typically used to test a large number for primality.

Necessary Background Concepts

- Any positive odd integer n ≥ 3 can be expressed as

  $n - 1 = 2^k q$ with **$k > 0$, q odd**

To see this, note that **n - 1** is an even integer. Then, divide **(n - 1**) by **2** until the result is an odd number **q**, for a total of **k** divisions. If **n** is expressed as a binary number, then the result is achieved by shifting the number to the right until the rightmost digit is a 1, for a total of **k** shifts

Two Properties of Prime Numbers

The first property is stated as follows:

If **p** is prime and **a is** a positive integer less than **p**, then $\mathbf{a^2 \bmod p = 1}$ if and only if either

a mod p = 1 or **a mod p = -1 mod p = p - 1**.

By the rules of modular arithmetic

$\mathbf{(a \bmod p)\ (a \bmod p) = a^2 \bmod p.}$

Thus, if either **a mod p = 1** or **a mod p = -1,** then $\mathbf{a^2 \bmod p = 1}$.

Conversely, if $\mathbf{a^2 \bmod p = 1}$, **then** $\mathbf{(a \bmod p)^2 = 1,}$ which is true only for

a mod p = 1 or **a mod p = -1.**

The second property is stated as follows:

Let **p** be a prime number greater than 2. We can then write $\mathbf{p - 1 = 2^k q}$ with **k > 0, q odd.**

Let **a** be any integer in the range **1 < a < p - 1**.

Then one of the two following conditions is true.

1. $\mathbf{a^q}$ is congruent to **1 modulo p**. That is, $\mathbf{a^q \bmod p = 1}$, or equivalently,

$$\mathbf{a^q \equiv 1 (mod\ p).}$$

2. One of the number $a^q,\ a^{2q},\ a^{4q}, \ldots, a^{2^{k-1}q}$ is congruent to **-1 modulo p**.

That is, there is some number j in the range ($1 \leq j \leq k$) such that

$a^{2^{j-1}q} \mod p = -1 \mod p = p - 1$

or equivalently,

$$a^{2^{j-1}q} \equiv -1 \pmod{p}$$

Details of the Algorithm

These considerations lead to the conclusion that, if **n** is **prime**, then either the first element in the list of residues, or remainders,

$(a^q, a^{2q}, \ldots, a^{2^{k-1}q}, a^{2^k q})$ modulo n equals 1;

or some element in the list equals **(n - 1);** otherwise **n is composite** (i.e., not a prime). On the other hand, **if the condition is met**, that **does not necessarily mean that n is prime**. For example, if n = 2047 = 23 * 89, then n - 1 = 2 * 1023. We compute 2^{1023} mod 2047 = 1, so that 2047 meets the condition but is not prime.

- We can use the preceding property to devise a test for primality.

  The procedure TEST takes a candidate integer **n** as input and returns the result **composite** if **n** is **definitely not a prime**, and the result **inconclusive** if **n may or may not be a prime**.

Algorithm

```
TEST (n)
1. Find integers k, q, with k > 0, q odd, so that
   (n − 1 = 2^k q);
2. Select a random integer a, 1 < a < n − 1;
3. if a^q mod n = 1 then return("inconclusive");
4. for j = 0 to k − 1 do
5. if a^(2^j q) mod n = n − 1 then return("inconclusive");
6. return("composite");
```

Let us apply the test to the prime number $n = 29$. We have $(n - 1) = 28 = 2^2(7) = 2^k q$. First, let us try $a = 10$. We compute $10^7 \bmod 29 = 17$, which is neither 1 nor 28, so we continue the test. The next calculation finds that $(10^7)^2 \bmod 29 = 28$, and the test returns `inconclusive` (i.e., 29 may be prime). Let's try again with $a = 2$. We have the following calculations: $2^7 \bmod 29 = 12$; $2^{14} \bmod 29 = 28$; and the test again returns `inconclusive`. If we perform the test for all integers a in the range 1 through 28, we get the same `inconclusive` result, which is compatible with n being a prime number.

Now let us apply the test to the composite number $n = 13 \times 17 = 221$. Then $(n - 1) = 220 = 2^2(55) = 2^k q$. Let us try $a = 5$. Then we have $5^{55} \bmod 221 = 112$, which is neither 1 nor $220 (5^{55})^2 \bmod 221 = 168$. Because we have used all values of j (i.e., $j = 0$ and $j = 1$) in line 4 of the TEST algorithm, the test returns `composite`, indicating that 221 is definitely a composite number. But suppose we had selected $a = 21$. Then we have $21^{55} \bmod 221 = 200$; $(21^{55})^2 \bmod 221 = 220$; and the test returns `inconclusive`, indicating that 221 may be prime. In fact, of the 218 integers from 2 through 219, four of these will return an inconclusive result, namely 21, 47, 174, and 200.

Discrete Logarithms

Discrete logarithms are fundamental to a number of public-key algorithms, including

Diffie-Hellman key exchange and the digital signature algorithm (DSA).

Primitive roots

- For any prime number p, if we have a number **a** such that **powers of a mod p** generate all the numbers between 1 to p-1 then a is called **Primitive Root** of **p**
- In terms of Group terminology a is the **generator element** of the multiplicative group of the finite field formed by **mod p**

Definition : A primitive root modulo a prime p is an integer r in $\mathbf{Z}_p$ such that every nonzero element of $\mathbf{Z}_p$ is a power of r.

Example : Since every element of $\mathbf{Z}_{11}$ is a power of 2, 2 is a primitive root of 11.

Powers of 2 modulo 11: $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, $2^4 = 5$, $2^5 = 10$, $2^6 = 9$, $2^7 = 7$, $2^8 = 3$, $2^{10} = 2$.

Example : Since not all elements of $\mathbf{Z}_{11}$ are powers of 3, 3 is not a primitive root of 11.

Powers of 3 modulo 11: $3^1 = 3$, $3^2 = 9$, $3^3 = 5$, $3^4 = 4$, $3^5 = 1$, and the pattern repeats for higher powers.

Important Fact : There is a primitive root modulo p for every prime number p.

Logarithms for Modular Arithmetic

→ With ordinary positive real numbers, the logarithm function is the inverse of exponentiation.

→ An analogous function exists for modular arithmetic.

Properties of ordinary logarithms.

The logarithm of a number is defined to be **the power to which some positive base (except 1) must be raised in order to equal the number**. That is, for base x and for a value y,

$$y = x^{\log_x(y)}$$

The properties of logarithms include

$$\log_x(1) = 0$$
$$\log_x(x) = 1$$

$$\log_x(yz) = log_x(y) + \log_x(z)$$
$$\log_x(y^r) = r \times \log_x(y)$$

- Consider a primitive root **a** for some prime number **p** (the argument can be developed for nonprimes as well).
  - — Then we know that the powers of **a** from 1 through (p - 1) produce each integer from 1 through (p - 1) exactly once.
- By the definition of modular arithmetic, any integer **b** satisfies

$$b \equiv r \pmod{p} \text{ for some } r, \text{ where } 0 \leq r \leq (p - 1)$$

- **For any integer *b* and a primitive root *a* of prime number *p*, we can find a unique exponent *i* such that**

$$b \equiv a^i \pmod{p} \quad \text{where } 0 \leq i \leq (p - 1)$$

This exponent *i* is referred to as the *discrete logarithm* of the number *b* for the base *a (mod p).*

  - — *We denote this value as* $\boldsymbol{dlog_{a,p}(b)}$.

- Note the following:

$$\text{dlog}_{a,p}(1) = 0 \text{ because } a^0 \bmod p = 1 \bmod p = 1$$
$$\text{dlog}_{a,p}(a) = 1 \text{ because } a^1 \bmod p = a$$

(a) Discrete logarithms to the base 2, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$log_{2,19}(a)$	18	1	13	2	16	14	6	3	8	17	12	15	5	7	11	4	10	9

(b) Discrete logarithms to the base 3, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$log_{3,19}(a)$	18	7	1	14	4	8	6	3	2	11	12	15	17	13	5	10	16	9

(c) Discrete logarithms to the base 10, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$log_{10,19}(a)$	18	17	5	16	2	4	12	15	10	1	6	3	13	11	7	14	8	9

(d) Discrete logarithms to the base 13, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$log_{13,19}(a)$	18	11	17	4	14	10	12	15	16	7	6	3	1	5	13	8	2	9

(e) Discrete logarithms to the base 14, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$log_{14,19}(a)$	18	13	7	8	10	2	6	3	14	5	12	15	11	1	17	16	4	9

(f) Discrete logarithms to the base 15, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$log_{15,19}(a)$	18	5	11	10	8	16	12	15	4	13	6	3	7	17	1	2	14	9

Calculation of Discrete Logarithms

Consider the equation

$$\mathbf{y = g^x \bmod p}$$

Given **g, x**, and **p**, it is a straightforward matter to calculate **y**. At the worst, we must perform **x** repeated multiplications, and algorithms exist for achieving greater efficiency

However, given **y, g**, and **p,** it is, in general, **very difficult** to calculate **x** (take the discrete logarithm).

Discrete logarithm problem (DLP)

Given a prime p and values g and y, find x such that

$$y = g^x \bmod p$$

Ingredients of Public Key Cryptosystems

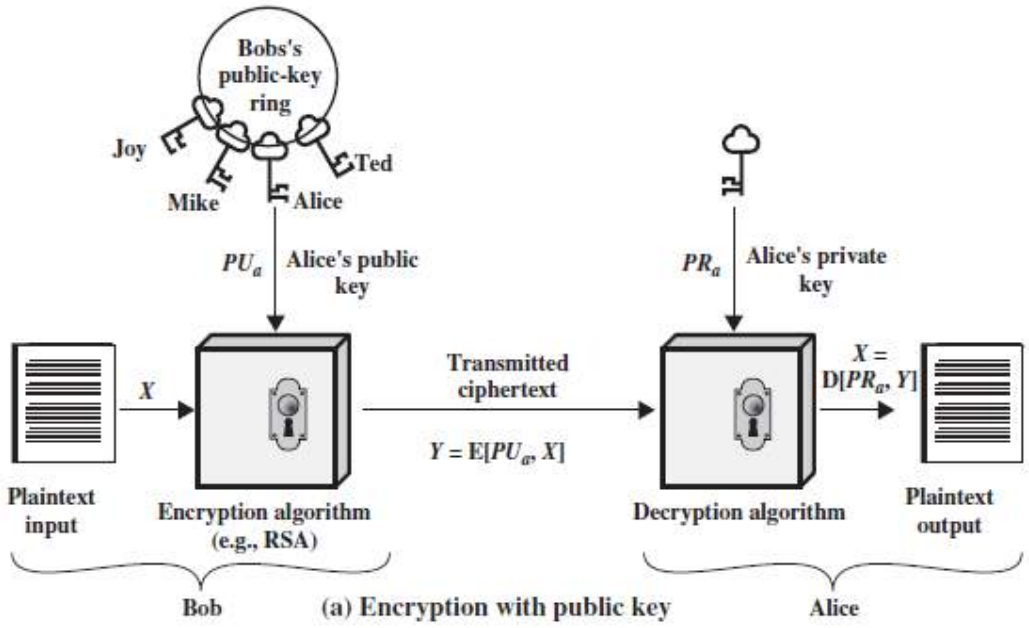
A public-key encryption scheme has following six ingredients

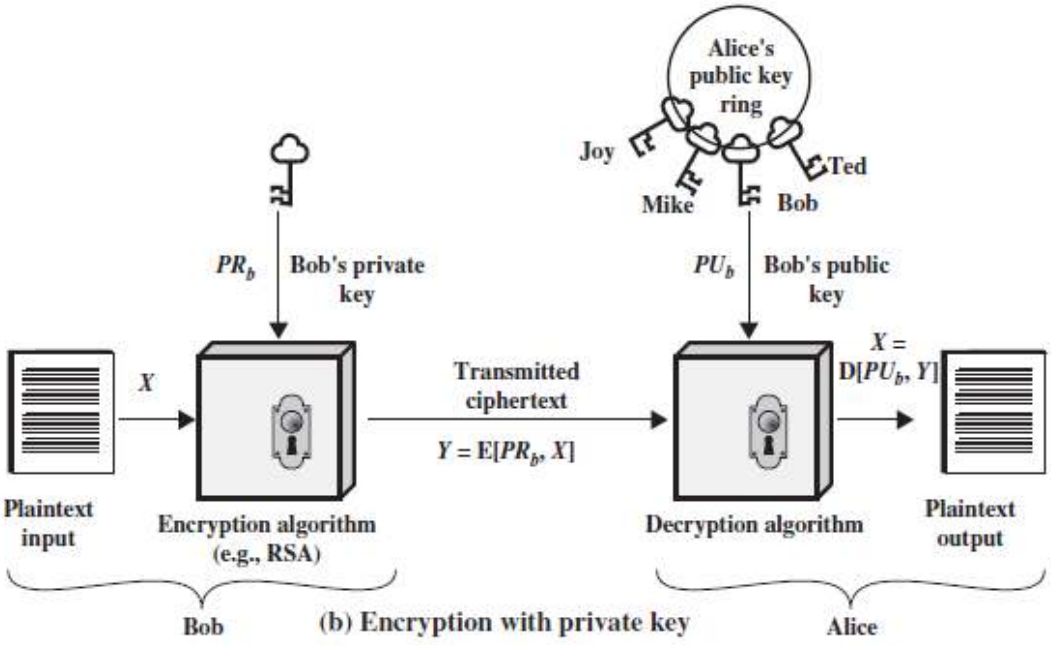
1. **Plaintext:** This is the readable message or data that is fed into the algorithm as input.
2. **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
3. **Public and private keys**: This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input.
4. **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.
5. **Decryption algorithm**: This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

The essential steps are the following.

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. As in Figure blow suggests, each user maintains a collection of public keys obtained from others.
3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.

4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.





Conventional Encryption	Public-Key Encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key. *Needed for Security:* 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if the key is kept secret. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	*Needed to Work:* 1. One algorithm is used for encryption and a related algorithm for decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one). *Needed for Security:* 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if one of the keys is kept secret. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

A closer look at the essential elements of a public-key encryption scheme

Let us take a closer look at the essential elements of a public-key encryption scheme, using following figure.

Public-Key Cryptosystem: Secrecy

Main Idea: *Encryption by Receiver's public key and Decryption by Receiver's private key***.**

- There is some source **A** that produces a message in plaintext, $\boldsymbol{X = [X_1, X_2, \ldots, X_M]}$**.**
- The $\boldsymbol{M}$ elements of $\boldsymbol{X}$ are letters in some finite alphabet.
- The message is intended for destination **B**.
- **B** generates a related pair of keys: a public key, $\mathbf{PU_b}$, and a private key, $\mathbf{PR_b}$.
- $\mathbf{PR_b}$ is known only to **B**, whereas $\mathbf{PU_b}$ is publicly available and therefore accessible by **A**.
- With the message **X** and the encryption key $\mathbf{PU_b}$ as input, **A** forms the ciphertext $\mathbf{Y = [Y_1, Y_2, \ldots, Y_N]}$**:**

$$\mathbf{Y = E(PU_b, X)}$$

- The intended receiver, in possession of the matching private key, is able to invert the transformation:

$$\mathbf{X = D(PR_b, Y)}$$

- An adversary, observing **Y** and having access to $\mathbf{PU_b}$, but not having access to $\mathbf{PR_b}$ or **X**, must attempt to recover **X** and/or $\mathbf{PR_b}$.
    - It is assumed that the adversary does have knowledge of the encryption **(E)** and decryption **(D)** algorithms. If the adversary is interested only in this particular message, then the focus of effort is to recover **X** by generating a plaintext estimate $\widehat{\boldsymbol{X}}$. Often, however, the adversary is interested in being able to read future messages as well, in which case an attempt is made to recover $\mathbf{PR_b}$ by generating an estimate $\widehat{\boldsymbol{PR_b}}$

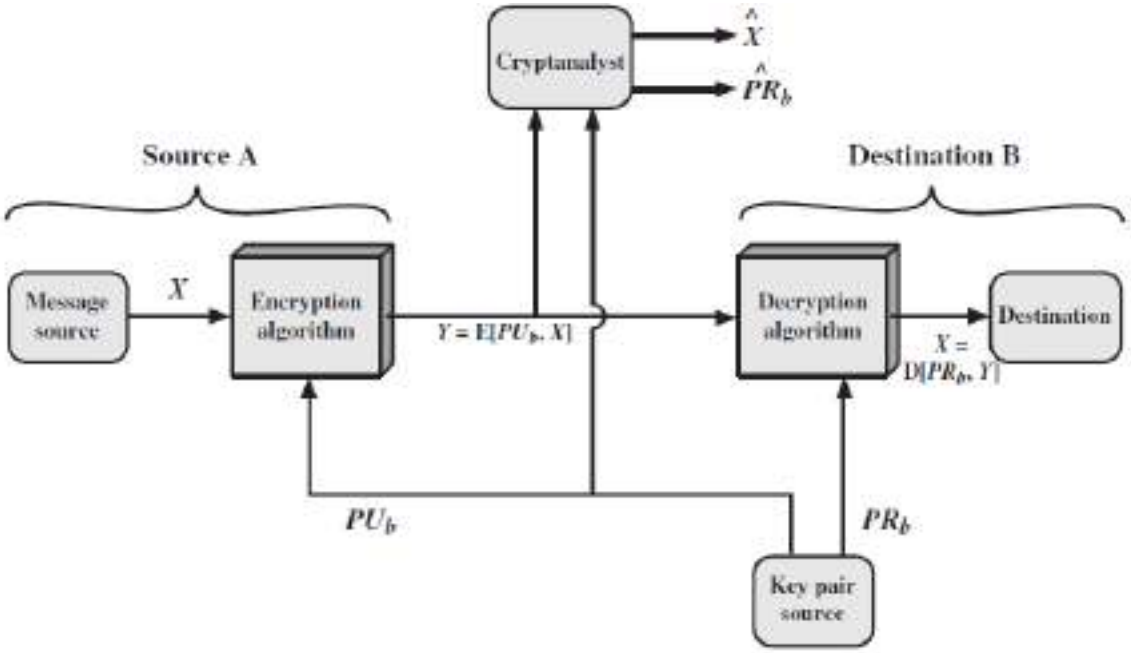


Figure : Public-Key Cryptosystem: Secrecy

- This scheme is for confidentiality

Public-Key Cryptosystem: Authentication

Main Idea: *Encryption by Sender's private key and Decryption by Sender's public key.*

$$Y = \mathrm{E}(PR_a, X)$$
$$X = \mathrm{D}(PU_a, Y)$$

- In this case, **A** prepares a message to **B** and encrypts it using **A's** private key before transmitting it.
- **B** can decrypt the message using **A's** public key.
- Because the message was encrypted using **A's** private key, only A could have prepared the message. Therefore, the entire encrypted message serves as **a digital signature**.
- In addition, it is impossible to alter the message without access to **A's** private key, so the message is *authenticated both in terms of source and in terms of data integrity*.

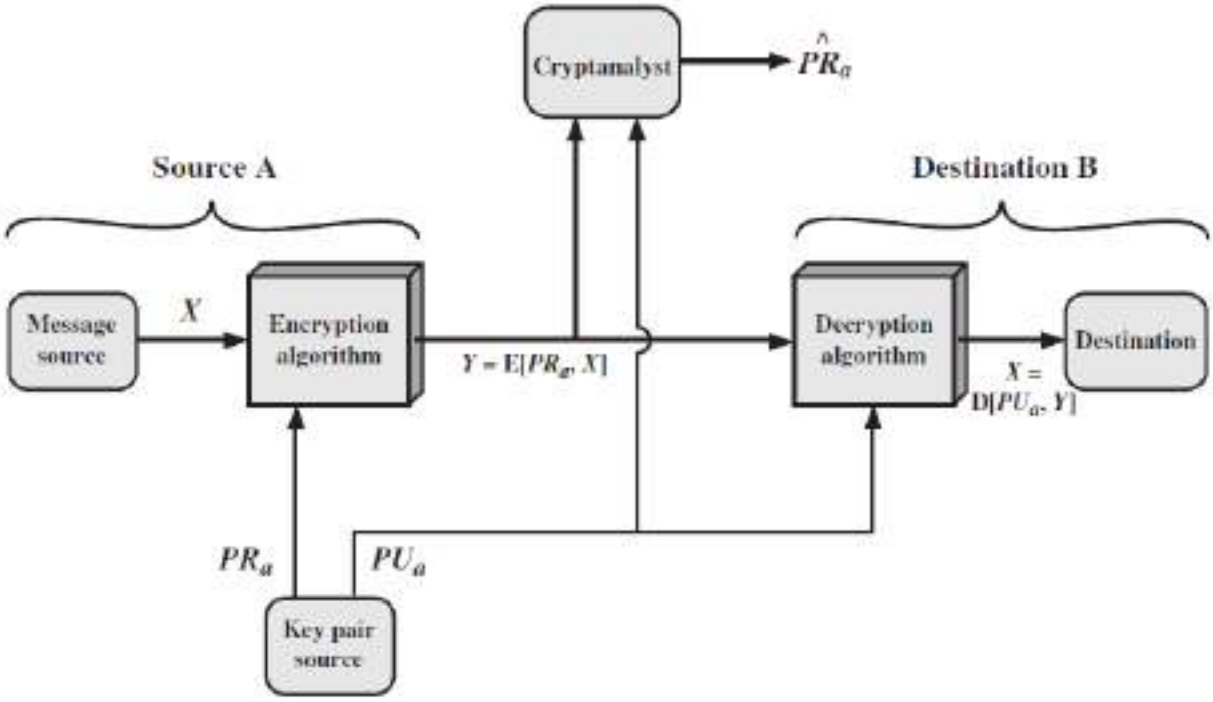


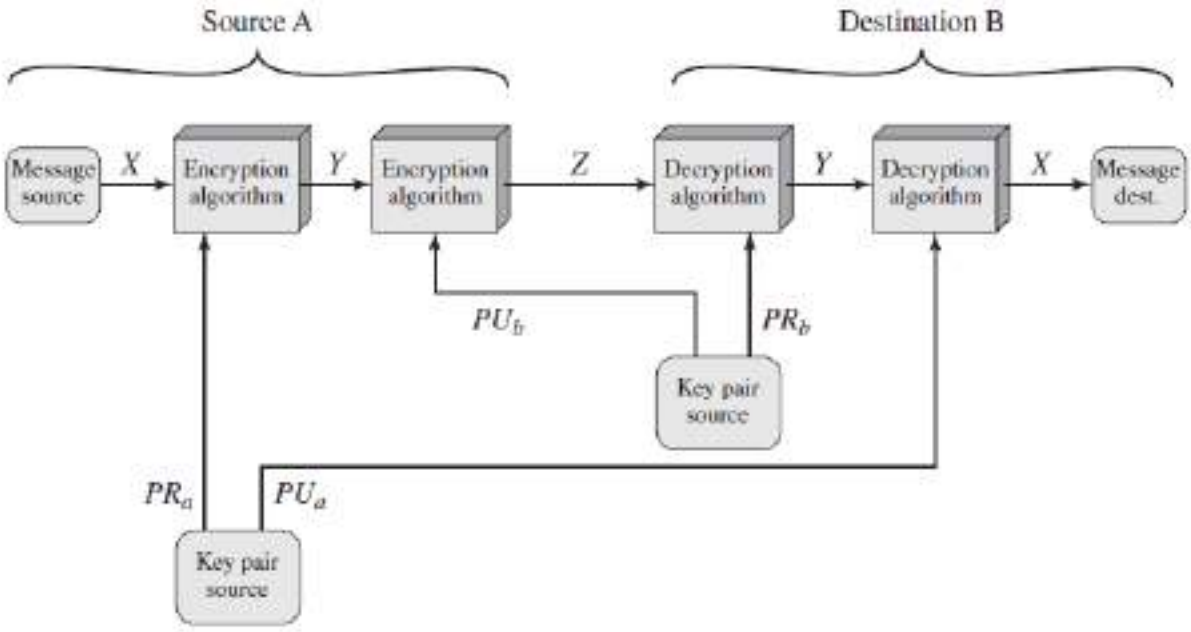
Figure: Public-Key Cryptosystem: Authentication

Public-Key Cryptosystem: Authentication and Secrecy

It is possible to provide both the authentication function and confidentiality by a double use of the public-key scheme

Main Idea: *First encrypt using Sender's private key and again encrypt the result by Receiver's public key similarly first decrypt by using Receiver's private key and again decrypt the result by Sender's public key.*

$$Z = \mathrm{E}(PU_b, \mathrm{E}(PR_a, X))$$
$$X = \mathrm{D}(PU_a, \mathrm{D}(PR_b, Z))$$



- In this case, we begin as before by encrypting a message, using the sender's private key. This provides the **digital signature**.
- Next, we encrypt again, using the receiver's public key. The final ciphertext can be decrypted only by the intended receiver, who alone has the matching private key. Thus, **confidentiality** is provided.
- The disadvantage of this approach is that the public-key algorithm, which is complex, must be exercised four times rather than two in each communication.

Applications for Public-Key Cryptosystems

- Public-key systems are characterized by the use of a cryptographic algorithm with two keys, one held private and one available publicly.
- Depending on the application, the sender uses either the sender's private key or the receiver's public key, or both, to perform some type of cryptographic function.
- In broad terms, we can classify the use of public-key cryptosystems into following three categories:
    1. **Encryption/decryption:** The sender encrypts a message with the recipient's public key.
    2. **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.
    3. **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.
- Some algorithms are suitable for all uses, others are specific to one

Requirements for Public-Key Cryptography

A public- key cryptographic algorithm must satisfy following conditions:

1. It is computationally easy for a party **B** to generate a pair (public key $\mathbf{PU_b}$, private key $\mathbf{PR_b}$).
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:

$$\mathbf{C = E(PU_b, M)}$$

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

$$\mathbf{M = D(PR_b,C) = D[PR_b, E(PU_b,M)]}$$

4. It is computationally infeasible for an adversary, knowing the public key, $\mathbf{PU_b}$, to determine the private key, $\mathbf{PR_b}$.
5. It is computationally infeasible for an adversary, knowing the public key, $\mathbf{PU_b}$, and a ciphertext, **C**, to recover the original message, **M**.

We can add a sixth requirement that, although useful, is not necessary for all public-key applications:

6. The two keys can be applied in either order:

$$\mathbf{M = D[PU_b, E(PR_b, M)] = D[PR_b, E(PU_b, M)]}$$

The RSA Algorithm

- RSA is the best known, and by far the most widely used general public key encryption algorithm, and was first published by Rivest, Shamir & Adleman of MIT in 1978
- Since that time RSA has reigned supreme as the most widely accepted and implemented general-purpose approach to public-key encryption.
- It is based on exponentiation in a finite (Galois) field over integers modulo a prime, using large integers (eg. 1024 bits).
- Its security is due to the cost of factoring large numbers.
- The RSA scheme is a cipher in which the plaintext and ciphertext are integers between 0 and n - 1 for some n. A typical size for n is 1024 bits, or 309 decimal digits. That is, n is less than 2^{1024}.

Algorithm

Alice generates a public/private key pair; Bob encrypts using Alice's public key; and Alice decrypts using her private key.

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	$PR = \{d, n\}$

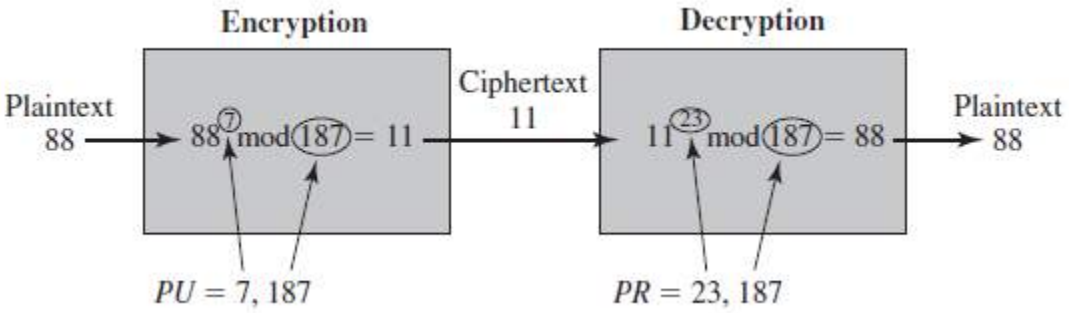
Encryption by Bob with Alice's Public Key

Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Private Key

Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

Example of RSA algorithm



In this simple example, the plaintext is an alphanumeric string. Each plaintext symbol is assigned a unique code of two decimal digits (e.g., a = 00, A = 26). A plaintext block consists of four decimal digits, or two alphanumeric characters.

For this example, the keys were generated as follows

1. Select two prime numbers, $p = 17$ and $q = 11$.
2. Calculate $n = pq = 17 \times 11 = 187$.
3. Calculate $\phi(n) = (p - 1)(q - 1) = 16 \times 10 = 160$.
4. Select e such that e is relatively prime to $\phi(n) = 160$ and less than $\phi(n)$; we choose $e = 7$.
5. Determine d such that $de \equiv 1 \pmod{160}$ and $d < 160$. The correct value is $d = 23$, because $23 \times 7 = 161 = (1 \times 160) + 1$; d can be calculated using the extended Euclid's algorithm

The resulting keys are **public key PU = {7, 187}** and

Private key PR = {23, 187}.

The example shows the use of these keys for a plaintext input of **M = 88**.

For encryption, we need to calculate $\mathbf{C = 88^7 \bmod 187}$.

Exploiting the properties of modular arithmetic, we can do this as follows.

$$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$$

$$88^1 \bmod 187 = 88$$

$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$

$$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$$

$$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11$$

For decryption, we calculate **M = 11^{23} mod 187**:

$$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$$

$$11^1 \bmod 187 = 11$$

$$11^2 \bmod 187 = 121$$

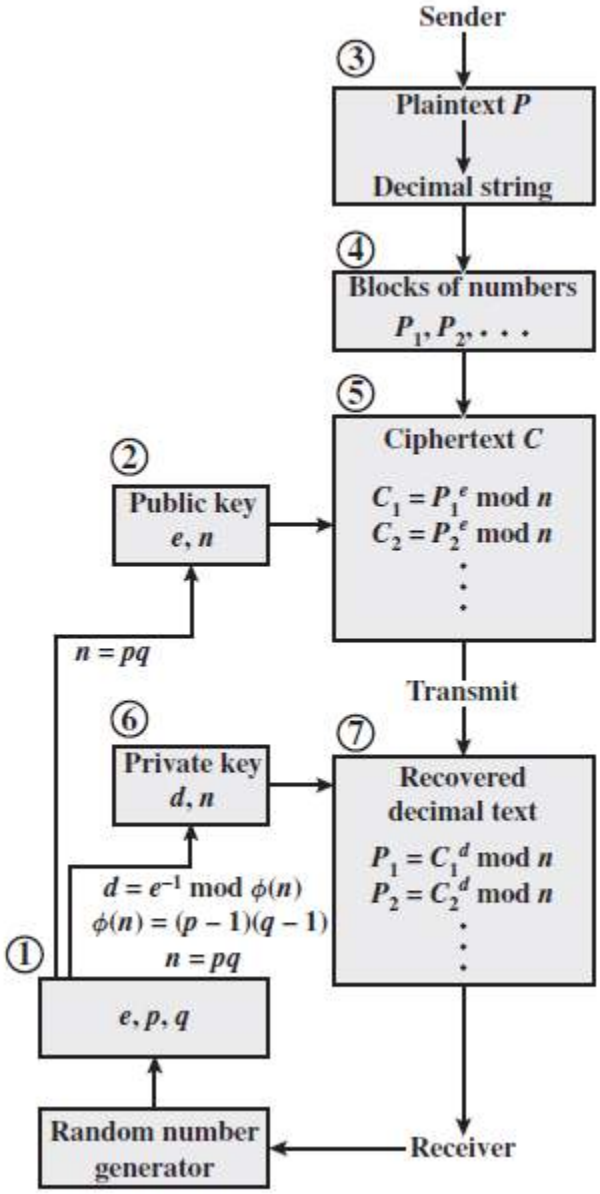
$$11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$$

$$11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$$

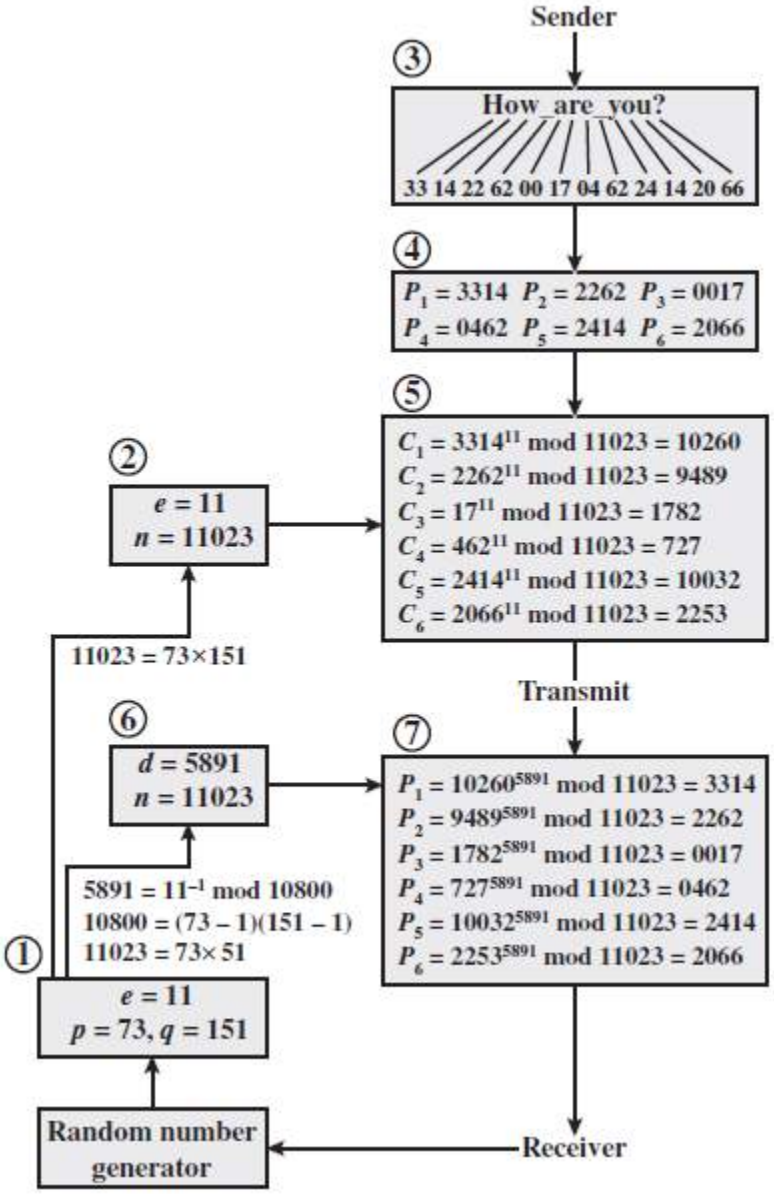
$$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79{,}720{,}245 \bmod 187 = 88$$

Encryption of multiple blocks using RSA

Following figures illustrates the sequence of events for the encryption of multiple blocks, figure (b) gives a specific example. The circled numbers indicate the order in which operations are performed.



Figure(a) : RSA Processing of Multiple Blocks(general approach)



Figure(b) : RSA Processing of Multiple Blocks(an example)

Exponentiation in Modular Arithmetic

- Both encryption and decryption in RSA involve **raising an integer to an integer power, mod n.**
- If **the exponentiation is done over the integers and then reduced modulo n, the intermediate values would be gargantuan** (extremely large).
- Fortunately, as the preceding example shows, we can make use of a property of modular arithmetic

$$[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$$

- Thus, we can reduce intermediate results modulo n. This makes the calculation practical.

Another consideration is the efficiency of exponentiation, because with RSA, we are dealing with potentially large exponents. To see how efficiency might be increased, consider that we wish to compute x^{16}. A straightforward approach requires 15 multiplications:

$$x^{16} = x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x$$

However, we can achieve the same final result with only four multiplications if we repeatedly take the square of each partial result, successively forming (x^2, x^4, x^8, x^{16}).

— As another example, suppose we wish to calculate x^{11} mod n for some integers x and n. Observe that $x^{11} = x^{1+2+8} = (x)(x^2)(x^8)$. In this case, we compute x mod n, x^2 mod n, x^4 mod n, and x^8 mod n and then calculate [(x mod n) * (x^2 mod n) *(x^8 mod n)] mod n.

- More generally, suppose we wish to find the value $\mathbf{a^b \bmod n}$ with **a, b**, and **m** positive integers. If we express **b** as a binary number $b_k b_{k-1} \ldots b_0$, then we have

$$b = \sum_{b_i \neq 0} 2^i$$

Therefore,

$$a^b = a^{\left(\sum 2^t \atop b_t \neq 0\right)} = \prod_{b_t \neq 0} a^{(2^t)}$$

$$a^b \bmod n = \left[\prod_{b_t \neq 0} a^{(2^t)}\right] \bmod n = \left(\prod_{b_t \neq 0} \left[a^{(2^t)} \bmod n\right]\right) \bmod n$$

We can therefore develop the following algorithm for computing a^b mod n

```
c ← 0; f ← 1
for i ← k downto 0
    do  c ← 2 × c
        f ← (f × f) mod n
    if  bi = 1
        then c ← c + 1
             f ← (f × a) mod n
return f
```

Note: The integer b is expressed as a binary number $b_k b_{k-1} \ldots b_0$.

Example: Result of the Fast Modular Exponentiation Algorithm for ab mod n, where a = 7, b = 560 = 1000110000, and n = 561

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
f	7	49	157	526	160	241	298	166	67	1

Efficient Operation Using the Public Key

- To speed up the operation of the RSA algorithm using the public key, a specific choice of e is usually made.
- The most common choice is **65537** ($2^{16} + 1$); two other popular choices are **3** and **17**. Each of **these choices has only two 1 bits**, so the **number of multiplications required to perform exponentiation is minimized**.
- However, with a very small public key, such as **e** = 3, RSA becomes vulnerable to a simple attack.

Efficient Operation Using the Private Key

- We cannot similarly choose a small constant value of **d** for efficient operation. A small value of **d** is vulnerable to a brute-force attack and to other forms of cryptanalysis.
- However, there is a way to speed up computation using the CRT (Chinese remainder theorem).

(Note: Please read book (page 169-170 of "Cryptography and Network Security Principles and Practice "- Sixth Edition by *William Stallings*) for detail explanation.

The Security of RSA

Five possible approaches to attacking the RSA algorithm are:

- **Brute force:** This involves trying all possible private keys.
- **Mathematical attacks:** There are several approaches, all equivalent in effort to factoring the product of two primes.
- **Timing attacks:** These depend on the running time of the decryption algorithm.
- **Hardware fault-based attack:** This involves inducing hardware faults in the processor that is generating digital signatures.
- **Chosen ciphertext attacks**: This type of attack exploits properties of the RSA algorithm.

Note: Please read book (page 172-177 of "Cryptography and Network Security Principles and Practice"- Sixth Edition by *William Stallings*) for detail explanation.

Distribution of public key

- **Key Distribution** - Distributing the keys over communicating parties.

Several techniques have been proposed for the distribution of public keys. Virtually all these proposals can be grouped into the following general schemes:

- **Public announcement**
- **Publicly available directory**
- **Public-key authority**
- **Public-key certificates**

Public Announcement of Public Keys

- On the face of it, the point of public-key encryption is that the public key is public.
- Thus, if there is some broadly accepted public-key algorithm, such as RSA, any participant can send his or her public key to any other participant or broadcast the key to the community at large as in figure below.

Figure: Uncontrolled Public-Key Distribution

- Although this approach is convenient, it has a major weakness. Anyone can forge such a public announcement. That is, some user could pretend to be user A and send a public key to another participant or broadcast such a public key.

Publicly Available Directory

- A greater degree of security can be achieved by maintaining a publicly available dynamic directory of public keys.
- Maintenance and distribution of the public directory would have to be the responsibility of some trusted entity or organization.

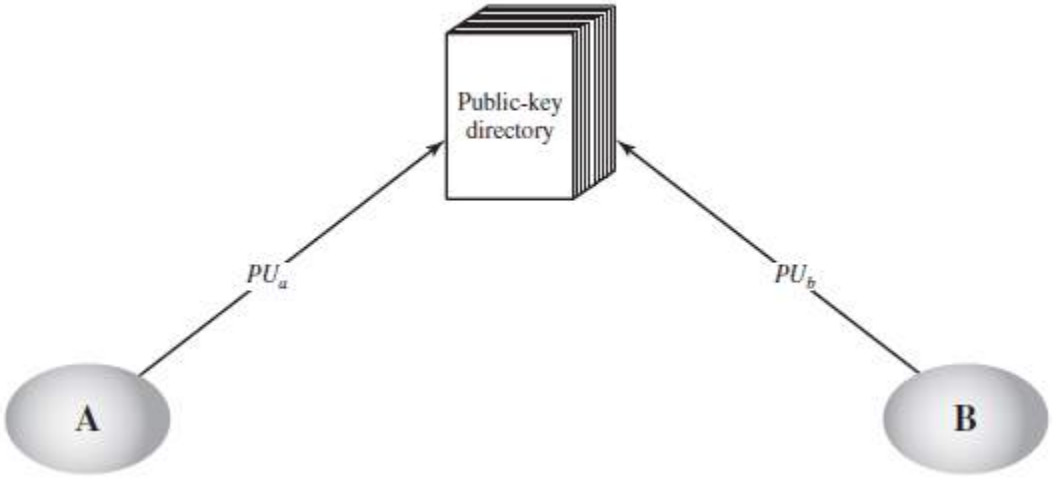


Figure: Public-Key Publication

- Such a scheme would include the following elements:
  1. The authority maintains **a directory** with a **{name, public key}** entry for each participant.
  2. Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.
  3. A participant may replace the existing key with a new one at any time, either because of the desire to replace a public key that has already been used for a large amount of data, or because the corresponding private key has been compromised in some way.
  4. Participants could also access the directory electronically. For this purpose secure, authenticated communication from the authority to the participant is mandatory.
- This scheme is clearly more secure than individual public announcements, but still has vulnerabilities.
    - If an opponent succeeds in obtaining or computing the private key of the directory authority, the opponent could authoritatively pass out counterfeit

public keys and subsequently impersonate any participant and eavesdrop on messages sent to any participant.

- Another way to achieve the same end is for the opponent to tamper with the records kept by the authority

Public-Key Authority

- Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory.
- As before, the scenario assumes that a central authority maintains a dynamic directory of public keys of all participants.
- In addition, each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key.

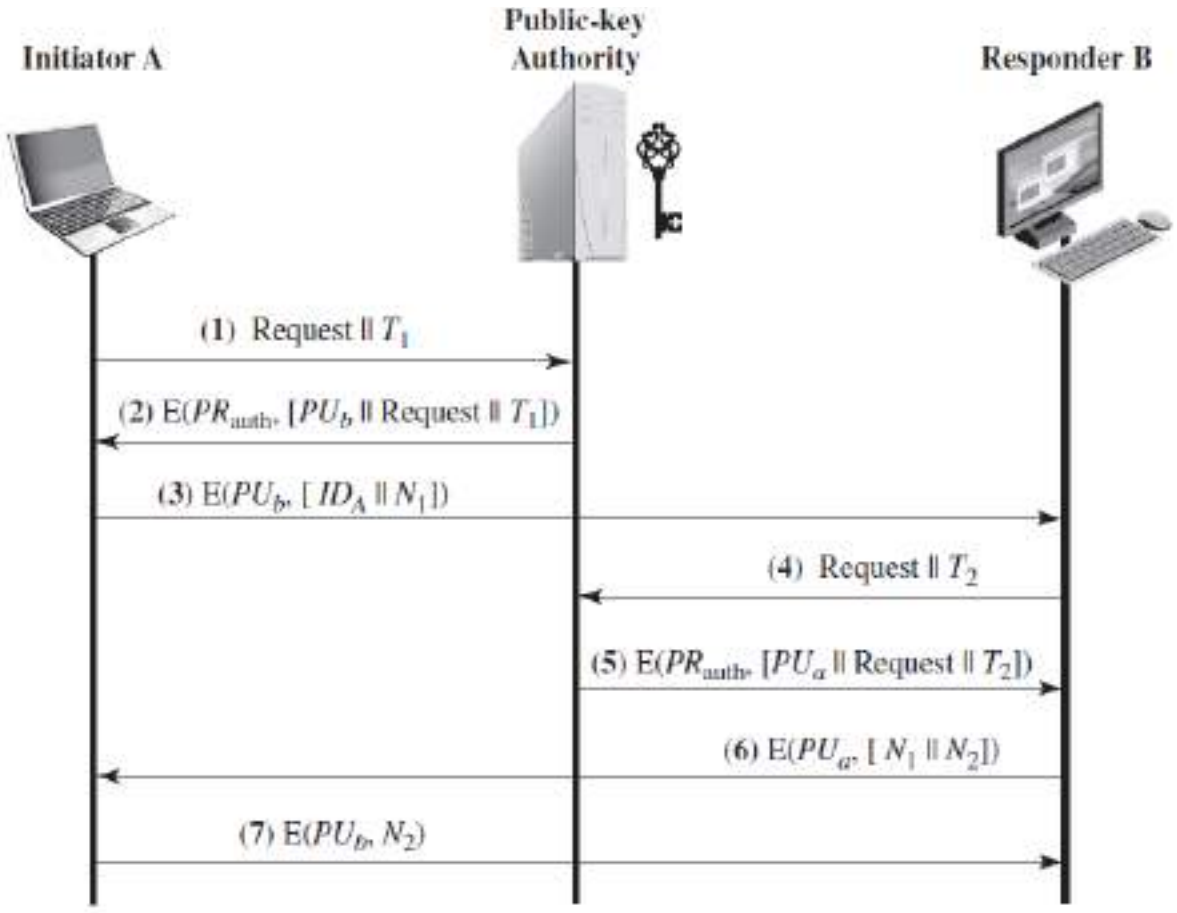


Figure: Public-Key Distribution Scenario

In this scheme the following steps (matched by number to figure above) occur.

1. A sends a timestamped message to the public-key authority containing a request for the current public key of **B**.

2. The authority responds with a message that is encrypted using the authority's private key, $\mathbf{PR_{auth}}$. Thus, **A** is able to decrypt the message using the authority's public key. Therefore, A is assured that the message originated with the authority. The message includes the following:

- **B'**s public key, $\mathbf{PU_b}$, which A can use to encrypt messages destined for **B**
- The original request used to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception by the authority
- The original timestamp given so A can determine that this is not an old message from the authority containing a key other than B's current public key identifier of **A** $\mathbf{(ID_A)}$ and a nonce $\mathbf{(N_1)}$, which is used to identify this transaction uniquely.

3. A stores **B'**s public key and also uses it to encrypt a message to **B** containing an identifier of A ($\mathbf{ID_A}$) and a nonce ($\mathbf{N_1}$), which is used to identify this transaction uniquely.

4, 5. B retrieves **A'**s public key from the authority in the same manner as **A** retrieved B's public key. At this point, public keys have been securely delivered to **A** and B, and they may begin their protected exchange. However, two additional steps are desirable:

6. B sends a message to **A** encrypted with $\mathbf{PU_a}$ and containing **A's** nonce ($\mathbf{N_1}$) as well as a new nonce generated by **B** ($\mathbf{N_2}$). Because only **B** could have decrypted message (3), the presence of $\mathbf{N_1}$ in message (6) assures A that the correspondent is **B**.

7. A returns $\mathbf{N_2}$, which is encrypted using **B's** public key, to assure **B** that its correspondent is A.

- Thus, a total of seven messages are required. However, the initial five messages need be used only infrequently because both A and B can save the other's public key for future use—

a technique known as caching. Periodically, a user should request fresh copies of the public keys of its correspondents to ensure currency.

- However this isn't perfect as the public-key authority could be somewhat of a bottleneck in the system. The reason for this is that a user must appeal to the authority for a public key for every other user that it wishes to contact. Also the directory of names and public keys maintained by the authority is vulnerable to tampering.

Public-Key Certificates

- An alternative approach to the above is the use of certificates that can be used by participants to exchange keys without contacting a public-key authority.
- Each certificate, containing a public key and other information, is created by a certificate authority and is given to the participant with the matching private key.
- A participant conveys its key information to another by transmitting its certificate.
- Other participants can verify that the certificate was created by the authority.
- Four requirements can be placed on this particular scheme:
    1. Any participant can read a certificate to determine the name and public key of the certificate's owner.
    2. Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.
    3. Only the certificate authority can create and update certificates.
    4. Any participant can verify the currency of the certificate.
- A certificate scheme is illustrated in following figure.

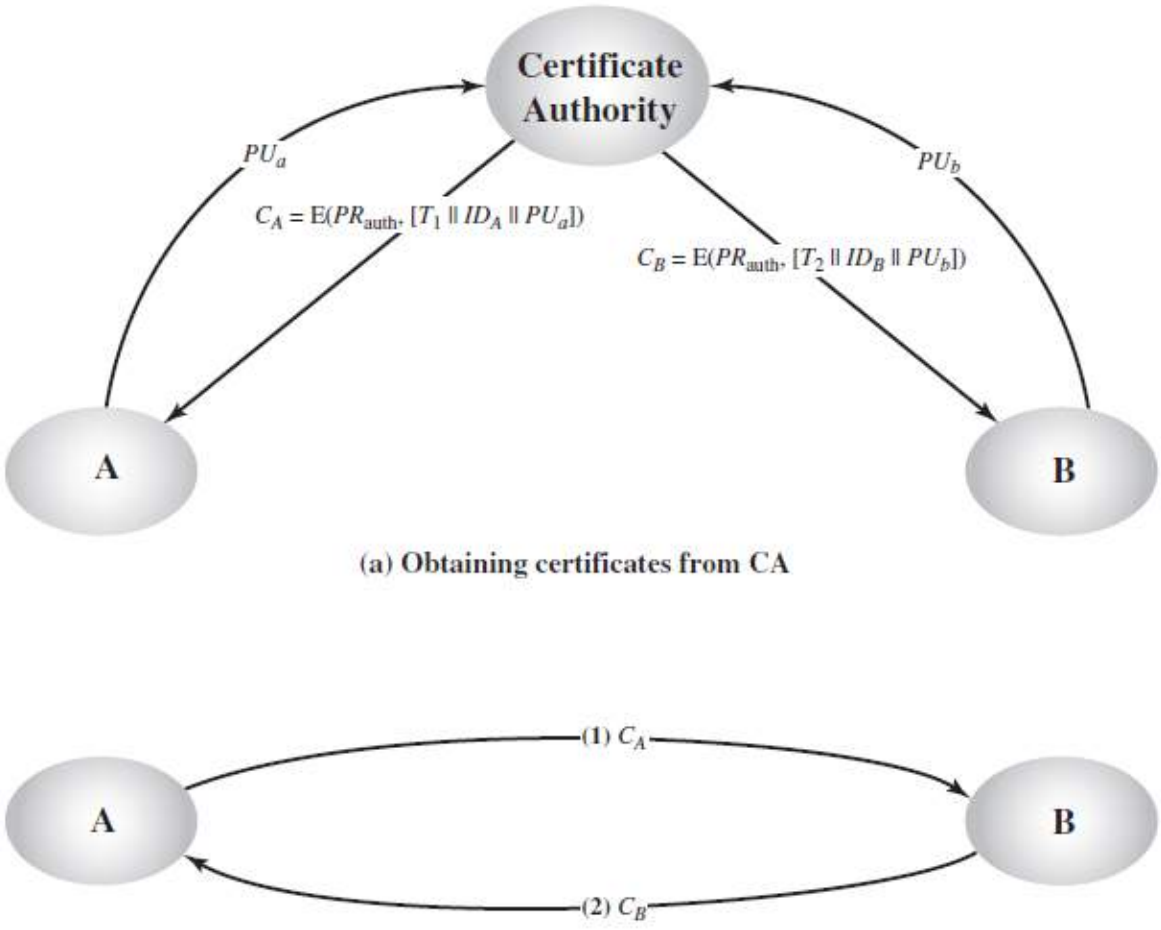


Figure: Exchange of Public-Key Certificates

- Each participant applies to the certificate authority, supplying a public key and requesting a certificate.
- Application must be in person or by some form of secure authenticated communication.
- For participant A, the authority provides a certificate of the form

$$C_A = \mathrm{E}(PR_{\text{auth}}, [T \| ID_A \| PU_a])$$

where **PR$_{\text{auth}}$** is the private key used by the authority and T is a timestamp.

- A may then pass this certificate on to any other participant, who reads and verifies the certificate as follows:

$$\mathrm{D}(PU_{\text{auth}}, C_A) = \mathrm{D}(PU_{\text{auth}}, \mathrm{E}(PR_{\text{auth}}, [T \| ID_A \| PU_a])) = (T \| ID_A \| PU_a)$$

- The recipient uses the authority's public key, **PU_{auth},** to decrypt the certificate.
- Because the certificate is readable only using the authority's public key, this verifies that the certificate came from the certificate authority.
- The elements **ID_A** and **PU_a** provide the recipient with the name and public key of the certificate's holder.
- The timestamp **T** validates the currency of the certificate. The timestamp counters the following scenario.
    - A's private key is learned by an adversary.
    - A generates a new private/public key pair and applies to the certificate authority for a new certificate.
    - Meanwhile, the adversary replays the old certificate to B.
    - If B then encrypts messages using the compromised old public key, the adversary can read those messages.
    - In this context, the compromise of a private key is comparable to the loss of a credit card. The owner cancels the credit card number but is at risk until all possible communicants are aware that the old credit card is obsolete. Thus, the timestamp serves as something like an expiration date. If a certificate is sufficiently old, it is assumed to be expired
- **X.509** certificates are used in most network security applications, including IP security, transport layer security (TLS), and S/MIME.

Distribution of secret key by using public key cryptography

- One of the most important uses of a public-key cryptosystem is to encrypt secret keys for distribution.

Simple Secret Key Distribution

- An extremely simple scheme was put forward by Merkle as in figure below.

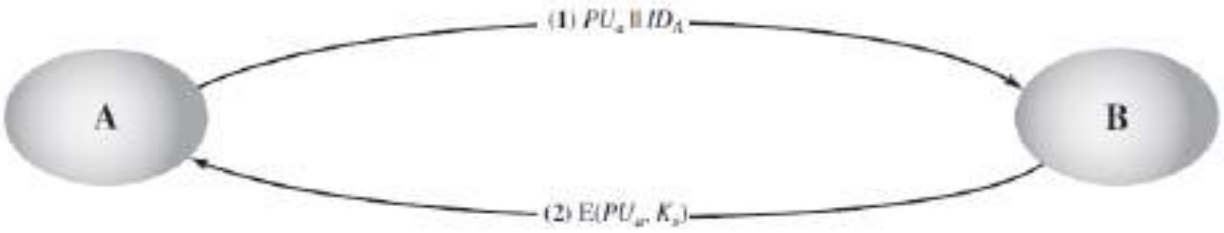


Figure: Simple Use of Public-Key Encryption to Establish a Session Key

If **A** wishes to communicate with **B**, the following procedure is employed:

1. **A** generates a public/private key pair **{PU_a, PR_a}** and transmits a message to **B** consisting of **PU_a** and an identifier of **A**, **ID_A**.
2. **B** generates a secret key, **K_s**, and transmits it to **A**, which is encrypted with **A's** public key.
3. **A** computes **D(PR_a, E(PU_a, K_s))** to recover the secret key. Because only **A** can decrypt the message, only **A** and **B** will know the identity of **K_s**.
4. **A** discards **PU_a** and **PR_a** and **B** discards **PU_a**.

- A and B can now securely communicate using conventional encryption and the session key **K_s**. At the completion of the exchange, both **A** and **B** discard **K_s**.
- Despite its simplicity, this is an attractive protocol. No keys exist before the start of the communication and none exist after the completion of communication. Thus, the risk of compromise of the keys is minimal.
- At the same time, the communication is secure from eavesdropping.

- The protocol is vulnerable to an active attack. If an opponent, **D**, has control of the intervening communications channel, then he can compromise the communications in the following way without being detected:

    1. A generates a public/private key pair **{PU_a, PR_a}** and transmits a message intended for B consisting of **PU_a** and an identifier of **A**, **ID_A**.

    2. **D** intercepts the message, creates its own public/private key pair **{PU_d, PR_d}** and transmits **PU_d || ID_A** to **B**.

    3. **B** generates a secret key, **K_s**, and transmits **E(PU_d, K_s).**

    4. **D** intercepts the message and learns **K_s** by computing **D(PR_d, E(PU_d, K_s))**.

    5. **D** transmits **E(PU_a, K_s)** to **A.**

- The result is that both **A** and **B** know **K_s** and are unaware that **K_s** has also been revealed to **D**. **A** and **B** can now exchange messages using **K_s**. **D** no longer actively interferes with the communications channel but simply eavesdrops. Knowing **K_s**, **D** can decrypt all messages, and both **A** and **B** are unaware of the problem.
- Thus, this simple protocol **is only useful in an environment where the only threat is eavesdropping**

Secret Key Distribution with Confidentiality and Authentication

An approach as in following figure, provides protection against both active and passive attacks. We begin at a point when it is assumed that **A** and **B** have exchanged public keys by one of the schemes. Then the following steps occur.

1. A uses B's public key to encrypt a message to B containing an identifier of A(ID_A) and a nonce (N_1), which is used to identify this transaction uniquely.
2. B sends a message to A encrypted with PU_a and containing A's nonce (N_1) as well as a new nonce generated by B (N_2). Because only B could have decrypted message (1), the presence of N_1 in message (2) assures A that the correspondent is B.
3. A returns N_2, encrypted using B's public key, to assure B that its correspondent is A.
4. A selects a secret key K_s and sends $M = \text{E}(PU_b, \text{E}(PR_a, K_s))$ to B. Encryption of this message with B's public key ensures that only B can read it; encryption with A's private key ensures that only A could have sent it.
5. B computes $\text{D}(PU_a, \text{D}(PR_b, M))$ to recover the secret key.

The result is that this scheme ensures both confidentiality and authentication in the exchange of a secret key.

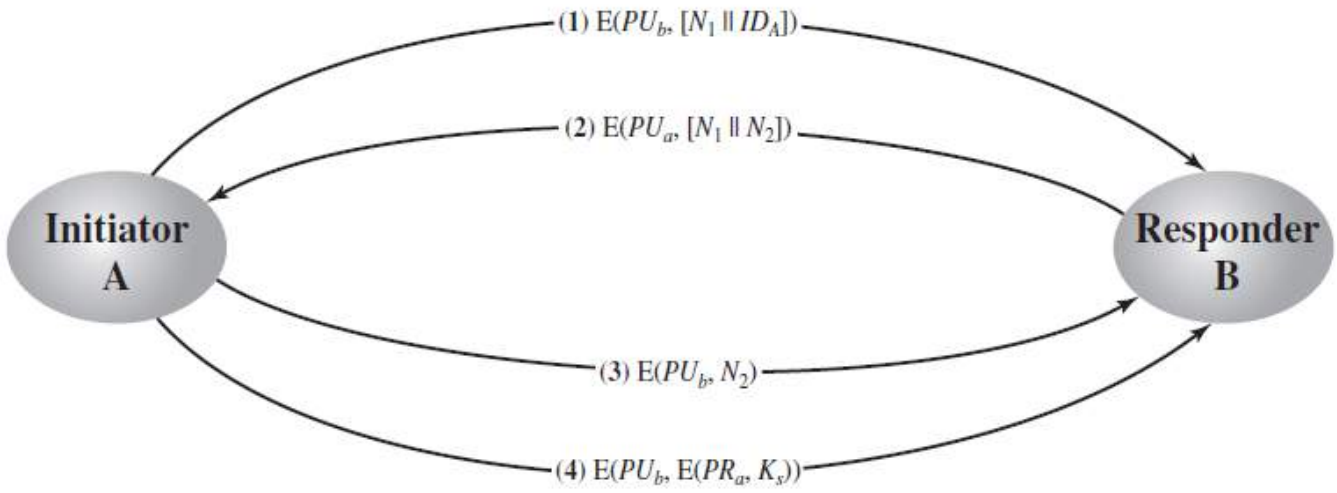


Figure: Public-Key Distribution of Secret Keys

Diffie-Helman Key Exchange

- The first published public-key algorithm appeared in the seminal paper by Diffie and Hellman that defined public-key cryptography and is generally referred to as

  Diffi- Hellman key exchange
- A number of commercial products employ this key exchange technique.
- The purpose of the algorithm is to enable two users to securely exchange a key that can then be used for subsequent symmetric encryption of messages.
- The algorithm itself is limited to the exchange of secret values.
- The Diffie-Hellman algorithm **depends for its effectiveness** on the **difficulty of computing discrete logarithmm**

The algorithm

Following figure summarizes the **Diffie-Hellman key exchange algorithm**. For this scheme, there are two publicly known numbers: a prime number **q** and an integer **a** that is a *primitive root* of q. Suppose the users **A** and **B** wish to create a shared key.

- User **A** selects a random integer $X_A < q$ and computes $\boldsymbol{Y_A = \alpha^{X_A} \bmod q}$.
- Similarly, user **B** independently selects a random integer $X_B < q$ and computes $\boldsymbol{Y_B = \alpha^{X_B} \bmod q}$.
- Each side keeps the **X** *value private* and makes the **Y** *value available publicly* to the other side.
  - Thus, $\mathbf{X_A}$ is **A'**s private key and $\mathbf{Y_A}$ is **A**'s corresponding public key, and similarly for B.
- User **A** computes the key as

$$\boldsymbol{K = (Y_B)^{X_A} mod\ q}$$

and user B computes the key as

$$\boldsymbol{K = (Y_A)^{X_B} mod\ q}$$

These two calculations produce identical results:

$$
\begin{aligned}
K &= (Y_B)^{X_A} \bmod q \\
&= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\
&= (\alpha^{X_B})^{X_A} \bmod q \\
&= \alpha^{X_B X_A} \bmod q \\
&= (\alpha^{X_A})^{X_B} \bmod q \\
&= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\
&= (Y_A)^{X_B} \bmod q
\end{aligned}
$$

by the rules of modular arithmetic

Alice **Bob**

Alice	Bob
Alice and Bob share a prime number q and an integer α, such that $\alpha < q$ and α is a primitive root of q	Alice and Bob share a prime number q and an integer α, such that $\alpha < q$ and α is a primitive root of q
Alice generates a private key X_A such that $X_A < q$	Bob generates a private key X_B such that $X_B < q$
Alice calculates a public key $Y_A = \alpha^{X_A} \bmod q$	Bob calculates a public key $Y_B = \alpha^{X_B} \bmod q$
Alice receives Bob's public key Y_B in plaintext	Bob receives Alice's public key Y_A in plaintext
Alice calculates shared secret key $K = (Y_B)^{X_A} \bmod q$	Bob calculates shared secret key $K = (Y_A)^{X_B} \bmod q$

Figure: The Diffie-Hellman Key Exchange

- The result is that the two sides have exchanged a secret value. Typically, this secret value is used as ***shared symmetric secret key***.
- Now consider an adversary who can observe the key exchange and wishes to determine the secret key **K**.
  - — Because $\mathbf{X_A}$ and $\mathbf{X_B}$ are private, an adversary only has the following ingredients to work with:

    q, α, $\mathbf{Y_A}$, and $\mathbf{Y_B}$.

  - — Thus, the adversary is forced to take a discrete logarithm to determine thekey. For example, to determine the private key of user **B**, an adversary must compute

$$X_B = \mathrm{d}\log_{\alpha,q}(Y_B)$$

  - — The adversary can then calculate the key **K** in the same manner as user **B** calculates it. That is, the adversary can calculate **K** as

$$K = (Y_A)^{X_B} \bmod q$$

NOTE: The **security** of the Diffie-Hellman key exchange lies in the fact that, **while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms**. For large primes, the latter task is considered infeasible.

Here is an example. Key exchange is based on the use of the prime number $q = 353$ and a primitive root of 353, in this case $\alpha = 3$. A and B select private keys $X_A = 97$ and $X_B = 233$, respectively. Each computes its public key:

A computes $Y_A = 3^{97} \bmod 353 = 40$.
B computes $Y_B = 3^{233} \bmod 353 = 248$.

After they exchange public keys, each can compute the common secret key:

A computes $K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160$.
B computes $K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$.

We assume an attacker would have available the following information:

$$q = 353; \alpha = 3; Y_A = 40; Y_B = 248$$

In this simple example, it would be possible by brute force to determine the secret key 160. In particular, an attacker **E** can determine the common key by discovering a solution to the equation $3^a \bmod 353 = 40$ or the equation $3^b \bmod 353 = 248$. The brute-force approach is to calculate powers of 3 modulo 353, stopping when the result equals either 40 or 248. The desired answer is reached with the exponent value of 97, which provides $3^{97} \bmod 353 = 40$. With larger numbers, the problem becomes impractical.

Man-in-the-Middle Attack

The Diffie-Helman Key Exchang protocol depicted in above figure is insecure against a **Man-in-the-middle attack**.

Suppose Alice and Bob wish to exchange keys, and Darth is the adversary. The attack proceeds as follows

1. Darth prepares for the attack by generating two random private keys $\mathbf{X_{D1}}$ and $\mathbf{X_{D2}}$ and then computing the corresponding public keys $\mathbf{Y_{D1}}$ and $\mathbf{Y_{D2}}$.
2. Alice transmits $\mathbf{Y_A}$ to Bob.
3. Darth intercepts $\mathbf{Y_A}$ and transmits $\mathbf{Y_{D1}}$ to Bob. Darth also calculates
$$K2 = (Y_A)^{X_{D2}} \bmod q.$$
4. Bob receives $\mathbf{Y_{D1}}$ and calculates
$$K1 = (Y_{D1})^{X_B} \bmod q.$$
5. . Bob transmits $\mathbf{Y_B}$ to Alice.
6. Darth intercepts $\mathbf{Y_B}$ and transmits $\mathbf{Y_{D2}}$ to Alice. Darth calculates
$$K1 = (Y_B)^{X_{D1}} \bmod q.$$
7. Alice receives $\mathbf{Y_{D2}}$ and calculates

$$K2 = (Y_{D2})^{X_A} \bmod q.$$

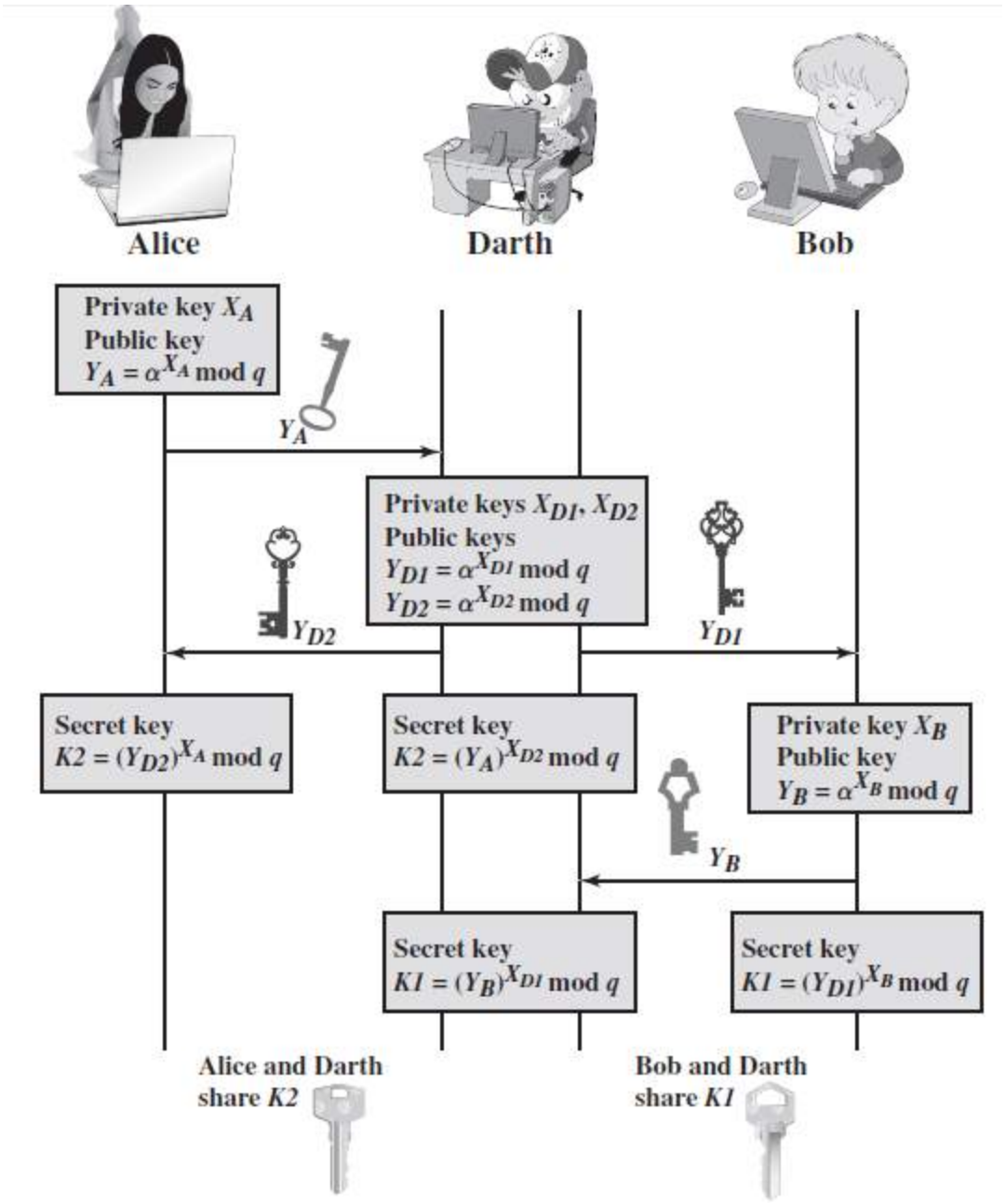


Figure: Man-in-the-Middle Attack

- At this point, Bob and Alice think that they share a secret key, but instead Bob and Darth share secret key **K1** and Alice and Darth share secret key **K2**. All future communication between Bob and Alice is compromised in the following way.

  1. Alice sends an encrypted message **M: E (K2, M).**

  2. Darth intercepts the encrypted message and decrypts it to recover **M**.

3. Darth sends Bob **E(K1, M)** or **E(K1, M′),** where **M′** is any message.

   — In the first case, Darth simply wants to eavesdrop on the communication without altering it.

   — In the second case, Darth wants to modify the message going to Bob.

❖ The key exchange protocol is vulnerable to such an attack because it does not authenticate the participants. This vulnerability can be overcome with the use of digital signatures and public-key certificates.

ElGamal Cryptographic System

- In 1984, T. Elgamal announced a public-key scheme based on discrete logarithms, closely related to the Diffie-Hellman technique
- The ElGamal cryptosystem is used in some form in a number of standards including the digital signature standard (DSS).
- ElGamal encryption is an public-key cryptosystem.
- It uses asymmetric key encryption for communicating between two parties and encrypting the message.
- This cryptosystem is based on the difficulty of finding discrete logarithm in a cyclic group.
- As with Diffie-Hellman, the global elements of Elgamal are a prime number **q** and $\boldsymbol{\alpha}$, which is a primitive root of **q**.

- The Steps in ElGamal Algorithm are summarized as follow:
  [Alice generates a public/private key pair; Bob encrypts using Alice's public key; and Alice decrypts using her private key.]

Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

Key Generation by Alice	
Select private X_A	$X_A < q - 1$
Calculate Y_A	$Y_A = \alpha^{X_A} \bmod q$
Public key	$\{q, \alpha, Y_A\}$
Private key	X_A

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < q$
Select random integer k	$k < q$
Calculate K	$K = (Y_A)^k \bmod q$
Calculate C_1	$C_1 = \alpha^k \bmod q$
Calculate C_2	$C_2 = KM \bmod q$
Ciphertext:	(C_1, C_2)

Decryption by Alice with Alice's Private Key	
Ciphertext:	(C_1, C_2)
Calculate K	$K = (C_1)^{X_A} \bmod q$
Plaintext:	$M = (C_2 K^{-1}) \bmod q$

From this, Elgamal process can be described as :

- Bob generates a random integer **k**.
- Bob generates a one-time key **K** using Alice's public-key components $\mathbf{Y_A}$, **q,** and **k**.
- Bob encrypts **k** using the public-key component a, yielding $\mathbf{C_1}$. $\mathbf{C_1}$ provides sufficient information for Alice to recover **K**.
- Bob encrypts the plaintext message **M** using **K**.
- Alice recovers **K** from $\mathbf{C_1}$ using her private key.
- Alice uses $\mathbf{K^{-1}}$ to recover the plaintext message from $\mathbf{C_2}$.

*NOTE: **K** functions as a one-time key, used to encrypt and decrypt the message*

Why the Elgamal scheme works?

Let us demonstrate why the Elgamal scheme works. First, we show how **K** is recovered by the decryption process:

$K = (Y_A)^k \bmod q$ — K is defined during the encryption process

$K = (\alpha^{X_A} \bmod q)^k \bmod q$ — substitute using $Y_A = \alpha^{X_A} \bmod q$

$K = \alpha^{kX_A} \bmod q$ — by the rules of modular arithmetic

$K = (C_1)^{X_A} \bmod q$ — substitute using $C_1 = \alpha^k \bmod q$

Next, using K, we recover the plaintext as

$$C_2 = KM \bmod q$$

$$(C_2K^{-1}) \bmod q = KMK^{-1} \bmod q = M \bmod q = M$$

Example: Let us start with the prime field **GF(19);** that is, **q** = 19. It has primitive roots {2, 3, 10, 13, 14, 15}. We choose $\boldsymbol{\alpha}$ = 10.

Alice generates a key pair as follows

1. Alice chooses $X_A = 5$.
2. Then $Y_A = \alpha^{X_A} \bmod q = \alpha^5 \bmod 19 = 3$
3. Alice's private key is 5 and Alice's public key is $\{q, \alpha, Y_A\} = \{19, 10, 3\}$.

Suppose Bob wants to send the message with the value $M = 17$. Then:

1. Bob chooses $k = 6$.
2. Then $K = (Y_A)^k \bmod q = 3^6 \bmod 19 = 729 \bmod 19 = 7$.
3. So
   $C_1 = \alpha^k \bmod q = \alpha^6 \bmod 19 = 11$
   $C_2 = \text{KM} \bmod q = 7 \times 17 \bmod 19 = 119 \bmod 19 = 5$
4. Bob sends the ciphertext (11, 5).

For decryption:

1. Alice calculates $K = (C_1)^{X_A} \bmod q = 11^5 \bmod 19 = 161051 \bmod 19 = 7$.
2. Then K^{-1} in GF(19) is $7^{-1} \bmod 19 = 11$.
3. Finally, $M = (C_2 K^{-1}) \bmod q = 5 \times 11 \bmod 19 = 55 \bmod 19 = 17$.

NOTE:

- If a message must be broken up into blocks and sent as a sequence of encrypted blocks, a unique value of **k** should be used for each block. If **k** is used for more than one block, knowledge of one block $\mathbf{M_1}$ of the message enables the user to compute other blocks as follows. Let

$$C_{1,1} = \alpha^k \bmod q;\ C_{2,1} = KM_1 \bmod q$$
$$C_{1,2} = \alpha^k \bmod q;\ C_{2,2} = KM_2 \bmod q$$

  Then

$$\frac{C_{2,1}}{C_{2,2}} = \frac{KM_1 \bmod q}{KM_2 \bmod q} = \frac{M_1 \bmod q}{M_2 \bmod q}$$

  If $\mathbf{M_1}$ is known, then $\mathbf{M_2}$ is easily computed as

$$M_2 = (C_{2,1})^{-1}\, C_{2,2}\, M_1 \bmod q$$

- The **security of Elgamal is based on the difficulty of computing discrete logarithms**.
  - To recover **A'**s private key, an adversary would have to compute

    $\mathbf{X_A = dlog_{a,q}(Y_A)}$.
  - Alternatively, to recover the one-time key **K**, an adversary would have to determine the random number **k**, and this would require computing the discrete logarithm

    $\mathbf{k = dlog_{a,q}(C_1)}$.
- These calculations are regarded as infeasible if **p** is at least 300 decimal digits and **q - 1** has at least one "large" prime factor.