Module 4: More Number Theory

Prime Numbers:

- **An integer p > 1 is a prime number if and only if its only divisors are 1 and p.**
- **Any integer a > 1 can be factored in a unique way**
- **where p1 < p2 <...< p_t are prime numbers and where each a_i is a positive integer.**

Any integer $a > 1$ can be factored in a unique way as

$$a = p_1^{a_1} \times p_2^{a_2} \times \cdots \times p_t^{a_t} \quad \textbf{(8.1)}$$

where $p_1 < p_2 < \ldots < p_t$ are prime numbers and where each a_i is a positive integer. This is known as the fundamental theorem of arithmetic; a proof can be found in any text on number theory.

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

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.

The value of any given positive integer can be specified by simply listing all the nonzero exponents in the foregoing formulation.

The integer 12 is represented by $\{a_2 = 2, \ a_3 = 1\}$.
The integer 18 is represented by $\{a_2 = 1, \ a_3 = 2\}$.
The integer 91 is represented by $\{a_7 = 1, a_{13} = 1\}$.

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 $k = ab$. We know that the integer k can be expressed as the product of powers of primes: $k = \prod_{p \in P} p^{k_p}$. It follows that $k_p = a_p + b_p$ for all $p \in P$.

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

What does it mean, in terms of the prime factors of a and b, to say that a divides b? Any integer of the form p^n can be divided only by an integer that is of a lesser or equal power of the same prime number, p^j with $j \leq n$. Thus, we can say the following.

Given

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

If a|b, then $a_p \leq b_p$ 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:

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

FERMAT'S AND EULER'S THEOREMS:

Two theorems that play important roles in public-key cryptography are Fermat's theorem and Euler's theorem.

Fermat's Theorem:

Fermat's theorem states the following: If p is prime and a is a positive integer not divisible by p, then,

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

Proof: Consider the set of positive integers less than p: $\{1, 2, \ldots, p-1\}$ and multiply each element by a, modulo p, to get the set $X = \{a \bmod p, 2a \bmod p, \ldots, (p-1)a \bmod p\}$. None of the elements of X is equal to zero because p does not divide a. Furthermore, no two of the integers in X are equal. To see this, assume that $ja \equiv ka \pmod{p}$, where $1 \leq j < k \leq p-1$. Because a is relatively prime to p,
resulting in $j \equiv k \pmod{p}$. This last equality is impossible, because j and k are both positive integers less than p. Therefore, we know that the $(p-1)$ elements of X are all positive integers with no two elements equal. We can conclude the X consists of the set of integers $\{1, 2, \ldots, p-1\}$ in some order. Multiplying the numbers in both sets (p and X) and taking the result mod p yields

$$a \times 2a \times \cdots \times (p-1)a \equiv [(1 \times 2 \times \cdots \times (p-1)] \pmod{p}$$
$$a^{p-1}(p-1)! \equiv (p-1)! \pmod{p}$$

We can cancel the $(p-1)!$ term because it is relatively prime to p

$a^{p-1} \equiv 1 \pmod{p}$ **which completes the proof.**

$a = 7, p = 19$
$7^2 = 49 \equiv 11 \pmod{19}$
$7^4 \equiv 121 \equiv 7 \pmod{19}$
$7^8 \equiv 49 \equiv 11 \pmod{19}$
$7^{16} \equiv 121 \equiv 7 \pmod{19}$
$a^{p-1} = 7^{18} = 7^{16} \times 7^2 \equiv 7 \times 11 \equiv 1 \pmod{19}$

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

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

Euler's Totient Function

Before presenting Euler's theorem, we need to introduce an important quantity in number theory, referred to as **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 $\phi(35) = 24$.

Table 8.2 lists the first 30 values of $\phi(n)$. The value $\phi(1)$ 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 $p \neq q$. Then we can show that, for $n = pq$,

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

To see that $\phi(n) = \phi(p) \times \phi(q)$, consider that the set of positive integers less that n is the set $\{1, \ldots, (pq - 1)\}$. The integers in this set that are not relatively prime to n are the set $\{p, 2p, \ldots, (q - 1)p\}$ and the set $\{q, 2q, \ldots, (p - 1)q\}$. Accordingly,

$$\begin{aligned}\phi(n) &= (pq - 1) - [(q - 1) + (p - 1)] \\ &= pq - (p + q) + 1 \\ &= (p - 1) \times (q - 1) \\ &= \phi(p) \times \phi(q)\end{aligned}$$

Table 8.2 Some Values of Euler's Totient Function $\phi(n)$

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

Euler's theorem states that for every a and n that are relatively prime:

$$a^{\phi(n)} \equiv 1 (\text{mod } n) \qquad \textbf{(8.4)}$$

Proof: Equation (8.4) is true if n is prime, because in that case, $\phi(n) = (n - 1)$ and Fermat's theorem holds. However, it also holds for any integer n. Recall that $\phi(n)$ is the number of positive integers less than n that are relatively prime to n. Consider the set of such integers, labeled as

$$R = \{x_1, x_2, \ldots, x_{\phi(n)}\}$$

That is, each element x_i of R is a unique positive integer less than n with $\gcd(x_i, n) = 1$. Now multiply each element by a, modulo n:

$$S = \{(ax_1 \bmod n), (ax_2 \bmod n), \ldots, (ax_{\phi(n)} \bmod n)\}$$

The set S is a permutation of R, by the following line of reasoning:

1. Because a is relatively prime to n and x_i is relatively prime to n, ax_i must also be relatively prime to n. Thus, all the members of S are integers that are less than n and that are relatively prime to n.
2. There are no duplicates in S. If $ax_i \bmod n = ax_j \bmod n$, then $x_i = x_j$.

Therefore,

$$\prod_{i=1}^{\phi(n)} (ax_i \bmod n) = \prod_{i=1}^{\phi(n)} x_i$$

$$\prod_{i=1}^{\phi(n)} ax_i \equiv \prod_{i=1}^{\phi(n)} x_i \pmod{n}$$

$$a^{\phi(n)} \times \left[\prod_{i=1}^{\phi(n)} x_i\right] \equiv \prod_{i=1}^{\phi(n)} x_i \pmod{n}$$

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

which completes the proof. This is the same line of reasoning applied to the proof of Fermat's theorem.

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

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

$$a^{\phi(n)+1} \equiv a \pmod{n} \qquad \textbf{(8.5)}$$

TESTING FOR PRIMALITY:

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

Miller-Rabin Algorithm:

The algorithm due to Miller and Rabin is typically used to test a large number for primality. First, any positive odd integer $n \geq 3$ can be expressed as,

$$n - 1 = 2^k q \qquad \text{with } k > 0, q \text{ odd}$$

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 $a^2 \bmod p = 1$ if and only if either $a \bmod p = 1$ or $a \bmod p = -1 \bmod p = p - 1$. By the rules of modular arithmetic $(a \bmod p)(a \bmod p) = a^2 \bmod p$. Thus, if either $a \bmod p = 1$ or $a \bmod p = -1$, then $a^2 \bmod p = 1$. Conversely, if $a^2 \bmod p = 1$, then $(a \bmod p)^2 = 1$, which is true only for $a \bmod p = 1$ or $a \bmod p = -1$.

The **second property** is stated as follows: Let p be a prime number greater than 2. We can then write $p - 1 = 2^k q$ 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. a^q is congruent to 1 modulo p. That is, $a^q \bmod p = 1$, or equivalently, $a^q \equiv 1 (\bmod p)$.
2. One of the numbers $a^q, a^{2q}, a^{4q}, \ldots, a^{2^{k-1}q}$ is congruent to -1 modulo p. That is, there is some number j in the range $(1 \le j \le k)$ such that $a^{2^{j-1}q} \bmod p = -1 \bmod p = p - 1$ or equivalently, $a^{2^{j-1}q} \equiv -1 (\bmod p)$.

Proof: Fermat's theorem states that $a^{n-1} \equiv 1 (\bmod n)$ if n is prime. We have $p - 1 = 2^k q$. Thus, we know that $a^{p-1} \bmod p = a^{2^k q} \bmod p = 1$. Thus, if we look at the sequence of numbers

$$a^q \bmod p,\ a^{2q} \bmod p, a^{4q} \bmod p, \ldots, a^{2^{k-1}q} \bmod p, a^{2^k q} \bmod p \qquad \textbf{(8.6)}$$

we know that the last number in the list has value 1. Further, each number in the list is the square of the previous number. Therefore, one of the following possibilities must be true.

1. The first number on the list, and therefore all subsequent numbers on the list, equals 1.
2. Some number on the list does not equal 1, but its square mod p does equal 1. By virtue of the first property of prime numbers defined above, we know that the only number that satisfies this condition is $p - 1$. So, in this case, the list contains an element equal to $p - 1$.

This completes the proof.

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 $n = 2047 = 23 \times 89$, then $n - 1 = 2 \times 1023$. We compute $2^{1023} \bmod 2047 = 1$, 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.

```
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.

Repeated Use of the Miller-Rabin Algorithm How can we use the Miller-Rabin algorithm to determine with a high degree of confidence whether or not an integer is prime? It can be shown that given an odd number n that is not prime and a randomly chosen integer, a with $1 < a < n - 1$, the probability that TEST will return `inconclusive` (i.e., fail to detect that n is not prime) is less than 1/4. Thus, if t different values of a are chosen, the probability that all of them will pass TEST (return inconclusive) for n is less than $(1/4)^t$. For example, for $t = 10$, the probability that a nonprime number will pass all ten tests is less than 10^{-6}. Thus, for a sufficiently large value of t, we can be confident that n is prime if Miller's test always returns `inconclusive`.

This gives us a basis for determining whether an odd integer n is prime with a reasonable degree of confidence. The procedure is as follows: Repeatedly invoke TEST (n) using randomly chosen values for a. If, at any point, TEST returns `composite`, then n is determined to be nonprime. If TEST continues to return `inconclusive` for t tests, then for a sufficiently large value of t, assume that n is prime.

THE CHINESE REMAINDER THEOREM:

One of the most useful results of number theory is the **Chinese remainder theorem** (CRT). In essence, the CRT says it is possible to reconstruct integers in a certain range from their residues modulo a set of pairwise relatively prime moduli.

> The 10 integers in Z_{10}, that is the integers 0 through 9, can be reconstructed from their two residues modulo 2 and 5 (the relatively prime factors of 10). Say the known residues of a decimal digit x are $r_2 = 0$ and $r_5 = 3$; that is, $x \bmod 2 = 0$ and $x \bmod 5 = 3$. Therefore, x is an even integer in Z_{10} whose remainder, on division by 5, is 3. The unique solution is $x = 8$.

Let,

$$M = \prod_{i=1}^{k} m_i$$

where the m_i are pairwise relatively prime; that is, $\gcd(m_i, m_j) = 1$ for $1 \le i, j \le k$, and $i \ne j$. We can represent any integer A in Z_M by a k-tuple whose elements are in Z_{m_i} using the following correspondence:

$$A \leftrightarrow (a_1, a_2, \ldots, a_k) \tag{8.7}$$

where $A \in Z_M$, $a_i \in Z_{m_i}$, and $a_i = A \bmod m_i$ for $1 \le i \le k$. The CRT makes two assertions.

1. The mapping of Equation (8.7) is a one-to-one correspondence (called a **bijection**) between Z_M and the Cartesian product $Z_{m_1} \times Z_{m_2} \times \ldots \times Z_{m_k}$. That is, for every integer A such that $0 \le A \le M$, there is a unique k-tuple $(a_1, a_2, \ldots, a_k)$ with $0 \le a_i < m_i$ that represents it, and for every such k-tuple $(a_1, a_2, \ldots, a_k)$, there is a unique integer A in Z_M.
2. Operations performed on the elements of Z_M can be equivalently performed on the corresponding k-tuples by performing the operation independently in each coordinate position in the appropriate system.

Let us demonstrate the **first assertion**. The transformation from A to $(a_1, a_2, \ldots, a_k)$, is obviously unique; that is, each a_i is uniquely calculated as $a_i = A \bmod m_i$. Computing A from $(a_1, a_2, \ldots, a_k)$ can be done as follows. Let $M_i = M/m_i$ for $1 \le i \le k$. Note that $M_i = m_1 \times m_2 \times \ldots \times m_{i-1} \times m_{i+1} \times \ldots \times m_k$, so that $M_i \equiv 0 \pmod{m_j}$ for all $j \ne i$. Then let

$$c_i = M_i \times \left(M_i^{-1} \bmod m_i\right) \qquad \text{for } 1 \le i \le k \tag{8.8}$$

By the definition of M_i, it is relatively prime to m_i and therefore has a unique multiplicative inverse mod m_i. So Equation (8.8) is well defined and produces a unique value c_i. We can now compute

$$A \equiv \left(\sum_{i-1}^{k} a_i c_i\right) \pmod{M} \tag{8.9}$$

To show that the value of A produced by Equation (8.9) is correct, we must show that $a_i = A \bmod m_i$ for $1 \le i \le k$. Note that $c_j \equiv M_j \equiv 0 \pmod{m_i}$ if $j \neq i$, and that $c_i \equiv 1 \pmod{m_i}$. It follows that $a_i = A \bmod m_i$.

The **second assertion** of the CRT, concerning arithmetic operations, follows from the rules for modular arithmetic. That is, the second assertion can be stated as follows: If

$$A \leftrightarrow (a_1, a_2, \ldots, a_k)$$
$$B \leftrightarrow (b_1, b_2, \ldots, b_k)$$

then

$$(A + B) \bmod M \leftrightarrow ((a_1 + b_1) \bmod m_1, \ldots, (a_k + b_k) \bmod m_k)$$
$$(A - B) \bmod M \leftrightarrow ((a_1 - b_1) \bmod m_1, \ldots, (a_k - b_k) \bmod m_k)$$
$$(A \times B) \bmod M \leftrightarrow ((a_1 \times b_1) \bmod m_1, \ldots, (a_k \times b_k) \bmod m_k)$$

One of the useful features of the Chinese remainder theorem is that it provides a way to manipulate (potentially very large) numbers mod M in terms of tuples of smaller numbers. This can be useful when M is 150 digits or more. However, note that it is necessary to know beforehand the factorization of M.

To represent 973 mod 1813 as a pair of numbers mod 37 and 49, define

$$m_1 = 37$$
$$m_2 = 49$$
$$M = 1813$$
$$A = 973$$

We also have $M_1 = 49$ and $M_2 = 37$. Using the extended Euclidean algorithm, we compute $M_1^{-1} = 34 \bmod m_1$ and $M_2^{-1} = 4 \bmod m_2$. (Note that we only need to compute each M_i and each M_i^{-1} once.) Taking residues modulo 37 and 49, our representation of 973 is (11, 42), because 973 mod 37 = 11 and 973 mod 49 = 42.

Now suppose we want to add 678 to 973. What do we do to (11, 42)? First we compute (678) $\leftrightarrow$ (678 mod 37, 678 mod 49) = (12, 41). Then we add the tuples element-wise and reduce (11 + 12 mod 37, 42 + 41 mod 49) = (23, 34). To verify that this has the correct effect, we compute

$$\begin{aligned}(23, 34) &\leftrightarrow a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1} \bmod M \\ &= [(23)(49)(34) + (34)(37)(4)] \bmod 1813 \\ &= 43350 \bmod 1813 \\ &= 1651\end{aligned}$$

and check that it is equal to (973 + 678) mod 1813 = 1651. Remember that in the above derivation, M_i^{-1} is the multiplicative inverse of M_1 modulo m_1 modulo M_2^{-1} is the multiplicative inverse of M_2 modulo m_2.

Suppose we want to multiply 1651 (mod 1813) by 73. We multiply (23, 34) by 73 and reduce to get (23 $\times$ 73 mod 37, 34 $\times$ 73 mod 49) = (14, 32). It is easily verified that

$$\begin{aligned}(14, 32) &\leftrightarrow [(14)(49)(34) + (32)(37)(4)] \bmod 1813 \\ &= 865 \\ &= 1651 \times 73 \bmod 1813\end{aligned}$$

DISCRETE LOGARITHMS:

Discrete logarithms are fundamental to a number of public-key algorithms, including Diffie-Hellman key exchange and the digital signature algorithm (DSA).

The Powers of an Integer, Modulo n:
Recall from Euler's theorem that, for every a and n that are relatively prime,

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

where $\phi(n)$, Euler's totient function, is the number of positive integers less than n and relatively prime to n. Now consider the more general expression:

$$a^m \equiv 1 \pmod{n} \qquad \textbf{(8.10)}$$

If a and n are relatively prime, then there is at least one integer m that satisfies Equation (8.10), namely, $M = \phi(n)$. The least positive exponent m for which Equation (8.10) holds is referred to in several ways:

- The order of a (mod n)
- The exponent to which a belongs (mod n)
- The length of the period generated by a

To see this last point, consider the powers of 7, modulo 19:

$$\begin{aligned}
7^1 &\equiv & &\equiv 7 \pmod{19} \\
7^2 &= 49 = 2 \times 19 + 11 & &\equiv 11 \pmod{19} \\
7^3 &= 343 = 18 \times 19 + 1 & &\equiv 1 \pmod{19} \\
7^4 &= 2401 = 126 \times 19 + 7 & &\equiv 7 \pmod{19} \\
7^5 &= 16807 = 884 \times 19 + 11 & &\equiv 11 \pmod{19}
\end{aligned}$$

There is no point in continuing because the sequence is repeating. This can be proven by noting that $7^3 \equiv 1 \pmod{19}$, and therefore, $7^{3+j} \equiv 7^3 7^j \equiv 7^j \pmod{19}$, and hence, any two powers of 7 whose exponents differ by 3 (or a multiple of 3) are congruent to each other (mod 19). In other words, the sequence is periodic, and the length of the period is the smallest positive exponent m such that $7^m \equiv 1 \pmod{19}$.

Table 8.3 shows all the powers of a, modulo 19 for all positive $a < 19$. The length of the sequence for each base value is indicated by shading. Note the following:

1. All sequences end in 1. This is consistent with the reasoning of the preceding few paragraphs.
2. The length of a sequence divides $\phi(19) = 18$. That is, an integral number of sequences occur in each row of the table.
3. Some of the sequences are of length 18. In this case, it is said that the base integer a generates (via powers) the set of nonzero integers modulo 19. Each such integer is called a primitive root of the modulus 19.

Table 8.3 Powers of Integers, Modulo 19

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

More generally, we can say that the highest possible exponent to which a number can belong (mod n) is $\phi(n)$. If a number is of this order, it is referred to as a **primitive root** of n. The importance of this notion is that if a is a primitive root of n, then its powers

$$a, a^2, \ldots, a^{\phi(n)}$$

are distinct (mod n) and are all relatively prime to n. In particular, for a prime number p, if a is a primitive root of p, then

$$a, a^2, \ldots, a^{p-1}$$

are distinct (mod p). For the prime number 19, its primitive roots are 2, 3, 10, 13, 14, and 15.

Not all integers have primitive roots. In fact, the only integers with primitive roots are those of the form 2, 4, p^α, and $2p^\alpha$, where p is any odd prime and α is a positive integer.

Logarithms for Modular Arithmetic

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

An analogous function exists for modular arithmetic. Let us briefly review the 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

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

$$\log_x(yz) = log_x(y) + \log_x(z) \quad \textbf{(8.11)}$$
$$\log_x(y^r) = r \times \log_x(y) \quad \textbf{(8.12)}$$

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. We also know that any integer b satisfies

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

by the definition of modular arithmetic. It follows that 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 $\text{dlog}_{a,p}(b)$.

Note the following:

$$\text{dlog}_{a,p}(1) = 0 \text{ because } a^0 \bmod p = 1 \bmod p = 1 \quad \textbf{(8.13)}$$
$$\text{dlog}_{a,p}(a) = 1 \text{ because } a^1 \bmod p = a \quad \textbf{(8.14)}$$

Here is an example using a nonprime modulus, $n = 9$. Here $\phi(n) = 6$ and $a = 2$ is a primitive root. We compute the various powers of a and find

$$\begin{aligned} 2^0 &= 1 & 2^4 &\equiv 7 \pmod 9 \\ 2^1 &= 2 & 2^5 &\equiv 5 \pmod 9 \\ 2^2 &= 4 & 2^6 &\equiv 1 \pmod 9 \\ 2^3 &= 8 \end{aligned}$$

This gives us the following table of the numbers with given discrete logarithms (mod 9) for the root $a = 2$:

Logarithm	0	1	2	3	4	5
Number	1	2	4	8	7	5

To make it easy to obtain the discrete logarithms of a given number, we rearrange the table:

Number	1	2	4	5	7	8
Logarithm	0	1	2	5	4	3

Now consider

$$\begin{aligned} x &= a^{\text{dlog}_{a,p}(x)} \bmod p \qquad y = a^{\text{dlog}_{a,p}(y)} \bmod p \\ xy &= a^{\text{dlog}_{a,p}(xy)} \bmod p \end{aligned}$$

Using the rules of modular multiplication,

$$\begin{aligned} xy \bmod p &= [(x \bmod p)(y \bmod p)] \bmod p \\ a^{\text{dlog}_{a,p}(xy)} \bmod p &= \left[\left(a^{\text{dlog}_{a,p}(x)} \bmod p\right)\left(a^{\text{dlog}_{a,p}(y)} \bmod p\right) \right] \bmod p \\ &= \left(a^{\text{dlog}_{a,p}(x)+\text{dlog}_{a,p}(y)}\right) \bmod p \end{aligned}$$

But now consider Euler's theorem, which states that, for every a and n that are relatively prime,

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

Any positive integer z can be expressed in the form $z = q + k\phi(n)$, with $0 \leq q < \phi(n)$. Therefore, by Euler's theorem,

$$a^z \equiv a^q \pmod n \qquad \text{if } z \equiv q \bmod \phi(n)$$

Applying this to the foregoing equality, we have

$$\text{dlog}_{a,p}(xy) \equiv [\text{dlog}_{a,p}(x) + \text{dlog}_{a,p}(y)] \pmod{\phi(p)}$$

and generalizing,

$$\text{dlog}_{a,p}(y^r) \equiv [r \times \text{dlog}_{a,p}(y)] \pmod{\phi(p)}$$

This demonstrates the analogy between true logarithms and discrete logarithms.

Keep in mind that unique discrete logarithms mod m to some base a exist only if a is a primitive root of m.

Table 8.4, which is directly derived from Table 8.3, shows the sets of discrete logarithms that can be defined for modulus 19.

Calculation of Discrete Logarithms

Consider the equation,

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

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). The difficulty seems to be on the same order of magnitude as that of factoring primes required for RSA. At the time of this writing, the asymptotically fastest known algorithm for taking discrete logarithms modulo a prime number is on the order of

$$e^{((\ln p)^{1/3}(\ln(\ln p))^{2/3})}$$

which is not feasible for large primes.

Table 8.4 Tables of Discrete Logarithms, Modulo 19

(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

Module 4: Principles of Public – key Cryptosystems

Principles of Public – key Cryptosystems:

Why Public-Key Cryptography?

- developed to address two key issues:
    - key distribution – how to have secure communications in general without having to trust a KDC with your key.
    - digital signatures – how to verify a message comes intact from the claimed sender
- public invention due to Whitfield Diffie & Martin Hellman at Stanford Uni in 1976
    - known earlier in classified community
- public-key/two-key/asymmetric cryptography involves the use of two keys:
    - a public-key, which may be known by anybody, and can be used to encrypt messages, and verify signatures
    - a related private-key, known only to the recipient, used to decrypt messages, and sign (create) signatures
- infeasible to determine private key from public
    - is asymmetric because, those who encrypt messages or verify signatures cannot decrypt messages or create signature

Table 9.1 Terminology Related to Asymmetric Encryption

Asymmetric Keys 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.
Public Key Certificate A digital document issued and digitally signed by the private key of a Certification Authority that binds the name of a subscriber to a public key. The certificate indicates that the subscriber identified in the certificate has sole control and access to the corresponding private key.
Public Key (Asymmetric) Cryptographic Algorithm 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.
Public Key Infrastructure (PKI) A set of policies, processes, server platforms, software and workstations used for the purpose of administering certificates and public-private key pairs, including the ability to issue, maintain, and revoke public key certificates.

Public – key Cryptosystems:

Asymmetric algorithms rely on one key for encryption and a different but related key for decryption. These algorithms have the following important characteristic.

- It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.

RSA exhibit following characteristics:

- Either of the two related keys can be used for encryption, with the other used for decryption.

  A **public-key encryption** scheme has six ingredients (Figure 9.1a)

- **Plaintext:** This is the readable message or data that is fed into the algorithm as input.

\

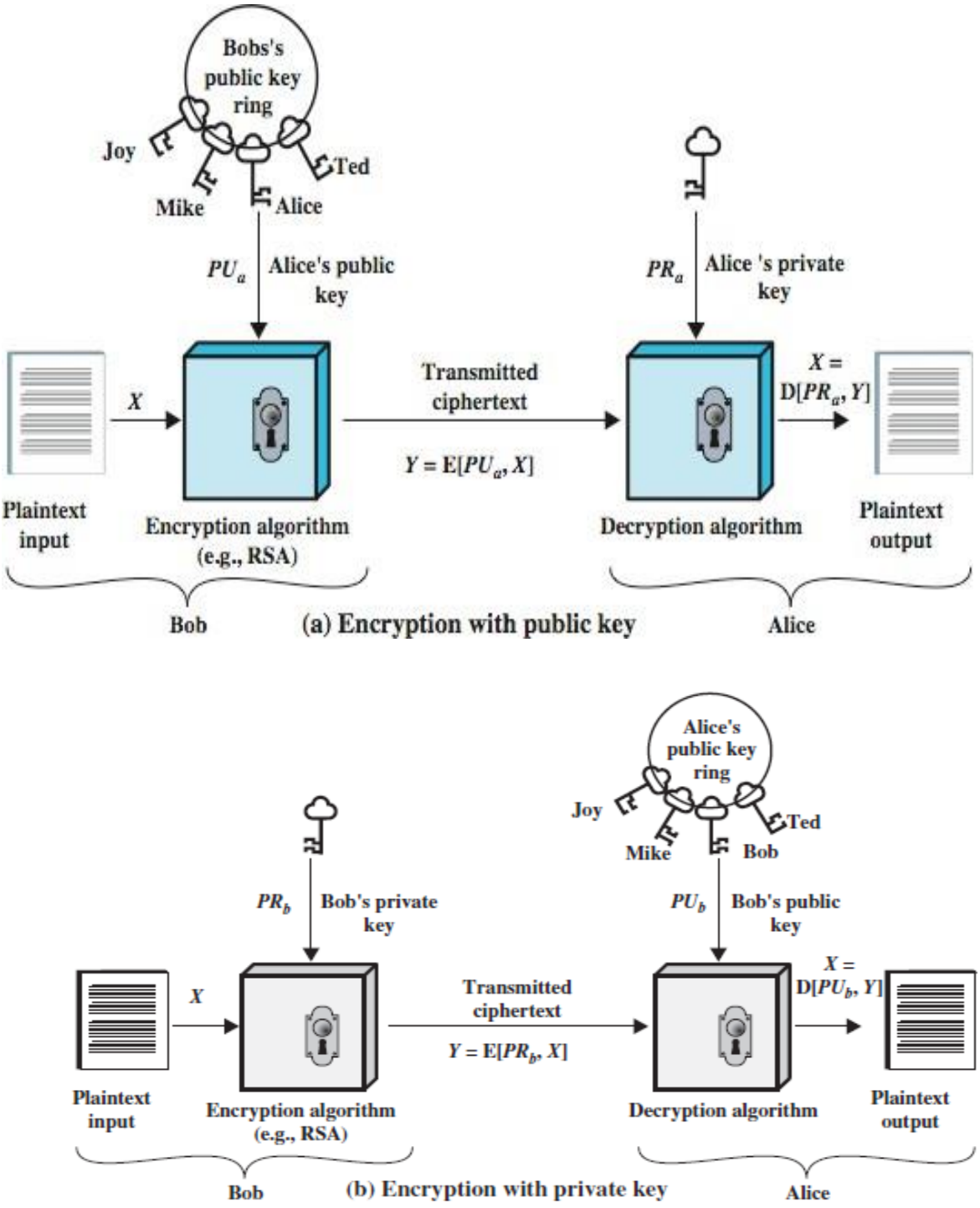


Figure 9.1 Public-Key Cryptography

- **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 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.

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 Figure 9.1a 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.

Table 9.2 Conventional and Public-Key Encryption

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.

Table 9.2 summarizes some of the important aspects of symmetric and public-key encryption. To discriminate between the two, we refer to the key used in symmetric encryption as a **secret key**. The two keys used for asymmetric encryption are referred to as the **public key** and the **private key**.

There is some source A that produces a message in plaintext, $X = [X_1, X_2, \ldots, X_M]$.

The M elements of X are letters in some finite alphabet.

The message is intended for destination B. B generates

a related pair of keys: a public key, PU_b, and a private key, PR_b. PR_b is known only to B, whereas PU_b is publicly available and therefore accessible by A.

With the message X and the encryption key PU_b as input, A forms the ciphertext $Y = [Y_1, Y_2, \ldots, Y_N]$:

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

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

An adversary, observing Y and having access to PU_b, but not having access to PR_b or X, must attempt to recover X and/or 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 $\hat{X}$. Often, however, the adversary is interested in being able to read future messages as well, in which case an attempt is made to recover PR_b by generating an estimate $\hat{PR}_b$.

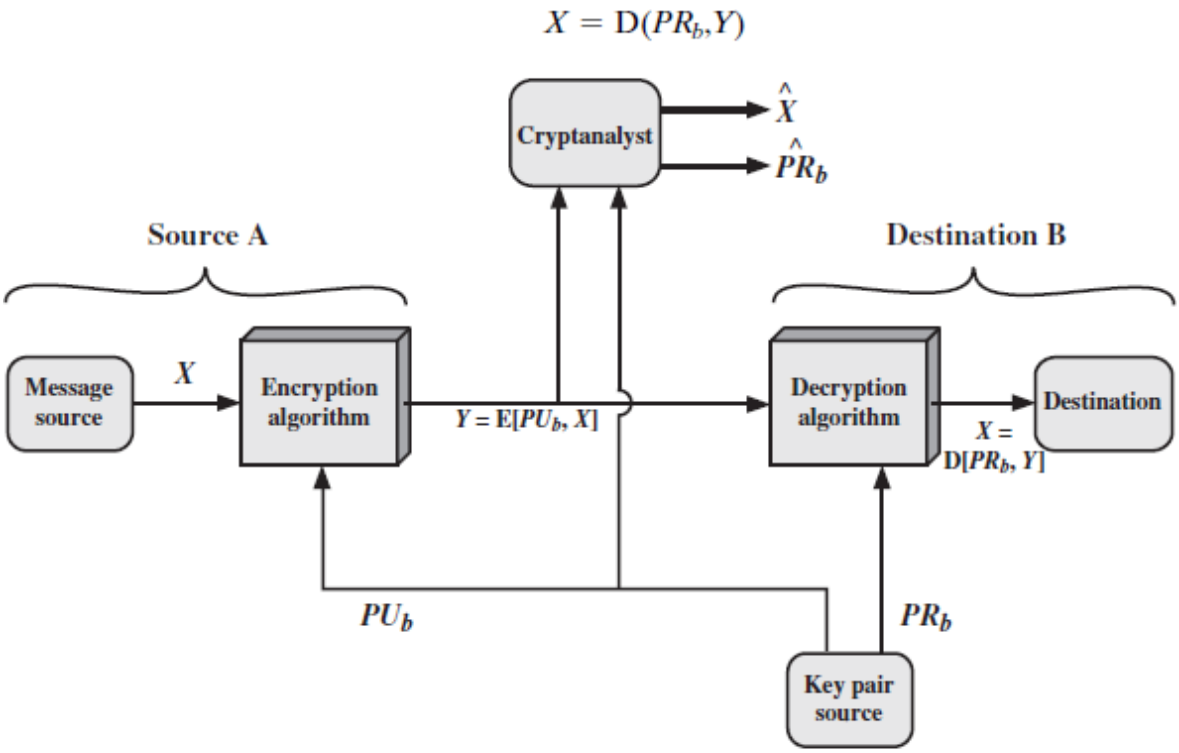


Figure 9.2 Public-Key Cryptosystem: Secrecy

An adversary, observing Y and having access to PU_b, but not having access to PR_b or X, must attempt to recover X and/or 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 $\hat{X}$. Often, however, the adversary is interested in being able to read future messages as well, in which case an attempt is made to recover PR_b by generating an estimate $\hat{PR}_b$.

$$Y = E(PR_a, X)$$
$$X = D(PU_a, Y)$$

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.

In the preceding scheme, the entire message is encrypted, which, although validating both author and contents, requires a great deal of storage. Each document

must be kept in plaintext to be used for practical purposes. A copy also must be stored in ciphertext so that the origin and contents can be verified in case of a dispute. A more efficient way of achieving the same results is to encrypt a small block of bits that is a function of the document. Such a block, called an authenticator, must have the property that it is infeasible to change the document without changing the authenticator. If the authenticator is encrypted with the sender's private key, it serves as a signature that verifies origin, content, and sequencing.

It is important to emphasize that the encryption process depicted in Figures 9.1b and 9.3 does not provide confidentiality.

It is, however, possible to provide both the authentication function and confidentiality by a double use of the public-key scheme (Figure 9.4):

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

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.

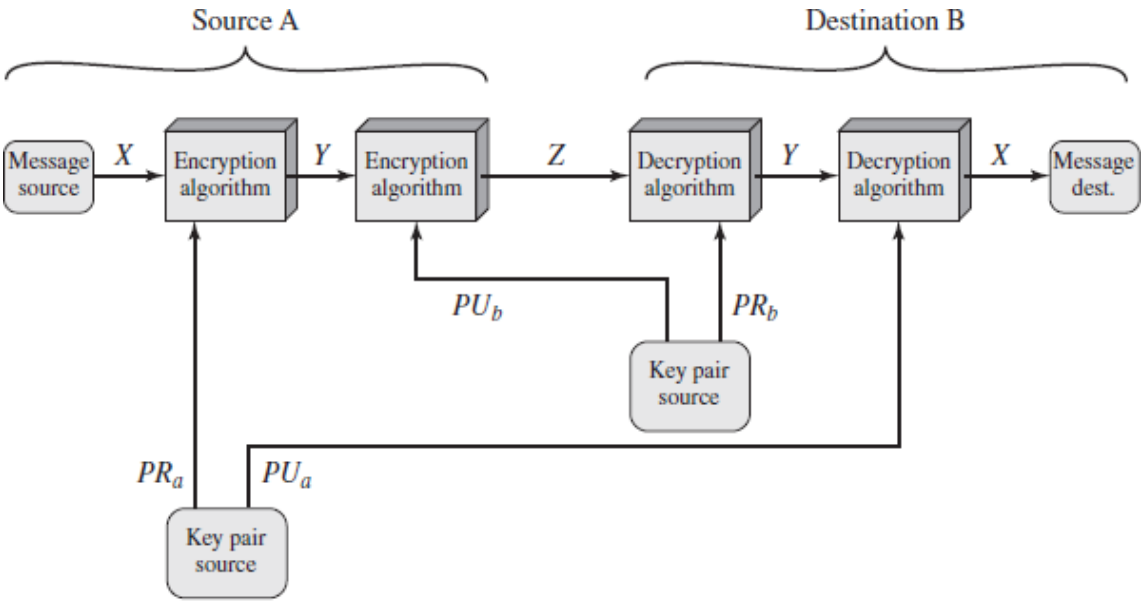


Figure 9.4 Public-Key Cryptosystem: Authentication and Secrecy

Applications for 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.

Requirements for Public-Key Cryptography

The cryptosystem illustrated in Figures 9.2 through 9.4 depends on a cryptographic algorithm based on two related keys.

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:

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

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

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

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

Table 9.3 Applications for Public-Key Cryptosystems

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

5. It is computationally infeasible for an adversary, knowing the public key, 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:

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

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:

$$Y = f(X) \quad \text{easy}$$
$$X = f^{-1}(Y) \quad \text{infeasible}$$

a **trap-door one-way function**, which 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. We can summarize as follows: A trap-door one-way function is a family of invertible functions f_k, such that

$$Y = f_k(X) \quad \text{easy, if } k \text{ and } X \text{ are known}$$
$$X = f_k^{-1}(Y) \quad \text{easy, if } k \text{ and } Y \text{ are known}$$
$$X = f_k^{-1}(Y) \quad \text{infeasible, if } Y \text{ is known but } k \text{ is not known}$$

Public-Key Cryptanalysis:

Public key schemes are no more or less secure than private key schemes - in both cases the size of the key determines the security. As with symmetric encryption, a public-key encryption scheme is vulnerable to a brute-force attack. The countermeasure is the same: **Use large keys**. However, there is a tradeoff to be considered. Public-key systems depend on the use of some sort of invertible mathematical function. The complexity of calculating these functions may not scale linearly with the number of bits in the key but grow more rapidly than that.

Thus, the key size must be large enough to make brute-force attack impractical but small enough for practical encryption and decryption. In practice, the key sizes that have been proposed do make brute-force attack impractical but result in encryption/decryption speeds that are too slow for general-purpose use. Instead, as was mentioned earlier, public-key encryption is currently confined to key management and signature applications.

Note also that you can't compare key sizes -> a 64-bit private key scheme has very roughly similar security to a 512-bit RSA - both could be broken given sufficient resources. But with public key schemes at least there is usually a firmer theoretical basis for determining the security since it’s based on well-known and well studied number theory problems.

THE RSA ALGORITHM:

One of the first successful responses to the challenge was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978.

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}. We examine RSA in this section in some detail, beginning with an explanation of the algorithm. Then we examine some of the computational and cryptanalytical implications of RSA.

RSA En/decryption

•To encrypt a message M the sender:
•Obtains **public key** of recipient PU={e,n}

•Computes: **$C=M^e \bmod n$,** where $0 \leq M < n$

•To decrypt the cipher text C the receiver:
•Uses their private key PR= {d,n}

•Computes: $M=C^d \bmod n$
•Note that the message M must be smaller than the modulus n(block if needed)

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:	$M = C^d \bmod n$

Figure 9.5 The RSA Algorithm

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 (Chapter 4).

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 $C = 88^7 \bmod 187$. 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 $M = 11^{23} \bmod 187$:

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

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

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

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

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

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

Refer extra problems on RSA from the given NOTES..

The following example shows the use of RSA to process multiple blocks of data. 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. Figure 9.7a illustrates the sequence of events for the encryption of multiple blocks, and Figure 9.7b gives a specific example. The circled numbers indicate the order in which operations are performed.

Computational Aspects: Complexity of the computation required to use RSA has two issues to consider: encryption/decryption and key generation.

Exponentiation in Modular Arithmetic Both encryption and decryption in RSA involve raising an integer to an integer power, mod n. Make use of a property of modular arithmetic:

$$[(a \bmod n) \text{ X } (b \bmod n)] \bmod n = (a \text{ X } b) \bmod n$$

To find the value ab mod 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,

$$a^b = a^{\left(\sum_{b_i \neq 0} 2^i\right)} = \prod_{b_i \neq 0} a^{(2^i)}$$

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

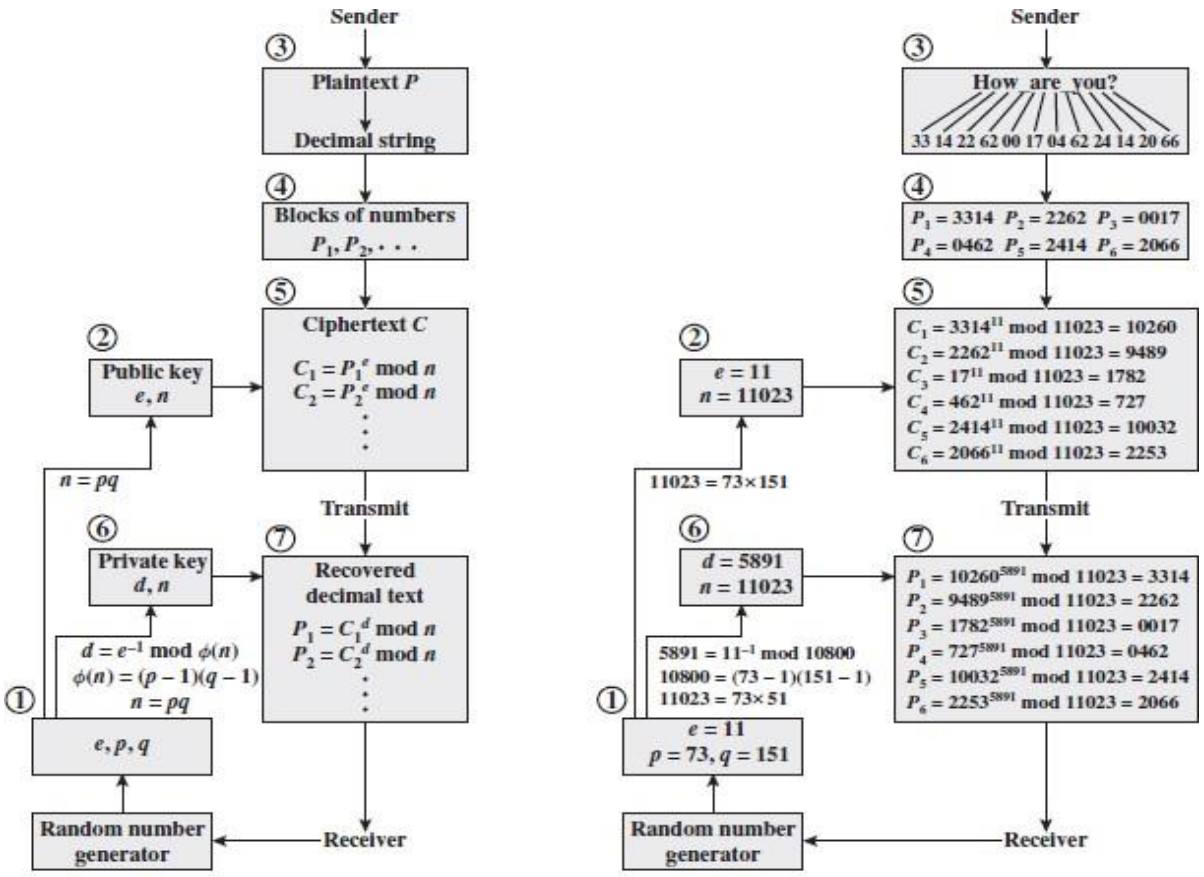


Figure 9.7 RSA Processing of Multiple Blocks

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.

```
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 $b_k b_{k-1} \ldots b_0$.

Figure 9.8 Algorithm for Computing a^b mod n

Table 9.4 Result of the Fast Modular Exponentiation Algorithm for $a^b \bmod n$, 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

However, with a very small public key, such as e = 3, RSA becomes vulnerable to a simple attack.

- Suppose we have three different RSA users who all use the value e = 3 but have unique values of n, namely (n1, n2, n3).
- If user A sends the same encrypted message M to all three users, then the three ciphertexts are $C_1 = M^3 \bmod n_1$, $C_2 = M^3 \bmod n_2$, and $C_3 = M^3 \bmod n_3$.
- It is likely that n_1, n_2, and n_3 are pairwise relatively prime. Therefore, one can use the Chinese remainder theorem (CRT) to compute $M^3 \bmod (n_1 n_2 n_3)$.
- By the rules of the RSA algorithm, M is less than each of the n_i; therefore $M^3 < n_1 n_2 n_3$. Accordingly, the attacker need only compute the cube root of M^3. This attack can be countered by adding a unique pseudorandom bit string as padding to each instance of M to be encrypted.

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. Need to compute the value $M = C^d \bmod n$. Let us define the following intermediate results:

$$V_p = C^d \bmod p \quad V_q = C^d \bmod q$$

Following the CRT using Equation (8.8), define the quantities

$$X_p = q \times (q^{-1} \bmod p) \quad X_q = p \times (p^{-1} \bmod q)$$

The CRT then shows, using Equation (8.9), that

$$M = (V_p X_p + V_q X_q) \bmod n$$

Furthermore, we can simplify the calculation of V_p and V_q using Fermat's theorem, which states that $a^{p-1} \equiv 1 \pmod p$ if p and a are relatively prime. Some thought should convince you that the following are valid.

$$V_p = C^d \bmod p = C^{d \bmod (p-1)} \bmod p \quad V_q = C^d \bmod q = C^{d \bmod (q-1)} \bmod q$$

Key Generation Before the application of the public-key cryptosystem, each participant must generate a pair of keys. This involves the following tasks.

• Determining two prime numbers, p and q.

• Selecting either e or d and calculating the other.

First, consider the selection of p and q. Because the value of **n = pq** will be known to any potential adversary, in order to prevent the discovery of p and q by exhaustive methods, these primes must be chosen from a sufficiently large set (i.e., p and q must be large numbers).

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.

Timing Attacks:

Timing attacks are applicable not just to RSA, but to other public-key cryptography systems. This attack is alarming for two reasons: It comes from a completely unexpected direction, and it is a ciphertext-only attack.

A **timing attack** is somewhat analogous to a burglar guessing the combination of a safe by observing how long it takes for someone to turn the dial from number to number. The attack using the modular exponentiation algorithm of Figure 9.8, but the attack can be adapted to work with any implementation that does not run in fixed time. In this algorithm, modular exponentiation is accomplished bit by bit, with one modular multiplication performed at each iteration and an additional modular multiplication performed for each 1 bit.

The timing attack is a serious threat, there are simple countermeasures that can be used, including the following.

• **Constant exponentiation time:** Ensure that all exponentiations take the same amount of time before returning a result. This is a simple fix but does degrade performance.

• **Random delay:** Better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack.

• **Blinding:** Multiply the ciphertext by a random number before performing exponentiation. This process prevents the attacker from knowing what ciphertext bits are being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing attack.

RSA Data Security incorporates a blinding feature into some of its products. The private-key operation $M = C_d \bmod n$ is implemented as follows.

1. Generate a secret random number r between 0 and $n - 1$.
2. Compute $C' = C(r^e) \bmod n$, where e is the public exponent.
3. Compute $M' = (C')^d \bmod n$ with the ordinary RSA implementation.
4. Compute $M = M'r^{-1} \bmod n$. In this equation, r^{-1} is the multiplicative inverse of $r \bmod n$; see Chapter 4 for a discussion of this concept. It can be demonstrated that this is the correct result by observing that $r^{ed} \bmod n = r \bmod n$.

RSA Data Security reports a 2 to 10% performance penalty for blinding.

Fault-Based Attack:

The faults cause the software to produce invalid signatures, which can then be analyzed by the attacker to recover the private key. The authors show how such an analysis can be done and then demonstrate it by extracting a 1024-bit private RSA key in approximately 100 hours, using a commercially available microprocessor. The attack algorithm involves inducing single-bit errors and observing the results.

Chosen Ciphertext Attack and Optimal Asymmetric Encryption Padding: The basic RSA algorithm is vulnerable to a **chosen ciphertext attack** (CCA). CCA is defined as an attack in which the adversary chooses a number of ciphertexts and is then given the corresponding plaintexts, decrypted with the target's private key. Thus, the adversary could select a plaintext, encrypt it with the target's public key, and then be able to get the plaintext back by having it decrypted with the private key.

A simple example of a CCA against RSA takes advantage of the following property of RSA:

$$\mathrm{E}(PU, M_1) \times \mathrm{E}(PU, M_2) = \mathrm{E}(PU, [M_1 \times M_2]) \qquad (9.2)$$

We can decrypt $C = M^e \bmod n$ using a CCA as follows.

1. Compute $X = (C \times 2^e) \bmod n$.
2. Submit X as a chosen ciphertext and receive back $Y = X^d \bmod n$.

But now note that

$$\begin{aligned} X &= (C \bmod n) \times (2^e \bmod n) \\ &= (M^e \bmod n) \times (2^e \bmod n) \\ &= (2M)^e \bmod n \end{aligned}$$

Therefore, $Y = (2M) \bmod n$. From this, we can deduce M. To overcome this simple attack, practical RSA-based cryptosystems randomly pad the plaintext prior to encryption. This randomizes the ciphertext so that Equation (9.2) no longer holds. However, more sophisticated CCAs are possible, and a simple padding with a random value has been shown to be insufficient to provide the desired security. To counter such attacks, RSA Security Inc., a leading RSA vendor and former holder of the RSA patent, recommends modifying the plaintext using a procedure known as **optimal asymmetric encryption padding** (OAEP).

Figure 9.10 depicts OAEP encryption.

- As a first step, the message M to be encrypted is padded.
- A set of optional parameters, *P*, is passed through a hash function, H.
- The output is then padded with zeros to get the desired length in the overall data block (DB).
- Next, a random seed is generated and passed through another hash function, called the mask generating function (MGF).
- The resulting hash value is bit-by-bit XORed with DB to produce a maskedDB. The maskedDB is in turn passed through the MGF to form a hash that is XORed with the seed to produce the maskedseed.
- The concatenation of the maskedseed and the maskedDB forms the encoded message EM. Note that the EM includes the padded message, masked by the seed, and the seed, masked by the maskedDB.
- The EM is then encrypted using RSA.

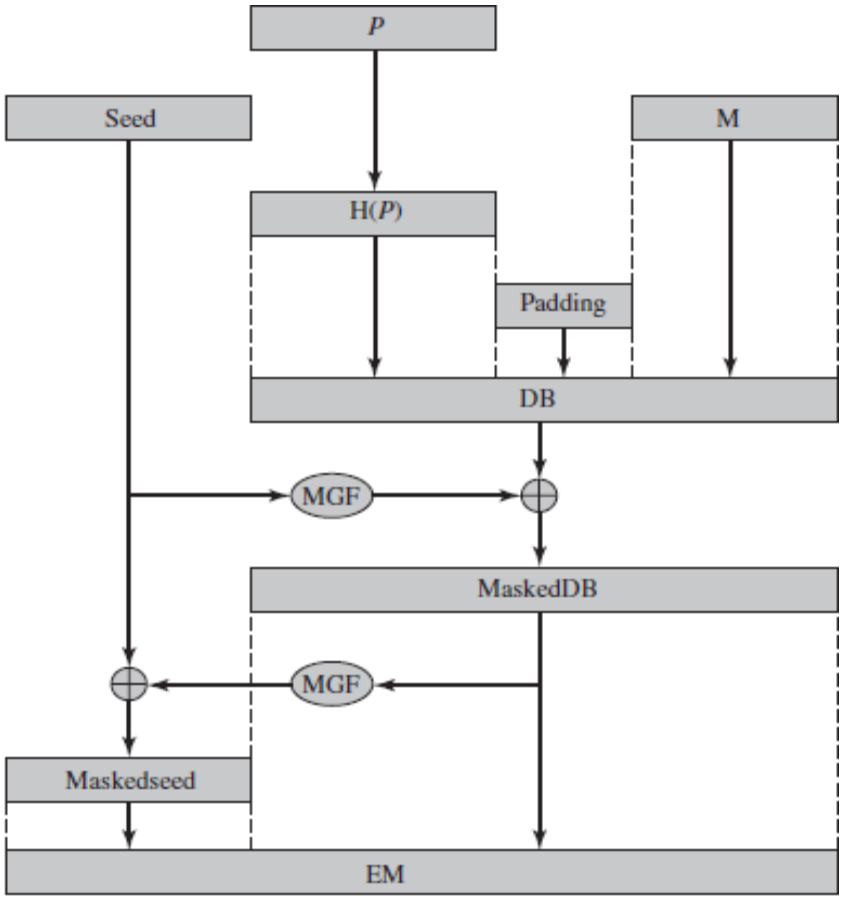


Figure 9.10 Encryption Using Optimal Asymmetric Encryption Padding (OAEP)

These are exercise problems mentioned in the text book..

9.2 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. $p = 17; q = 31, e = 7; M = 2$

Diffie-Hellman Key Exchange: 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 Algorithm:

Figure 10.1 summarizes the Diffie-Hellman key exchange algorithm. For this scheme, there are two publicly known numbers: a prime number q and an integer a is a primitive root of q. Suppose the users A and B wish to create a shared key.

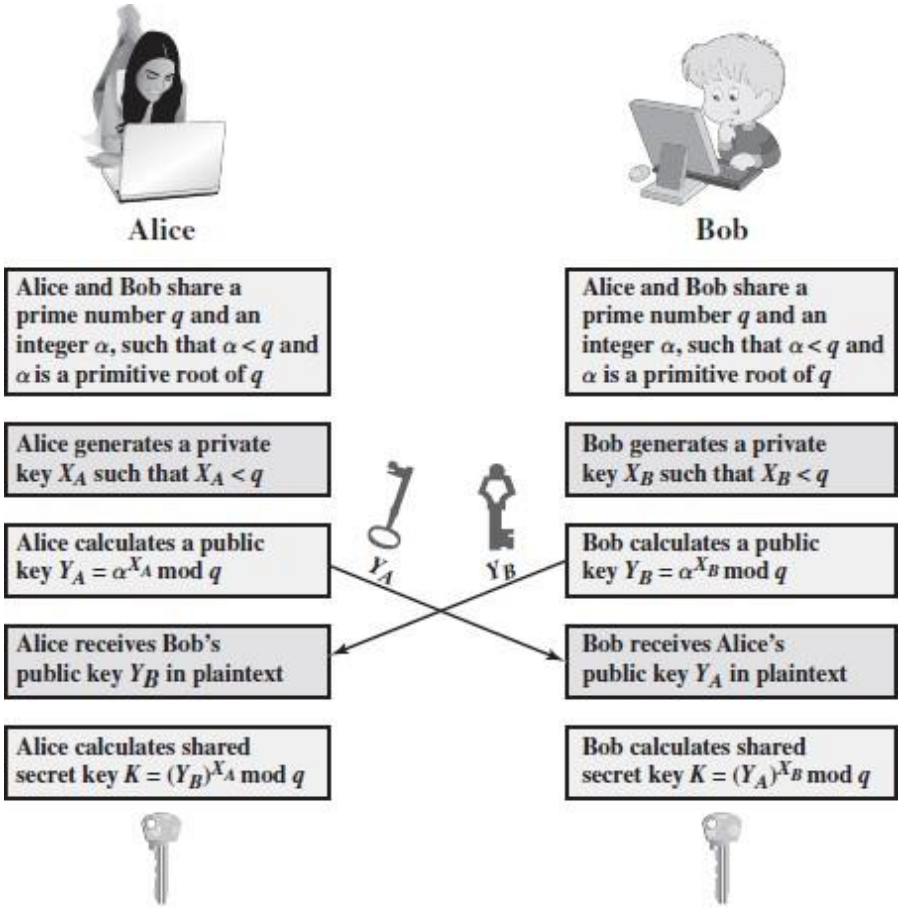


Figure 10.1 The Diffie-Hellman Key Exchange

User A selects a random integer $X_A < q$ and computes $Y_A = \alpha^{X_A} \bmod q$. Similarly, user B independently selects a random integer $X_B < q$ and computes $Y_B = \alpha^{X_B} \bmod q$. Each side keeps the X value private and makes the Y value available publicly to the other side. Thus, X_A is A's private key and Y_A is A's corresponding public key, and similarly for B. User A computes the key as $K = (Y_B)^{X_A} \bmod q$ and user B computes the key as $K = (Y_A)^{X_B} \bmod q$. These two calculations produce identical results:

$$
\begin{aligned}
K &= (Y_B)^{X_A} \bmod q \\
&= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\
&= (\alpha^{X_B})^{X_A} \bmod q && \text{by the rules of modular arithmetic} \\
&= \alpha^{X_B X_A} \bmod q \\
&= (\alpha^{X_A})^{X_B} \bmod q \\
&= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\
&= (Y_A)^{X_B} \bmod q
\end{aligned}
$$

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 X_A and X_B are private, an adversary only has the following ingredients to work with: q, α, Y_A, and Y_B. Thus, the adversary is forced to take a discrete logarithm to determine the key. For example, to determine the private key of user B, an adversary must compute

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

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

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

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.

Key Exchange Protocols:

Figure 10.1 shows a simple protocol that makes use of the Diffie-Hellman calculation.

- Suppose that user A wishes to set up a connection with user B and use a secret key to encrypt messages on that connection.
- User A can generate a one-time private key X_A, calculate Y_A, and send that to user B.
- User B responds by generating a private value X_B, calculating Y_B, and sending Y_B to user A.
- Both users can now calculate the key.
- The necessary public values q and a would need to be known ahead of time. Alternatively, user A could pick values for q and a and include those in the first message.

Man-in-the-Middle Attack:

The protocol depicted in Figure 10.1 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 (Figure 10.2).

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

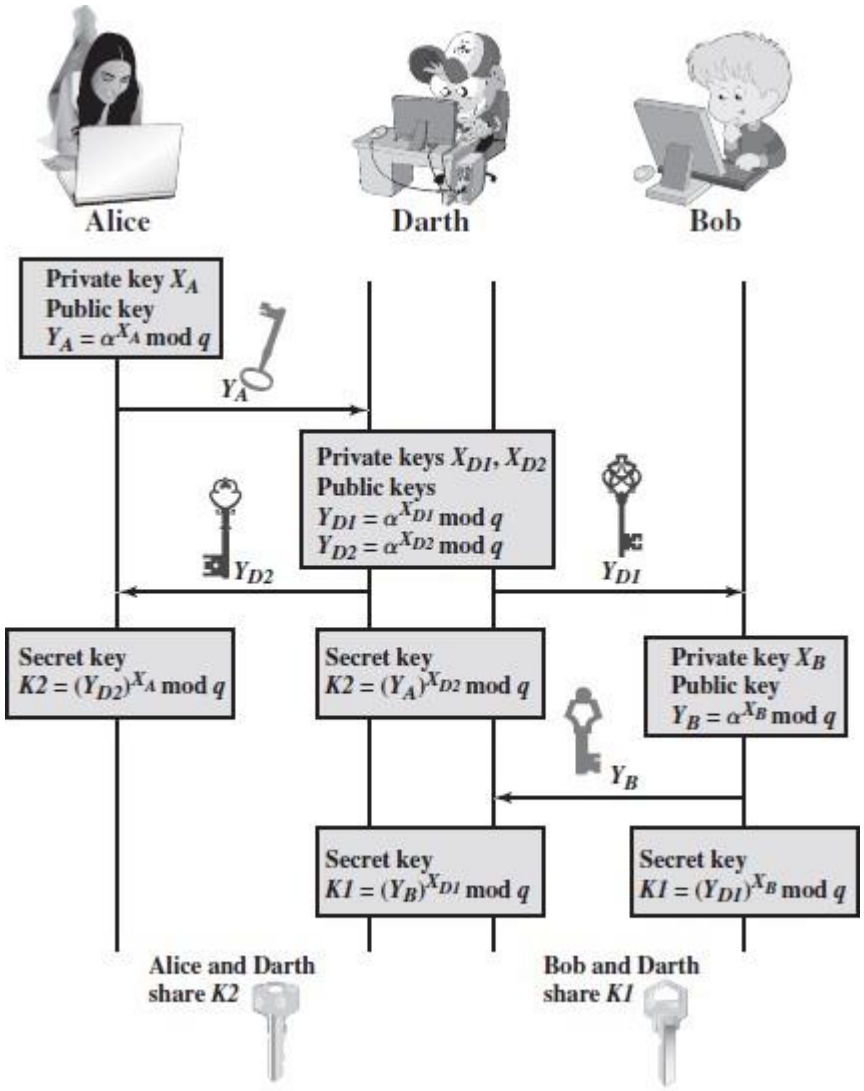


Figure 10.2 Man-in-the-Middle Attack

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.

Elliptic Curve Arithmetic:

Most of the products and standards that use public-key cryptography for encryption and digital signatures use RSA. The principal attraction of ECC, compared to RSA, is that it appears to offer equal security for a far smaller key size, thereby reducing processing overhead.

For elliptic curve cryptography, an operation over elliptic curves, called addition, is used. Multiplication is defined by repeated addition. For example,

$$a \times k = \underbrace{(a + a + \ldots + a)}_{k \text{ times}}$$

where the addition is performed over an elliptic curve. Cryptanalysis involves determining *k* given *a* and (a X k).

An **elliptic curve** is defined by an equation in two variables with coefficients. For cryptography, the variables and coefficients are restricted to elements in a finite field, which results in the definition of a finite abelian group.

Abelian Groups:

An **abelian group** *G*, sometimes denoted by {G, **.**}, is a set of elements with a binary operation, denoted by **.** that associates to each ordered pair (*a*, *b*) of elements in *G* an element (*a* **.** *b*) in *G*, such that the following axioms are obeyed:

(A1) Closure:	If a and b belong to G, then $a \cdot b$ is also in G.
(A2) Associative:	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$ for all a, b, c in G.
(A3) Identity element:	There is an element e in G such that $a \cdot e = e \cdot a = a$ for all a in G.
(A4) Inverse element:	For each a in G there is an element a' in G such that $a \cdot a' = a' \cdot a = e$.
(A5) Commutative:	$a \cdot b = b \cdot a$ for all a, b in G.

Elliptic Curves over Real Numbers:

Elliptic curves are not ellipses. They are so named because they are described by cubic equations, similar to those used for calculating the circumference of an ellipse. In general, cubic equations for elliptic curves take the following form, known as a **Weierstrass equation:**

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

where a, b, c, d, e are real numbers and x and y take on values in the real numbers. For our purpose, it is sufficient to limit ourselves to equations of the form

$$y^2 = x^3 + ax + b \qquad \textbf{(10.1)}$$

Such equations are said to be cubic, or of degree 3, because the highest exponent they contain is a 3. Also included in the definition of an elliptic curve is a single element denoted O and called the *point at infinity* or the *zero point*, which we discuss subsequently. To plot such a curve, we need to compute

$$y = \sqrt{x^3 + ax + b}$$

For given values of a and b, the plot consists of positive and negative values of y for each value of x. Thus, each curve is symmetric about $y = 0$. Figure 10.4 shows two examples of elliptic curves.

Now, consider the set of points E(a, b) consisting of all of the points (x, y) that satisfy Equation (10.1) together with the element O. Using a different value of the pair (a, b) results in a different set E(a, b). Using this terminology, the two curves in Figure 10.4 depict the sets E(-1, 0) and E(1, 1), respectively.

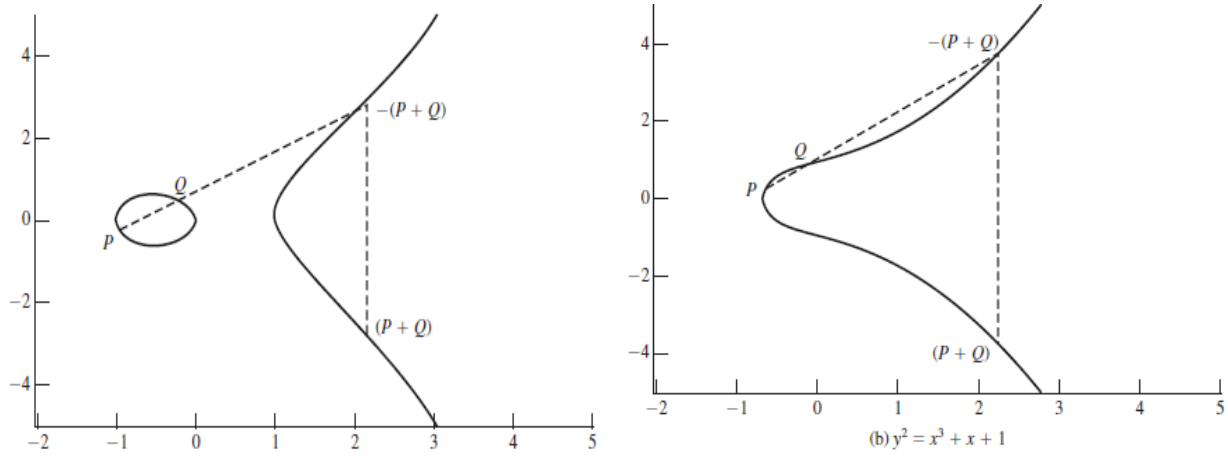


Figure 10.4 Example of Elliptic Curves

Geometric Description of Addition:

It can be shown that a group can be defined based on the set E(*a*, *b*) for specific values of *a* and *b* in Equation (10.1), provided the following condition is met:

$$4a^3 + 27b^2 \neq 0 \qquad \textbf{(10.2)}$$

To define the group, we must define an operation, called addition and denoted by +, for the set E(*a*, *b*), where *a* and *b* satisfy Equation (10.2). In geometric terms, the rules for addition can be stated as follows: If three points on an elliptic curve lie on a straight line, their sum is *O*. From this definition, we can define the rules of addition over an elliptic curve.

1. O serves as the additive identity. Thus $O = -O$; for any point P on the elliptic curve, $P + O = P$. In what follows, we assume $P \neq O$ and $Q \neq O$.
2. The negative of a point P is the point with the same x coordinate but the negative of the y coordinate; that is, if $P = (x, y)$, then $-P = (x, -y)$. Note that these two points can be joined by a vertical line. Note that $P + (-P) = P - P = O$.
3. To add two points P and Q with different x coordinates, draw a straight line between them and find the third point of intersection R. It is easily seen that there is a unique point R that is the point of intersection (unless the line is tangent to the curve at either P or Q, in which case we take $R = P$ or $R = Q$, respectively). To form a group structure, we need to define addition on these three points: $P + Q = -R$. That is, we define $P + Q$ to be the mirror image (with respect to the x axis) of the third point of intersection. Figure 10.4 illustrates this construction.
4. The geometric interpretation of the preceding item also applies to two points, P and $-P$, with the same x coordinate. The points are joined by a vertical line, which can be viewed as also intersecting the curve at the infinity point. We therefore have $P + (-P) = O$, which is consistent with item (2).
5. To double a point Q, draw the tangent line and find the other point of intersection S. Then $Q + Q = 2Q = -S$.

With the preceding list of rules, it can be shown that the set $E(a, b)$ is an abelian group.

Algebraic Description of Addition:

In this subsection, we present some results that enable calculation of additions over elliptic curves.

For two distinct points, $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$, that are not negatives of each other, the slope of the line l that joins them is $\Delta = (y_Q - y_P)/(x_Q - x_P)$. There is exactly one other point where l intersects the elliptic curve, and that is the negative of the sum of P and Q.

After some algebraic manipulation, we can express the sum $R = P + Q$ as

$$\begin{aligned} x_R &= \Delta^2 - x_P - x_Q \\ y_R &= -y_P + \Delta(x_P - x_R) \end{aligned} \tag{10.3}$$

We also need to be able to add a point to itself: $P + P = 2P = R$. When $y_P \neq 0$, the expressions are

$$\begin{aligned} x_R &= \left(\frac{3x_P^2 + a}{2y_P}\right)^2 - 2x_P \\ y_R &= \left(\frac{3x_P^2 + a}{2y_P}\right)(x_P - x_R) - y_P \end{aligned} \tag{10.4}$$

Elliptic Curves over Zp:

Elliptic curve cryptography makes use of elliptic curves in which the variables and coefficients are all restricted to elements of a finite field.

Two families of elliptic curves are used in cryptographic applications:

- ✓ prime curves over Zp : For a **prime curve** over Zp, we use a cubic equation in which the variables and coefficients all take on values in the set of integers from 0 through p - 1 and in which calculations are performed modulo p.
- ✓ binary curves over GF(2^m): For a **binary curve** defined over GF(2^m), the variables and coefficients all take on values in GF(2^m) and in calculations are performed over GF(2^m).

The **prime curves** are best for **software applications**, because the extended bit-fiddling operations needed by binary curves are not required; and that **binary curves** are best for **hardware applications**, where it takes remarkably few logic gates to create a powerful, fast cryptosystem.

For elliptic curves over Z_p, as with real numbers, we limit ourselves to equations of the form of Equation (10.1), but in this case with coefficients and variables limited to Z_p:

$$y^2 \bmod p = (x^3 + ax + b) \bmod p \qquad \textbf{(10.5)}$$

For example, Equation (10.5) is satisfied for $a = 1, b = 1, x = 9, y = 7, p = 23$:

$$7^2 \bmod 23 = (9^3 + 9 + 1) \bmod 23$$
$$49 \bmod 23 = 739 \bmod 23$$
$$3 = 3$$

Now consider the set $E_p(a, b)$ consisting of all pairs of integers (x, y) that satisfy Equation (10.5), together with a point at infinity O. The coefficients a and b and the variables x and y are all elements of Z_p.

For example, let $p = 23$ and consider the elliptic curve $y^2 = x^3 + x + 1$. In this case, $a = b = 1$. Note that this equation is the same as that of Figure 10.4b. The figure shows a continuous curve with all of the real points that satisfy the equation. For the set $E_{23}(1, 1)$, we are only interested in the nonnegative integers in the quadrant from (0, 0) through $(p - 1, p - 1)$ that satisfy the equation mod p. Table 10.1 lists the points (other than O) that are part of $E_{23}(1, 1)$. Figure 10.5 plots the points of $E_{23}(1, 1)$; note that the points, with one exception, are symmetric about $y = 11.5$.

Table 10.1 Points (other than O) on the Elliptic Curve E_{23} (1,1)

(0, 1)	(6, 4)	(12, 19)
(0, 22)	(6, 19)	(13, 7)
(1, 7)	(7, 11)	(13, 16)
(1, 16)	(7, 12)	(17, 3)
(3, 10)	(9, 7)	(17, 20)
(3, 13)	(9, 16)	(18, 3)
(4, 0)	(11, 3)	(18, 20)
(5, 4)	(11, 20)	(19, 5)
(5, 19)	(12, 4)	(19, 18)

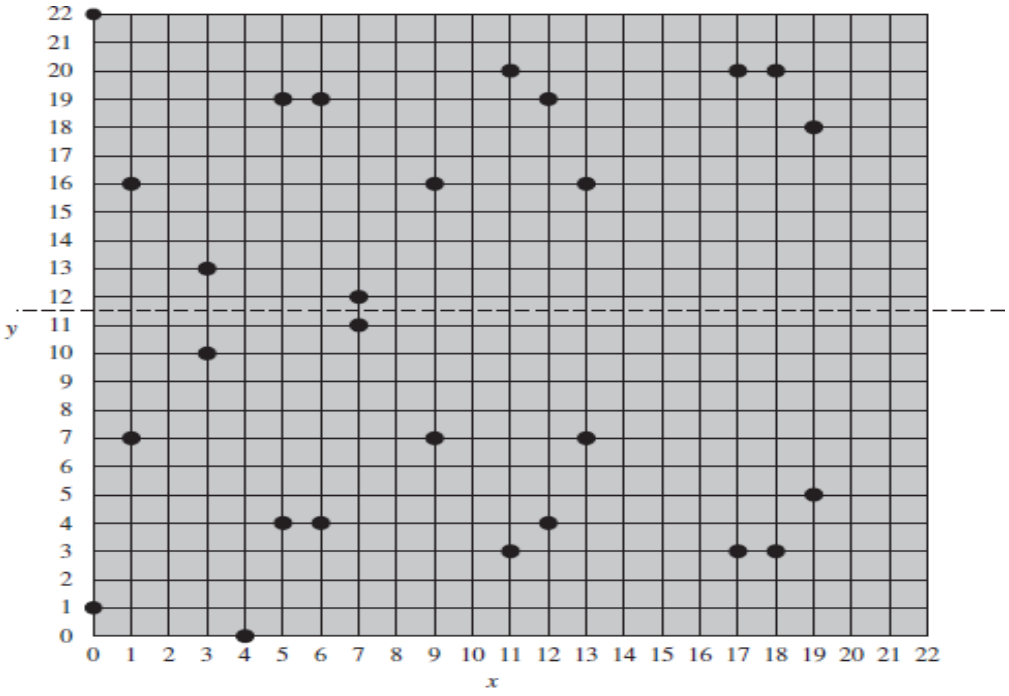


Figure 10.5 The Elliptic Curve $E_{23}(1, 1)$

It can be shown that a finite abelian group can be defined based on the set $E_p(a, b)$ provided that $(x^3 + ax + b) \bmod p$ has no repeated factors. This is equivalent to the condition

$$(4a^3 + 27b^2) \bmod p \neq 0 \bmod p \qquad \textbf{(10.6)}$$

Note that Equation (10.6) has the same form as Equation (10.2).

The rules for addition over $E_p(a, b)$, correspond to the algebraic technique described for elliptic curves defined over real numbers. For all points $P, Q \in E_p(a, b)$:

1. $P + O = P$.
2. If $P = (x_P, y_P)$, then $P + (x_P, -y_P) = O$. The point $(x_P, -y_P)$ is the negative of P, denoted as $-P$. For example, in $E_{23}(1, 1)$, for $P = (13, 7)$, we have $-P = (13, -7)$. But $-7 \bmod 23 = 16$. Therefore, $-P = (13,16)$, which is also in $E_{23}(1, 1)$.
3. If $P = (x_p, y_p)$ and $Q = (x_Q, y_Q)$ with $P \neq -Q$, then $R = P + Q = (x_R, y_R)$ is determined by the following rules:

$$x_R = (\lambda^2 - x_P - x_Q) \bmod p$$
$$y_R = (\lambda(x_P - x_R) - y_P) \bmod p$$

where

$$\lambda = \begin{cases} \left(\frac{y_Q - y_P}{x_Q - x_P}\right) \bmod p & \text{if } P \neq Q \\ \left(\frac{3x_P^2 + a}{2y_P}\right) \bmod p & \text{if } P = Q \end{cases}$$

4. Multiplication is defined as repeated addition; for example, $4P = P + P + P + P$.

For example, let $P = (3, 10)$ and $Q = (9, 7)$ in $E_{23}(1, 1)$. Then

$$\lambda = \left(\frac{7 - 10}{9 - 3}\right) \bmod 23 = \left(\frac{-3}{6}\right) \bmod 23 = \left(\frac{-1}{2}\right) \bmod 23 = 11$$
$$x_R = (11^2 - 3 - 9) \bmod 23 = 109 \bmod 23 = 17$$
$$y_R = (11(3 - 17) - 10) \bmod 23 = -164 \bmod 23 = 20$$

So $P + Q = (17, 20)$. To find $2P$,

$$\lambda = \left(\frac{3(3^2) + 1}{2 \times 10}\right) \bmod 23 = \left(\frac{5}{20}\right) \bmod 23 = \left(\frac{1}{4}\right) \bmod 23 = 6$$

The last step in the preceding equation involves taking the multiplicative inverse of 4 in Z_{23}. This can be done using the extended Euclidean algorithm defined in Section 4.4. To confirm, note that $(6 \times 4) \bmod 23 = 24 \bmod 23 = 1$.

$$x_R = (6^2 - 3 - 3) \bmod 23 = 30 \bmod 23 = 7$$
$$y_R = (6(3 - 7) - 10) \bmod 23 = (-34) \bmod 23 = 12$$

and $2P = (7, 12)$.

For determining the security of various elliptic curve ciphers, it is of some interest to know the number of points in a finite abelian group defined over an elliptic curve. In the case of the finite group $E_P(a, b)$, the number of points N is bounded by

$$p + 1 - 2\sqrt{p} \leq N \leq p + 1 + 2\sqrt{p}$$

Note that the number of points in $E_p(a, b)$ is approximately equal to the number of elements in Z_p, namely p elements.

Elliptic Curves over GF(2^m):

A **finite field** GF(2^m) consists of 2^m elements, together with addition and multiplication operations that can be defined over polynomials. For elliptic curves over GF(2^m), we use a cubic equation in which the variables and coefficients all take on values in GF(2^m) for some number m and in which calculations are performed using the rules of arithmetic in GF(2^m). It turns out that the form of cubic equation appropriate for cryptographic applications for elliptic curves is somewhat different for GF(2^m) than for Zp. The form is,

$$y^2 + xy = x^3 + ax^2 + b \qquad \textbf{(10.7)}$$

Table 10.2 Points (other than O) on the Elliptic Curve $E_{2^4}(g^4, 1)$

$(0, 1)$	(g^5, g^3)	(g^9, g^{13})
$(1, g^6)$	(g^5, g^{11})	(g^{10}, g)
$(1, g^{13})$	(g^6, g^8)	(g^{10}, g^8)
(g^3, g^8)	(g^6, g^{14})	$(g^{12}, 0)$
(g^3, g^{13})	(g^9, g^{10})	(g^{12}, g^{12})

where it is understood that the variables x and y and the coefficients a and b are elements of GF(2^m) and that calculations are performed in GF(2^m).

Now consider the set $E_{2^m}(a, b)$ consisting of all pairs of integers (x, y) that satisfy Equation (10.7), together with a point at infinity O.

For example, let us use the finite field GF(2^4) with the irreducible polynomial $f(x) = x^4 + x + 1$. This yields a generator g that satisfies $f(g) = 0$ with a value of $g^4 = g + 1$, or in binary, $g = 0010$. We can develop the powers of g as follows.

$g^0 = 0001$	$g^4 = 0011$	$g^8 = 0101$	$g^{12} = 1111$
$g^1 = 0010$	$g^5 = 0110$	$g^9 = 1010$	$g^{13} = 1101$
$g^2 = 0100$	$g^6 = 1100$	$g^{10} = 0111$	$g^{14} = 1001$
$g^3 = 1000$	$g^7 = 1011$	$g^{11} = 1110$	$g^{15} = 0001$

For example, $g^5 = (g^4)(g) = (g + 1)(g) = g^2 + g = 0110$.

Now consider the elliptic curve $y^2 + xy = x^3 + g^4x^2 + 1$. In this case, $a = g^4$ and $b = g^0 = 1$. One point that satisfies this equation is (g^5, g^3):

$$(g^3)^2 + (g^5)(g^3) = (g^5)^3 + (g^4)(g^5)^2 + 1$$
$$g^6 + g^8 = g^{15} + g^{14} + 1$$
$$1100 + 0101 = 0001 + 1001 + 0001$$
$$1001 = 1001$$

Table 10.2 lists the points (other than O) that are part of $E_{2^4}(g^4, 1)$. Figure 10.6 plots the points of $E_{2^4}(g^4, 1)$.

It can be shown that a finite abelian group can be defined based on the set $E_{2^m}(a, b)$, provided that $b \neq 0$. The rules for addition can be stated as follows. For all points $P, Q \in E_{2^m}(a, b)$:

1. $P + O = P$.
2. If $P = (x_P, y_P)$, then $P + (x_P, x_P + y_P) = O$. The point $(x_P, x_P + y_P)$ is the negative of P, which is denoted as $-P$.
3. If $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ with $P \neq -Q$ and $P \neq Q$, then $R = P + Q = (x_R, y_R)$ is determined by the following rules:

$$x_R = \lambda^2 + \lambda + x_P + x_Q + a$$
$$y_R = \lambda(x_P + x_R) + x_R + y_P$$

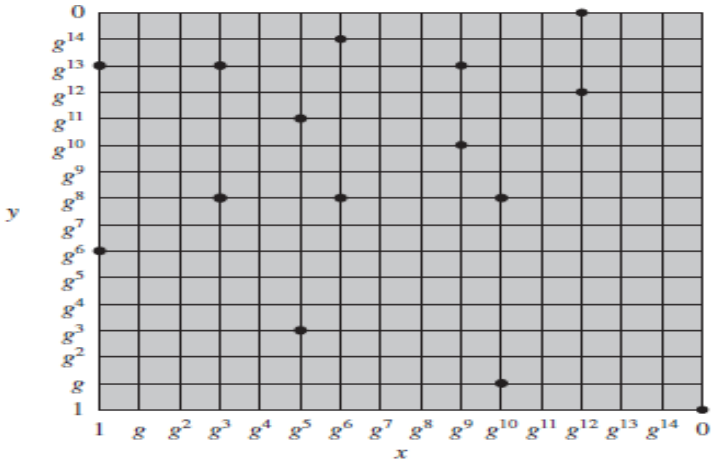


Figure 10.6 The Elliptic Curve $E_{2^4}(g^4, 1)$

where

$$\lambda = \frac{y_Q + y_P}{x_Q + x_P}$$

4. If $P = (x_P, y_P)$ then $R = 2P = (x_R, y_R)$ is determined by the following rules:

$$x_R = \lambda^2 + \lambda + a$$
$$y_R = x_P^2 + (\lambda + 1)x_R$$

where

$$\lambda = x_P + \frac{y_P}{x_P}$$

Elliptic Curve Cryptography:

The addition operation in ECC is the counterpart of modular multiplication in RSA, and multiple addition is the counterpart of modular exponentiation. To form a cryptographic system using elliptic curves, we need to find a "hard problem" corresponding to factoring the product of two primes or taking the discrete logarithm.

Consider the equation $Q = kP$ where $Q, P \in E_P(a, b)$ and $k < p$. It is relatively easy to calculate Q given k and P, but it is hard to determine k given Q and P. This is called the discrete logarithm problem for elliptic curves.

Consider the group $E_{23}(9,17)$. This is the group defined by the equation $y^2 \bmod 23 = (x^3 + 9x + 17) \bmod 23$. What is the discrete logarithm k of $Q = (4, 5)$ to the base $P = (16, 5)$? The brute-force method is to compute multiples of P until Q is found. Thus,

$$P = (16, 5); 2P = (20, 20); 3P = (14, 14); 4P = (19, 20); 5P = (13, 10);$$
$$6P = (7, 3); 7P = (8, 7); 8P = (12, 17); 9P = (4, 5)$$

Because $9P = (4, 5) = Q$, the discrete logarithm $Q = (4, 5)$ to the base $P = (16, 5)$ is $k = 9$. In a real application, k would be so large as to make the brute-force approach infeasible.

Analog of Diffie-Hellman Key Exchange:

Key exchange using elliptic curves can be done in the following manner.

- First pick a large integer q, which is either a prime number p or an integer of the form 2^m, and elliptic curve parameters a and b for Equation (10.5) or Equation (10.7).
- This defines the elliptic group of points Eq(a, b).
- Next, pick a base point G = (x_1, y_1) in Ep(a, b) whose order is a very large value n.
- The **order** n of a point G on an elliptic curve is the smallest positive integer n such that nG = 0 and G are parameters of the cryptosystem known to all participants.
- A key exchange between users A and B can be accomplished as follows (Figure 10.7).

1. A selects an integer n_A less than n. This is A's private key. A then generates a public key $P_A = n_A \times G$; the public key is a point in $E_q(a, b)$.
2. B similarly selects a private key n_B and computes a public key P_B.
3. A generates the secret key $k = n_A \times P_B$. B generates the secret key $k = n_B \times P_A$.

The two calculations in step 3 produce the same result because

$$n_A \times P_B = n_A \times (n_B \times G) = n_B \times (n_A \times G) = n_B \times P_A$$

To break this scheme, an attacker would need to be able to compute k given G and kG, which is assumed to be hard.

Elliptic Curve Encryption/Decryption: The first task in this system is to encode the plaintext message m to be sent as an (x, y) point P_m.

It is the point P_m that will be encrypted as a ciphertext and subsequently decrypted. As with the key exchange system, an encryption/decryption system requires a point G and an elliptic group Eq(a, b) as parameters. Each user A selects a private key n_A and generates a public key $P_A = n_A X G$.

To encrypt and send a message P_m to B, A chooses a random positive integer k and produces the ciphertext C_m consisting of the pair of points:

$$C_m = \{kG, P_m + kP_B\}$$

Note that A has used B's public key P_B. To decrypt the ciphertext, B multiplies the first point in the pair by B's private key and subtracts the result from the second point:

$$P_m + kP_B - n_B(kG) = P_m + k(n_BG) - n_B(kG) = P_m$$

A has masked the message P_m by adding kP_B to it. Nobody but A knows the value of k, so even though P_b is a public key, nobody can remove the mask kP_B. However, A also includes a "clue," which is enough to remove the mask if one knows the private key n_B. For an attacker to recover the message, the attacker would have to compute k given G and kG, which is assumed to be hard.

Table 10.3 Comparable Key Sizes in Terms of Computational Effort for Cryptanalysis (NIST SP-800-57)

Symmetric Key Algorithms	Diffie-Hellman, Digital Signature Algorithm	RSA (size of *n* in bits)	ECC (modulus size in bits)
80	$L = 1024$ $N = 160$	1024	160–223
112	$L = 2048$ $N = 224$	2048	224–255
128	$L = 3072$ $N = 256$	3072	256–383
192	$L = 7680$ $N = 384$	7680	384–511
256	$L = 15{,}360$ $N = 512$	15,360	512+

Note: L = size of public key, N = size of private key

Security of Elliptic Curve Cryptography:

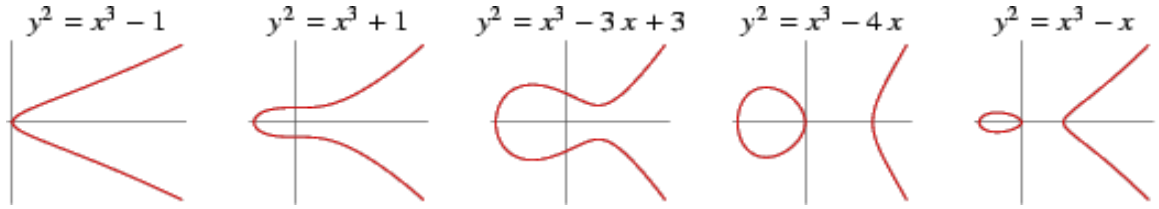
The security of ECC depends on how difficult it is to determine k given kP and P. This is referred to as the elliptic curve logarithm problem. The fastest known technique for taking the elliptic curve logarithm is known as the **Pollard rho method**. Table 10.3, from NIST SP800-57, compares various algorithms by showing comparable key sizes in terms of computational effort for cryptanalysis. A smaller key size can be used for ECC compared to RSA. For equal key lengths, the computational effort required for ECC and RSA is comparable. Thus, there is a computational **advantage to using ECC with a shorter key length than a comparably secure RSA**.

NOTE: General form of a EC

- **An *elliptic curve* is a plane curve defined by an equation of the form**

$$y^2 = x^3 + ax + b$$

Examples



Elliptic Curve on a finite set of Integers:

- Consider $y^2 = x^3 + 2x + 3 \pmod 5$

  $x = 0 \Rightarrow y^2 = 3 \Rightarrow$ no solution (mod 5)

  $x = 1 \Rightarrow y^2 = 6 = 1 \Rightarrow y = 1, 4 \pmod 5$

  $x = 2 \Rightarrow y^2 = 15 = 0 \Rightarrow y = 0 \pmod 5$

  $x = 3 \Rightarrow y^2 = 36 = 1 \Rightarrow y = 1, 4 \pmod 5$

  $x = 4 \Rightarrow y^2 = 75 = 0 \Rightarrow y = 0 \pmod 5$

- Then points on the elliptic curve are

  (1,1) (1,4) (2,0) (3,1) (3,4) (4,0)

  and the point at infinity: ∞

Using the finite fields we can form an Elliptic Curve Group.