DIFFIE HELLMAN KEY EXCHANGE

Example :

$q = 11$, $\alpha = 2$, $X_A = 6$, $X_B = 8$.

1. Choosen the prime number $q = 11$.
   choose α and it is a primitive root of q. $\alpha = 2$.

2. Choose any random value key for user A and User B.

   $X_A = 6$ $\qquad$ $X_B = 8$.

3. Find $Y_A = \alpha^{X_A} \bmod q$ $\qquad$ $Y_B = \alpha^{X_B} \bmod q$

   $Y_A = 2^6 \bmod 11$ $\qquad$ $Y_B = 2^8 \bmod 11$

   $= 64 \bmod 11$ $\qquad$ $= 256 \bmod 11$

   $\boxed{Y_A = 9}$ $\qquad$ $\boxed{Y_B = 3}$

Y_A is tranferred to User B.
Y_B is transferred to User A.

Shared Secret Key K of User A

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

$$K = 3^{6} \bmod 11$$

$$\equiv 729 \bmod 11$$

$$= 3.$$

Shared Secret Key K of User B.

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

$$= 9^{8} \bmod 11$$

$$= 43046721 \bmod 11 = 2.$$

To find $43046721 \bmod 11$

$43046721 / 11 = 3913338$

$3913338 \times 11 = 43046718$

$43046721 - 43046718 = \boxed{3}$

RSA Algorithm

Example: 2. $M = 8$, $e = 17$

1. Choose 2 prime numbers.

$p = 7$, $q = 11$, e.

2. $n = p \times q$

$= 7 \times 11$

$n = 77$

3. $\varphi(n) = (p-1)(q-1)$

$= (7-1)(11-1)$

$= 6 \times 10$

$\varphi(n) = 60$

4. choose e such that i) $1 < e < \varphi(n)$.

& $\gcd(e, \varphi(n)) = 1$. $1 < 17 < \varphi(n)$.

$\gcd(e, 60) = 1$

$\gcd(17, 60) = 1$.

5. $d = \dfrac{1 + k \cdot \varphi(n)}{e}$

$$d = \frac{1 + \phi(n)}{e}$$

$$d = \frac{1 + 60}{17} = \frac{61}{17} = 3.588$$

Substitute $k = 0$ to 16.

K=0 $d = \frac{1 + k \cdot \phi(n)}{e} = \frac{1 + 0.60}{17} = 1/17$.

K=1 $d = \frac{1+60}{17} = 3.58$

K=2 $d = \frac{1 + 2 \times 60}{17} = \frac{121}{17} = 7.11$

K=3 $d = \frac{1 + 3 * 60}{17} = \frac{181}{17} = 10.6$

K=4 $d = \frac{1 + 4 * 60}{17} = \frac{240}{17} = 14.1$

K=5 $d = \frac{1 + 5 * 60}{17} = \frac{301}{17} = 17.7$

- - - - -

K=15 $d = \frac{1 + 15 * 60}{17} = \frac{901}{17} = (53)$ whole no.

$d = 53$

Public Key $K_U = \{e, n\}$

$K_U = \{17, 77\}$

Private key $K_R = \{d, n\}$

$K_R = \{53, 77\}$.

Encryption:

$C = M^e \bmod n$

$C = 8^{17} \bmod 77$

$= 57$.

How to find: $8^{17} \bmod 77$.

$8^{17} \bmod 77 = 8^4 \bmod 77 \cdot (8 \cdot 8 \cdot 8 \cdot 8 \cdot 8)^4 \bmod 77$

$= [8^4 \bmod 77 \times 8^4 \bmod 77 \times 8^4 \bmod 77 \times 8^4 \bmod 77 \times 8^1 \bmod 77] \bmod 77$

$$= [15 \times 15 \times 15 \times 15 \times 8] \bmod 77$$

$$= 405000 \bmod 77$$

$8^{17} \bmod 77 = 57.$

Note:

$$8^4 \bmod 77 = 4096 \bmod 77$$
$$= 15.$$

$4096 \bmod 77 = ?$

1) $4096 / 77 = 53$

2) $53 \times 77 = 4081$

3) $4096 - 4081 = 15$

Decryption:

$$M = C^d \bmod n$$

$$= 57^{53} \bmod 77.$$

$$= (57^5 \cdot 57^5 \cdot 57^5 \cdot 57^5 \cdot 57^5 \cdot 57^5 \cdot 57^5 \cdot 57^5 \cdot 57^5 \cdot 57^5 \cdot 57^3) \bmod 77.$$

$$= (57^5 \bmod 77 \times 57^5 \bmod 77 \times 57^5 \bmod 77 \times$$
$$57^5 \bmod 77 \times 57^5 \bmod 77 \times 57^5 \bmod 77 \times$$
$$57^5 \bmod 77 \times 57^5 \bmod 77 \times 57^5 \bmod 77$$
$$\times 57^3 \bmod 77) \bmod 77$$

$$57^5 \bmod 77 = 6016920057 \bmod 77$$
$$= 43.$$

$57^3 \bmod 77 = 185193 \bmod 77$

$= 8$

$M = (43 \times 43 \times 43 \times 43 \times 43 \times 43 \times 43 \times 43 \times 43 \times 43 \times 43 \times 8) \bmod 77$

$= (43^5 \cdot 43^5 \cdot 8) \bmod 77$

$= (43^5 \bmod 77 \times 43^5 \bmod 77 \times 8 \bmod 77) \times \bmod 77$

$43^5 \bmod 77 = 147008443 \bmod 77$

$= 43$

$= (43 \times 43 \times 8) \bmod 77$

$= 14792 \bmod 77$

$M = 8.$

EULER'S THEOREM

- called as Euler totient function.

Theorem:

It states that if x and n are co-prime +ve integers, then $x^{\phi(n)} \equiv 1 \bmod n$.

where $\phi(n)$ is Euler Totient function.

$x^{\phi(n)} \bmod n = 1 \bmod n$

$\phi(n) = n - 1.$

$\phi(a * b) = \phi(a) * \phi(b)$

Example:

$x = 11$, $n = 10$. Both are co-prime.

∴ we can represent them as

$x^{\phi(n)} \equiv 1 \bmod n$.

$11^{\phi(10)} \equiv 1 \bmod 10.$

$\phi(10) = \phi(2 \times 5) = \phi(2) * \phi(5)$

$= 2 - 1 * 5 - 1$

$= 1 * 4$

$\phi(10) = 4$

Substitute $\varphi(10) = 4$

$$11^{\varphi(10)} \equiv 1 \bmod 10$$

$$11^{4} \equiv 1 \bmod 10.$$

$$14641 \equiv 1 \bmod 10 \quad \text{or}$$

$14641 \bmod 10 = 1.$

Note: $x^{\varphi(n)\cdot a} \equiv 1 \bmod n.$

ie) $11^{4\cdot 2} \equiv 1 \bmod n$

$$11^{8} \equiv 1 \bmod 10$$

$214358881 \equiv 1 \bmod 10$

$214358881 \bmod 10 = 1.$ true.

ie) Any multiple of $\varphi(n)$ will give the same result.

n	$\varphi(n)$	nos. coprime to n.	n	$\varphi(n)$	nos coprime to n
1	1	1	7	6	1, 2, 3, 4, 5, 6
2	1	1	8	4	1, 3, 5, 7
3	2	1, 2	9	6	1, 2, 4, 5, 7, 8
4	2	1, 3	10	4	1, 3, 7, 9
5	4	1, 2, 3, 4			
6	2	1, 5			

Note: Two integers a, b are said to be relatively prime, mutually prime or co-prime, if the only +ve integer / factor that divides both of them is 1.

* If n is prime, $\phi(n) = n - 1$.
* If n is not prime, $\phi(a * b) = \phi(a) * \phi(b)$.
  - a and b should be co-prime.

* Eg: $n = 11$.

$\phi(n) = n - 1$

$\phi(11) = 11 - 1 = 10$.

$\boxed{\phi(11) = 10}$

* Eg: $n = 35$

$\phi(a * b) = \phi(a) * \phi(b)$

$\phi(35) = \phi(5 * 7)$

$= \phi(5) * \phi(7)$

$= 4 * 6$

$\boxed{\phi(35) = 24}$

* Euler totient function ϕn is defined as the no. of +ve integers less than n that are co-prime to n.

(1) $\phi(6) = \{1, 5\}$. $n = 6$.

No. of +ve integers less than n } 1, 2, 3, 4, 5.
6 6 6
× × ×

These integers should not have common divisor with n.

$\phi(6) = \{1, 5\}$.

No. of elements in these set is the totient function.

(2) $\phi(8) = ?$ $n = 8$.

No. of +ve ints less than $n(8) = 1, \underline{2}, 3, \underline{4}, 5, \underline{6}, 7$

Nos. have common divisor with 8 = 2, 4, 6

Except these numbers = $\{1, 3, 5, 7\}$.

$\phi(8) = \{1, 3, 5, 7\}$.

$\phi(8) = 4$.

CHINESE REMAINDER THEOREM

Given :

$x \equiv a_1 \pmod{m_1}$

$x \equiv a_2 \pmod{m_2}$

$x \equiv a_3 \pmod{m_3}$

$x \equiv a_4 \pmod{m_4}$ ...

* These are called as simultaneous eqns.
* CRT is used to solve simultaneous equations.

* Given $x \equiv a_i \pmod{m_i}$
* Check whether m_1, m_2, m_3, m_4 .. are relatively prime or coprime. For coprime $\gcd(m_1, m_2) = 1$
* Calculate $M = m_1 \times m_2 \times m_3 \ldots m_i$
* Calculate $M_i = \frac{M}{m_i}$ $M_1 = \frac{M}{m_1}$, $M_2 = \frac{M}{m_2}$ $M_3 = \frac{M}{m_3}$ ...
* Calculate M_i^{-1}.

$$M_i \cdot M_i^{-1} \equiv 1 \pmod{m_i}$$

* Calculate $x = \sum \left[a_i \times M_i \times M_i^{-1} \right] \bmod M$.

NOTE:

If m_i is prime, $M_i^{-1} = M_i^{P-2} \bmod P$

> By fermat's little theorem,
>
> $$a^{-1} = a^{P-2} \pmod P$$

CRT states that there always exists x an 'x' that satisfies the given congruence.

$x \equiv a_1 \bmod m_1$

$x \equiv a_2 \bmod m_2$ ..

and m_1, m_2.. must be co-prime to one another.

Example:

$x \equiv 1 \bmod 5$ * find x?

$x \equiv 1 \bmod 7$

$x \equiv 3 \bmod 11$.

Solution:

Given:

$a_1 = 1 \quad a_2 = 1 \quad a_3 = 3$

$m_1 = 5 \quad m_2 = 7 \quad m_3 = 11$.

i) Check m_1, m_2 & m_3 are co-prime to each other.

$\gcd(5, 7) = 1$

$\gcd(5, 11) = 1$

$\gcd(7, 11) = 1$.

So m_1, m_2, m_2 are co-prime to each other.

ii) Calculate $M = m_1 \times m_2 \times m_3$

$M = 5 \times 7 \times 11$

$\boxed{M = 385}$

iii) Calculate M_1, M_2, M_3.

$M_1 = \frac{M}{m_1}$, $M_2 = \frac{M}{m_2}$, $M_3 = \frac{M}{m_3}$

$$M_1 = \frac{385}{5} = 77$$

$$M_2 = \frac{385}{7} = 55$$

$$M_3 = \frac{385}{11} = 35$$

$M_1 = 77$
$M_2 = 55$
$M_3 = 35$

$M = 385$
$m_1 = 5$
$m_2 = 7$
$m_3 = 11$

iv) Calculate M_i^{-1}.

Since m_1, m_2 and m_3 are prime numbers, we can find multiplicative inverse using Fermat theorem.

$$a^{-1} = a^{p-2} \bmod p.$$

To find M_1^{-1}:

$M_1 \cdot M_1^{-1} \equiv 1 \bmod m_i$

$M_1 = 77 \quad M_1^{-1} = ? \quad m_1 = 5$

$a = 77 \quad a^{-1} = ? \quad p = 5$

$$a^{-1} = a^{p-2} \bmod p$$

$$M_1^{-1} = M_1^{m_1-2} \bmod m_1$$

$$M_1^{-1} = 77^{5-2} \bmod 5$$

$$M_1^{-1} = 77^3 \bmod 5 = 456533 \bmod 5$$

$$M_1^{-1} = 3$$

$M_1 \cdot M_1^{-1} \equiv 1 \bmod m_1$ ∴

$x \times \frac{1}{x} = 1$

Find M_2^{-1}:

$M_2 = 55$, $M_2^{-1} = ?$ $m_2 = 7$

$$M_2^{-1} = M_2^{m_2-2} \bmod 7$$

$$M_2^{-1} = 55^{7-2} \bmod 7$$

$$= 55^5 \bmod 7$$

$$= 503,284,375 \bmod 7$$

$$\boxed{M_2^{-1} = 6.}$$

Find M_3^{-1}:

$M_3 = 35$, $M_3^{-1} = ?$ $m_3 = 11$.

$$M_3^{-1} = M_3^{m_3-2} \bmod m_3$$

$$= 35^{11-2} \bmod 11$$

$$= 35^9 \bmod 11 = \left(35^4 \times 35^5\right) \bmod 11$$

4774715 136420

$= (35^{4} \bmod 11 \times 35^{5} \bmod 11) \bmod 11$

$= (1500625 \bmod 11 \times 52521875 \bmod 11) \bmod 11$

$= (5 \times 10) \bmod 11$

$= 50 \bmod 11$

$M_3^{-1} = 6$

$M_1^{-1} = 3$

$M_2^{-1} = 6$

$M_3^{-1} = 6$

v) Calculate x.

$x = (a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1} + a_3 M_3 M_3^{-1}) \bmod M$

$a_1 = 1$	$a_2 = 1$	$a_3 = 3$	$M = 385$
$M_1 = 77$	$M_2 = 55$	$M_3 = 35$	
$M_1^{-1} = 3$	$M_2^{-1} = 6$	$M_3^{-1} = 6$	

$x = (1 \times 77 \times 3 + 1 \times 55 \times 6 + 3 \times 35 \times 6) \bmod 385$

$= (231 + 330 + 630) \bmod 385$

$= 1191 \bmod 385$

$x = 36$

We will verify the answer.

$x \equiv 1 \bmod 5$	$36 \equiv 1 \bmod 5$	$36 \bmod 5 = 1$
$x \equiv 1 \bmod 7$	$36 \equiv 1 \bmod 7$	$36 \bmod 7 = 1$
$x \equiv 3 \bmod 11$	$36 \equiv 3 \bmod 11$	$36 \bmod 11 = 3$

UNIT III
PUBLIC KEY CRYPTOGRAPHY

MATHEMATICS OF ASYMMETRIC KEY CRYPTOGRAPHY: Primes – Primality Testing – Factorization – Euler‘s totient function, Fermat‘s and Euler‘s Theorem – Chinese Remainder Theorem – Exponentiation and logarithm – ASYMMETRIC KEY CIPHERS: RSA cryptosystem – Key distribution – Key management – Diffie Hellman key exchange – ElGamal cryptosystem – Elliptic curve arithmetic-Elliptic curve cryptography.

MATHEMATICS OF ASYMMETRIC KEY CRYPTOGRAPHY

5.1. PRIMES

- 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.. In particular, note the number of primes in each range of 100 numbers.

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

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

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

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

- It is useful for what follows to express this another way. 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 ap 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.

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.

5.2. PRIMALITY TESTING

Contents
• **Testing for primality** ✓ **Miller-Rabin Algorithm** • **Two Properties of Prime Numbers** • **Details of the Algorithm** • **A Deterministic Primality Algorithm** • **Distribution of Primes**

Testing for primality:

Miller-Rabin Algorithm

- The algorithm due to Miller and Rabin [MILL75, RABI80] is typically used to test a large number for primality. Before explaining the algorithm, we need some background.
- First, any positive odd integer n ≥ 3 can be expressed as

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

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 mod 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 (a mod p) (a mod p) = a^2 mod p.

The second property is stated as follows:

1. a^q is congruent to *1 modulo p*. That is, a^q mod p = 1, or equivalently, $a^q \equiv 1 (\text{mod } p)$.
2. One of the numbers a^q, $a^{2^{q}a^{2k-l}}q$ is congruent to −1 modulo *p*.

Details of the Algorithm

- 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");
```

A Deterministic Primality Algorithm

- All of the algorithms in use, including the most popular (Miller-Rabin), produced a probabilistic result.
- AKS developed a relatively simple deterministic algorithm that efficiently determines whether a given large number is a prime. The algorithm, known as the AKS algorithm, does not appear to be as efficient as the Miller- Rabin algorithm.

Distribution of Primes

- A result from number theory, known as the prime number theorem, states that the primes near *n* are spaced on the average one every ln(*n*) integers.
- Thus, on average, one would have to test on the order of ln(*n*) integers before a prime is found. Because all even integers can be immediately rejected, the correct figure is 0.5 ln(*n*).

5.3. FACTORIZATION

5.4. EULER'S TOTIENT FUNCTION

- ✓ Euler's totient function Φ(n) defined as the number of positive integers less than n and Relatively prime to n. by conversion Φ(1)=1.

5.5. FERMAT'S AND EULER'S THEOREM

Contents
• **Fermat's Theorem**   • Proof: • **Euler's Totient Function**

- **Euler's Theorem**
  - Proof:

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

$$a^{p-1} \equiv 1 \pmod{p} \qquad \textbf{(8.2)}$$

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.
- 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 .

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

- Hence proved.
- Example

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

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

$$a^p \equiv a \pmod{p} \qquad \textbf{(8.3)}$$

Euler's Theorem

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

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

Proof:

- Equation (8.4) is true if n is prime, because in that case, f(n) = (n - 1) and Fermat's theorem holds.
- However, it also holds for any integer n. Recall that f(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 primeto *n*.
2. There are no duplicates in *S*. Refer to Equation (4.5). If ax_i mod n = ax_j mod *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.

5.5. CHINESE REMAINDER THEOREM

Let $m_1,\ldots,m_k$ be integers that are pairwise relatively prime integers. Define *M* to be the product of all the m_i's. Let $a_1, \ldots, a_k$ be integers. Then the set of congruences.

$x \equiv a1 (mod\ m1)$
$x \equiv a2 (mod\ m2)$
.
.
.
$x \equiv ak (mod\ mk)$
has a unique solution modulo M.

Proof:

- Put M= $m_1, m_2, \ldots\ldots m_r$ and for each k= 1,2,……r.

Let ,

$$M_k = \frac{M}{m_k}$$

- Then gcd (M_K, m_k) = 1 for all k.
- Let, y_k be an inverse of M_K modulo m_k for each k.
- Then by definition of inverse we have

$$M_K y_k = 1 (mod\ m_k)$$

- Let , $x = a_1 M_1 y_1 + a_2 M_2 y_2 + \ldots\ldots + a_k M_k y_k$.

- Then x is a simultaneous solution to all of the congruence.
- Since the modulo $m_1, m_2, \ldots\ldots m_r$ are pairwise relatively prime, any two simultaneous solution to the system must be congruent modulo M.
- Thus , the solution is a unique congruence class modulo M, and the value of x computed above is in that class.

5.6. EXPONENTIATION AND LOGARITHM

Contents
• **Introduction** • **The Powers of an Integer, Modulo *n*** • **Logarithms for Modular Arithmetic** • **Calculation of Discrete Logarithms**

Introduction

- 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 [Equation (8.4)] that, for every a and n that are relatively prime,

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

- where$\varphi(n)$, 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, $M = \varphi(n)$. 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*

- 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 $\varphi(19) = 18$. 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

Logarithms for Modular Arithmetic:

- With ordinary positive real numbers, the logarithm function is the inverse of exponentiation.An analogous function exists for modular arithmetic.
- 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

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

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

- 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 dloga,p(b).
  Note the following:

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

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.

(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

ASYMMETRIC KEY CIPHERS

5.7. RSA CRYPTOSYSTEM

Contents

- introduction
- Description of the Algorithm
- Computational Aspects
- The Security of RSA

Introduction:

- It was developed by **Rivest, Shamir and Adleman**. This algorithm makes use of an expression with exponentials.
- Plaintext is encrypted in blocks, with each block having a binary value less than some number n.
- The RSA scheme is a cipher in which the plaintext and cipher text 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 21024.

Description of the Algorithm

- That is, the block size must be less than or equal to $\log_2$ (n); in practice, the block size is k-bits, where $2^k < n < 2^{k+1}$.Encryption and decryption are of the following form, for some Plaintext block M and Cipher text block C:

$$\mathbf{C = M^e \bmod n}$$

$$M = C^d \bmod n$$

- Both the sender and receiver know the value of n. the sender knows the value of e and only the receiver knows the value of d. thus, this is a public key encryption algorithm with a public key of KU = {e, n} and a private key of KR = {d, n}.
- Let us focus on the first requirement. We need to find the relationship of the form:

$$\mathbf{M^{ed} = M \bmod n}$$

- A corollary to Euler's theorem fits the bill: Given two prime numbers p and q Integers,

n and m, such that n=pq and 0<m<n, and arbitrary integer k, the following relationship holds

$$m^{k\Phi(n)+1} = m^{k(p-1)(q-1)+1} = m \bmod n$$

- where Φ(n) – Euler totient function, which is the number of positive integers less than n and relatively prime to n.we can achieve the desired relationship, if

$$d = e^{-1} \bmod \Phi(n)$$

- That is, e and d are multiplicative inverses mod Φ(n). According to the rule of modular arithmetic, this is true only if d (and therefore e) is relatively prime to Φ(n). Equivalently, gcd (Φ(n), d) = 1.

The steps involved in RSA algorithm for generating the key are

- Select two prime numbers, p = 17 and q = 11.
- Calculate n = p*q = 17*11 = 187
- Calculate Φ (n) = (p-1) (q-1) = 16*10 = 160.
- Select e such that e is relatively prime to Φ (n) = 160 and less than Φ(n);
- we choose e =7.
- Determine d such that ed ≡ 1 mod Φ (n) and d<160. The correct value is d = 23, because 23*7 = 161 = 1 mod 160.

RSA algorithm is summarized below.

Figure 9.5. The RSA Algorithm

Key Generation	
Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
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	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

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

5.8. KEY DISTRIBUTION AND KEY MANAGEMENT

Distribution of Public Keys

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 (Figure 10.1).

- **For example**, because of the growing popularity of **PGP (pretty good privacy),** which makes use of RSA, many PGP users have adopted the practice of appending their public key to messages that they send to public forums, such as USENET newsgroups and Internet mailing lists.

Figure 10.1. 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.**

- Until such time as user A discovers the forgery and alerts other participants, the forger is able to read all encrypted messages intended for A and can use the forged keys for authentication.

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 10.2). 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.

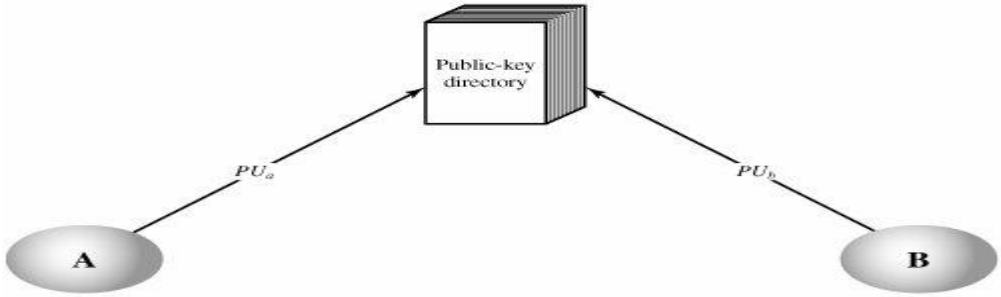


Figure 10.2. Public-Key Publication

- This scheme is clearly more secure than individual public announcements but still has vulnerabilities.
- If an adversary succeeds in obtaining or computing the private key of the directory authority, the adversary 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 adversary 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.
- A typical scenario is illustrated in Figure 10.3, 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.

The following steps:

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, 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, $\boldsymbol{P_{Ub}}$ which A can use to encrypt messages destined for B
- The original request, 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, so A can determine that this is not an old message from the authority containing a key other than B's current public key

3. A stores B's public key and also uses it to encrypt a message to B containing an identifier of A (*IDA*) and a nonce (*N*1), which is used to identify this transaction uniquely.

4. B retrieves A's public key from the authority in the same manner as A retrieved B's public key.

5. 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 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 (3), the presence of *N*1 in message (6) assures A that the correspondent is B.

7. A returns *N*2, 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 four 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.

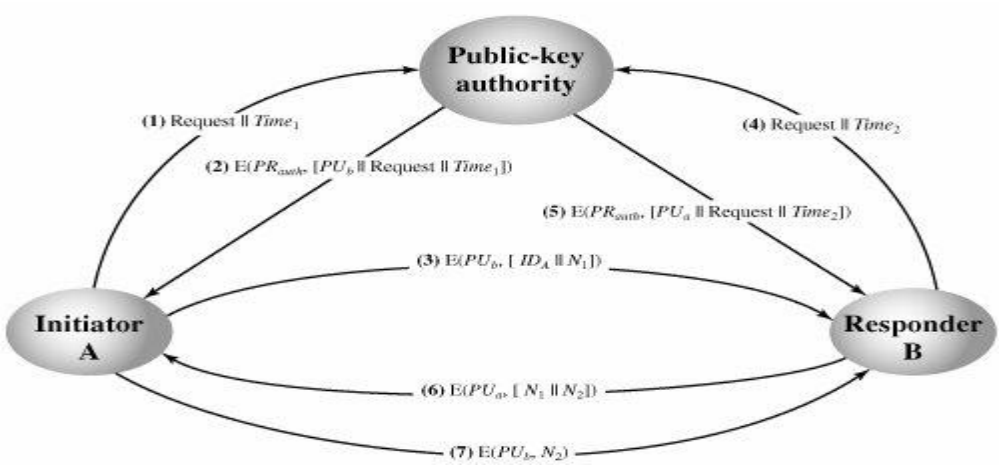


Figure 10.3. Public-Key Distribution Scenario

Public-Key Certificates

The scenario of Figure 10.3 is attractive, yet it has **some drawbacks**.

- The public-key authority could be somewhat of a bottleneck in the system, for a user must appeal to the authority for a public key for every other user that it wishes to contact.

- As before, the directory of names and public keys maintained by the authority is vulnerable to tampering.
- **An alternative approach,** first suggested by Kohnfelder [KOHN78], is to use **certificates** that can be used by participants to exchange keys without contacting a public-key authority, in a way that is as reliable as if the keys were obtained directly from a public-key authority.

- In essence, a certificate consists of a public key plus an identifier of the key owner, with the whole block signed by a trusted third party. Typically, the third party is a certificate authority, **such as a government agency or a financial institution,** that is trusted by the user community.

- A user can present his or her public key to the authority in a secure manner, and obtain a certificate. The user can then publish the certificate.

- Anyone needed this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature. A participant can also convey its key information to another by transmitting its certificate. Other participants can verify that the certificate was created by the authority.

We can place the following requirements on this 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.

Following additional requirement:

4. Any participant can verify the currency of the certificate.

- A certificate scheme is illustrated in Figure 10.4. Each participant applies to the certificate authority, supplying a public key and requesting a certificate.

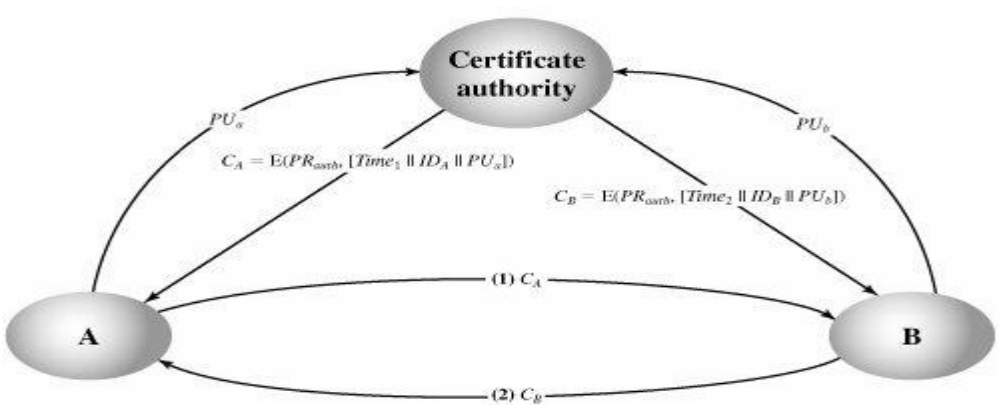


Figure 10.4. Exchange of Public-Key Certificates

- Application must be in person or by some form of secure authenticated communication. For participant A, the authority provides a certificate of the form

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

- Where PR_{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:

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

- 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.

- One scheme has become universally accepted for formatting public-key certificates: the X.509 standard. X.509 certificates are used in most network security applications, including IP security, **secure sockets layer (SSL), secure electronic transactions (SET), and S/MIME,**

5.9. KEY MANAGEMENT

Contents
• Introduction • Distribution of Public Keys • Distribution of Secret Keys Using Public-Key Cryptography

Introduction:

One of the major roles of public-key encryption has been to address the problem of key distribution. There are actually two distinct aspects to the use of public-key cryptography in this regard:

- The distribution of public keys
- The use of public-key encryption to distribute secret keys.

Distribution of Secret Keys Using Public-Key Cryptography

- Once public keys have been distributed or have become accessible, secure communication that thwarts However, few users will wish to make exclusive use of public-key encryption for communication because of the relatively slow data rates that can be achieved.

- Accordingly, public-key encryption provides for the distribution of secret keys to be used for conventional encryption.

Simple Secret Key Distribution:

An extremely simple scheme was put forward by Merkle [MERK79], as illustrated in Figure 10.5. 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, 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.

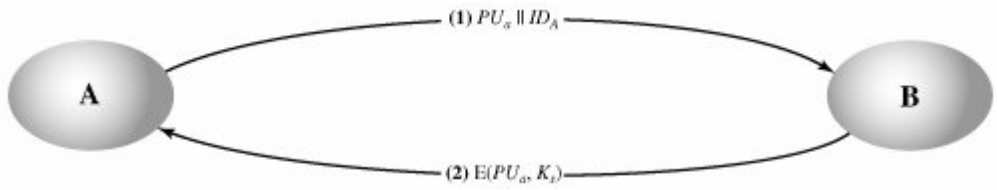


Figure 10.5. Simple Use of Public-Key Encryption to Establish a Session Key

- A and B can now securely communicate using conventional encryption and the session key *Ks*. At the completion of the exchange, both A and B discard *Ks*. 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 depicted in Figure 10.5 is insecure against an adversary who can intercept messages and then either relay the intercepted message or substitute another message (see Figure 1.4c). Such an attack is known as a **man-in-the-middle attack .**

In this case, if an adversary, E, has control of the intervening communication channel, then E can compromise the communication in the following fashion 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. E intercepts the message, creates its own public/private key pair $\{PU_e, PR_e\}$ and transmits $PUe \| ID_A$ to B.
3. B generates a secret key, *Ks*, and transmits $E(PU_e, K_s)$.
4. E intercepts the message, and learns *Ks* by computing $D(PR_e, E(PU_e, K_s))$.
5. E 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 E. A and B can now exchange messages using K_s E no longer actively interferes with the communications channel but simply eavesdrops.
- Knowing K_s E 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:

Figure 10.6, based on an approach, 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 described earlier in this section.

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 $\boldsymbol{M = E(PU_b, E(PR_a, K_s))}$ 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 D(PU_a, D(PR_b, M)) to recover the secret key.

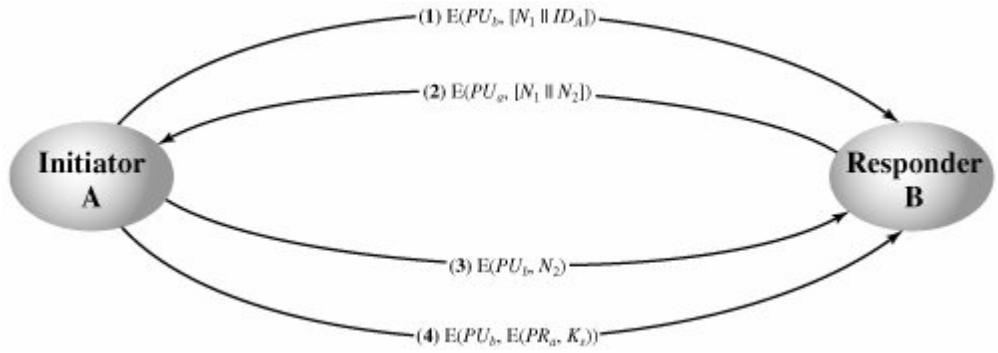


Figure 10.6. Public-Key Distribution of Secret Keys

Notice that the first three steps of this scheme are the same as the last three steps of Figure 10.3. The
result is that this scheme ensures both confidentiality and authentication in the exchange of a secret key.

A Hybrid Scheme

- Yet another way to use public-key encryption to distribute secret keys is a hybrid approach in use on IBM mainframes scheme retains the use of a **key distribution center (KDC**) that shares a secret master key with each user and distributes secret session keys encrypted with the master key.
- A public key scheme is used to distribute the master keys. The following rationale is provided for using this **three-level approach:**

● **Performance:** There are many applications, especially transaction-oriented applications, in which the session keys change frequently.Distribution of session keys by public-key encryption could degrade overall system performance because of the relatively high computational load of public-key encryption and decryption.

With a three-level hierarchy, public-key encryption is used only occasionally to update the master key between a user and the KDC.

● **Backward compatibility:** The hybrid scheme is easily overlaid on an existing KDC scheme, with
minimal disruption or software changes.

The addition of a public-key layer provides a secure, efficient means of distributing master keys. This is
an advantage in a configuration in which a single KDC serves a widely distributed set of users.

5.10. DIFFIE HELLMAN KEY EXCHANGE

Contents

Introduction

- The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms.
- That is, if a is a primitive root of the prime number p, then the numbers a mod p, a2 mod p,c, ap-1 mod p are distinct and consist of the integers from 1 through p - 1 in some permutation. For any integer b and a primitive root a of prime number p, we can find a unique exponent i such that **b K ai (mod p)** where 0 … i … (p - 1).
- The exponent i is referred to as the discrete logarithm of b for the base a, mod p.

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 that is a primitive root of q. Suppose the users A and B wish to create a shared key.

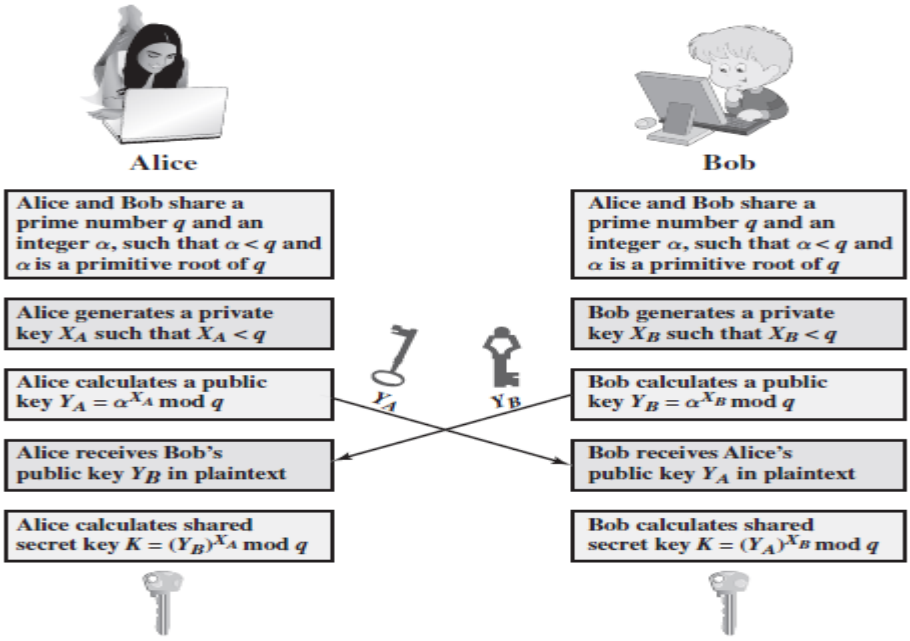


Figure 10.1 The Diffie-Hellman Key Exchange

- User A selects a random integer XA 6 q and computes **YA = aXA mod q.** Similarly, user B independently selects a random integer XB 6 q and computes YB = aXB mod q. Each side keeps the X value private and makes the Y value available publicly to the other side.

- Thus, XA is A's private key and YA is A's corresponding public key, and similarly for B. User A computes the key as K = (YB)XA mod q and user B computes the key as K = (YA)XB mod q. These two calculations produce identical results:

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

- 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 XA and XB are private, an adversary only has the following ingredients to work with: q, a, YA, and YB. Thus, the adversary is forced to take a discrete logarithm to determine the key.
- 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

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

- 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:

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

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 XA, calculate YA, and send that to user B. User B responds by generating a private value XB, calculating YB, and sending YB to user A. Both users can now calculate the key.
- The necessary public values q and α 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 $K2 = (Y_{D2})^{X_A} \bmod q$.

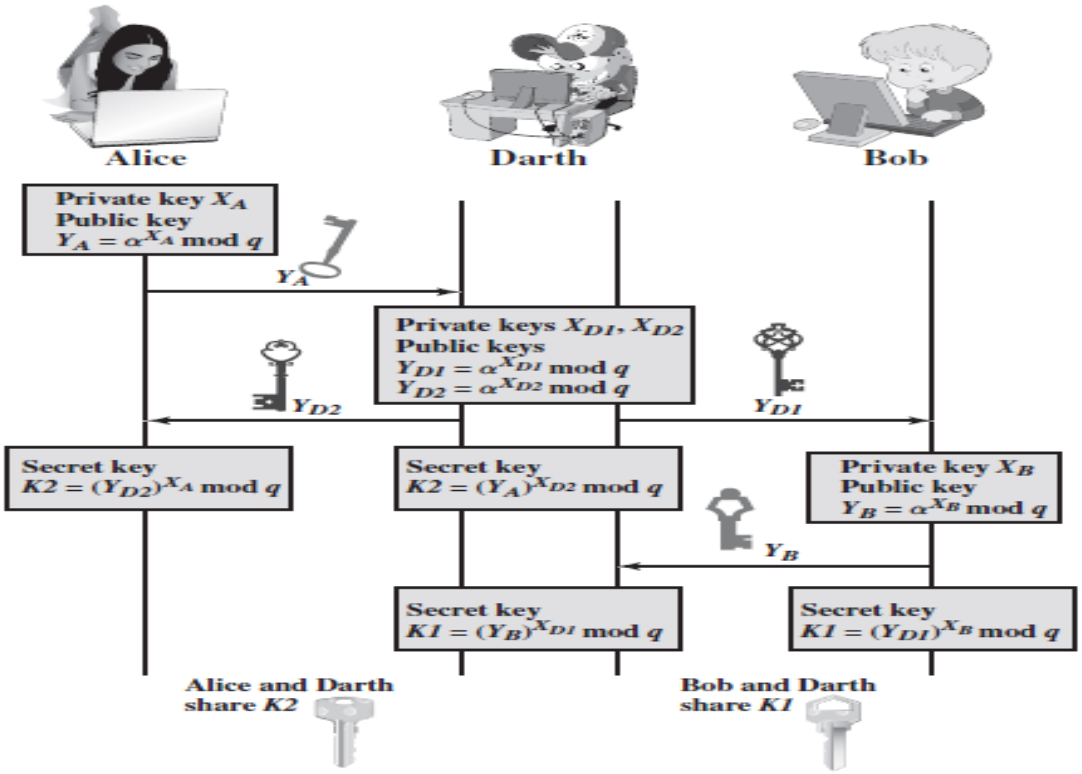


Figure 10.2 Man-in-the-Middle Attack

- 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.

5.11. ELGAMAL CRYPTOSYSTEM

In 1984, T. Elgamal announced a public-key scheme based on discrete logarithms, closely related to the Diffie-Hellman technique. The ElGamal2 cryptosystem is used in some form in a number of standards including the digital signature standard (DSS), and the S/MIME e-mail standard.
As with Diffie-Hellman, the global elements of ElGamal are a prime number and , which is a primitive root of .

User A generates a private/public key pair as follows:

1. Generate a random integer X_A, such that $1 < X_A < q - 1$.
2. Compute $Y^A = \alpha^{X_A} \bmod q$.
3. A's private key is X_A; A's pubic key is $\{q, \alpha, Y_A\}$.

Any user B that has access to A's public key can encrypt a message as follows:

1. Represent the message as an integer M in the range $0 \leq M \leq q - 1$. Longer messages are sent as a sequence of blocks, with each block being an integer less than q.
2. Choose a random integer k such that $1 \leq k \leq q - 1$.
3. Compute a one-time key $K = (Y_A)^k \bmod q$.
4. Encrypt M as the pair of integers (C_1, C_2) where

$$C_1 = \alpha^k \bmod q; \; C_2 = KM \bmod q$$

User A recovers the plaintext as follows:

1. Recover the key by computing $K = (C_1)^{X_A} \bmod q$.
2. Compute $M = (C_2 K^{-1}) \bmod q$.

These steps are summarized in following figure.
It corresponds to following scenario :
Alice generates a public/private key pair; Bob encrypts using Alice's public key; and Alice decrypts using her private key.

Let us demonstrate why the ElGamal scheme works. First, we show how 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

$$C_2 = KM \bmod q$$

$$(C_2 K^{-1}) \bmod q = KMK^{-1} \bmod q = M \bmod q = M$$

We can restate the ElGamal process as follows, using Figure 10.3.

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

The, K functions as a one-time key, used to encrypt and decrypt the message.*K C*1

Elgamal Cryptosystem

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	$Y_A = \alpha^{XA} \bmod q$
Public key	$PU = \{q, \alpha, Y_A\}$
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	$K = (C_1)^{XA} \bmod q$
Plaintext:	$M = (C_2 K^{-1}) \bmod q$

5.12. ELLIPTIC CURVE ARITHMETIC

Contents

- Elliptic Curves over Real Numbers
- Elliptic Curves over Zp
- Elliptic Curves over GF(2m)

- Most of the products and standards that use public-key cryptography for encryption and digital signatures use RSA. As we have seen, the key length for secure RSA use has increased over recent years, and this has put a heavier processing load on applications using RSA.
- This burden has ramifications, especially for electronic commerce sites that conduct large numbers of secure transactions. A competing system challenges RSA: **elliptic curve cryptography (ECC).** ECC is showing up in standardization efforts, including the IEEE P1363 Standard for Public-Key Cryptography.

- 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.

- On the other hand, although the theory of ECC has been around for some time, it is only recently that products have begun to appear and that there has been sustained cryptanalytic interest in probing for weaknesses. Accordingly, the confidence level in ECC is not yet as high as that in RSA.
- ECC is fundamentally more difficult to explain than either RSA or Diffie- Hellman, and a full mathematical description is beyond the scope of this book. This section and the next give some background on elliptic curves and ECC.

- We begin with a brief review of the concept of abelian group. Next, we examine the concept of elliptic curves defined over the real numbers. This is followed by a look at elliptic curves defined over finite fields. Finally, we are able to examine elliptic curve ciphers.

- 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:3

(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 $a \cdot \mathrm{e} = \mathrm{e} \cdot a = a$ for all a in G.
(A4) Inverse element:	For each a in G there is an element a' in G such that $a \cdot a' = a' \cdot a = \mathrm{e}$.
(A5) Commutative:	$a \cdot b = b \cdot a$ for all a, b in G.

- A number of public-key ciphers are based on the use of an abelian group.

- **For example, Diffie-Hellman key exchang**e involves multiplying pairs of nonzero integers modulo a prime number q. Keys are generated by exponentiation

the group, with exponentiation defined as repeated multiplication. For example, $a^k \bmod q = \underbrace{(a \times a \times \ldots \times a)}_{k \text{ times}} \bmod q$. To attack Diffie-Hellman, the attacker must determine k given a and a^k; this is the discrete logarithm problem.

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 $(a \times k)$.

- 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.
- Before looking at this, we first look at elliptic curves in which the variables and coefficients are real numbers. This case is perhaps easier to visualize.

Elliptic Curves over Real Numbers:1

- 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.4 For our purpose, it is sufficient to limit ourselves to equations of the form

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

- 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. As you can see, the formula sometimes produces weirdlooking 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.

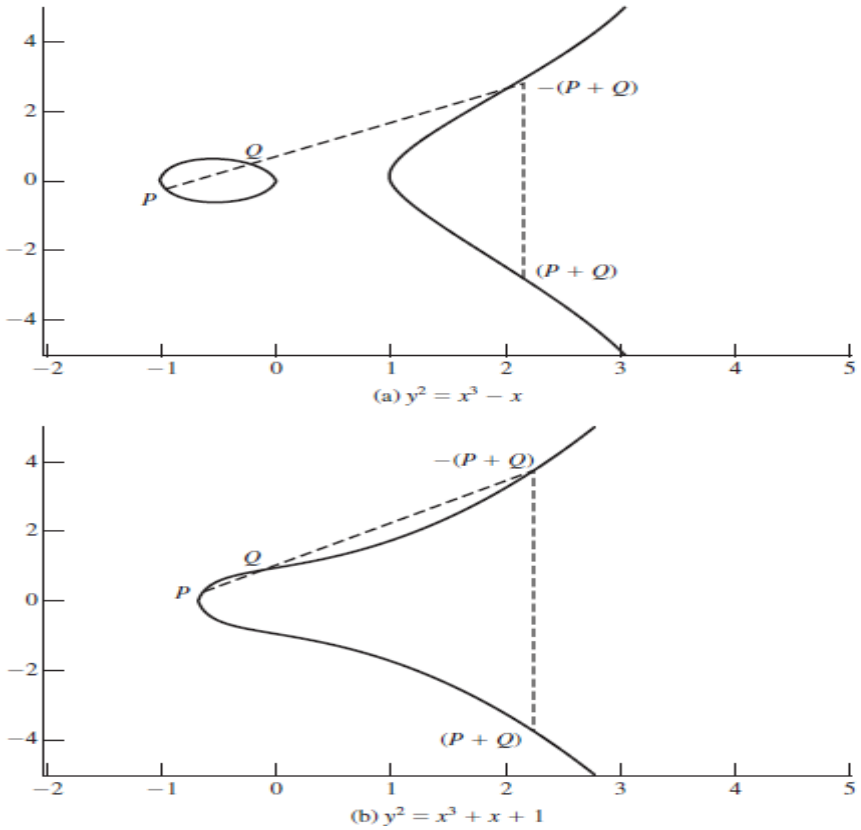


Figure 10.4 Example of Elliptic Curves

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 and binary
- curves over GF(2m). 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.

- For a **binary curve defined** over GF(2m), the variables and coefficients all take on values in GF(2m) and in calculations are performed over GF(2m).
- There is no obvious geometric interpretation of elliptic curve arithmetic over finite fields. The algebraic interpretation used for elliptic curve arithmetic over real numbers does readily carry over, and this is the approach we take.
- For elliptic curves over Zp, 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 Zp:

$$y^2 \bmod p = (x^3 + ax + b) \bmod p$$

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 Ep(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 Zp.

- **For example,** let p = 23 and consider the elliptic curve y2 = x3 + 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 E23(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 E23(1, 1). Figure 10.5 plots the points of E23(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)

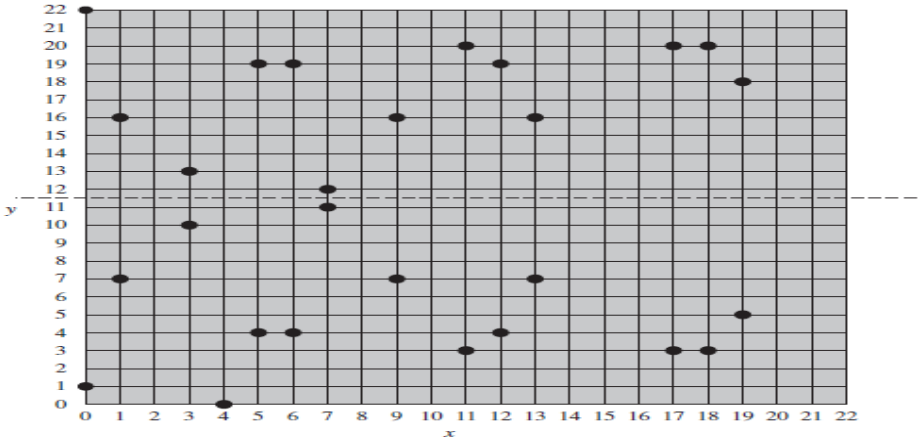


Figure 10.5 The Elliptic Curve $E_{23}(1, 1)$

- It can be shown that a finite abelian group can be defined based on the set Ep(a, b) provided that (x3 + ax + b) mod p has no repeated factors. This is equivalent to the condition

$$(4a^3 + 27b^2) \bmod p \neq 0 \bmod p$$

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 Z23. This can be done using the extended Euclidean algorithm defined in Section 4.4. To confirm, note that (6 * 4) mod 23 = 24 mod 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 EP(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 Ep(a, b) is approximately equal to the number of elements in Zp, namely p elements.

Elliptic Curves over GF(2m)

- Recall from Chapter 4 that a finite field GF(2m) consists of 2m elements, together with addition and multiplication operations that can be defined over polynomials.

- For elliptic curves over GF(2m), we use a cubic equation in which the variables and coefficients all take on values in GF(2m) for some number m and in which calculations are performed using the rules of arithmetic in GF(2m).

- It turns out that the form of cubic equation appropriate for cryptographic applications for elliptic curves is somewhat different for GF(2m) than for Zp. The form is
  - $y^2 + xy = x^3 + ax^2 + b$

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(2m) and that calculations are performed in GF(2m).

- Now consider the set E2m(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(24) with the irreducible polynomial f(x) = x4 + x + 1. This yields a generator g that satisfies f(g) = 0 with a value of g4 = 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:

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

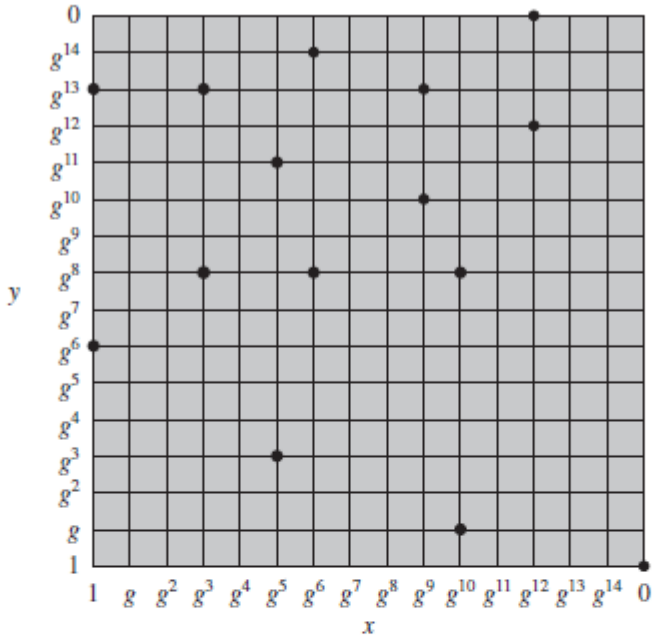


Figure 10.6 The Elliptic Curve $E_{24}(g^4, 1)$

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

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

5.13. ELLIPTIC CURVE CRYPTOGRAPHY

Contents
• Analog of Diffie-Hellman Key Exchange • Elliptic Curve Encryption/Decryption • Security of 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 $Q, P \in EP(a, b)$ and k 6 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.

- This is the group defined by the equation $y2 \bmod 23 = (x3 + 9x + 17) \bmod 23$. 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.

In the remainder of this section, we show two approaches to ECC that give the flavor of this technique.

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 2m, 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 = (x1, y1) 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

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

Elliptic Curve Encryption/Decryption

- Several approaches to encryption/decryption using elliptic curves have been analyzed in the literature. In this subsection, we look at perhaps the simplest. The first task in this system is to encode the plaintext message m to be sent as an (x, y) point Pm

- It is the point Pm that will be encrypted as a ciphertext and subsequently decrypted. Note that we cannot simply encode the message as the x or y coordinate of a point, because not all such coordinates are in Eq(a, b);

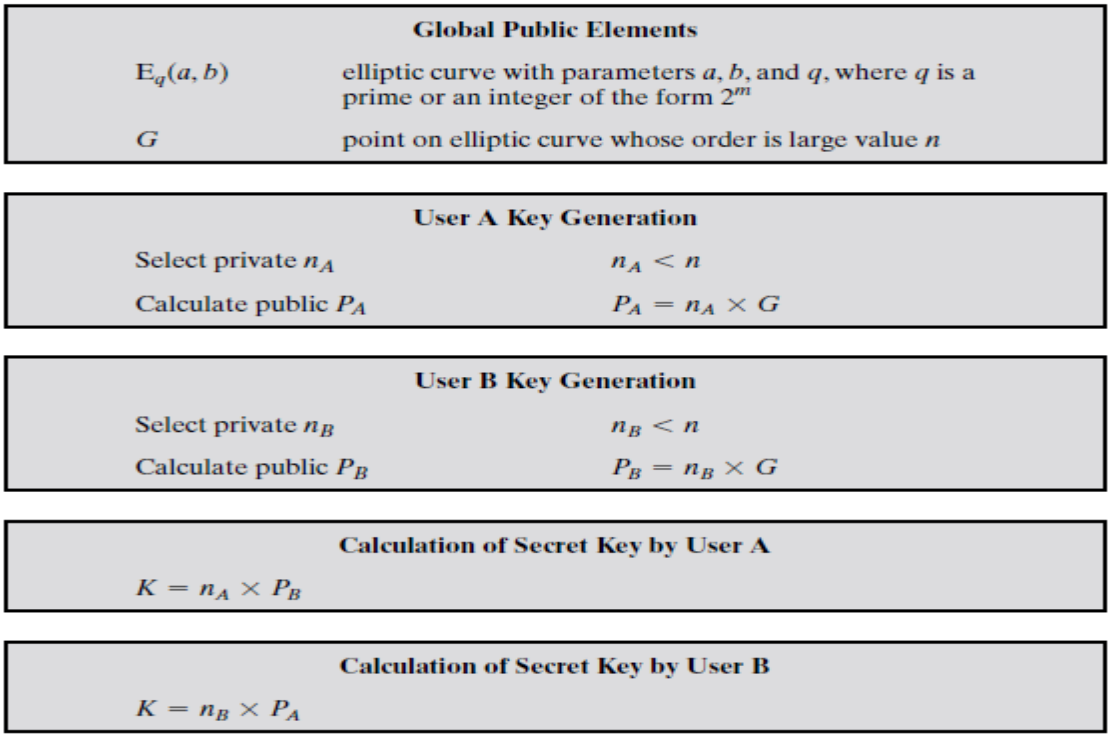


Figure 10.7 ECC Diffie-Hellman Key Exchange

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..

UNIT V
SECURITY PRACTICE AND SYSTEM SECURITY

Electronic Mail security – PGP, S/MIME – IP security – Web Security – SYSTEM SECURITY: Intruders – Malicious software – viruses – Firewalls.

5.1. ELECTRONIC MAIL SECURITY

- In virtually all distributed environments, electronic mail is the most heavily used network- based application. Users expect to be able to, and do, send e-mail to others who are connected directly or indirectly to the Internet, regardless of host operating system or communications suite.
- With the explosively growing reliance on e-mail, there grows a demand for authentication and confidentiality services. Two schemes stand out as approaches that enjoy widespread use: Pretty Good Privacy (PGP) and S/MIME. Both are examined in this chapter and Domain Keys Identified Mail.

Contents
• **Pretty Good Privacy** o Notation o Operational Description • **S/MIME** o RFC 5322 o Multipurpose Internet Mail Extensions o S/MIME Functionality o S/MIME Messages o S/MIME Certificate Processing o Enhanced Security Services

5.1.1. PGP

Contents
• **Pretty Good Privacy** o Notation o Operational Description

Pretty Good Privacy(PGP)

- PGP is a remarkable phenomenon. Largely the effort of a single person, Phil Zimmermann, PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications.

- **In essence, Zimmermann has done the following**:

1. Selected the best available cryptographic algorithms as building blocks.
2. Integrated these algorithms into a general-purpose application that is independent of operating system and processor and that is based on a small set of easy-to-use commands.
3. Made the package and its documentation, including the source code, freely available via the Internet, bulletin boards, and commercial networks such as AOL (America On Line).
4.Entered into an agreement with a company (Viacrypt, now Network Associates) to provide a fully compatible, low-cost commercial version of PGP.

- **Characteristics of PGP or PGP has grown explosively and is now widely used. A number of reasons can be cited for this growth.**
  1. It is available free worldwide in versions that run on a variety of platforms, including Windows, UNIX, Macintosh, and many more.
     .
  2. It is based on algorithms that have survived extensive public review and are considered extremely secure. Specifically, the package includes RSA, DSS, and Diffie-Hellman for public-key encryption; CAST-128, IDEA, and 3DES for symmetric encryption; and SHA-1 for hash coding.
  3. It has a wide range of applicability
  4. It was not developed by, nor is it controlled by, any governmental or standards organization.
  .
  5. PGP is now on an Internet standards track (RFC 3156; *MIME Security with OpenPGP*).
  6. The algorithms used are extremely secure

Notation

- Most of the notation used in this chapter has been used before, but a few terms are new. It is perhaps best to summarize those at the beginning. The following symbols are used.
  Ks = session key used in symmetric encryption scheme
  PRa = private key of user A, used in public-key encryption scheme
  PUa = public key of user A, used in public-key encryption scheme
  EP = public-key encryption
  DP = public-key decryption
  EC = symmetric encryption
  DC = symmetric decryption
  H = hash function
  } = concatenation
  Z = compression using ZIP algorithm
  R64 = conversion to radix 64 ASCII format1

- The PGP documentation often uses the term ***secret key*** to refer to a key paired with a public key in a public-key encryption scheme.
- As was mentioned earlier, this practice risks confusion with a secret key used for symmetric encryption. Hence, we use the term ***private key*** instead.

Operational Description in PGP

- The actual operation of PGP, as opposed to the management of keys, consists of four services**:**
  - **Authentication,**
  - **Confidentiality,**
  - **Confidentiality and Authentication,**
  - **E-mail**
  - **Compatibility**
- (Table 19.1). We examine each of these in turn.

Authentication

- Figure 19.1a illustrates the digital signature service provided by PGP. This is the digital signature scheme discussed in Chapter 13 and illustrated in Figure 13.2. The sequence is as follows.
  1. The sender creates a message.
  2. SHA-1 is used to generate a 160-bit hash code of the message.
  3. The hash code is encrypted with RSA using the sender's private key, and the result is prepended to the message.
  4. The receiver uses RSA with the sender's public key to decrypt and recover the hash code.
  5. The receiver generates a new hash code for the message and compares it with the decrypted hash code. If the two match, the message is accepted as authentic

Table 19.1 Summary of PGP Services

Function	Algorithms Used	Description
Digital signature	DSS/SHA or RSA/SHA	A hash code of a message is created using SHA-1. This message digest is encrypted using DSS or RSA with the sender's private key and included with the message.
Message encryption	CAST or IDEA or Three-key Triple DES with Diffie-Hellman or RSA	A message is encrypted using CAST-128 or IDEA or 3DES with a one-time session key generated by the sender. The session key is encrypted using Diffie-Hellman or RSA with the recipient's public key and included with the message.
Compression	ZIP	A message may be compressed for storage or transmission using ZIP.
E-mail compatibility	Radix-64 conversion	To provide transparency for e-mail applications, an encrypted message may be converted to an ASCII string using radix-64 conversion.

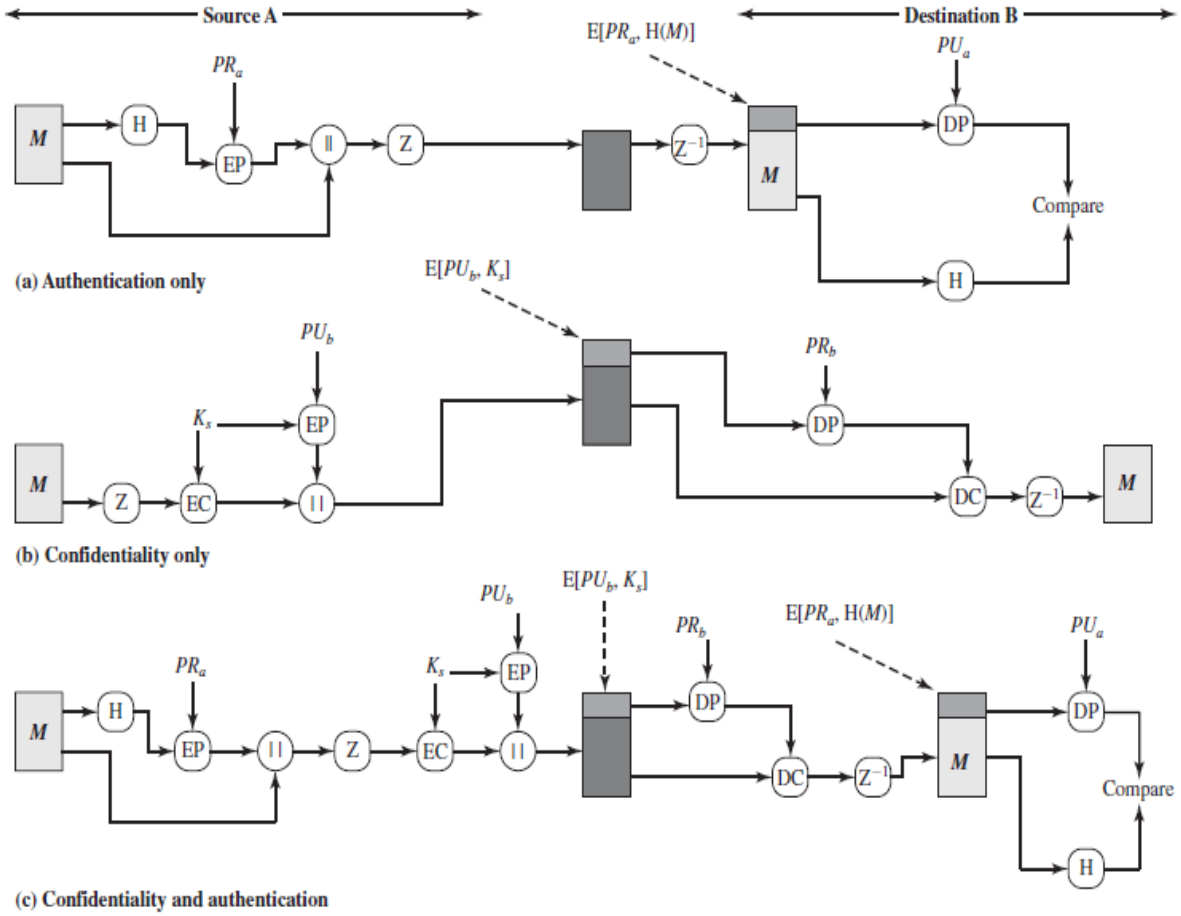


Figure 19.1 PGP Cryptographic Functions

- The combination of **SHA-1 and RSA** provides an effective digital signature scheme. Because of the strength of RSA, the recipient is assured that only the possessor of the matching private key can generate the signature.
- Because of the strength of SHA-1, the recipient is assured that no one else could generate a new message that matches the hash code and, hence, the signature of the original message. As an alternative, signatures can be generated using **DSS/SHA-1**.
- Although signatures normally are found attached to the message or file that they sign, this is not always the case: Detached signatures are supported.
- A detached signature may be stored and transmitted separately from the message it signs. This is useful in several contexts. A user may wish to maintain a separate signature log of all messages sent or received. A detached signature of an executable program can detect subsequent virus infection.
- Finally, detached signatures can be used when more than one party must sign a document, such as a legal contract. Each person's signature is independent and therefore is applied only to the document. Otherwise, signatures would have to be nested, with the second signer signing both the document and the first signature, and so on.

Confidentiality

- Another basic service provided by PGP is confidentiality, which is provided by encrypting messages to be transmitted or to be stored locally as files.
- In both cases, the symmetric encryption algorithm CAST-128 may be used.

- Alternatively, IDEA or 3DES may be used. The 64-bit cipher feedback (CFB) mode is used.
- As always, one must address the problem of key distribution. In PGP, each symmetric key is used only once.
- That is, a new key is generated as a random 128-bit number for each message. Thus, although this is referred to in the documentation as a session key, it is in reality a one-time key. Because it is to be used only once, the session key is bound to the message and transmitted with it.
- To protect the key, it is encrypted with the receiver's public key. Figure 19.1b illustrates the sequence,

Which can be described as follows?

1. The sender generates a message and a random 128-bit number to be used as a session key for this message only.
2. The message is encrypted using CAST-128 (or IDEA or 3DES) with the session key.
3. The session key is encrypted with RSA using the recipient's public key and is prepended to the message.
4. The receiver uses RSA with its private key to decrypt and recover the session key.
5. The session key is used to decrypt the message.

Confidentiality and Authentication

- As Figure 19.1c illustrates, both services may be used for the same message. First, a signature is generated for the plaintextmessage and prepended to the message.
- Then the plaintext message plus signatureis encrypted using CAST-128 (or IDEA or 3DES), and the session key is encrypted using RSA (or ElGamal).
- This sequence is preferable to the opposite: encrypting the message and then generating a signature for the encrypted message.
- It is generally more convenient to store a signature with a plaintext version of a message. Furthermore, for purposes of third-party verification, if the signature is performed first, a third party need not be concerned with the symmetric key when verifying the signature.

Compression

- As a default, PGP compresses the message after applying the signature but before encryption. This has the benefit of saving space both for e-mail transmission and for file storage.
- The placement of the compression algorithm, indicated by Z for compression and Z-1 for decompression in Figure 19.1, is critical.

  1. The signature is generated before compression for two reasons:

  a. It is preferable to sign an uncompressed message so that one can store only the uncompressed message together with the signature for future verification. If one signed a compressed document, then it would be necessary either to store a compressed version of the message for later verification or to recompress the message when verification is required.

  b. Even if one were willing to generate dynamically a recompressed message for verification, PGP's compression algorithm presents a difficulty. The algorithm is not deterministic; various implementations of the algorithm achieve different tradeoffs in

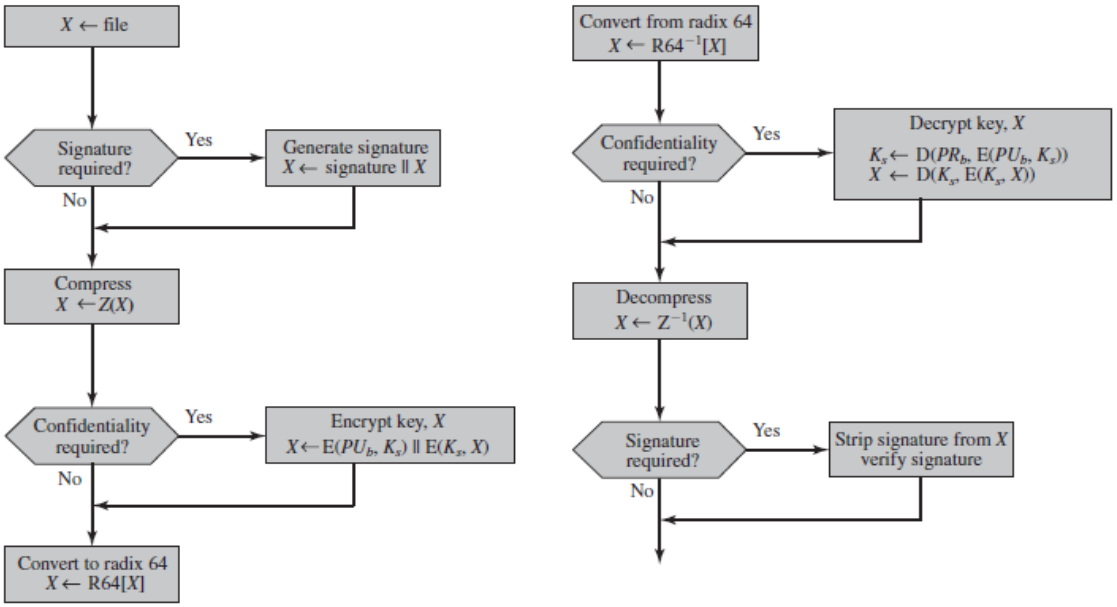
running speed versus compression ratio and, as a result, produce different compressed forms. However, these different compression algorithms are interoperable because any version of the algorithm can correctly decompress the output of any other version. Applying the hash function and signature after compression would constrain all PGP implementations to the same version of the compression algorithm.

2. Message encryption is applied after compression to strengthen cryptographic security. Because the compressed message has less redundancy than the original plaintext, cryptanalysis is more difficult.

.

E-mail Compatibility

- When PGP is used, at least part of the block to be transmitted is encrypted. If only the signature service is used, then the message digest is encrypted (with the sender's private key). If the confidentiality service is used, the message plus signature (if present) are encrypted (with a one-time symmetric key).
- Thus, part or the entire resulting block consists of a stream of arbitrary 8-bit octets. However, many electronic mail systems only permit the use of blocks consisting of ASCII text. To accommodate this restriction, PGP provides the service of converting the raw 8-bit binary stream to a stream of printable ASCII characters.
- The scheme used for this purpose is radix-64 conversion. Each group of three octets of binary data is mapped into four ASCII characters. This format also appends a CRC to detect transmission errors.
- The use of radix 64 expands a message by 33%. Fortunately, the session key and signature portions of the message are relatively compact, and the plaintext message has been compressed. In fact, the compression should be more than enough to compensate for the radix-64 expansion.



(a) Generic transmission diagram (from A)

(b) Generic reception diagram (to B)

Figure 19.2 Transmission and Reception of PGP Messages

5.1.2. S/MIME

Contents

S/MIME

- Secure/Multipurpose Internet Mail Extension (S/MIME) is a security enhancement to the MIME Internet e-mail format standard based on technology from RSA Data Security.
- Although both PGP and S/MIME are on an IETF standards track, it appears likely that S/MIME will emerge as the industry standard for commercial and organizational use, while PGP will remain the choice for personal e-mail security for many users. S/MIME is defined in a number of documents—most importantly RFCs 3370, 3850, 3851, and 3852.

RFC 5322

- RFC 5322 defines a format for text messages that are sent using electronic mail. It has been the standard for Internet-based text mail messages and remains in common use.
- In the RFC 5322 context, messages are viewed as having an envelope and contents. The envelope contains whatever information is needed to accomplish transmission and delivery. The contents compose the object to be delivered to the recipient.
- The RFC 5322 standard applies only to the contents. However, the content standard includes a set of header fields that may be used by the mail system to create the envelope, and the standard is intended to facilitate the acquisition of such information by programs.
- The overall structure of a message that conforms to RFC 5322 is very simple. A message consists of some number of header lines **(*the header*)** followed by unrestricted text **(*the body*).**
- The header is separated from the body by a blank line. Put differently, a message is ASCII text, and all lines up to the first blank line are assumed to be header lines used by the user agent part of the mail system.
- A header line usually consists of a keyword, followed by a colon, followed by the keyword's arguments; the format allows a long line to be broken up into several lines. The most frequently used keywords **are *From, To, Subject,* and *Date*.**
- **Here is an example message**:

```
Date: October 8, 2009 2:15:49 PM EDT
From: "William Stallings" <ws@shore.net>
Subject: The Syntax in RFC 5322
To: Smith@Other-host.com
Cc: Jones@Yet-Another-Host.com

Hello. This section begins the actual
message body, which is delimited from the
message heading by a blank line.
```

- Another field that is commonly found in RFC 5322 headers is *Message-ID*. This field contains a unique identifier associated with this message.

Multipurpose Internet Mail Extensions

- Multipurpose Internet Mail Extension (MIME) is an extension to the RFC 5322 framework that is intended to address some of the problems and limitations of the use of Simple Mail Transfer Protocol (SMTP), defined in RFC 821, or some other mail transfer protocol and RFC 5322 for electronic mail. [PARZ06]
- **Lists the following limitations of the SMTP/5322 scheme**.

  1. SMTP cannot transmit executable files or other binary objects. A number of schemes are in use for converting binary files into a text form that can be used by SMTP mail systems, including the popular UNIX UUencode/ UUdecode scheme. However, none of these is a standard or even a *de facto* standard.
  2. SMTP cannot transmit text data that includes national language characters, because these are represented by 8-bit codes with values of 128 decimal or higher, and SMTP is limited to 7-bit ASCII.
  3. SMTP servers may reject mail message over a certain size.
  4. SMTP gateways that translate between ASCII and the character code EBCDIC do not use a consistent set of mappings, resulting in translation problems.
  5. SMTP gateways to X.400 electronic mail networks cannot handle nontextual
  6. Some SMTP implementations do not adhere completely to the SMTP **standards defined in RFC 821. Common problems include:**
     - Deletion, addition, or reordering of carriage return and linefeed
     - Truncating or wrapping lines longer than 76 characters
     - Removal of trailing white space (tab and space characters)
     - Padding of lines in a message to the same length
     - Conversion of tab characters into multiple
- **Overview the MIME specification includes the following elements.**

  1. Five new message header fields are defined, which may be included in an RFC 5322 header. These fields provide information about the body of the message.
  2. A number of content formats are defined, thus standardizing representations that support multimedia electronic mail.
  3. Transfer encodings are defined that enable the conversion of any content format into a form that is protected from alteration by the mail system.

- **The five header fields defined in MIME are**

- **MIME-Version:** Must have the parameter value 1.0. This field indicates that the message conforms to RFCs 2045 and 2046.
- **Content-Type:** Describes the data contained in the body with sufficient detail that the receiving user agent can pick an appropriate agent or mechanism to represent the data to the user or otherwise deal with the data in an appropriate manner.
- **Content-Transfer-Encoding:** Indicates the type of transformation that has been used to represent the body of the message in a way that is acceptable for mail transport.
- **Content-ID:** Used to identify MIME entities uniquely in multiple contexts.
- **Content-Description:** A text description of the object with the body; this is useful when the object is not readable (e.g., audio data).

Mail Message Header

```
MIME-Version: 1.0
Content-type: multipart/mixed; boundary="simple boundary"
This is the preamble. It is to be ignored, though it is a
handy place for mail composers to include an explanatory
note to non-MIME conformant readers.
–simple boundary
This is implicitly typed plain ASCII text. It does NOT
end with a linebreak.
–simple boundary
Content-type: text/plain; charset=us-ascii
This is explicitly typed plain ASCII text. It DOES end
with a linebreak.
–simple boundary–
This is the epilogue. It is also to be ignored.
```

- **There are four subtypes of the multipart type,** all of which have the same overall syntax.
- The **multipart/mixed subtype** is used when there are multiple independent body parts that need to be bundled in a particular order.
- For the **multipart/ parallel subtype**, the order of the parts is not significant. If the recipient's system is appropriate, the multiple parts can be presented in parallel.
- For example, a picture

Table 19.2 MIME Content Types

Type	Subtype	Description
Text	Plain	Unformatted text; may be ASCII or ISO 8859.
	Enriched	Provides greater format flexibility.
Multipart	Mixed	The different parts are independent but are to be transmitted together. They should be presented to the receiver in the order that they appear in the mail message.
	Parallel	Differs from Mixed only in that no order is defined for delivering the parts to the receiver.
	Alternative	The different parts are alternative versions of the same information. They are ordered in increasing faithfulness to the original, and the recipient's mail system should display the "best" version to the user.
	Digest	Similar to Mixed, but the default type/subtype of each part is message/rfc822.
Message	rfc822	The body is itself an encapsulated message that conforms to RFC 822.
	Partial	Used to allow fragmentation of large mail items, in a way that is transparent to the recipient.
	External-body	Contains a pointer to an object that exists elsewhere.
Image	jpeg	The image is in JPEG format, JFIF encoding.
	gif	The image is in GIF format.
Video	mpeg	MPEG format.
Audio	Basic	Single-channel 8-bit ISDN mu-law encoding at a sample rate of 8 kHz.
Application	PostScript	Adobe Postscript format.
	octet-stream	General binary data consisting of 8-bit bytes.

- For the **multipart/alternative subtype**, the various parts are different representations of the same information. The following is an example:

```
From: Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Subject: Formatted text mail
MIME-Version: 1.0
Content-Type: multipart/alternative;
boundary = boundary42

   –boundary42
Content-Type: text/plain; charset = us-ascii
   ...plain text version of message goes here....

   –boundary42
Content-Type: text/enriched
   ...RFC 1896 text/enriched version of same message
goes here...

   –boundary42–
```

- The **multipart/digest subtype** is used when each of the body parts is interpreted as an RFC 5322 message with headers

MIME Transfer Encodings

.

- The MIME standard defines two methods of encoding data. The Content- Transfer-Encoding field can actually take on six values, as listed in Table 19.3. However, three of these values (7bit, 8bit, and binary) indicate that no encoding has been done but provide some information about the nature of the data.

Table 19.3 MIME Transfer Encodings

7bit	The data are all represented by short lines of ASCII characters.
8bit	The lines are short, but there may be non-ASCII characters (octets with the high-order bit set).
binary	Not only may non-ASCII characters be present, but the lines are not neces-sarily short enough for SMTP transport.
quoted-printable	Encodes the data in such a way that if the data being encoded are mostly ASCII text, the encoded form of the data remains largely recognizable by humans.
base64	Encodes data by mapping 6-bit blocks of input to 8-bit blocks of output, all of which are printable ASCII characters.
x-token	A named nonstandard encoding.

Mail Message Format

MIME-Version: 1.0
From: Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Subject: A multipart example
Content-Type: multipart/mixed;
Content-type: text/plain; charset=US-ASCII

Content-Type: multipart/parallel; boundary=unique-boundary-2
Content-Type: audio/basic
Content-Transfer-Encoding: base64
Content-type: text/enriched
Content-Type: message/rfc822
From: (mailbox in US-ASCII)
To: (address in US-ASCII)
Subject: (subject in US-ASCII)
Content-Type: Text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: Quoted-printable

Figure 19.3 Example MIME Message Structure

S/MIME Functionality

- In terms of general functionality, S/MIME is very similar to PGP. Both offer the ability to sign and/or encrypt messages.

S/MIME provides the following functions.

- **Enveloped data:** This consists of encrypted content of any type and encryptedcontent encryption keys for one or more recipients.

- **Signed data:** A digital signature is formed by taking the message digest of the content to be signed and then encrypting that with the private key of the signer. The content plus signature are then encoded using base64 encoding. A signed data message can only be viewed by a recipient with S/MIME capability.
- **Clear-signed data:** As with signed data, a digital signature of the content is formed. However, in this case, only the digital signature is encoded using base64. As a result, recipients without S/MIME capability can view the message content, although they cannot verify the signature.
- **Signed and enveloped data:** Signed-only and encrypted-only entities may be nested, so that encrypted data may be signed and signed data or clear-signed data may be encrypted.

Cryptographic Algorithms

- Table 19.5 summarizes the cryptographic algorithms used in S/MIME. S/MIME uses the following terminology taken from RFC 2119 **(*Key Words for use in RFCs to Indicate Requirement Levels*)** to specify the requirement level:

  - **MUST:** The definition is an absolute requirement of the specification. An implementation must include this feature or function to be in conformance with the specification.
  - **SHOULD:** There may exist valid reasons in particular circumstances to ignore this feature or function, but it is recommended that an implementation include the feature or function.

Table 19.5 Cryptographic Algorithms Used in S/MIME

Function	Requirement
Create a message digest to be used in forming a digital signature.	MUST support SHA-1. Receiver SHOULD support MD5 for backward compatibility.
Encrypt message digest to form a digital signature.	Sending and receiving agents MUST support DSS. Sending agents SHOULD support RSA encryption. Receiving agents SHOULD support verification of RSA signatures with key sizes 512 bits to 1024 bits.
Encrypt session key for transmission with a message.	Sending and receiving agents SHOULD support Diffie-Hellman. Sending and receiving agents MUST support RSA encryption with key sizes 512 bits to 1024 bits.
Encrypt message for transmission with a one-time session key.	Sending and receiving agents MUST support encryption with tripleDES. Sending agents SHOULD support encryption with AES. Sending agents SHOULD support encryption with RC2/40.
Create a message authentication code.	Receiving agents MUST support HMAC with SHA-1. Sending agents SHOULD support HMAC with SHA-1.

S/MIME Messages

- The general procedures for S/MIME message preparation

1. Securing a MIME Entity

- S/MIME secures a MIME entity with a signature, encryption, or both.
- A MIME entity may be an entire message (except for the RFC 5322 headers), or if the MIME content type is multipart, then a MIME entity is one or more of the subparts of the message. The MIME entity is prepared according to the normal rules for MIME message preparation. Then the MIME entity plus some security-related data, such as algorithm identifiers and certificates, are processed by S/MIME to produce what is known as a PKCS object.
- A PKCS object is then treated as message content and wrapped in MIME (provided with appropriate MIME headers).
- The message to be sent is converted to canonical form. In particular, for a given type and subtype, the appropriate canonical form is used for the message content. For a multipart message, the appropriate canonical form is used for each subpart.

2. Enveloped Data

- The steps for preparing an envelopedData MIME entity are
  1. Generate a pseudorandom session key for a particular symmetric encryption algorithm (RC2/40 or triple DES).
  2. For each recipient, encrypt the session key with the recipient's public RSA key.
  3. For each recipient, prepare a block known as RecipientInfo that contains an identifier of the recipient's public-key certificate,2 an identifier of the algorithm used to encrypt the session key, and the encrypted session key.
  4. Encrypt the message content with the session key.

A sample message (excluding the RFC 5322 headers) is

Content-Type: application/pkcs7-mime; smime-type=envelopeddata;
name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m
rfvbnj756tbBghyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
7n8HHGghyHhHUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H
f8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
0GhIGfHfQbnj756YT64V

- To recover the encrypted message, the recipient first strips off the base64 encoding. Then the recipient's private key is used to recover the session key. Finally, the message content is decrypted with the session key.

3. SignedData

- . The steps for preparing a signedData MIME entity are
  1. Select a message digest algorithm (SHA or MD5).
  2. Compute the message digest (hash function) of the content to be signed.
  3. Encrypt the message digest with the signer's private key.
  4. Prepare a block known as SignerInfo that contains the signer's public-key certificate, an identifier of the message digest algorithm, an identifier of the algorithm used to encrypt the message digest, and the encrypted message digest.

A sample message as follows

Content-Type: application/pkcs7-mime; smime-type=
signed-data; name=smime.p7m

Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m
567GhIGfHfYT6ghyHhHUujpfyF4f8HHGTrfvhJhjH776tbB9HG4VQbnj7
77n8HHGT9HG4VQpfyF467GhIGfHfYT6rfvbnj756tbBghyHhHUujhJhjH
HUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H7n8HHGghyHh
6YT64V0GhIGfHfQbnj75

- The recipient independently computes the message digest and compares it to the decrypted message digest to verify the signature.

4. Clear Signing

- Clear signing is achieved using the multipart content type with a signed subtype. As was mentioned, this signing process does not involve transforming the message to be signed, so that the message is sent "in the clear." Thus, recipients with MIME capability but not S/MIME capability are able to read the incoming message.

- A multipart/signed message has two parts.
    - The first part can be any **MIME type** but must be prepared so that it will not be altered during transfer from source to destination. This means that if the first part is not 7bit, then it needs to be encoded using base64 or quoted-printable.
    - This second part has **a MIME content type** of application and a subtype of pkcs7-signature. Here is a sample message:

Content-Type: multipart/signed;
protocol="application/pkcs7-signature";
micalg=sha1; boundary=boundary42
—boundary42
Content-Type: text/plain
This is a clear-signed message.
—boundary42
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s
ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHf
YT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrf
vbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujp
fyF4
7GhIGfHfYT64VQbnj756
—boundary42—

4. ***Registration Request***

- **The certification request includes**

1. Certification Request Info block
2. Identifier of the public key encryption algorithm
3. Signature of the certification RequestInfo block

S/MIME Certificate Processing

- S/MIME uses public-key certificates that conform to version 3 of X.509 . The key-management scheme used by S/MIME is in some ways a hybrid between a strict X.509 certification hierarchy and PGP's web of trust.
- As with the PGP model, S/MIME managers and/or users must configure each client with a list of trusted keys and with certificate revocation lists.

1. User Agent Role

- An S/MIME user has several key-management functions to perform.
  - **Key generation:**. A user agent SHOULD generate RSA key pairs with a length in the range of 768 to 1024 bits and MUST NOT generate a length of less than 512 bits.
  - **Registration:** A user's public key must be registered with a certification authority in order to receive an X.509 public-key certificate.
  - **Certificate storage and retrieval:** A user requires access to a local list of certificates in order to verify incoming signatures and to encrypt outgoing messages.

2. VeriSign Certificates

- VeriSign provides a CA service that is intended to be compatible with S/MIME and a variety of other applications. VeriSign issues X.509 certificates with the product name VeriSign Digital ID.
- Each Digital ID contains the following :
  - Owner's public key
  - Owner's name or alias
  - Expiration date of the Digital ID
  - Serial number of the Digital ID
  - Name of the certification authority that issued the Digital ID
  - Digital signature of the certification authority that issued the Digital ID

- Digital IDs can also contain other user-supplied information, including
  - Address
  - E-mail address
  - Basic registration information (country, zip code, age, and gender)

Enhanced Security Services

The details of these may change, and additional services may be added**. The three services are**

- **Signed receipts:** A signed receipt may be requested in a SignedData object. Returning a signed receipt provides proof of delivery to the originator of a message and allows the originator to demonstrate to a third party that the recipient received the message
- **Security labels:** A security label may be included in the authenticated attributes of a SignedData object. A security label is a set of security information regarding the sensitivity of the content that is protected by S/MIME encapsulation. The labels may be used for access control, by indicating which users are permitted access to an object.
- **Secure mailing lists:** When a user sends a message to multiple recipients, a certain amount of per-recipient processing is required, including the use of each recipient's public key. The user can be relieved of this work by employing the services of an S/MIME Mail List Agent (MLA). An MLA can take a single incoming message, perform the recipient-specific encryption for each recipient, and forward the message..

5.2. IP SECURITY

Contents
• **IP Security Overview** ○ Applications of IPsec ○ Benefits of IPsec ○ Routing Applications • **IP Security Architecture** ○ IPsec Documents ○ IPsec Services ○ Security Associations (SA) • **Transport and Tunnel Modes**

IP Security Overview

- To provide security, the IAB included authentication and encryption as necessary security features in the next-generation IP, which has been issued as IPv6. Fortunately, these security capabilities were designed to be usable both with the current IPv4 and the future IPv6. This means that vendors can begin offering these features now, and many vendors now do have some IPsec capability in their products.
- The IPsec specification now exists as a set of Internet standards.

Applications of IPSec

- IPSec provides the capability to secure communications across a LAN, across private and public WANs, and across the Internet. Examples of its use include the following:
  - **Secure branch office connectivity over the Internet:** A company can build a secure virtual private network over the Internet or over a public WAN. This enables a business to rely heavily on the Internet and reduce its need for private networks, saving costs and network management overhead.
  - **Secure remote access over the Internet:** An end user whose system is equipped with IP security protocols can make a local call to an Internet service provider (ISP) and gain secure access to a company network. This reduces the cost of toll charges for traveling employees and telecommuters.
  - **Establishing extranet and intranet connectivity with partners:** IPSec can be used to secure communication with other organizations, ensuring authentication and confidentiality and providing a key exchange mechanism.

- **Enhancing electronic commerce security:** Even though some Web and electronic commerce applications have built-in security protocols, the use of IPSec enhances that security.

- The principal feature of IPSec that enables it to support these varied applications is that it can encrypt and/or authenticate *all* traffic at the IP level.
- Thus, all distributed applications, including remote logon, client/server, e-mail, file transfer, Web access, and so on, can be secured.
- Figure 20.1 is a typical scenario of IPsec usage. An organization maintains LANs at dispersed locations. Nonsecure IP traffic is conducted on each LAN. For traffic offsite, through some sort of private or public WAN, IPsec protocols are used.
- These protocols operate in networking devices, such as a router or firewall, that connect each LAN to the outside world. The IPsec networking device will typically encrypt and compress all traffic going into the WAN and decrypt and decompress traffic coming from the WAN; these operations are transparent to workstations and servers on the LAN.
- Secure transmission is also possible with individual users who dial into the WAN. Such user workstations must implement the IPsec protocols to provide security.

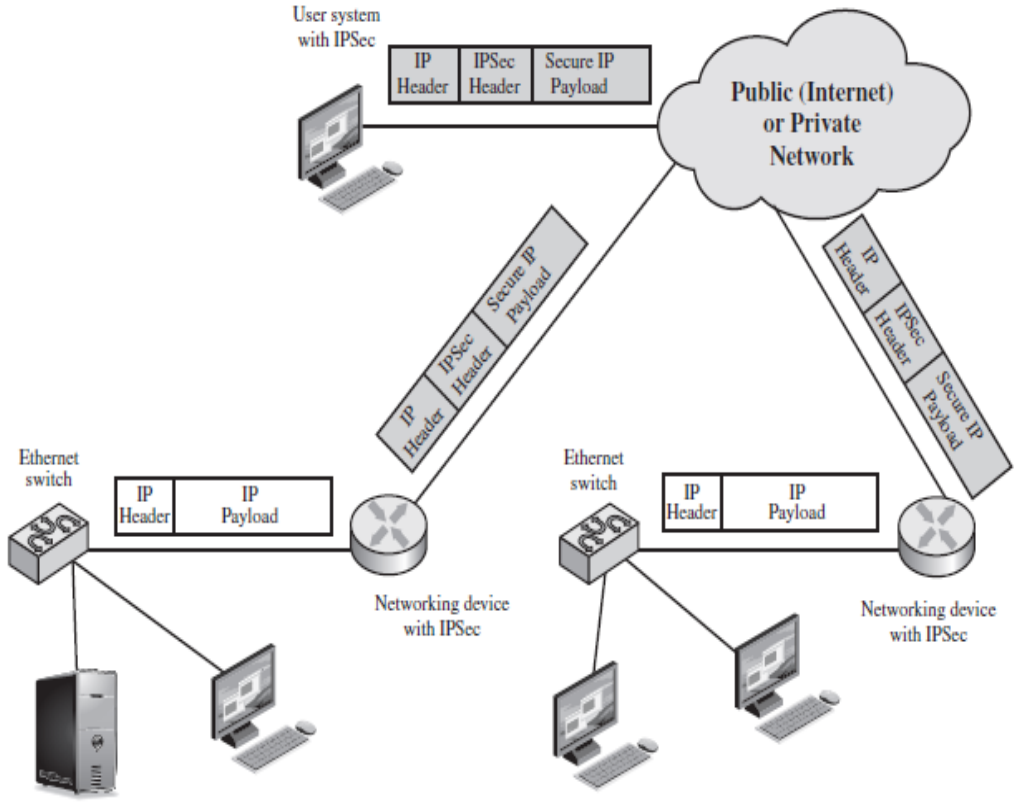


Figure 20.1 An IP Security Scenario

Benefits of IPSec

- [MARK97] lists the following benefits of IPSec: When IPSec is implemented in a firewall or router, it provides strong security that can be applied to all traffic crossing

the perimeter. Traffic within a company or workgroup does not incur the overhead of security-related processing.

- IPSec in a firewall is resistant to bypass if all traffic from the outside must use IP, and the firewall is the only means of entrance from the Internet into the organization.
- IPSec is below the transport layer (TCP, UDP) and so is transparent to applications.
- There is no need to change software on a user or server system when IPSec is implemented in the firewall or router.
- Even if IPSec is implemented in end systems, upper-layer software, including applications, is not affected.
- IPSec can be transparent to end users. There is no need to train users on security mechanisms, issue keying material on a per-user basis, or revoke keying material when users leave the organization.
- IPSec can provide security for individual users if needed. This is useful for offsite workers and for setting up a secure virtual sub network within an organization for sensitive applications.

Routing Applications

- A router advertisement (a new router advertises its presence) comes from an authorized router.
- neighbor advertisement (a router seeks to establish or maintain a neighbor
- relationship with a router in another routing domain) comes from an authorized router.
- A redirect message comes from the router to which the initial IP packet was sent.
- A routing update is not forged.

IP Security Architecture

- The IPSec specification has become quite complex. To get a feel for the overall architecture, we begin with a look at the **documents that define IPSec**. Then we discuss **IPSec services** and introduce the concept of **security association(SA).**

1. IPsec Documents

- The IPSec specification consists of numerous documents. The most important of these, issued in November of 1998, are RFCs 2401, 2402, 2406, and 2408:
  - **RFC 2401: An overview of a security architecture**
  - **RFC 2402: Description of a packet authentication extension to IPv4 and IPv6**
  - **RFC 2406: Description of a packet encryption extension to IPv4 and IPv6**
  - **RFC 2408: Specification of key management capabilities**
- Support for these features is mandatory for IPv6 and optional for IPv4. In both cases, the security features are implemented as extension headers that follow the main IP header.
- The extension header for authentication is known as the Authentication header; that for encryption is known as the Encapsulating Security Payload (ESP) header.

- IPsec encompasses three functional areas: authentication, confidentiality, and key anagement. The totality of the IPsec specification is scattered across dozens of RFCs and draft IETF documents, making this the most complex and difficult to grasp of all IETF specifications.

- The best way to grasp the scope of IPsec is to consult the latest version of the IPsec document roadmap, which as of this writing is RFC 6071 **[*IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap,***
- **The documents can be categorized into the following groups**.

    - **Architecture:** Covers the general concepts, security requirements, definitions, and mechanisms defining IPsec technology. The current specification is RFC 4301, ***Security Architecture for the Internet Protocol***.

    - **Authentication Header (AH):** AH is an extension header to provide message authentication. The current specification is RFC 4302, ***IP Authentication***

    - ***Heade****r*. Because message authentication is provided by ESP, the use of AH is deprecated. It is included in IPsecv3 for backward compatibility but should not be used in new applications.

    - **Encapsulating Security Payload (ESP):** ESP consists of an encapsulating header and trailer used to provide encryption or combined encryption/authentication. The current specification is RFC 4303, *IP **Encapsulating Security Payload (ESP)*.

    - **Internet Key Exchange (IKE):** This is a collection of documents describing the key management schemes for use with IPsec. The main specification is RFC 5996, ***Internet Key Exchange (IKEv2) Protocol***, but there are a number of related RFCs.

    - **Cryptographic algorithms:** This category encompasses a large set of documents that define and describe cryptographic algorithms for encryption, message authentication, pseudorandom functions (PRFs), and cryptographic key exchange.

    - **Other:** There are a variety of other IPsec-related RFCs, including those dealing with security policy and management information base (MIB) content.

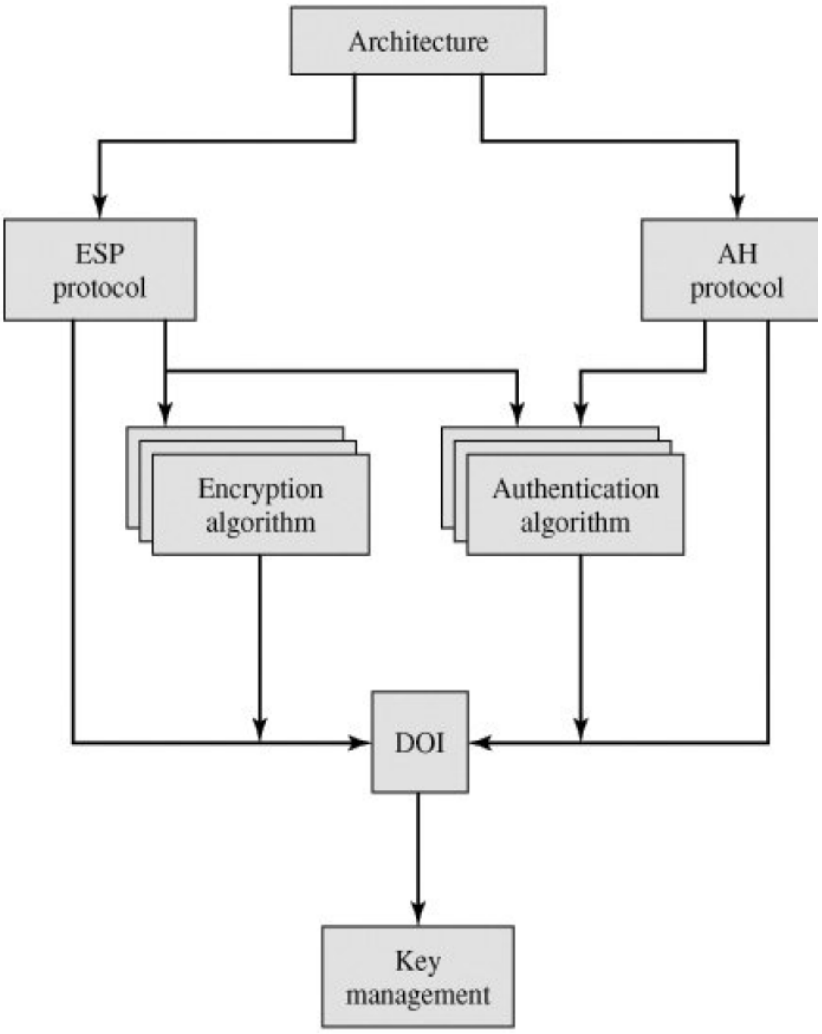


Figure 16.2. IPSec Document Overview

2. IPsec Services

- IPsec provides security services at the IP layer by enabling a system to select required ecurity protocols, determine the algorithm(s) to use for the service(s), and put in place any cryptographic keys required to provide the requested services.
- Two protocols are used to provide security: an authentication protocol designated by the header of the protocol, **Authentication Header (AH);** and a combined encryption/ authentication protocol designated by the format of the packet for that protocol, **Encapsulating Security Payload (ESP).**
- RFC 4301 lists the following services:
    - Access control
    - Connectionless integrity
    - Data origin authentication
    - Rejection of replayed packets (a form of partial sequence integrity)
    - Confidentiality (encryption)
    - Limited traffic flow confidentiality

	AH	ESP (encryption only)	ESP (encryption plus authentication)
Access control	✔	✔	✔
Connectionless integrity	✔		✔
Data origin authentication	✔		✔
Rejection of replayed packets	✔	✔	✔
Confidentiality		✔	✔
Limited traffic flow confidentiality		✔	✔

3. Security Associations (SA)

- A key concept that appears in both the authentication and confidentiality mechanisms for IP is the security association (SA). An association is a one-way relationship between a sender and a receiver that affords security services to the traffic carried on it.
- If a peer relationship is needed, for two-way secure exchange, then two security associations are required.
- A security association is uniquely identified by three parameters:

Security Parameters Index (SPI): A bit string assigned to this SA and having local significance only. The SPI is carried in AH and ESP headers to enable the receiving system to select the SA under which a received packet will be processed.

IP Destination Address: Currently, only unicast addresses are allowed; this is the address of the destination endpoint of the SA, which may be an end user system or a network system such as a firewall or router.

Security Protocol Identifier: This indicates whether the association is an AH or ESP security Association

SA Parameters

- In each IPSec implementation, there is a nominal Security Association Database that defines the parameters associated with each SA. A security association is normally defined by the following parameters:
  - [**Sequence Number Counter:** A 32-bit value used to generate the Sequence Number field in AH or ESP headers,
  - **Sequence Counter Overflow:** A flag indicating whether overflow of the Sequence Number Counter should generate an auditable event and prevent further transmission of packets on this SA (required for all implementations).
  - **ESP Information:** Encryption and authentication algorithm, keys, initialization values, key lifetimes, and related parameters being used with ESP (required for ESP implementations).
  - **Lifetime of This Security Association:** A time interval or byte count after which an SA must be replaced with a new SA (and new SPI) or terminated, plus an indication of which of these actions should occur (required for all implementations).
  - **IPSec Protocol Mode:** Tunnel, transport, or wildcard (required for all implementations). These modes are discussed later in this section.
  - **Path MTU:** Any observed path maximum transmission unit (maximum size of a packet that can
  - be transmitted without fragmentation) and aging variables (required for all implementations). **Anti-Replay Window:** Used to determine whether an inbound AH or ESP packet is a replay,
  - described in Section 16.3 (required for all implementations).

- **AH Information:** Authentication algorithm, keys, key lifetimes, and related parameters being used with AH (required for AH implementations).

SA Selectors

- IPSec provides the user with considerable flexibility in the way in which IPSec services are applied to IP traffic.
- The means by which IP traffic is related to specific SAs (or no SA in the case of traffic allowed to bypass IPSec) is the nominal Security Policy Database (SPD).
- In its simplest form, an SPD contains entries, each of which defines a subset of IP traffic and points to an SA for that traffic.
- In more complex environments, there may be multiple entries that potentially relate to a single SA or multiple SAs associated with a single SPD entry. The reader is referred to the relevant IPSec documents for a full discussion.
- Each SPD entry is defined by a set of IP and upper-layer protocol field values, called ***selectors*.**
- In effect, these selectors are used to filter outgoing traffic in order to map it into a particular SA. Outbound processing obeys the following general sequence for each IP packet:

Transport and Tunnel Modes

- Both **AH and ESP support** two modes of use: **transport and tunnel mode**. The operation of these two modes is best understood in the context of a description of ESP. Here we provide a brief overview.

1. Transport Mode

- Transport mode provides protection primarily for upper-layer protocols. That is, transport mode protection extends to the payload of an IP packet.1 Examples include a TCP or UDP segment or an ICMP packet,
- The transport mode is used for end - to- end communication between two hosts (e.g., a client and a server, or two workstations).
- When a host runs AH or ESP over IPv4, the payload is the data that normally follow the IP header. For IPv6, the payload is the data that normally follow both the IP header and any IPv6 extensions headers that are present, with the possible exception of the destination options header, which may be included in the protection.
- ESP in transport mode encrypts and optionally authenticates the IP payload but not the IP header. AH in transport mode authenticates the IP payload and selected portions of the IP header.

2. Tunnel Mode

- Tunnel mode provides protection to the entire IP packet. To achieve this, after the AH or ESP fields are added to the IP packet, the entire packet plus security fields is treated as the payload of new outer IP packet with a new outer IP header.
- The entire original, inner, packet travels through a tunnel from one point of an IP network to another; no routers along the way are able to examine the inner IP header. Because the original packet is encapsulated, the new, larger packet may have totally different source and destination addresses, adding to the security.

- Tunnel mode is used when one or both ends of a security association (SA) are a security gateway, such as a firewall or router that implements IPsec.
- In tunnel mode, a number of hosts on networks behind firewalls may engage in secure communications without implementing IPsec. The unprotected packets generated by such hosts are tunneled through external networks by tunnel mode SAs set up by the IPsec software in the firewall or secure router at the boundary of the local network.
- ESP in tunnel mode encrypts and optionally authenticates the entire inner IP packet, including the inner IP header.
- AH in tunnel mode authenticates the entire inner IP packet and selected portions of the outer IP header.

Table 20.1 Tunnel Mode and Transport Mode Functionality

	Transport Mode SA	Tunnel Mode SA
AH	Authenticates IP payload and selected portions of IP header and IPv6 extension headers.	Authenticates entire inner IP packet (inner header plus IP payload) plus selected portions of outer IP header and outer IPv6 extension headers.
ESP	Encrypts IP payload and any IPv6 extension headers following the ESP header.	Encrypts entire inner IP packet.
ESP with Authentication	Encrypts IP payload and any IPv6 extension headers following the ESP header. Authenticates IP payload but not IP header.	Encrypts entire inner IP packet. Authenticates inner IP packet.

-

Authentication header (AH)

Contents

Authentication header (AH)

- .The Authentication Header provides support for data integrity and authentication of IP packets. The data integrity feature ensures that undetected modification to a packet's content in transit is not possible.

- The authentication feature enables an end system or network device to authenticate the user or application and filter traffic accordingly; it also prevents the address spoofing attacks
- Authentication is based on the use of a message authentication code (MAC), protocol hence the two parties must share a secret key.
- The Authentication Header consists of the following fields (Figure 16.3):
  - **Next Header (8 bits):** Identifies the type of header immediately following this header.
  - **Payload Length (8 bits):** Length of Authentication Header in 32-bit words, minus 2. For

- example, the default length of the authentication data field is 96 bits, or three 32-bit words. With a three-word fixed header, there are a total of six words in the header, and the Payload Length field has a value of 4.
- **Reserved (16 bits):** For future use.
- **Security Parameters Index (32 bits):** Identifies a security association.
- **Sequence Number (32 bits):** A monotonically increasing counter value,
- **Authentication Data (variable):** A variable-length field (must be an integral number of 32-bit words) that contains the Integrity Check Value (ICV), or MAC, for this packet,

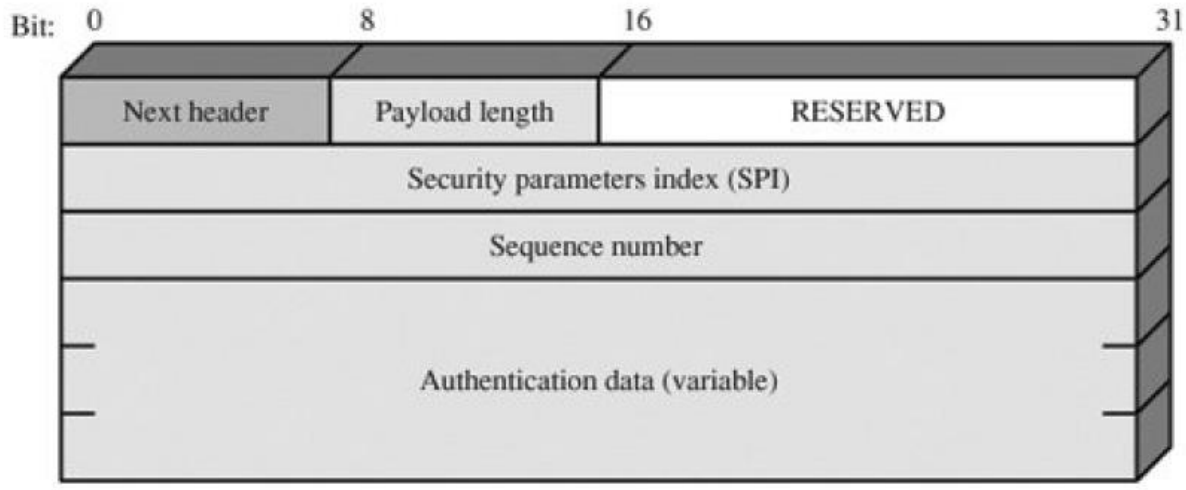


Figure - IPSec Authentication Header

Anti-Replay Service

- A replay attack is one in which an attacker obtains a copy of an authenticated packet and later transmits

- It to the intended destination. The receipt of duplicate, authenticated IP packets may disrupt service in some way or may have some other undesired consequence. The Sequence Number field is designed to thwart such attacks. First, we discuss sequence number generation by the sender, and then we look at how it is processed by the recipient.
- When a new SA is established, the **sender** initializes a sequence number counter to 0. Each time that a packet is sent on this SA, the sender increments the counter and places the value in the Sequence Number field. Thus, the first value to be used is 1. If anti-replay is enabled (the default), the sender must not allow the sequence number to cycle past 232 1 back to zero. Otherwise, there would be multiple valid packets with the same sequence number. If the limit of 232 1 is reached, the sender should terminate this SA and negotiate a new SA with a new key. Because IP is a connectionless, unreliable service, the protocol does not guarantee that packets will be delivered in order and does not guarantee that all packets will be delivered.
- Therefore, the IPSec authentication document dictates that the **receiver** should implement a window of size *W*, with a default of $W = 64$. The right edge of the window represents the highest sequence number, *N*, so far received for a valid packet. For any packet with a sequence number in the range from $N\ W + 1$ to *N* that has been correctly

received (i.e., properly authenticated), the corresponding slot in the window is marked (Figure 16.4). Inbound processing proceeds as follows when a packet is received:

1.If the received packet falls within the window and is new, the MAC is checked. If the packet is authenticated, the corresponding slot in the window is marked.

2. If the received packet is to the right of the window and is new, the MAC is checked. If the packet is authenticated, the window is advanced so that this sequence number is the right edge of the window, and the corresponding slot in the window is marked.

3. If the received packet is to the left of the window, or if authentication fails, the packet is discarded; this is an auditable event.

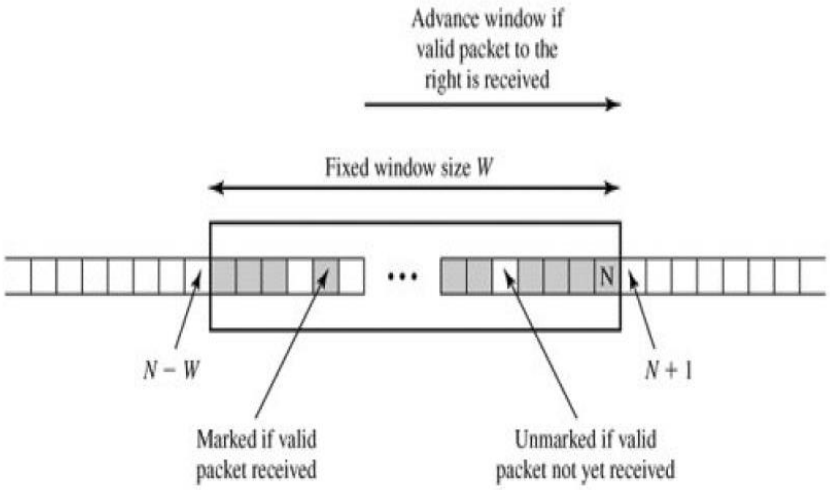


Figure 16.4. Antireplay Mechanism

Integrity Check Value

- The Authentication Data field holds a value referred to as the Integrity Check Value. The ICV is a message authentication code or a truncated version of a code produced by a MAC algorithm. The current specification dictates that a compliant implementation must support
    - HMAC-MD5-96
    - HMAC-SHA-1-96
- Both of these use the HMAC algorithm, the first with the MD5 hash code and the second with the SHA-1 hash code
- In both cases, the full HMAC value is calculated but then truncated by using the first 96 bits, which is the default length for the Authentication Data field.

Transport and Tunnel Modes

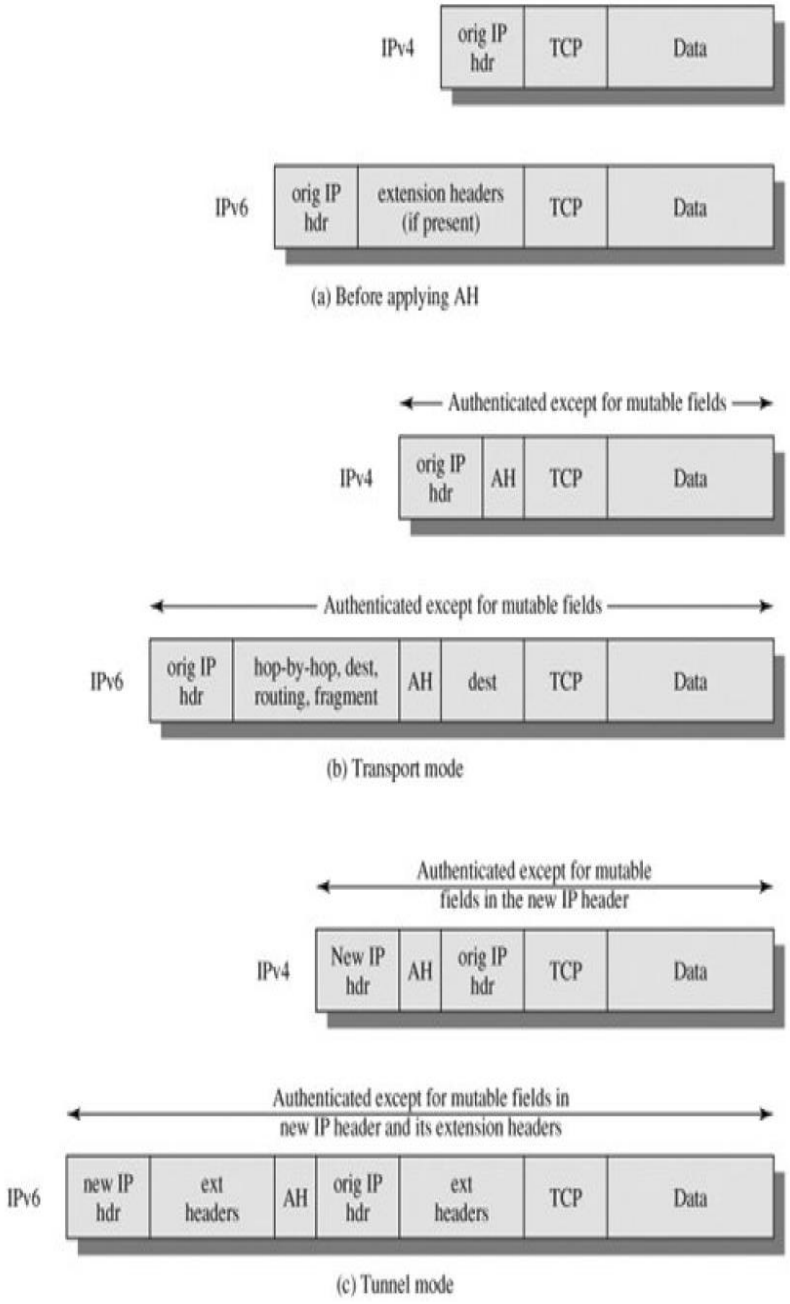
- Figure 16.5 shows two ways in which the IPSec authentication service can be used. In one case, authentication is provided directly between a server and client workstations; the workstation can be either on the same network as the server or on an external network.

- As long as the workstation and the server share a protected secret key, the authentication process is secure. This case uses a transport mode SA.
- In the other case, a remote workstation authenticates itself to the corporate firewall, either for access to the entire internal network or because the requested server does not support the authentication feature. This case uses a tunnel mode SA.
- In this subsection, we look at the scope of authentication provided by AH and the authentication header location for the two modes. The considerations are somewhat different for IPv4 and IPv6. Figure 16.6a shows typical IPv4 and IPv6 packets.
- In this case, the IP payload is a TCP segment; it could also be a data unit for any other protocol that uses IP, such as UDP or ICMP.

.

- For **transport mode AH** using IPv4, the AH is inserted after the original IP header and before the IP payload (e.g., a TCP segment); this is shown in the upper part of Figure 16.6b. Authentication covers the entire packet, excluding mutable fields in the IPv4 header that are set to zero for MAC calculation

Figure 16.6. Scope of AH Authentication



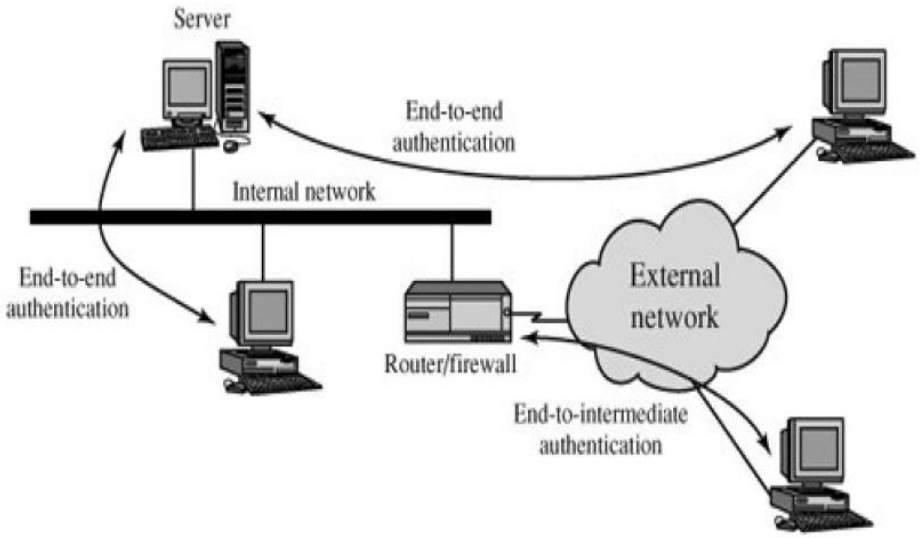


Figure 16.5. End-to-End versus End-to-Intermediate Authentication

- In the context of IPv6, AH is viewed as an end-to-end payload; that is, it is not examined or processedherefore, the AH appears after the IPv6 base header and the hop-by-hop, routing, and fragment extension headers. The destination options extension header could appear before or after the AH header, depending on the semantics desired. Again, authentication covers the entire packet, excluding mutable fields that are set to zero for MAC calculation.
- For **tunnel mode AH**, the entire original IP packet is authenticated, and the AH is inserted between the original IP header and a new outer IP header (Figure 16.6c). The inner IP header carries the ultimate source and destination addresses, while an outer IP header may contain different IP addresses (e.g., addresses of firewalls or other security gateways). With tunnel mode, the entire inner IP packet, including the entire inner IP header is protected by AH.
- The outer IP header (and in the case of IPv6, the outer IP extension headers) is protected except for mutable and unpredictable fields.

Encapsulating Security Payload (ESP)

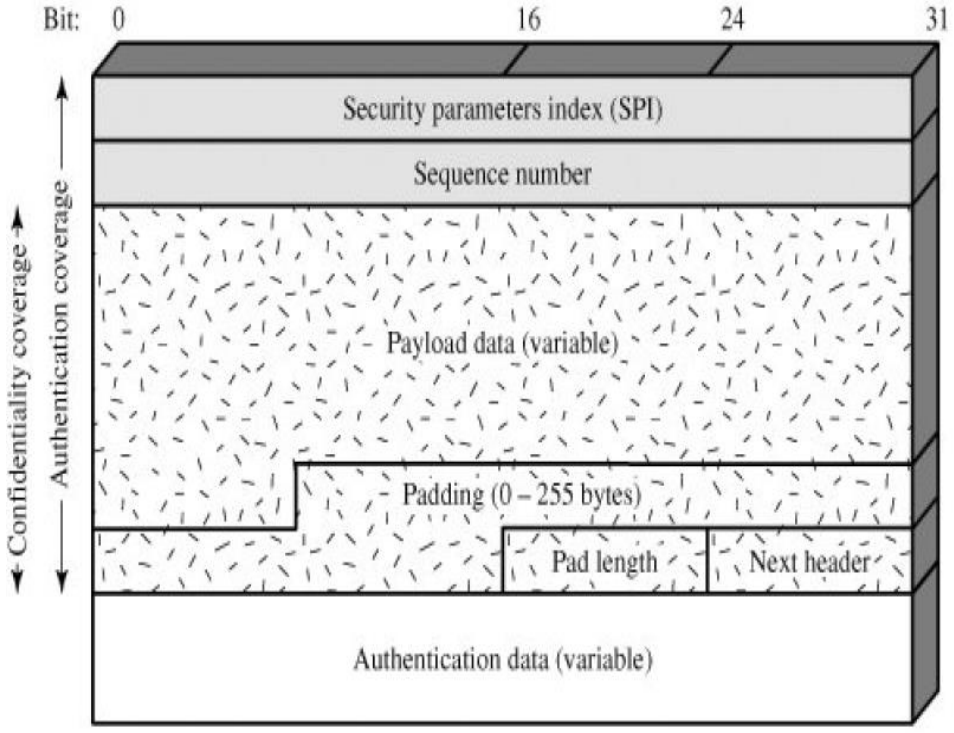
Contents
Encapsulating Security Payload • ESP Format • Encryption and Authentication Algorithms • Padding • Anti-Replay Service • Transport and Tunnel Modes

Encapsulating Security Payload

- The Encapsulating Security Payload provides confidentiality services, including confidentiality of message contents and limited traffic flow confidentiality. As an optional feature, ESP can also provide an authentication service.

ESP Format

- Figure 16.7 shows the format of an ESP packet. It contains the following fields:
  - **Security Parameters Index (32 bits):** Identifies a security association. **Sequence Number (32 bits):** A monotonically increasing counter value; this provides an antireplay function, as discussed for AH.
  - **Payload Data (variable):** This is a transport-level segment (transport mode) or IP packet (tunnel mode) that is protected by encryption.
  - **Padding (0255 bytes):** The purpose of this field is discussed later.
  - **Pad Length (8 bits):** Indicates the number of pad bytes immediately preceding this field.
  - **Next Header (8 bits):** Identifies the type of data contained in the payload data field by identifying the first header in that payload (for example, an extension header in IPv6, or an upper-layer protocol such as TCP).



  - **Authentication Data (variable):** A variable-length field (must be an integral number of 32-bit words) that contains the Integrity Check Value computed over the ESP packet minus the Authentication Data field.

Encryption and Authentication Algorithms

- The Payload Data, Padding, Pad Length, and Next Header fields are encrypted by the ESP service. If the algorithm used to encrypt the payload requires cryptographic

synchronization data, such as an initialization vector (IV), then these data may be carried explicitly at the beginning of the Payload Datafield.

- If included, an IV is usually not encrypted, although it is often referred to as being part of the ciphertext. The current specification dictates that a compliant implementation must support DES in cipher block chaining (CBC) mode).
- A number of other algorithms have been assigned identifiers in the DOI document and could therefore easily be used for encryption; these include
    - Three-key triple DES
    - RC5
    - IDEA
    - Three-key triple IDEA
    - CAST
    - Blowfish
- As with AH, ESP supports the use of a MAC with a default length of 96 bits. Also as with AH, the current specification dictates that a compliant implementation must support HMAC-MD5-96 and HMAC-SHA-1-96.

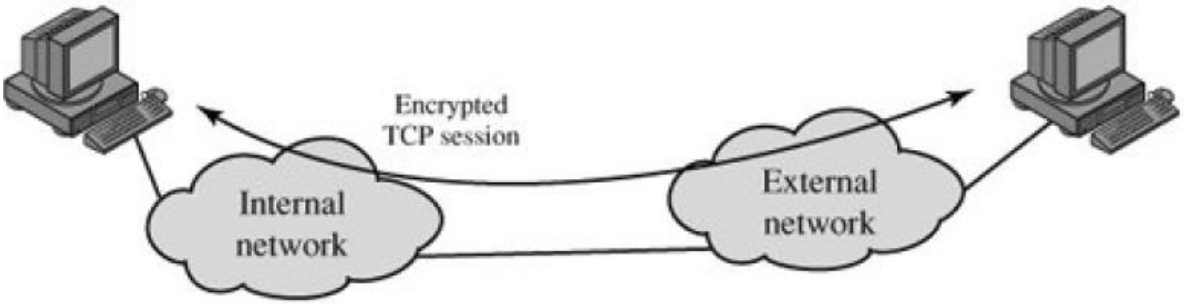
Padding

The Padding field serves several purposes:

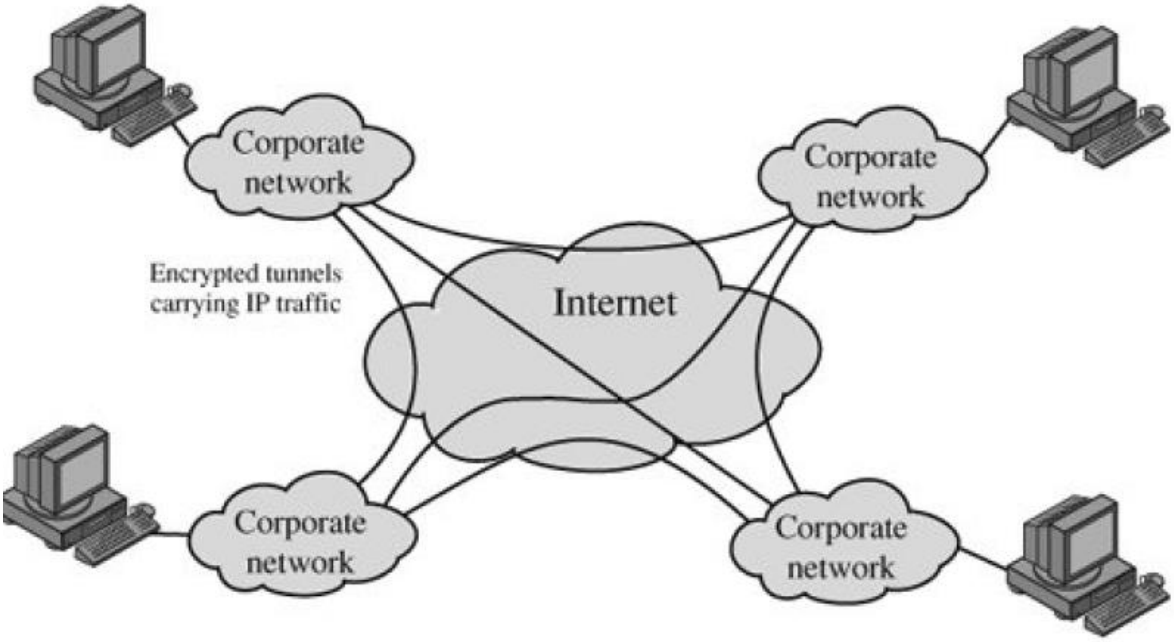
- If an encryption algorithm requires the plaintext to be a multiple of some number of bytes (e.g., the multiple of a single block for a block cipher), the Padding field is used to expand the plaintext (**consisting of the Payload Data, Padding, Pad Length, and Next Header fields**) to the required length.
- The ESP format requires that the Pad Length and Next Header fields be right aligned within a 32- bit word. Equivalently, the ciphertext must be an integer multiple of 32 bits. The Padding field is used to assure this alignment.
- Additional padding may be added to provide partial traffic flow confidentiality by concealing the actual length of the payload.

Transport and Tunnel Modes

- Figure 16.8 shows two ways in which the IPSec ESP service can be used. In the upper part of the figure, encryption (and optionally authentication) is provided directly between two hosts. Figure 16.8b shows how tunnel mode operation can be used to set up a *virtual private network.*
- In this example, an organization has four private networks interconnected across the Internet. Hosts on the internal networks use the Internet for transport of data but do not interact with other Internet-based hosts.
- By terminating the tunnels at the security gateway to each internal network, the configuration allows the hosts to avoid implementing the security capability. The former technique is support by a transport mode SA, while the latter technique uses a tunnel mode SA.



(a) Transport-level security



(b) A virtual private network via tunnel mode

Transport Mode ESP

- Transport mode ESP is used to encrypt and optionally authenticate the data carried by IP (e.g., a TCP segment), as shown in Figure 16.9a. For this mode using IPv4, the ESP header is inserted into the IP packet immediately prior to the transport-layer header (e.g., TCP, UDP, ICMP) and an ESP trailer (Padding, Pad Length, and Next Header fields) is placed after the IP packet; if authentication is selected, the ESP Authentication Data field is added after the ESP trailer.
- The entire transport-level segment plus the ESP trailer are encrypted. Authentication covers all of the ciphertext plus the ESP header.

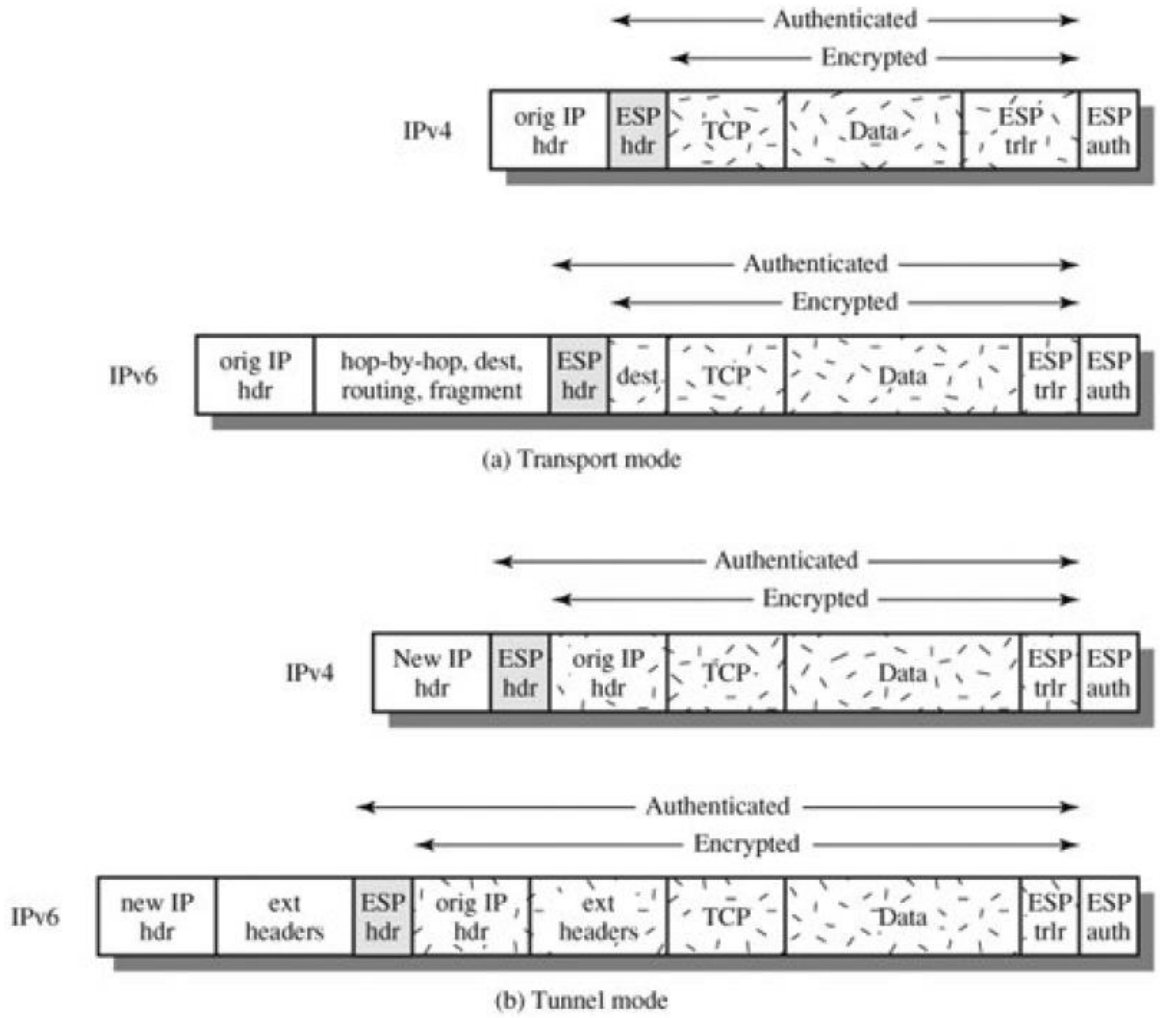


Figure 16.9. Scope of ESP Encryption and Authentication

- In the context of IPv6, ESP is viewed as an end-to-end payload; that is, it is not examined or processed by intermediate routers. Therefore, the ESP header appears after the IPv6 base header and the hop-byhop, routing, and fragment extension headers.
- The destination options extension header could appear before or after the ESP header, depending on the semantics desired. For IPv6, encryption covers the entire transport-level segment plus the ESP trailer plus the destination options extension header if it occurs after the ESP header. Again, authentication covers the ciphertext plus the ESP header.

Tunnel Mode ESP

- Tunnel mode ESP is used to encrypt an entire IP packet (Figure 16.9b). For this mode, the ESP header is prefixed to the packet and then the packet plus the ESP trailer is encrypted. This method can be used to counter traffic analysis. Because the IP header contains the destination address and possibly source routing directives and hopby- hop option information, it is not possible simply to transmit the encrypted IP packet prefixed by the ESP header. Intermediate routers would be unable to process such a packet.
- Therefore, it is necessary to encapsulate the entire block (ESP header plus ciphertext plus Authentication Data, if present) with a new IP header that will contain sufficient information for routing but not for traffic analysis.Whereas the transport mode is suitable for protecting connections between hosts that support the ESP feature, the tunnel mode is useful in a configuration that includes a firewall or other sort of security

gateway that protects a trusted network from external networks. In this latter case, encryption occurs only between an external host and the security gateway or between two security gateways.

- This relieves hosts on the internal network of the processing burden of encryption and simplifies the key distribution task by reducing the number of needed keys. Further, it thwarts traffic analysis based on ultimate destination.

VARIOUS ASPECTS OF IPV6

Contents
IPv6 (Internet Protocol Version 6) o Advantages of IPv6: o IPv6 Addresses o CIDR Notation: o IPv6 Packet Format:

IPv6 (Internet Protocol Version 6)

- IPv6 is the next generation Internet Protocol designed as a successor to the IP version 4.
- IPv6 was designed to enable high-performance, scalable Internet.
- This was achieved by overcoming many of the weaknesses of IPv4 protocol and by adding several new features.

Advantages of IPv6:

1. *Larger address space*
   - IPv6 has 128-bit address space, which is 4 times wider in bits in compared to IPv4's 32-bit address space.
   - So there is a huge increase in the address space.
2. *Better header forma*
   - IPv6 uses a better header format. In its header format the options are separated from the base header.
   - The options are inserted when needed, between the base header and upper layer data.
   - The helps in speeding up the routing process.
3. *New option*
   - New options have been added in IPv6 to increase the functionality.

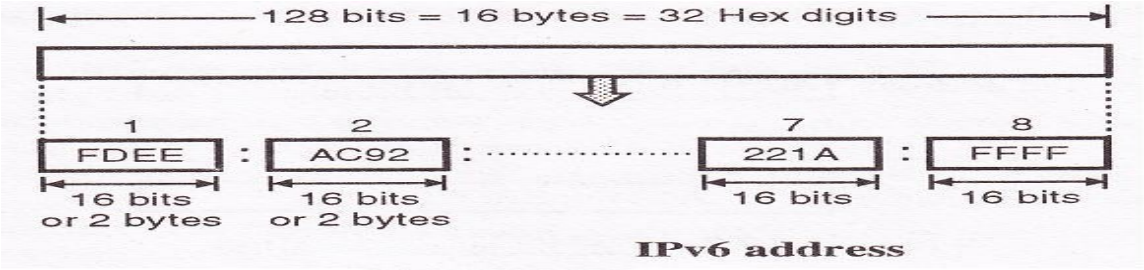
4. *Possibility of extension*
   - IPv6 has been designed in such a way that there is a possibility of extension of protocol if required.
5. *More security*

- IPv6 includes security in the basic specification.
- It includes encryption of packets (ESP: Encapsulated Security Payload) and authentication of the sender of packets (AH: Authentication Header).

6. *Support to resource allocation*
   - To implement better support for real time traffic (such as video conference), IPv6 includes flow label in the specification.
   - With flow label mechanism, routers can recognize to which end-to-end flow the packets belongs.
7. *Plug and play*
   - IPv6 includes plug and play in the standard specification.
   - It therefore must be easier for novice users to connect their machines to the network, it will be done automatically.
8. *Clearer specification and optimization*
   - IPv6 follows good practices of IPv4, and rejects minor flaws/obsolete items of IPv4.

IPv6 Addresses

An IPv6 addresses consists of 16 bytes (octets) i.e. it is 128 bits long as shown in the below figure.



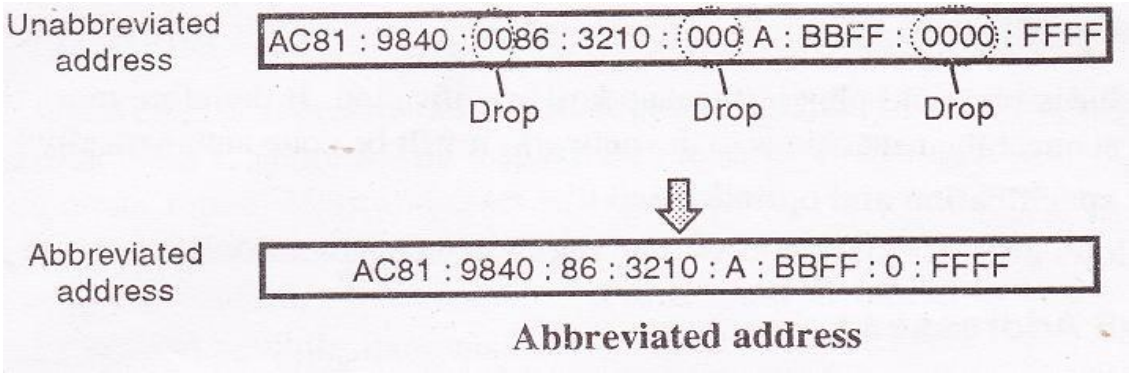
IPv6 address

Hexadecimal colon notation:

- IPv6 uses a special notation called hexadecimal colon notation. In this, the 128 bits are divided into 8 sections; each one is 2 bytes long.
- 2 bytes correspond to 16 bits. So in hexadecimal notation will require four hexadecimal digits.
- Hence the IPv6 address consists of 32 hex digits and every group of 4 digits is separated by a colon as shown in the above figure.
- IPv6 uses 128-bit addresses. Only about 15% of the address space is initially allocated, the remaining 85% being reserved for future use.
- This remainder may be used in the future for expanding the address spaces of existing address types or for totally new uses.

Abbreviation:

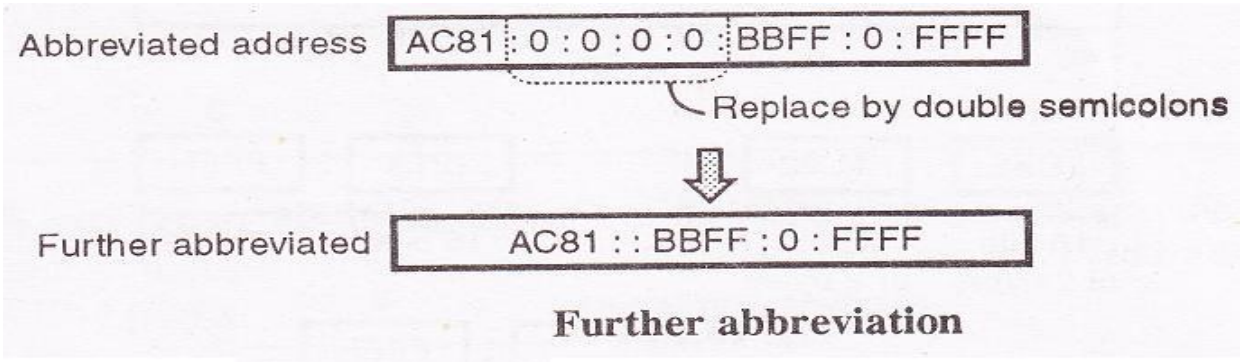
- The IPv6 address, even in hexadecimal format is very long. But in this address there are many of the zero digits in it.
- In such a case, we can abbreviate the address. The leading zeros of a section (four digits between two colons) can be omitted.
- Note that oOnly the leading zeros can be dropped but the trailing zeros can not drop. This is illustrated in the below figure.



Abbreviated address

Further abbreviation:

- Further abbreviations are possible if there is consecutive section consisting of only zeros.
- We can remove the zeros completely and replace them with double semicolon as shown in the below figure.

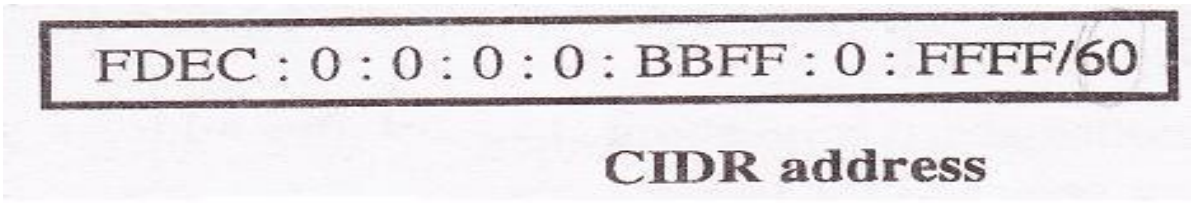


Further abbreviation

- It is important to note abbreviation is allowed only once per address. Also note that if there are two runs of zero sections, then only one of them can be abbreviated.

CIDR Notation:

- IPv6 protocol allows classless addressing and CIDR notation.
- The below figure shows how to define a prefix of 60 bits using CIDR.



CIDR address

Categories of Address:

IPv6 defines three different types of addresses.

Unicast

- A unicast address defines a single computer.
- A packet sent to a unicast address is delivered to that specific computer.

Anycast

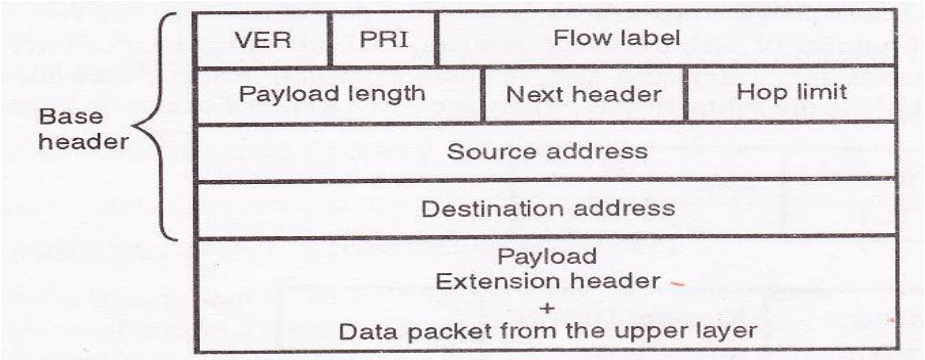
- This is a type of address defines a group of computers with addresses which have the same prefix.
- A packet sent to an anycast address must be delivered to exactly one of the members of the group which is the closest or the most easily accessible.

Multicast Addresses:

- A multicast address defines a group of computers which may or may not share the same prefix and may or may not be connected to the same physical network.
- A packet sent to a multicast address must be delivered to each member of the set.
- There are no broadcast addresses in IPv6, because multicast addresses can perform the same function. The type of address is determined by the leading bits.
- Multicast addresses all start with FF (1111 1111) and all other addresses are unicast addresses.
- Anycast addresses are assigned from the unicast address space and they do not differ syntactically from unicast addresses.
- Anycast addressing is a rather new concept and there is little experience with the widespread use of anycast addresses.
- Therefore, some restrictions apply to anycast addressing in IPv6 until more experience is gained.
- An anycast address may not be used as the Source Address of an IPv6 packet and anycast addresses may not be assigned to hosts but to routers only.

IPv6 Packet Format:

- The IPv6 packet is shown in the below figure. Each packet consists of a base header which is mandatory followed by the payload.
- The payload is made up of two parts
    - Optional extension headers and
    - Data from an upper layer



Format of an IPv6 datagram

- The base header is 40 byte length whereas the extension header and the data from upper layer contain upto 65,535 bytes of information.

Base header

In the base header have eight fields.

These fields are as follows:

1) **Version (VER):** It is a 4 bit field which defines the version of IP such as IPv4 or IPv6. For IPv6 the value of this field is 6.
2) **Priority:** It is a 4 bit field which defines the priority of the packet which is important in connection with the traffic congestion.
3) **Flow label:** It is a 24 bit (3 byte) field which is designed for providing special handling for a particular flow of data.
4) **Payload length:** This is a 2 byte length field which is used to define the total length of the IP datagram excluding the base header.
5) **Next header:** It is an 8 bit field which defines the header which follows the base header in the datagram.
6) **Hop limit:** This is an 8 bit field which has the same purpose as TTL in IPv4.
7) **Source address:** It is a 16 byte (128) Internet address which identifies the original source of datagram.
8) **Destination address:** This is a 16 byte (128) internet address which identifies the final destination of datagram. But this field will contain the address of the next router if source routing is being used.

KEY MANAGEMENT OF IPSEC OR INTERNET KEY EXCHANGE PROTOCOL

Contents
Internet Key Exchange • Key Determination Protocol • Header and Payload Formats

Internet Key Exchange

- The key management portion of IPsec involves the determination and distribution of secret keys. A typical requirement is four keys for communication between two applications: transmit and receive pairs for both integrity and confidentiality.
- The IPsec Architecture document mandates **support for two types of key management:**
  - **Manual:** A system administrator manually configures each system with its own keys and with the keys of other communicating systems. This is practical for small, relatively static environments.
  - **Automated:** An automated system enables the on-demand creation of keys for SAs and facilitates the use of keys in a large distributed system with an evolving configuration.
  - The default automated key management protocol for IPsec is referred to as ISAKMP/Oakley and **consists of the following elements:**

- **Oakley Key Determination Protocol:**
  - ✓ Oakley is a key exchange protocol based on the Diffie-Hellman algorithm but providing added security. Oakley is generic in that it does not dictate specific formats.
- **Internet Security Association and Key Management Protocol (ISAKMP):**
  - ✓ ISAKMP provides a framework for Internet key management and provides the specific protocol support, including formats, for negotiation of security attributes.
  - ✓ ISAKMP by itself does not dictate a specific key exchange algorithm; rather, ISAKMP consists of a set of message types that enable the use of a variety of key exchange algorithms. Oakley is the specific key exchange algorithm mandated for use with the initial version of ISAKMP.
  - ✓ In IKEv2, the terms Oakley and ISAKMP are no longer used, and there are significant differences from the use of Oakley and ISAKMP in IKEv1. Nevertheless, the basic functionality is the same. In this section, we describe the IKEv2 specification.

Key Determination Protocol

- IKE key determination is a refinement of the Diffie-Hellman key exchange algorithm. Recall that Diffie-Hellman involves the following interaction between users A and B.
- There is prior agreement on two global parameters: *q*, a large prime number; and a, a primitive root of *q*. A selects a random integer *XA* as its private key and transmits to B its public key *_A* = a*XA* mod *q*. Similarly, B selects a random integer *XB* as its private key and transmits to A its public key *_B* = a*XB* mod *q*. Each side can now compute the secret session key:

$$K = (\mathrm{Y}_B)^{X_A} \bmod q = (\mathrm{Y}_A)^{X_B} \bmod q = \alpha^{X_A X_B} \bmod q$$

The Diffie-Hellman algorithm has two attractive features:

- Secret keys are created only when needed. There is no need to store secret keys for a long period of time, exposing them to increased vulnerability.
- The exchange requires no pre-existing infrastructure other than an agreement on the global parameters. However, there are a number of weaknesses to Diffie-Hellman, as pointed out in [HUIT98]. • It does not provide any information about the identities of the parties.
- It is subject to a man-in-the-middle attack, in which a third party C impersonates B while communicating with A and impersonates A while communicating with B. Both A and B end up negotiating a key with C, which can then listen to and pass on traffic.
  The man-in-the-middle attack proceeds as
  1. B sends his public key *YB* in a message addressed to A (see Figure 10.2).
  2. The enemy (E) intercepts this message. E saves B's public key and sends a message to A that has B's User ID but E's public key *YE*. This message is sent in such a way that it appears as though it was sent from B's host system. A receives E's message and stores E's public key with B's User ID. Similarly, E sends a message to B with E's public key, purporting to come

3. from A.
4. B computes a secret key *K*1 based on B's private key and *YE*. A computes a secret key *K*2 based on A's private key and *YE*. E computes *K*1 using E's secret key *XE* and *YB* and computers *K*2 using *XE* and *YA*.
5. From now on, E is able to relay messages from A to B and from B to A, appropriately changing their encipherment en route in such a way that neither A nor B will know that they share their communication with E.

- It is computationally intensive. As a result, it is vulnerable to a clogging attack, in which an opponent requests a high number of keys. The victim spends considerable computing resources doing useless modular exponentiation rather than real work.
- IKE key determination is designed to retain the advantages of Diffie-Hellman, while countering its weaknesses.

Features of IKE key determination

- **The IKE key determination algorithm is characterized by five important features**:
  - ✓ .It employs a mechanism known as cookies to thwart clogging attacks.
  - ✓ It enables the two parties to negotiate a *group*; this, in essence, specifies the global parameters of the Diffie-Hellman key exchange.
  - ✓ It uses nonces to ensure against replay attacks.
  - ✓ It enables the exchange of Diffie-Hellman public key values.. It authenticates the Diffie-Hellman exchange to thwart man-in-the-middle attacks.

- **IKE mandates that cookie generation satisfy three basic requirements**:
  - ✓ The cookie must depend on the specific parties. This prevents an attacker from obtaining a cookie using a real IP address and UDP port and then using it to swamp the victim with requests from randomly chosen IP addresses or ports.
  - ✓ It must not be possible for anyone other than the issuing entity to generate cookies that will be accepted by that entity. This implies that the issuing entity will use local secret information in the generation and subsequent verification of a cookie. It must not be possible to deduce this secret information from any particular cookie. The point of this requirement is that the issuing entity need not save copies of its cookies, which are then more vulnerable to discovery, but can verify an incoming cookie acknowledgment when it needs to.
  - ✓ The cookie generation and verification methods must be fast to thwart attacks intended to sabotage processor resources.

- IKE key determination employs **nonces** to ensure against replay attacks. Each nonce is a locally generated pseudorandom number. Nonces appear in responses and are encrypted during certain portions of the exchange to secure their use.

- Three different **authentication** methods can be used with IKE key determination:

  - **Digital signatures:** The exchange is authenticated by signing a mutually obtainable hash; each party encrypts the hash with its private key. The hash is generated over important parameters, such as user IDs and nonces.
  - **Public-key encryption:** The exchange is authenticated by encrypting parameters such as IDs and nonces with the sender's private key.
  - **Symmetric-key encryption:** A key derived by some out-of-band mechanism can be used to authenticate the exchange by symmetric encryption of exchange parameters.

Header and Payload Formats

- IKE defines procedures and packet formats to establish, negotiate, modify, and delete security associations. As part of SA establishment, IKE defines payloads for exchanging key generation and authentication data.
- These payload formats provide a consistent framework independent of the specific key exchange protocol, encryption algorithm, and authentication mechanism.

1. IKE Header Format

- An IKE message consists of an IKE header followed by one or more payloads
- .All of this is carried in a transport protocol. The specification dictates that implementations must support the use of UDP for the transport protocol.

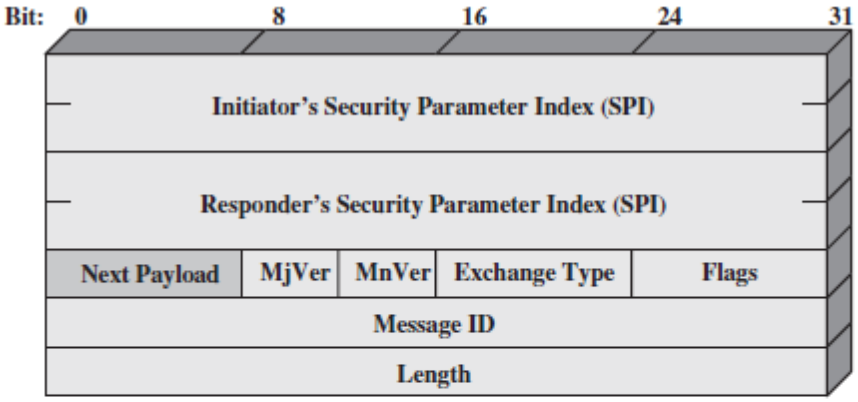


Figure 20.12 IKE Formats

- Figure 20.12a shows the header format for an IKE message. It consists of the following fields.
  - **Initiator SPI (64 bits):** A value chosen by the initiator to identify a unique IKE security association (SA).
  - **Responder SPI (64 bits):** A value chosen by the responder to identify a unique IKE SA.
  - **Next Payload (8 bits):** Indicates the type of the first payload in the message; payloads are discussed in the next subsection.
  - **Major Version (4 bits):** Indicates major version of IKE in use.
  - **Minor Version (4 bits):** Indicates minor version in use.
  - **Exchange Type (8 bits):** Indicates the type of exchange
  - **Flags (8 bits):** Indicates specific options set for this IKE exchange. Three bits are defined so far. The initiator bit indicates whether this packet is sent by the SA initiator. The version bit indicates whether the transmitter is capable of using a higher major version number than the one currently indicated. The response bit indicates whether this is a response to a message containing the same message ID.
  - **Message ID (32 bits):** Used to control retransmission of lost packets and matching of requests and responses.

- **Length (32 bits):** Length of total message (header plus all payloads) in octets.

2. IKE Payload Types

- All IKE payloads begin with the same generic payload header shown in Figure 20.12b. The Next Payload field has a value of 0 if this is the last payload in the message; otherwise its value is the type of the next payload. The Payload Length field indicates the length in octets of this payload, including the generic payload header.
- The critical bit is 0 if the sender wants the recipient to skip this payload if it does not understand the payload type code in the Next Payload field of the previous payload. It is set to 1 if the sender wants the recipient to reject this entire message if it does not understand the payload type.

- These elements are formatted as substructures within the payload as follows.

  - **Proposal:** This substructure includes a proposal number, a protocol ID (AH, ESP, or IKE), an indicator of the number of transforms, and then a transform substructure. If more than one protocol is to be included in a proposal, then there is a subsequent proposal substructure with the same proposal number.
  - **Transform:** Different protocols support different transform types. The transforms are used primarily to define cryptographic algorithms to be used with a particular protocol.
  - **Attribute:** Each transform may include attributes that modify or complete the specification of the transform. An example is key length.

Table 20.3 IKE Payload Types

Type	Parameters
Security Association	Proposals
Key Exchange	DH Group #, Key Exchange Data
Identification	ID Type, ID Data
Certificate	Cert Encoding, Certificate Data
Certificate Request	Cert Encoding, Certification Authority
Authentication	Auth Method, Authentication Data
Nonce	Nonce Data
Notify	Protocol-ID, SPI Size, Notify Message Type, SPI, Notification Data
Delete	Protocol-ID, SPI Size, # of SPIs, SPI (one or more)
Vendor ID	Vendor ID
Traffic Selector	Number of TSs, Traffic Selectors
Encrypted	IV, Encrypted IKE payloads, Padding, Pad Length, ICV
Configuration	CFG Type, Configuration Attributes
Extensible Authentication Protocol	EAP Message

  -

- The **Key Exchange payload** can be used for a variety of key exchange techniques, including Oakley, Diffie-Hellman, and the RSA-based key exchange used by PGP. The Key Exchange data field contains the data required to generate a session key and is dependent on the key exchange algorithm used.

- The **Identification payload** is used to determine the identity of communicating peers and may be used for determining authenticity of information. Typically the ID Data field will contain an IPv4 or IPv6 address.
- The **Certificate payload** transfers a public-key certificate. The Certificate Encoding field indicates the type of certificate or certificate-related information, which may include the following:
    - PKCS #7 wrapped X.509 certificate
    - PGP certificate
    - DNS signed key
    - X.509 certificate—signature
    - X.509 certificate—key exchange
    - Kerberos tokens
    - Certificate Revocation List (CRL)
    - Authority Revocation List (ARL)
    - SPKI certificate
- At any point in an IKE exchange, the sender may include a **Certificate Request** payload to request the certificate of the other communicating entity. The payload may list more than one certificate type that is acceptable and more than one certificate authority that is acceptable.
- The **Authentication** payload contains data used for message authentication purposes. The authentication method types so far defined are RSA digital signature, shared-key message integrity code, and DSS digital signature.
- The **Nonce** payload contains random data used to guarantee liveness during an exchange and to protect against replay attacks.
- The **Notify** payload contains either error or status information associated with this SA or this SA negotiation. The following table lists the IKE notify messages.

Error Messages	Status Messages
Unsupported Critical	Initial Contact
Payload	Set Window Size
Invalid IKE SPI	Additional TS Possible
Invalid Major Version	IPCOMP Supported
Invalid Syntax	NAT Detection Source IP
Invalid Payload Type	NAT Detection Destination IP
Invalid Message ID	Cookie
Invalid SPI	Use Transport Mode

- The **Delete** payload indicates one or more SAs that the sender has deleted from its database and that therefore are no longer valid.
- The **Vendor ID** payload contains a vendor-defined constant. The constant is used by vendors to identify and recognize remote instances of their implementations. This mechanism allows a vendor to experiment with new features while maintaining backward compatibility.
- The **Traffic Selector** payload allows peers to identify packet flows for processing by IPsec services.

- The **Encrypted** payload contains other payloads in encrypted form. The encrypted payload format is similar to that of ESP. It may include an IV if the encryption algorithm requires it and an ICV if authentication is selected.
- The **Configuration** payload is used to exchange configuration information between IKE peers.
- The **Extensible Authentication Protocol (EAP)** payload allows IKE SAs to be authenticated using EAP,

5.3. WEB SECURITY

Contents
Web Security
Web Security Considerations
Web Security Threats
Web Traffic Security Approaches
Secure Socket Layer and Transport Layer Security
SSL Architecture
SSL Record Protocol
Change Cipher Spec Protocol
Alert Protocol
Handshake Protocol
Cryptographic Computations
Transport Layer Security
Secure Electronic Transaction
SET Overview
Dual Signature
Payment Processing

Web Security Considerations

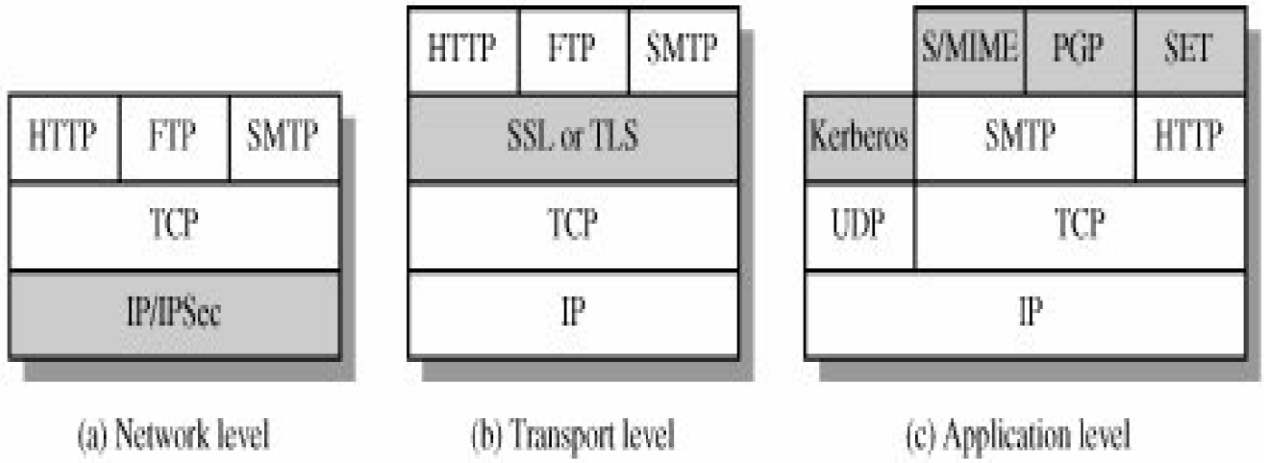
- The World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets.
- As such, the security tools and approaches discussed so far in this book are relevant to the issue of Web security.
- But, as pointed out in [GARF97], the Web presents new challenges not generally appreciated in the context of computer and network security:
- The Internet is two way. Unlike traditional publishing environments, even electronic publishing systems involving tele text, voice response, or fax-back, the Web is vulnerable to attacks on the Web servers over the Internet.
- The Web is increasingly serving as a highly visible outlet for corporate and product information and as the platform for business transactions.
- Reputations can be damaged and money can be lost if the Web servers are subverted.

Web Security Threats

- Table 17.1 provides a summary of the types of security threats faced in using the Web. One way to group these threats is in terms of passive and active attacks.
- Passive attacks include eavesdropping on network traffic between browser and server and gaining access to information on a Web site that is supposed to be restricted.
- Active attacks include impersonating another user, altering messages in transit between client and server, and altering information on a Web site.

Web Traffic Security Approaches

- A number of approaches to providing Web security are possible. The various approaches that have been considered are similar in the services they provide and, to some extent, in the mechanisms that they use, but they differ with respect to their scope of applicability and their relative location within the TCP/IP protocol stack.
- The advantage of using IPSec is that it is transparent to end users and applications and provides a general-purpose solution. Further, IPSec includes a filtering capability so that only selected traffic need incur the overhead of IPSec processing.



	Threats	Consequences	Countermeasures
Integrity	•Modification of user data •Trojan horse browser •Modification of memory •Modification of message traffic in transit	•Loss of information •Compromise of machine •Vulnerabilty to all other threats	Cryptographic checksums
Confidentiality	•Eavesdropping on the Net •Theft of info from server •Theft of data from client •Info about network configuration •Info about which client talks to server	•Loss of information •Loss of privacy	Encryption, web proxies
Denial of Service	•Killing of user threads •Flooding machine with bogus requests •Filling up disk or memory •Isolating machine by DNS attacks	•Disruptive •Annoying •Prevent user from getting work done	Difficult to prevent
Authentication	•Impersonation of legitimate users •Data forgery	•Misrepresentation of user •Belief that false information is valid	Cryptographic techniques

Table 17.1. A Comparison of Threats on the Web

SECURE SOCKET LAYER SECURITY(SSL)

Contents
• **Secure Socket Layer Security** ○ SSL Architecture ○ SSL Record Protocol ○ Change Cipher Spec Protocol ○ Alert Protocol ○ Handshake Protocol ○ Cryptographic Computations • **Transport Layer Security**

Secure Socket Layer

- SSL protocol is an internet protocol for secure exchange of information between a web browser and web server
- SSL is Designed to make use of TCP to provide a reliable end to end secure service
- SSL provides security services between TCP and applications that use TCP. The SSL protocol is an internet protocol for secure exchange of information between a web browser and web server
- Subsequently, when a consensus was reached to submit the protocol for Internet standardization, the TLS working group was formed within IETF to develop a common standard. This first published version of TLS can be viewed as essentially an SSLv3.1 and is very close to and backward compatible with SSLv3.
- The bulk of this section is devoted to a discussion of SSLv3. At the end of the section, the principal differences between SSLv3 and TLS are described.

SSL Architecture

- SSL is designed to make use of TCP to provide a reliable end-to-end secure service. SSL is not a single protocol but rather two layers of protocols, as illustrated in Figure 17.2.
- The SSL Record Protocol provides basic security services to various higher-layer protocols. In particular, the Hypertext Transfer Protocol (HTTP), which provides the transfer service for Web client/server interaction, can operate on top of SSL. Three higher-layer protocols are defined as part of SSL:
- The Handshake Protocol, The Change Cipher Spec Protocol, and the Alert Protocol.

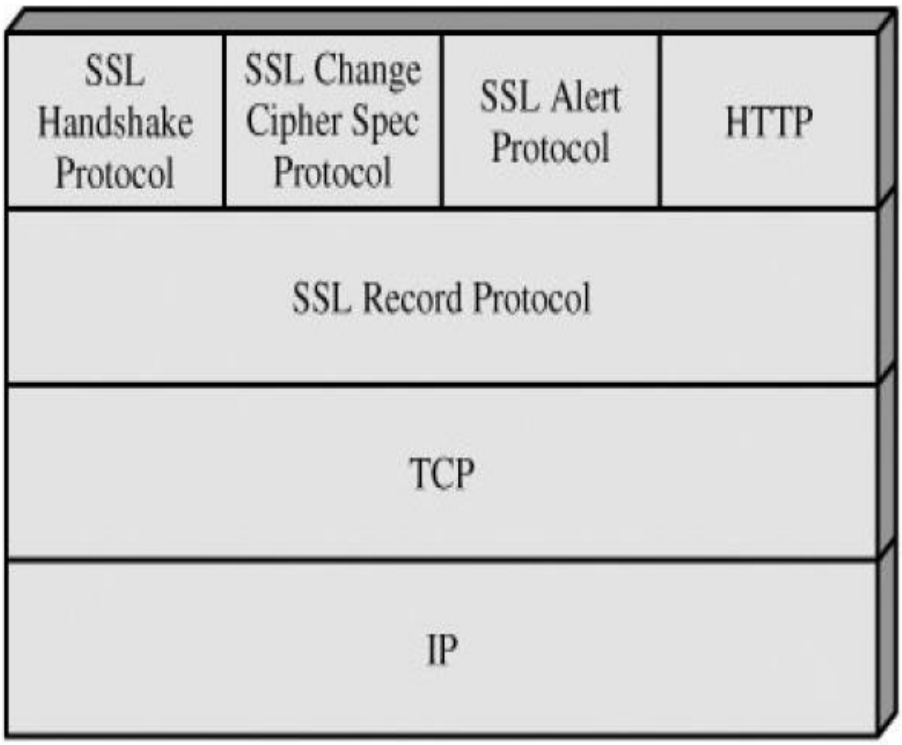


Figure 17.2. SSL Protocol Stack

- Two important SSL concepts are the SSL session and the SSL connection, which are defined in the specification as follows:
    - **Connection:** A connection is a transport (in the OSI layering model definition) that provides a suitable type of service. For SSL, such connections are peer-to-peer relationships. The connections are transient. Every connection is associated with one session.
    - **Session:** An SSL session is an association between a client and a server. Sessions are created by the Handshake Protocol. Sessions define a set of cryptographic security parameters, which can be shared among multiple connections.
- Sessions are used to avoid the expensive negotiation of new security parameters for each connection. Between any pair of parties (applications such as HTTP on client and server), there may be multiple secure connections.
- In theory, there may also be multiple simultaneous sessions between parties, but this feature is not used in practice.
- There are actually a number of states associated with each session.
- Once a session is established, there is a current operating state for both read and write (i.e., receive and send). In addition, during the Handshake Protocol, pending read and write states are created.

- Upon successful conclusion of the Handshake Protocol, the pending states become the current states. A session state is defined by the following parameters (definitions taken from the SSL specification):
    - **Session identifier:** An arbitrary byte sequence chosen by the server to identify an active or resumable session state.
    - **Peer certificate:** An X509.v3 certificate of the peer. This element of the state may be null.
    - **Compression method:** The algorithm used to compress data prior to encryption.
    - **Cipher spec:** Specifies the bulk data encryption algorithm (such as null, AES, etc.) and a hash algorithm (such as MD5 or SHA-1) used for MAC calculation. It also defines cryptographic attributes such as the hash size.
    - **Master secret:** 48-byte secret shared between the client and server.
    - **Is resumable:** A flag indicating whether the session can be used to initiate new connections.
- A connection state is defined by the following parameters:
    - **Server and client random:** Byte sequences that are chosen by the server and client for each connection.
    - **Server write MAC secret:** The secret key used in MAC operations on data sent by the server.
    - **Client write MAC secret:** The secret key used in MAC operations on data sent by the client.
    - **Server write key:** The conventional encryption key for data encrypted by the server and decrypted by the client.
    - **Client write key:** The conventional encryption key for data encrypted by the client and decrypted by the server.
    - **Initialization vectors:** When a block cipher in CBC mode is used, an initialization vector (IV) is maintained for each key. This field is first initialized by the SSL Handshake Protocol. Thereafter the final ciphertext block from each record is preserved for use as the IV with the following record.
    - **Sequence numbers:** Each party maintains separate sequence numbers for transmitted and received messages for each connection. When a party sends or receives a change cipher spec message, the appropriate sequence number is set to zero. Sequence numbers may not exceed 264

- **Confidentiality:** The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads.
- **Message Integrity:** The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

- **SSL components:**
  - SSL Record Protocol
  - SSL Handshake Protocol
  - SSL Alert Protocol
  - SSL Change Cipher Spec Protocol

SSL Record Protocol

- The SSL Record Protocol provides two services for SSL connections:
  - **Confidentiality:** The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads.
  - **Message Integrity:** The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

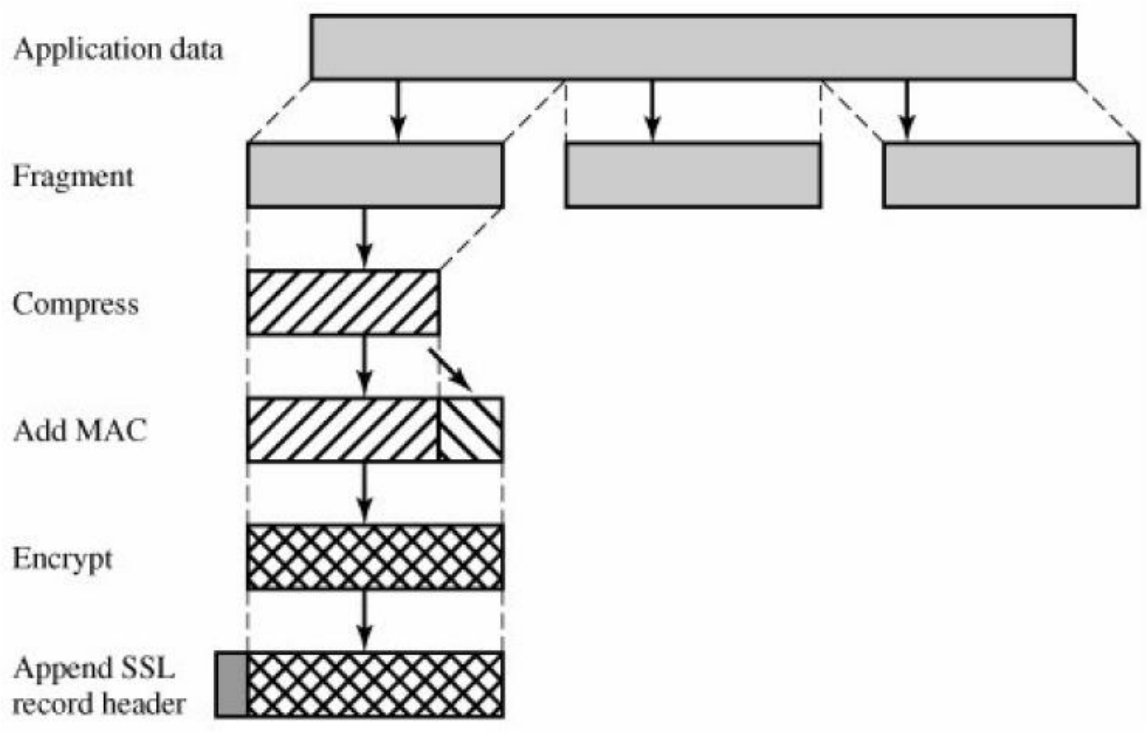


Figure 17.3. SSL Record Protocol Operation

- Figure 17.3 indicates the overall operation of the SSL Record Protocol. The Record Protocol takes an application message to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, adds a header, and transmits the resulting unit in a TCP segment.

- Received data are decrypted, verified, decompressed, and reassembled and then delivered to higher-level users.
- The first step is **fragmentation**. Each upper-layer message is fragmented into blocks of 214 bytes (16384 bytes) or less.
- Next, **compression** is optionally applied. Compression must be lossless and may not increase the content length by more than 1024 bytes. In SSLv3 (as well as the current version of TLS), no compression algorithm is specified, so the default compression algorithm is null.

- The next step in processing is to compute a **message authentication code** over the compressed data. For this purpose, a shared secret key is used. The calculation is defined as

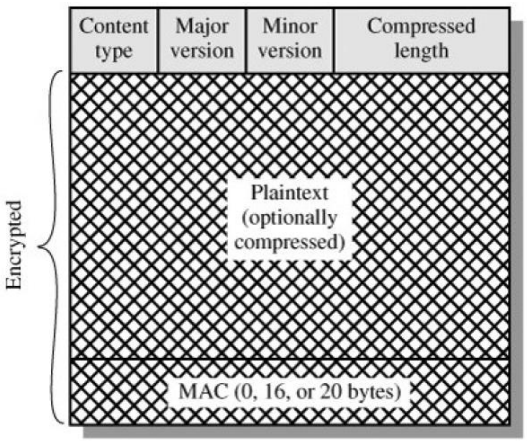
hash(MAC_write_secret || pad_2 ||
hash(MAC_write_secret || pad_1 || seq_num ||
SSLCompressed.type ||
SSLCompressed.length || SSLCompressed.fragment))

Where

|| = concatenation
MAC_write_secret = shared secret key
hash = cryptographic hash algorithm; either MD5 or SHA-1
pad_1 = the byte 0x36 (0011 0110) repeated 48 times (384 bits) for MD5 and 40 times (320 bits) for SHA-1
pad_2 = the byte 0x5C (0101 1100) repeated 48 times for MD5 and 40 times for
SHA-1
seq_num = the sequence number for this message
SSLCompressed.type = the higher-level protocol used to process this fragment
SSLCompressed.length = the length of the compressed fragment
SSLCompressed.fragment = the compressed fragment (if compression is not used, the plaintext Fragment

- Next, the compressed message plus the MAC are **encrypted** using symmetric encryption. Encryption may not increase the content length by more than 1024 bytes, so that the total length may not exceed 214 + 2048.
- The final step of SSL Record Protocol processing is to prepend a header, consisting of the following fields:
    - **Content Type (8 bits):** The higher layer protocol used to process the enclosed fragment.
    - **Major Version (8 bits):** Indicates major version of SSL in use. For SSLv3, the value is 3.
    - **Minor Version (8 bits):** Indicates minor version in use. For SSLv3, the value is 0.
    - **Compressed Length (16 bits):** The length in bytes of the plaintext fragment (or compressed
    - fragment if compression is used). The maximum value is 214 + 2048.

Figure 17.4 illustrates the SSL record format.



Change Cipher Spec Protocol

- The Change Cipher Spec Protocol is one of the three SSL-specific protocols that use the SSL Record Protocol, and it is the simplest. This protocol consists of a single message (Figure 17.5a), which consists of a single byte with the value 1.
- The sole purpose of this message is to cause the pending state to be copied into the current state, which updates the cipher suite to be used on this connection.

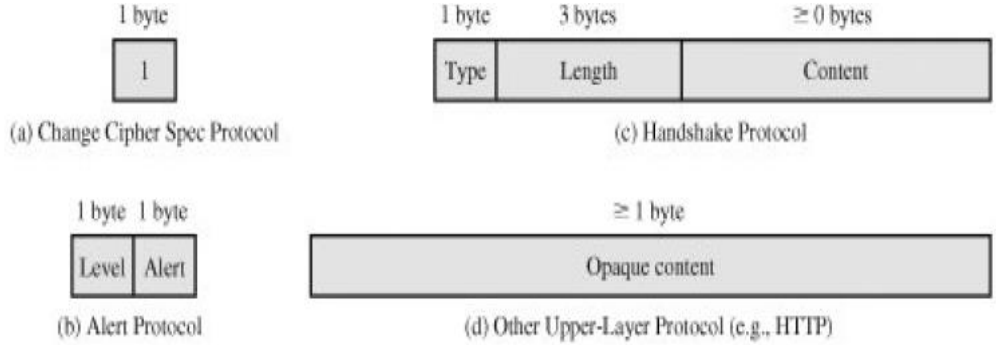


Figure 17.5. SSL Record Protocol Payload

Alert Protocol

- The Alert Protocol is used to convey SSL-related alerts to the peer entity. As with other applications that use SSL, alert messages are compressed and encrypted, as specified by the current state.
- Each message in this protocol consists of two bytes (Figure 17.5b). The first byte takes the value warning(1) or fatal(2) to convey the severity of the message.
- If the level is fatal, SSL immediately terminates the connection. Other connections on the same session may continue, but no new connections on this session may be established. The second byte contains a code that indicates the specific alert.
- First, we list those alerts that are always fatal (definitions from the SSL specification):
    - **Unexpected message:** An inappropriate message was received.

- **bad_record_mac:** An incorrect MAC was received.
- **Decompression failure:** The decompression function received improper input (e.g., unable to decompress or decompress to greater than maximum allowable length).
- **handshake failure:** Sender was unable to negotiate an acceptable set of security parameters given the options available.
- **illegal parameter:** A field in a handshake message was out of range or inconsistent with other fields.

- **The remainder of the alerts is the following:**
  - **close notify:** Notifies the recipient that the sender will not send any more messages on this connection. Each party is required to send a close_notify alert before closing the write side of a connection.
  - **no certificate:** May be sent in response to a certificate request if no appropriate certificate is available.
  - **bad certificate:** A received certificate was corrupt (e.g., contained a signature that did not verify).
  - **unsupported certificate:** The type of the received certificate is not supported.
  - **certificate revoked:** A certificate has been revoked by its signer.
  - **certificate expired:** A certificate has expired.
  - **certificate unknown:** Some other unspecified issue arose in processing the certificate, rendering it unacceptable.

Handshake Protocol

- The most complex part of SSL is the Handshake Protocol. This protocol allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record.
- The Handshake Protocol is used before any application data is transmitted.
- Each message has three fields:
  - **Type (1 byte):** Indicates one of 10 messages. Table 17.2 lists the defined message types.
  - **Length (3 bytes):** The length of the message in bytes.
  - **Content (0 bytes):** The parameters associated with this message; these are listed in Table17.2.

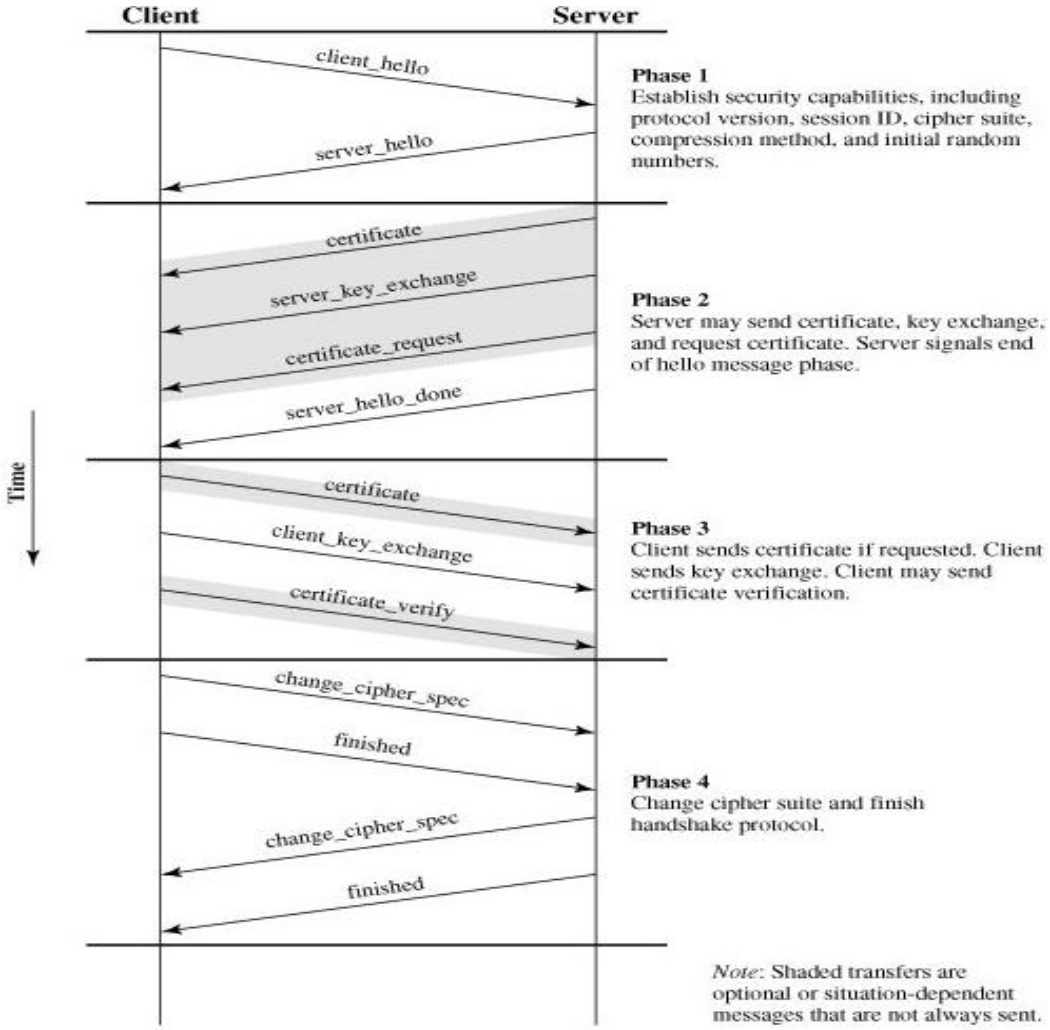


Figure 17.6. Handshake Protocol Action

Cryptographic Computations

- Two further items are of interest: the creation of a shared master secret by means of the key exchange, and the generation of cryptographic parameters from the master secret.

Master Secret Creation

- The shared master secret is a one-time 48-byte value (384 bits) generated for this session by means of secure key exchange. The creation is in two stages. First, a pre_master_secret is exchanged.
- Second, the master_secret is calculated by both parties. For pre_master_secret exchange, there are two possibilities:
    - **RSA:** A 48-byte pre_master_secret is generated by the client, encrypted with the server's public RSA key, and sent to the server. The server decrypts the ciphertext using its private key to recover the pre_master_secret.
    - **Diffie-Hellman:** Both client and server generate a Diffie-Hellman public key. After these are exchanged, each side performs the Diffie-Hellman calculation to create the shared pre_master_secret.

      Both sides now compute the master_secret as follows:
      master_secret = MD5(pre_master_secret || SHA('A' ||

pre_master_secret ||ClientHello.random ||
ServerHello.random)) ||
MD5(pre_master_secret || SHA('BB' ||
pre_master_secret || ClientHello.random ||
ServerHello.random)) ||
MD5(pre_master_secret || SHA('CCC' ||
pre_master_secret || ClientHello.random ||
ServerHello.random))

- where ClientHello.random and ServerHello.random are the two nonce values exchanged in the initial hello messages

Transport Layer Security(TLS)

- TLS is an IETF standardization initiative whose goal is to produce an Internet standard version of SSL.
- TLS is defined as a Proposed Internet Standard in RFC 2246. RFC 2246 is very similar to SSLv3. In this section, we highlight the differences.

Version Number

- The TLS Record Format is the same as that of the SSL Record Format (Figure 17.4), and the fields in the header have the same meanings. The one difference is in version values. For the current version of TLS, the Major Version is 3 and the Minor Version is 1.

Message Authentication Code

- There are two differences between the SSLv3 and TLS MAC schemes: the actual algorithm and the scope of the MAC calculation. TLS makes use of the HMAC algorithm defined in RFC 2104.
- HMAC is defined as follows:

$$\text{HMAC}_K(M) = H[(K^+ \oplus \text{opad}) \| H[(K^+ \oplus \text{ipad}) \| M]]$$

Where

H = embedded hash function (for TLS, either MD5 or SHA-1)
M = message input to HMAC
K+ = secret key padded with zeros on the left so that the result is equal to the block length of the hash code(for MD5 and SHA-1, block length = 512 bits)
ipad = 00110110 (36 in hexadecimal) repeated 64 times (512 bits)
opad = 01011100 (5C in hexadecimal) repeated 64 times (512 bits)

- SSLv3 uses the same algorithm, except that the padding bytes are concatenated with the secret key rather than being XORed with the secret key padded to the block length. The level of security should be about the same in both cases.
- For TLS, the MAC calculation encompasses the fields indicated in the following expression:

HMAC_hash(MAC_write_secret, seq_num || TLSCompressed.type ||
TLSCompressed.version || TLSCompressed.length ||

TLSCompressed.fragment)

- The MAC calculation covers all of the fields covered by the SSLv3 calculation, plus the field
- TLSCompressed.version, which is the version of the protocol being employed.

Pseudorandom Function

- TLS makes use of a pseudorandom function referred to as PRF to expand secrets into blocks of data for purposes of key generation or validation.
- The objective is to make use of a relatively small shared secret value but to generate longer blocks of data in a way that is secure from the kinds of attacks made on hash functions and MACs.The PRF is based on the data expansion function (Figure 16.7) given as

P_hash(secret, seed)= HMAC_hash(secret,A(1) || seed) ||
HMAC_hash(secret, A(2) || seed) ||
HMAC_hash(secret, A(3) || seed) || . . .

where

A() is defined as

A(0) = seed

A(i) = HMAC_hash(secret,A(i – 1))

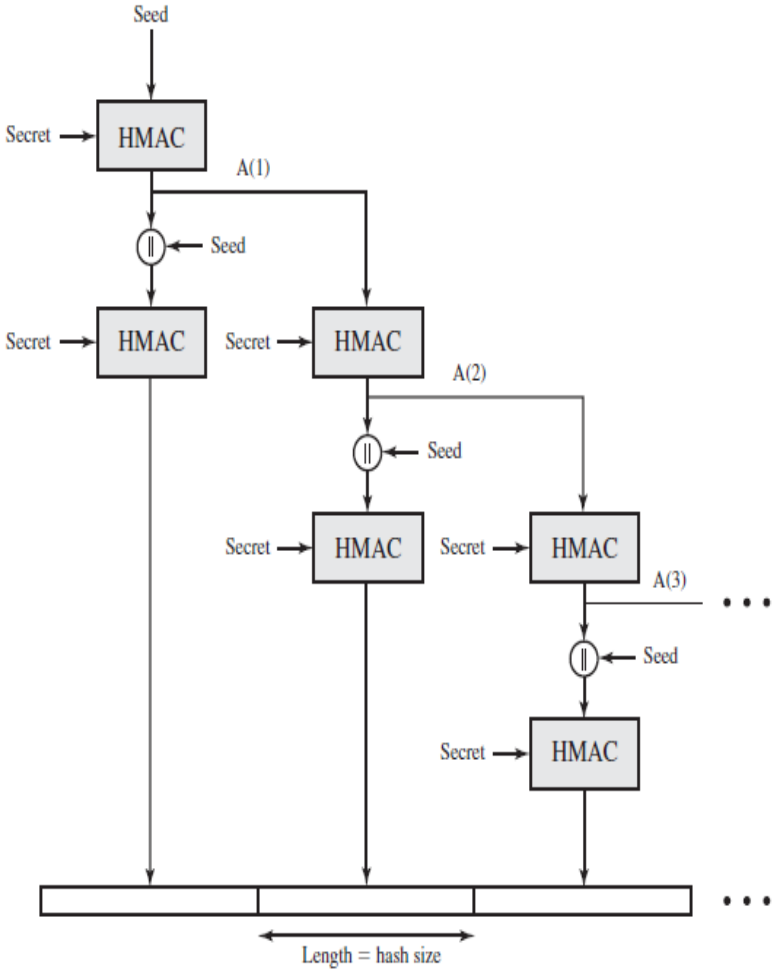


Figure 16.7 TLS Function `P_hash(secret, seed)`

Alert Codes

- TLS supports all of the alert codes defined in SSLv3 with the exception of no_certificate.A number of additional codes are defined in TLS; of these, the following are always fatal.
  - **record_overflow:** A TLS record was received with a payload (ciphertext) whose length exceeds bytes, or the ciphertext decrypted to a length of greater than bytes.
  - **unknown_ca:** A valid certificate chain or partial chain was received, but the certificate was not accepted because the CA certificate could not be located or could not be matched with a known, trusted CA.
  - **access_denied:** A valid certificate was received, but when access control was applied, the sender decided not to proceed with the negotiation.
  - **decode_error:** A message could not be decoded, because either a field was out of its specified range or the length of the message was incorrect.
  - **protocol_version:** The protocol version the client attempted to negotiate is recognized but not supported.
  - **insufficient_security:** Returned instead of handshake failure when a negotiation has failed specifically because the server requires ciphers more secure than those supported by the client.
  - **unsupported_extension:** Sent by clients that receive an extended server hello containing an extension not in the corresponding client hello.

- **internal_error:** An internal error unrelated to the peer or the correctness of the protocol makes it impossible to continue.
- **decrypt_error:** A handshake cryptographic operation failed, including being unable to verify a signature, decrypt a key exchange, or validate a finished message.

- **The remaining alerts include the following**.
  - **user_canceled:** This handshake is being canceled for some reason unrelated to a protocol failure.
  - n**o_renegotiation:** Sent by a client in response to a hello request or by the server in response to a client hello after initial handshaking. Either of these messages would normally result in renegotiation, but this alert indicates that the sender is not able to renegotiate.This message is always a warning.

Cipher Suites

- There are several small differences between the cipher suites available under SSLv3 and under TLS:
  - K**ey Exchange:** TLS supports all of the key exchange techniques of SSLv3 with the exception of Fortezza.
  - **Symmetric Encryption Algorithms:** TLS includes all of the symmetric encryption algorithms found in SSLv3, with the exception of Fortezza

5.4. SYSTEM SECURITY

5.4.1. INTRUDERS

Contents
• **Intruder** ✓ **Masquerader:** ✓ **Misfeasor:** ✓ **Clandestine user:** • **Intruder Behavior Patterns** ✓ **Hackers** ✓ **Criminals** ✓ **Insider Attacks** • **Intrusion Techniques** ✓ **One-way function:** ✓ **Access control:**

Intruder

One of the two most publicized threats to security is the intruder, often referred to as a hacker or cracker.

- **The identified three classes of intruders:**

  • **Masquerader:** An individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account

• **Misfeasor:** A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges
• **Clandestine user:** An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection.

- The masquerader is likely to be an outsider; the misfeasor generally is an insider; and the clandestine user can be either an outsider or an insider.

The following are examples of intrusion:

• Performing a remote root compromise of an e-mail server

• Defacing a Web server

• Guessing and cracking passwords

• Copying a database containing credit card numbers

• Viewing sensitive data, including payroll records and medical information, without authorization

• Running a packet sniffer on a workstation to capture usernames and passwords

• Using a permission error on an anonymous FTP server to distribute pirated software and music files

• Dialing into an unsecured modem and gaining internal network access

• Posing as an executive, calling the help desk, resetting the executive's e-mail password, and learning the new password

• Using an unattended, logged-in workstation without permission.

Intruder Behavior Patterns:

The three broad examples of intruder behavior patterns are

- **Hackers** Traditionally, those who hack into computers do so for the thrill of it or for status. The hacking community is a strong meritocracy in which status is determined by level of competence.

(a) Hacker

1. Select the target using IP lookup tools such as NSLookup, Dig, and others.
2. Map network for accessible services using tools such as NMAP.
3. Identify potentially vulnerable services (in this case, pcAnywhere).
4. Brute force (guess) pcAnywhere password.
5. Install remote administration tool called DameWare.
6. Wait for administrator to log on and capture his password.
7. Use that password to access remainder of network.

- **Criminals** Organized groups of hackers have become a widespread and common threat to Internet-based systems. These groups can be in the employ of a corporation or government but often are loosely affiliated gangs of hackers.

(b) Criminal Enterprise

1. Act quickly and precisely to make their activities harder to detect.
2. Exploit perimeter through vulnerable ports.
3. Use Trojan horses (hidden software) to leave back doors for reentry.
4. Use sniffers to capture passwords.
5. Do not stick around until noticed.
6. Make few or no mistakes.

- **Insider Attacks** Insider attacks are among the most difficult to detect and prevent. Employees already have access and knowledge about the structure and content of corporate databases. Insider attacks can be motivated by revenge or simply a feeling of entitlement.

(c) Internal Threat

1. Create network accounts for themselves and their friends.
2. Access accounts and applications they wouldn't normally use for their daily jobs.
3. E-mail former and prospective employers.
4. Conduct furtive instant-messaging chats.
5. Visit Web sites that cater to disgruntled employees, such as f'dcompany.com.
6. Perform large downloads and file copying.
7. Access the network during off hours.

Intrusion Techniques:

The password file can be protected in one of two ways:

• **One-way function:** The system stores only the value of a function based on the user's password. When the user presents a password, the system transforms that password and compares it with the stored value.
• **Access control:** Access to the password file is limited to one or a very few accounts.

The techniques for learning passwords:

1. Try default passwords used with standard accounts that are shipped with the system. Many administrators do not bother to change these defaults.
2. Exhaustively try all short passwords (those of one to three characters).
3. Try words in the system's online dictionary or a list of likely passwords. Examples of the latter are readily available on hacker bulletin boards.
4. Collect information about users, such as their full names, the names of their spouse and children, pictures in their office, and books in their office that are related to hobbies.
5. Try users' phone numbers, Social Security numbers, and room numbers.
6. Try all legitimate license plate numbers for this state.
7. Use a Trojan horse to bypass restrictions on access.
8. Tap the line between a remote user and the host system.

Contents

Intrusion detection system:

1. If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised.

2. An effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions.

3. Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.

The following approaches to intrusion detection:

1. **Statistical anomaly detection:** Involves the collection of data relating to the behavior of legitimate users over a period of time.

   a. ***Threshold detection:*** This approach involves defining thresholds, independent of user, for the frequency of occurrence of various events.
   b. ***Profile based:*** A profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts.

2. **Rule-based detection:** Involves an attempt to define a set of rules that can be used to decide that a given behavior is that of an intruder.

   a. ***Anomaly detection:*** Rules are developed to detect deviation from previous usage patterns.
   b. ***Penetration identification:*** An expert system approach that searches for suspicious behavior.

Audit Records

- A fundamental tool for intrusion detection is the audit record. Some record of ongoing activity by users must be maintained as input to an intrusion detection system. Basically, two plans are used:

  • **Native audit records:** Virtually all multiuser operating systems include accounting software that collects information on user activity.

  • **Detection-specific audit records:** A collection facility can be implemented that generates audit records containing only that information required by the intrusion detection system.

Each audit record contains the following fields:

• **Subject:** Initiators of actions.
• **Action:** Operation performed by the subject on or with an object.
• **Object:** Receptors of actions.
• **Exception-Condition:** Denotes which, if any, exception condition is raised on return.
• **Resource-Usage:** A list of quantitative elements in which each element gives the amount used of some resource.
• **Time-Stamp:** Unique time-and-date stamp identifying when the action took place.

Statistical Anomaly Detection:

Statistical anomaly detection techniques fall into two broad categories:

- ***Threshold detection systems:*** Threshold detection involves counting the number of occurrences of a specific event type over an interval of time.
- ***Profile-based systems:*** Profile-based anomaly detection focuses on characterizing the past behavior of individual users or related groups of users and then detecting significant deviations.

Examples of metrics that are useful for profile-based intrusion detection are the following:

• **Counter:** A nonnegative integer that may be incremented but not decremented until it is reset by management action.
• **Gauge:** A nonnegative integer that may be incremented or decremented. Typically, a gauge is used to measure the current value of some entity.
• **Interval timer:** The length of time between two related events.
• **Resource utilization:** Quantity of resources consumed during a specified period.

Given these general metrics, various tests can be performed to determine whether current activity fits within acceptable limits. the following approaches that may be taken:

- **Mean and standard deviation**- of a parameter over some historical period. This gives a reflection of the average behavior and its variability.

- **Multivariate** - A **multivariate** model is based on correlations between two or more variables.
    - **Markov process**- A **Markov process** model is used to establish transition probabilities among various states.
    - **Time series**- A **time series** model focuses on time intervals, looking for sequences of events that happen too rapidly or too slowly.
    - **Operational**- Finally, an **operational model** is based on a judgment of what is considered abnormal, rather than an automated analysis of past audit records.

Rule-Based Intrusion Detection:

- Rule-based techniques detect intrusion by observing events in the system and applying a set of rules that lead to a decision regarding whether a given pattern of activity is or is not suspicious.
- **Rule-based anomaly detection** is similar in terms of its approach and strengths to statistical anomaly detection.
- With the rule-based approach, historical audit records are analyzed to identify usage patterns and to generate automatically rules that describe those patterns. Rules may represent past behavior patterns of users, programs, privileges, time slots, terminals, and so on.
- **Rule-based penetration identification** takes a very different approach to intrusion detection. The key feature of such systems is the use of rules for identifying known penetrations or penetrations that would exploit known weaknesses.
- Rules can also be defined that identify suspicious behavior, even when the behavior is within the bounds of established patterns of usage.

Distributed Intrusion Detection

- Until recently, work on intrusion detection systems focused on single-system standalone facilities.
- Porras points out the following major issues in the design of a distributed intrusion detection system :
    - A distributed intrusion detection system may need to deal with different audit record formats.
    - One or more nodes in the network will serve as collection and analysis points for the data from the systems on the network.
    - Either a centralized or decentralized architecture can be used. With a centralized architecture, there is a single central point of collection and analysis of all audit data.

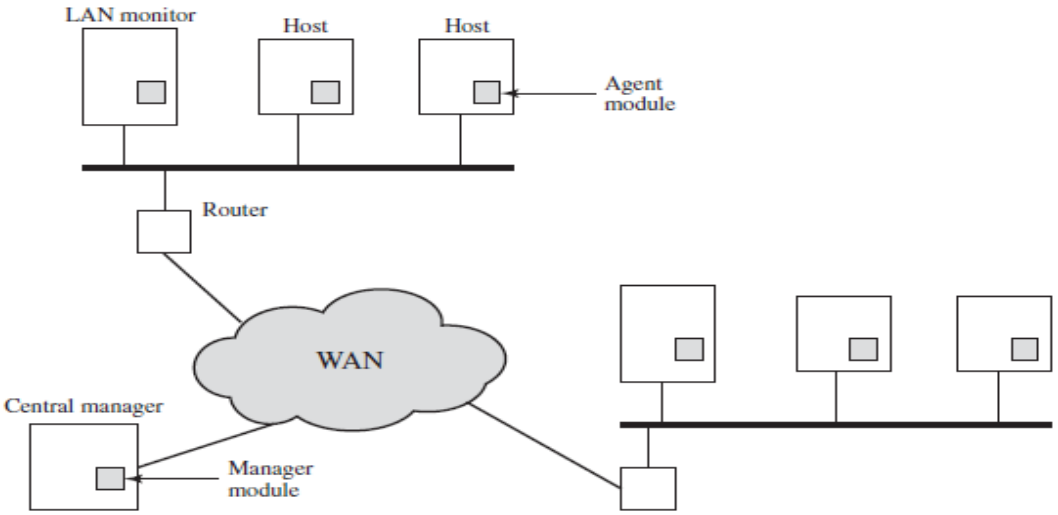


Figure 20.2 Architecture for Distributed Intrusion Detection

A good example of a distributed intrusion detection system, which consists of three main components:

• **Host agent module:** An audit collection module operating as a background process on a monitored system. Its purpose is to collect data on security related events on the host and transmit these to the central manager.
• **LAN monitor agent module:** Operates in the same fashion as a host agent module except that it analyzes LAN traffic and reports the results to the central manager.
• **Central manager module:** Receives reports from LAN monitor and host agents and processes and correlates these reports to detect intrusion.

- Figure 20.3 shows the general approach that is taken. The agent captures each audit record produced by the native audit collection system. A filter is applied that retains only those records that are of security interest.
- At the lowest level, the agent scans for notable events that are of interest independent of any past events.
- At the next higher level, the agent looks for sequences of events, such as known attack patterns (signatures).
- Finally, the agent looks for anomalous behavior of an individual user based on a historical profile of that user, such as number of programs executed, number of files accessed, and the like.

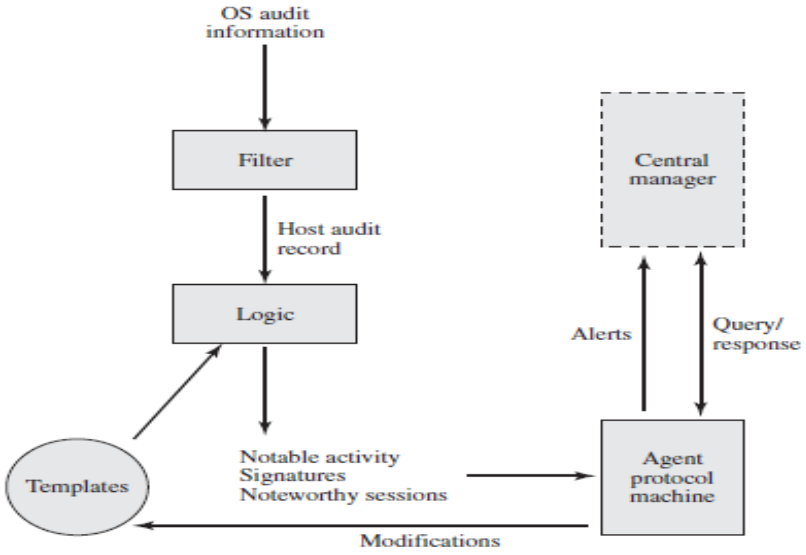


Figure 20.3 Agent Architecture

Honeypots

- A relatively recent innovation in intrusion detection technology is the honeypot. Honeypots are decoy systems that are designed to *lure a potential attacker* away from critical systems.

- Honeypots are designed to
  - divert an attacker from accessing critical systems
  - collect information about the attacker's activity
  - encourage the attacker to stay on the system long enough for administrators to respond

- Initial efforts involved a single honeypot computer with IP addresses designed to attract hackers. More recent research has focused on building entire honeypot networks that emulate an enterprise, possibly with actual or simulated traffic and data. Once hackers are within the network, administrators can observe their behavior in detail and figure out defenses.

Intrusion Detection Exchange Format:

- To facilitate the development of distributed intrusion detection systems that can function across a wide range of platforms and environments, standards are needed to support interoperability. Such standards are the focus of the IETF Intrusion Detection Working Group.

- The purpose of the working group is to define data formats and exchange procedures for sharing information of interest to intrusion detection and response systems and to management systems that may need to interact with them.
  The outputs of this working group include:
  1. A requirements document, which describes the high-level functional requirements
  2. A common intrusion language specification, which describes data formats that satisfy the requirements.
  3. A framework document, which identifies existing protocols best used for communication

5.4.2. MALICIOUS SOFTWARE

Malicious software can be divided into two categories:

- Those that need a host program, and those that are independent.
- The former are essentially fragments of programs that cannot exist independently of some actual application program, utility, or system program.
- Viruses, logic bombs, and backdoors are examples. The latter are self-contained programs that can be scheduled and run by the operating system. Worms and zombie programs are examples.

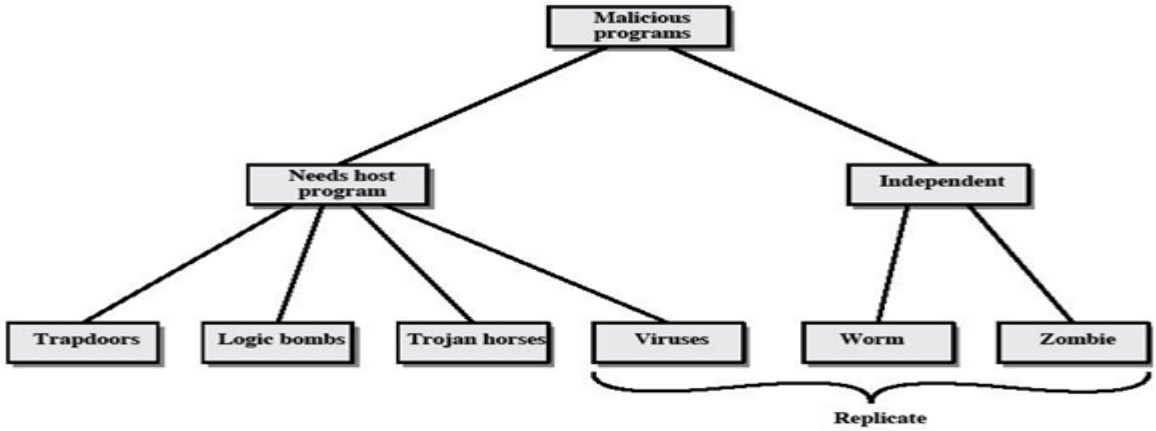


Figure 19.1 Taxonomy of Malicious Programs

5.4.3. VIRUSES

Contents
• **The Nature of Viruses** ✓ **Infection mechanism:** ✓ **Trigger:** ✓ **Payload:** • **Virus Structure** • **Viruses Classification** ✓ **classification by target** ✓ **classification by concealment strategy** • **Virus Kits** • **Macro Viruses** • **E-Mail Viruses**

- Perhaps the most sophisticated types of threats to computer systems are presented by programs that exploit vulnerabilities in computing systems.

Backdoor:

- A backdoor, also known as a **trapdoor,** is a secret entry point into a program that allows someone that is aware of the backdoor to gain access without going through the usual security access procedures.
- Programmers have used backdoors legitimately for many years to debug and test programs.
- The backdoor is code that recognizes some special sequence of input or is triggered by being run from a certain user ID or by an unlikely sequence of events.
- Backdoors become threats when unscrupulous programmers use them to gain unauthorized access.
- The backdoor was the basic idea for the vulnerability portrayed in the movie *War Games*.

Logic Bomb

- One of the oldest types of program threat, predating viruses and worms, is the logic bomb.
- The logic bomb is code embedded in some legitimate program that is set to "explode" when certain conditions are met.
- Examples of conditions that can be used as triggers for a logic bomb are the presence or absence of certain files, a particular day of the week or date, or a particular user running the application.
- Once triggered, a bomb may alter or delete data or entire files, cause a machine halt, or do some other damage.

Trojan Horses

- A Trojan horse is a useful, or apparently useful, program or command procedure containing hidden code that, when invoked, performs some unwanted or harmful function.
- Trojan horse programs can be used to accomplish functions indirectly that an unauthorized user could not accomplish directly.
- For example, to gain access to the files of another user on a shared system, a user could create a Trojan horse program that, when executed,changed the invoking user's file permissions so that the files are readable by any user.
- The author could then induce users to run the program by placing it in a common directory and naming it such that it appears to be a useful utility.
- The code creates a backdoor in the login program that permits the author to log on to the system using a special password.
- This Trojan horse can never be discovered by reading the source code of the login program.

Zombie

- A zombie is a program that secretly takes over another Internet-attached computer and then uses that computer to launch attacks that are difficult to trace to the zombie's creator.
- Zombies are used in denial-of-service attacks, typically against targeted Web sites.

- The zombie is planted on hundreds of computers belonging to unsuspecting third parties, and then used to overwhelm the target Web site by launching an overwhelming onslaught of Internet traffic.

The Nature of Viruses

- A computer virus is a piece of software that can **"infect**" other programs by modifying them; the modification includes injecting the original program with a routine to make copies of the virus program, which can then go on to infect other programs.
- A virus can do anything that other programs do. The difference is that a virus attaches itself to another program and executes secretly when the host program is run.

A computer virus has three parts:

- **Infection mechanism:** The means by which a virus spreads, enabling it to replicate. The mechanism is also referred to as the **infection vector**.
- **Trigger:** The event or condition that determines when the payload is activated or delivered.
- **Payload:** What the virus does, besides spreading. During its lifetime.

A typical virus goes through the following four phases:

- ✓ **Dormant phase:** The virus will eventually be activated by some event, such as a date, the presence of another program or file, or the capacity of the disk exceeding some limit.
- ✓ **Propagation phase:** The virus places a copy of itself into other programs or into certain system areas on the disk.
- ✓ **Triggering phase:** As with the dormant phase, the triggering phase can be caused by a variety of system events, including a count of the number of times that this copy of the virus has made copies of itself.
- ✓ **Execution phase:** The function may be harmless, such as a message on the screen, or damaging, such as the destruction of programs and data files.

Virus Structure A virus can be prepended or postpended to an executable program, or it can be embedded in some other fashion. In this case, the virus code,V, is prepended to infected programs, and it is assumed that the entry point to the program, when invoked, is the first line of the program.

```
     program V :=

{goto main;
     1234567;

     subroutine infect-executable :=
          {loop:
          file := get-random-executable-file;
          if (first-line-of-file = 1234567)
             then goto loop
             else prepend V to file; }

     subroutine do-damage :=
          {whatever damage is to be done}

     subroutine trigger-pulled :=
          {return true if some condition holds}

main:    main-program :=
          {infect-executable;
          if trigger-pulled then do-damage;
          goto next;}
next:

}
```

Figure 21.1 A Simple Virus

When this program is invoked, control passes to its virus, which performs the following steps:

1. For each uninfected file P2 that is found, the virus first compresses that file to produce ,which is shorter than the original program by the size of the virus.
2. A copy of the virus is prepended to the compressed program.
3. The compressed version of the original infected program is uncompressed.
4. The uncompressed original program is executed.

Viruses Classification:

Viruses are classified along two orthogonal axes:

- ✓ The type of target the virus tries to infect and
- ✓ The method the virus uses to conceal itself from detection by users and antivirus software.

A virus **classification by target** includes the following categories:

• **Boot sector infector:** Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.

• **File infector:** Infects files that the operating system or shell consider to be executable.

• **Macro virus:** Infects files with macro code that is interpreted by an application.

- A virus classification **by concealment strategy** includes the following categories:

• **Encrypted virus:** A typical approach is as follows. A portion of the virus creates a random encryption key and encrypts the remainder of the virus. The key is stored with the virus.

• **Stealth virus:** A form of virus explicitly designed to hide itself from detection by antivirus software. Thus, the entire virus, not just a payload is hidden.

• **Polymorphic virus:** A virus that mutates with every infection, making detection by the "signature" of the virus impossible.

• **Metamorphic virus:** As with a polymorphic virus, a metamorphic virus mutates with every infection. Metamorphic viruses may change their behavior as well as their appearance.

Virus Kits

- Another weapon in the virus writers' armory is the virus-creation toolkit. Such a toolkit enables a relative novice to quickly create a number of different viruses.
- Although viruses created with toolkits tend to be less sophisticated than viruses designed from scratch, the sheer number of new viruses that can be generated using a toolkit creates a problem for antivirus schemes.
-

Macro Viruses

- In the mid-1990s, macro viruses became by far the most prevalent type of virus. Macro viruses are particularly threatening for a number of reasons:
  1. A macro virus is platform independent.
  2. Macro viruses infect documents, not executable portions of code.
  3. Macro viruses are easily spread. A very common method is by electronic mail.
  4. Because macro viruses infect user documents rather than system programs, traditional file system access controls are of limited use in preventing their spread.

E-Mail Viruses:

- A more recent development in malicious software is the e-mail virus. The first rapidly spreading e-mail viruses, such as Melissa, made use of a Microsoft Word macro embedded in an attachment. If the recipient opens the e-mail attachment, the Word macro is activated. Then
  1. The e-mail virus sends itself to everyone on the mailing list in the user's e-mail package.
  2. The virus does local damage on the user's system.

WORM COUNTERMEASURES OR DIGITAL IMMUNE SYSTEM

Contents
• **Countermeasures**

- **Antivirus Approaches**
  - ✓ **Detection:**
  - ✓ **Identification:**
  - ✓ **Removal:**
- **First generation: simple scanners**
- **Second generation: heuristic scanners**
- **Third generation: activity traps**
- **Fourth generation: full-featured protection**
- **Advanced Antivirus Techniques**
  - **Generic Decryption:**
    - ➢ **CPU emulator:**
    - ➢ **Virus signature scanner:**
    - ➢ **Emulation control module:**
  - **Digital Immune System:**
    - ➢ **Integrated mail systems:**
    - ➢ **Mobile-program systems:**
  - **Behavior-Blocking Software:**

Countermeasures:
Antivirus Approaches:

- The ideal solution to the threat of viruses is prevention:
  - Do not allow a virus to get into the system in the first place, or
  - block the ability of a virus to modify any files containing executable code or macros.

- The next best approach is to be able to do the following:

  • **Detection:** Once the infection has occurred, determine that it has occurred and locate the virus.
  • **Identification:** Once detection has been achieved, identify the specific virus that has infected a program.
  • **Removal:** Once the specific virus has been identified, remove all traces of the virus from the infected program and restore it to its original state. Remove the virus from all infected systems so that the virus cannot spread further.

- The four generations of antivirus software:
  - **• First generation: simple scanners**
  - **• Second generation: heuristic scanners**
  - **• Third generation: activity traps**
  - **• Fourth generation: full-featured protection**

- A **first-generation** scanner requires a virus signature to identify a virus. The virus may contain "wildcards" but has essentially the same structure and bit pattern in all copies.
- A **second-generation** scanner does not rely on a specific signature. Rather, the scanner uses heuristic rules to search for probable virus infection.
- **Third-generation** programs are memory-resident programs that identify a virus by its actions rather than its structure in an infected program.

- **Fourth-generation** products are packages consisting of a variety of antivirus techniques used in conjunction.

Advanced Antivirus Techniques

- More sophisticated antivirus approaches and products continue to appear. In this subsection, we highlight some of the most important.

Generic Decryption:

- Generic decryption (GD) technology enables the antivirus program to easily detect even the most complex polymorphic viruses while maintaining fast scanning speeds. In order to detect such a structure, executable files are run through a GD scanner,

Which contains the following elements?

• **CPU emulator:** A software-based virtual computer. Instructions in an executable file are interpreted by the emulator rather than executed on the underlying processor.

• **Virus signature scanner:** A module that scans the target code looking for known virus signatures.

• **Emulation control module:** Controls the execution of the target code.

Digital Immune System:

- The motivation for this development has been the rising threat of Internet-based virus propagation.
- **The two major trends in Internet technology are**

• **Integrated mail systems:** Systems such as Lotus Notes and Microsoft Outlook make it very simple to send anything to anyone and to work with objects that are received.

• **Mobile-program systems:** Capabilities such as Java and ActiveX allow programs to move on their own from one system to another.

- Figure 21.4 illustrates the typical steps in digital immune system operation:

1. A monitoring program on each PC uses a variety of heuristics based on system behavior.
2. The administrative machine encrypts the sample and sends it to a central virus analysis machine.
3. This machine creates an environment in which the infected program can be safely run for analysis.
4. The resulting prescription is sent back to the administrative machine.
5. The administrative machine forwards the prescription to the infected client.
6. The prescription is also forwarded to other clients in the organization.
7. Subscribers around the world receive regular antivirus updates that protect them from the new virus.

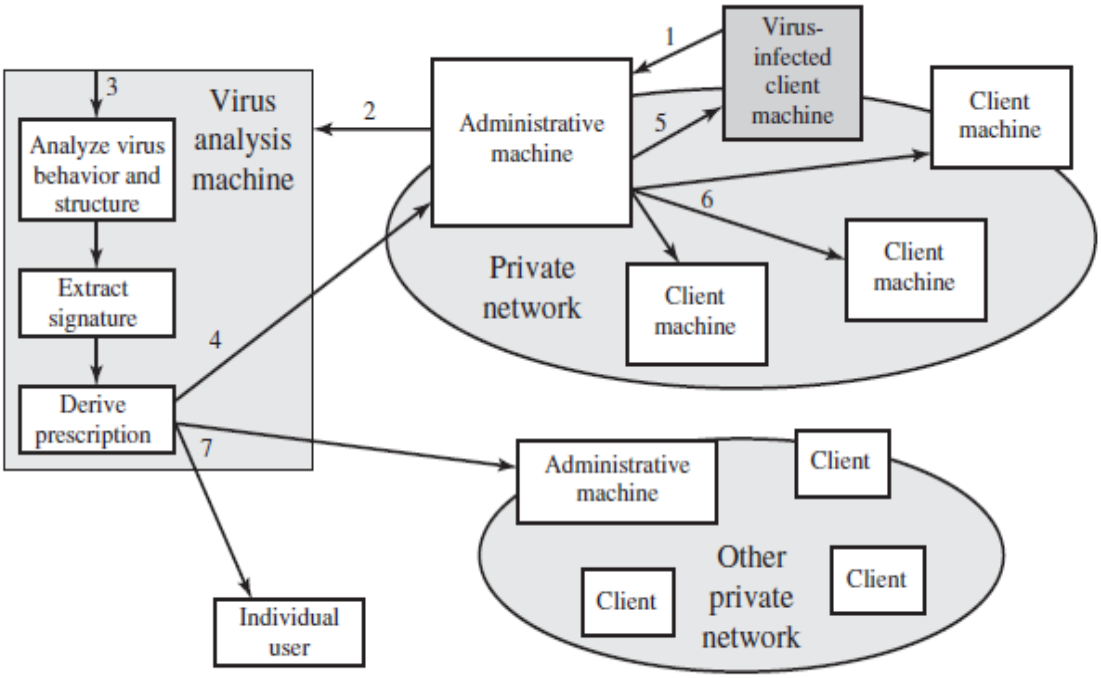


Figure 21.4 Digital Immune System

- ***Behavior-Blocking Software:***
    - The behavior blocking software blocks potentially malicious actions before they have a chance to affect the system.
    - Monitored behaviors can include
        - Attempts to open, view, delete, and/or modify files;
        - Attempts to format disk drives and other unrecoverable disk operations;
        - Modifications to the logic of executable files or macros;
        - Modification of critical system settings, such as start-up settings;
        - Scripting of e-mail and instant messaging clients to send executable content; and
        - Initiation of network communications.

    - Figure 21.5 illustrates the operation of a behavior blocker. Behavior-blocking software runs on server and desktop computers and is instructed through policies set by the network administrator to let benign actions take place but to intercede when unauthorized or suspicious actions occur.

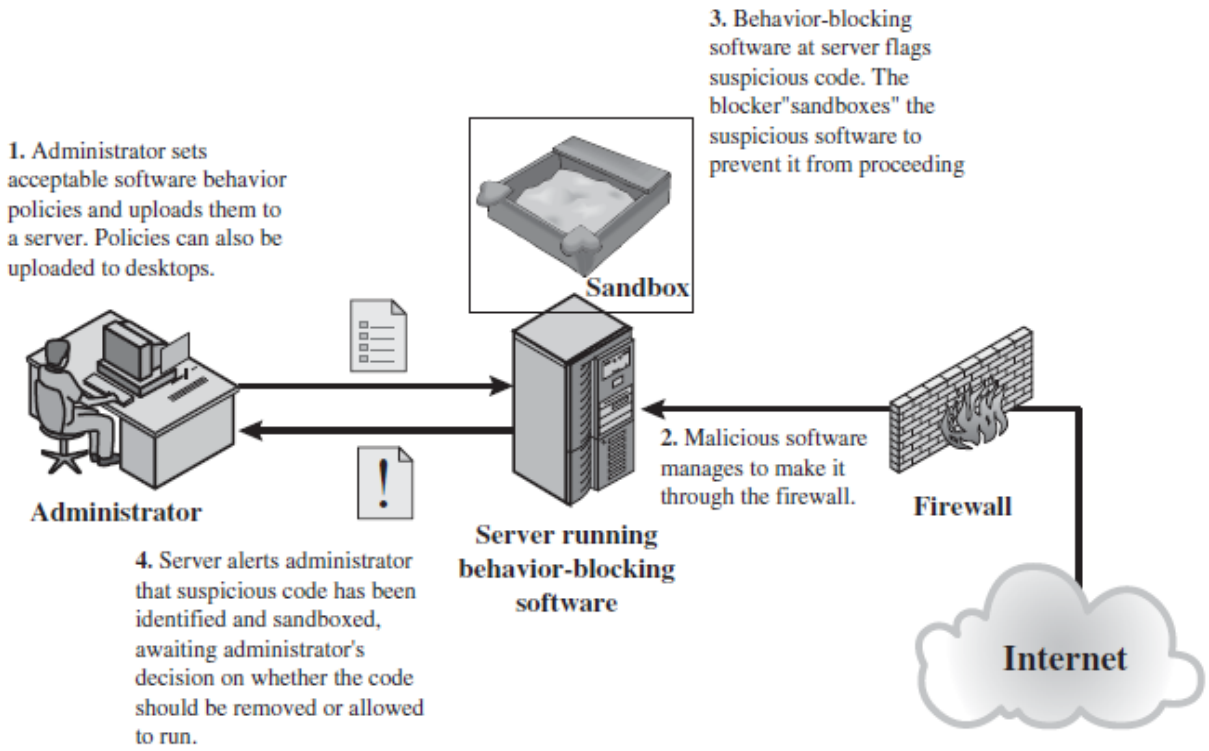


Figure 21.5 Behavior-Blocking Software Operation

5.4.4. FIREWALLS

Internet Firewalls for Trusted Systems

- A firewall is a device or group of devices that controls access between networks.
- A firewall generally consists of filters and gateway(s), varying from firewall to firewall.
- It is a security gateway that controls access between the public Internet and an intranet (a private internal network) and is a secure computer system placed between a trusted network and an untrusted internet.
- Firewalls act as an intermediate server in handling SMTP and HTTP connections in either direction.
- Firewalls can be classified into three main categories: packet filters, circuit-level gateways and application-level gateways

Contents
• Role of Firewalls • Firewall-Related Terminology • Types of Firewalls • Firewall Designs

Screened Host Firewall (Single-homed Bastion Host)

Screened Host Firewall (Dual-homed Bastion Host)

Screened Subnet Firewall

Role of Firewalls:

- The firewall itself must be immune to penetration.

- Firewalls create checkpoints (or choke points) between an internal private network and an untrusted Internet .

- The firewall may filter on the basis of IP source and destination addresses and TCP port number.

- The firewall also enforces logging, and provides alarm capacities as well.

- Firewalls may block TELNET or RLOGIN connections from the Internet to the intranet.

- They also block SMTP and FTP connections to the Internet from internal systems not authorised to send e-mail or to move files.

- The firewall provides protection from various kinds of IP spoofing and routing attacks.

Firewall-Related Terminology:

To design and configure a firewall, some familiarity with the basic terminology is required.
.

- Bastion Host
- Proxy Server
- SOCKS
- Choke Point
- De-militarised Zone (DMZ)
- Logging and Alarms
- VPN

Bastion Host

- A bastion host is a publicly accessible device for the network's security, which has a direct connection to a public network such as the Internet.

- The bastion host serves as a platform for any one of the three types of firewalls: packet filter, circuit-level gateway or application-level gateway.
- Bastion hosts must check all incoming and outgoing traffic and enforce the rules specified in the security policy. They must be prepared for attacks from external and possibly internal sources. They should be built with the least amount of hardware and software in order for a potential hacker to have less opportunity to overcome the firewall.

- **The bastion host's role falls into the following three common types:**

    - ***Single-homed bastion host***: This is a device with only one network interface, normally used for an **application-level gateway**. The external router is configured to send all incoming data to the bastion host, and all internal clients are configured to send all outgoing data to the host.
    - ***Dual-homed bastion host*:** This is a firewall device with at least two network interfaces. Dual-homed bastion hosts serve as **application-level gateways**, and as **packet filters and circuit-level gateways as well**. The advantage of using such hosts is that they create a complete break between the external network and the internal network.
    - ***Multihomed bastion host***: Single-purpose or internal bastion hosts can be classified as either single-homed or multihomed bastion hosts. The latter are used to allow the user to enforce strict security mechanisms.

Proxy Server

- When the security policy requires all inbound and outbound traffic to be sent through a proxy server, a new proxy server should be created for the new streaming application. On the new proxy server, it is necessary to implement strict security mechanisms such as authentication.

SOCKS

- The SOCKS protocol version 4 provides for unsecured firewall traversal for TCP-based client/server applications, including HTTP, TELNET and FTP.
- The new protocol extends the SOCKS version 4 model to include UDP, and allows the framework to include provision for generalized strong authentication schemes, and extends the addressing scheme to encompass domain name and IPv6 addresses.

Choke Point

- The most important aspect of firewall placement is to create choke points. A choke point is the point at which a public internet can access the internal network. The most comprehensive and extensive monitoring tools should be configured on the choke points.
- Proper implementation requires that all traffic be funnelled through these choke points

De-militarised Zone (DMZ)

- The DMZ is an expression that originates from the Korean War. It meant a strip of land forcibly kept clear of enemy soldiers. In terms of a firewall, the DMZ is a network that lies between an internal private network and the external public network.
- DMZ networks are sometimes called perimeter networks. A DMZ is used as an additional buffer to further separate the public network from the internal network.
- A gateway is a machine that provides relay services to compensate for the effects of a filter. The network inhabited by the gateway is often called the DMZ. A gateway in the DMZ is sometimes assisted by an internal gateway.

Logging and Alarms

- Logging is usually implemented at every device in the firewall, but these individual logs combine to become the entire record of user activity. Packet filters normally do not enable logging by default so as not to degrade performance. Packet filters as well as circuit-level gateways log only the most basic information.

VPN

- Some firewalls are now providing VPN services. VPNs are appropriate for any organization requiring secure external access to internal resources. All VPNs are tunneling protocols in the sense that their information packets or payloads are encapsulated or tunneled into the network packets.
- All data transmitted over a VPN is usually encrypted because an opponent with access to the Internet could eavesdrop on the data as it travels over the public network. The VPN encapsulates all the encrypted data within an IP packet. Authentication, message integrity and encryption are very important fundamentals for implementing a VPN.

.

Types of firewall

Contents
• **Types of Firewalls** ✓ **Packet Filtering Firewall** ✓ **Stateful Inspection Firewalls** ✓ **Application-Level Gateway** ✓ **Circuit-Level Gateway**

Types of Firewalls

- A firewall may act as a packet filter. It can operate as a positive filter, allowing to pass only packets that meet specific criteria, or as a negative filter, rejecting any packet that meets certain criteria.

Packet Filtering Firewall

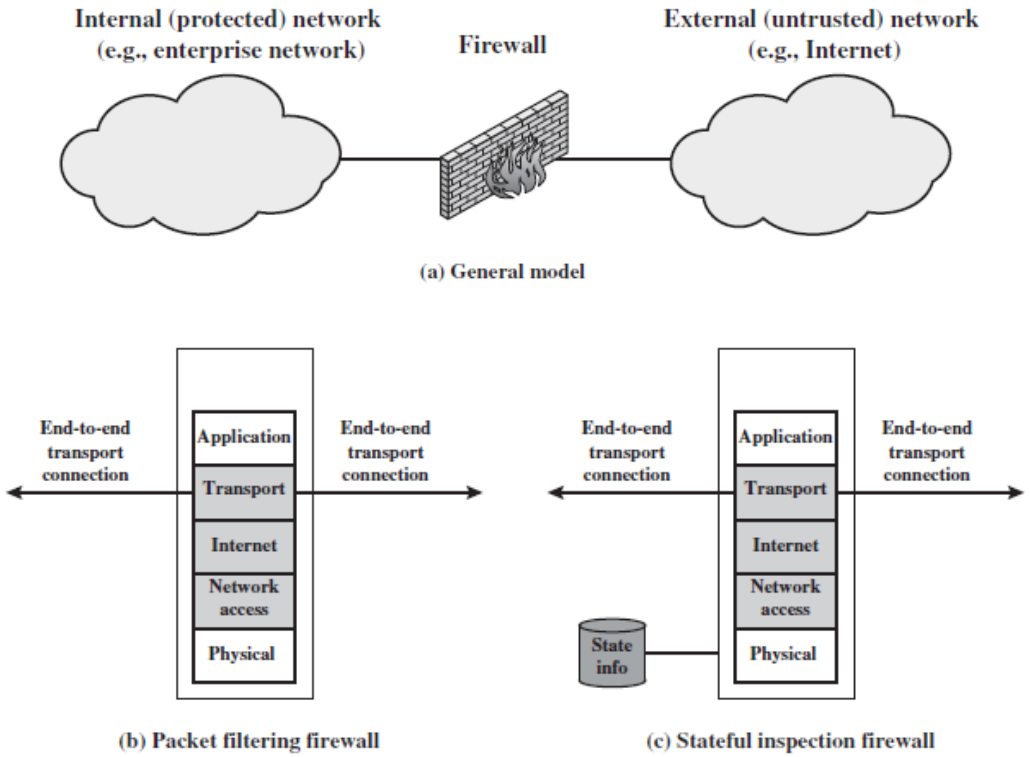
- A packet filtering firewall applies a set of rules to each incoming and outgoing IP packet and then forwards or discards the packet.
- The firewall is typically configured to filter packets going in both directions (from and to the internal network). Filtering rules are based on information contained in a network packet:
    - **Source IP address:** The IP address of the system that originated the IP packet

- **Destination IP address:** The IP address of the system the IP packet is trying to reach
- **Source and destination transport-level address:** The transport-level (e.g., TCP or UDP) port number, which defines applications such as SNMP or TELNET
- **IP protocol field:** Defines the transport protocol
- **Interface:** For a firewall with three or more ports, which interface of the firewall the packet came from or which interface of the firewall the packet is destined for

- The packet filter is typically set up as a list of rules based on matches to fields in the IP or TCP header. **Two default policies are possible:**

  • **Default = discard:** That which is not expressly permitted is prohibited.
  • **Default = forward:** That which is not expressly prohibited is permitted.

- The default discard policy is more conservative. Initially, everything is blocked, and services must be added on a case-by-case basis.
- The default forward policy increases ease of use for end users but provides reduced security; the security administrator must, in essence, react to each new security threat as it becomes known.



A. Inbound mail is allowed (port 25 is for SMTP incoming), but only to a gateway host. However, packets from a particular external host, SPIGOT, are blocked because that host has a history of sending massive files in e-mail messages.

B. This is an explicit statement of the default policy. All rulesets include this rule implicitly as the last rule.

- **One *advantage* of** a packet filtering firewall is its **simplicit**y. Also, packet filters typically are transparent to users and are very fast.

- The following are ***disadvantages*** of packet filter firewalls:

    - Because packet filter firewalls do not examine upper-layer data, they ***cannot prevent attacks*** that employ application-specific vulnerabilities or functions.
    - Because of the ***limited information*** available to the firewall, the logging functionality present in packet filter firewalls is limited.
    - Most packet filter firewalls ***do not support advanced user*** authentication schemes.
    - Packet filter firewalls are generally *vulnerable to attacks*, such as ***network layer address spoofing*.**
    - Finally, due to the small number of variables used in access control decisions, packet filter firewalls **are *susceptible to security breaches*** caused by improper configurations.

- Some of the attacks that can be made on packet filtering firewalls and the appropriate countermeasures are the following:
-
    - **IP address spoofing:** The intruder transmits packets from the outside with a source IP address field containing an address of an internal host.
    - **Source routing attacks:** The source station specifies the route that a packet should take as it crosses the Internet, in the hopes that this will bypass security measures that do not analyze the source routing information.
    - **Tiny fragment attacks:** The intruder uses the IP fragmentation option to create extremely small fragments and force the TCP header information into a separate packet fragment.

<u>Stateful Inspection Firewalls</u>

- A simple packet filtering firewall must permit inbound network traffic on all these high-numbered ports for TCP-based traffic to occur. This creates a vulnerability that can be exploited by unauthorized users.

- A stateful inspection packet firewall tightens up the rules for TCP traffic by creating a directory of outbound TCP connections. There is an entry for each currently established connection. The packet filter will now allow incoming traffic to high-numbered ports only for those packets that fit the profile of one of the entries in this directory.

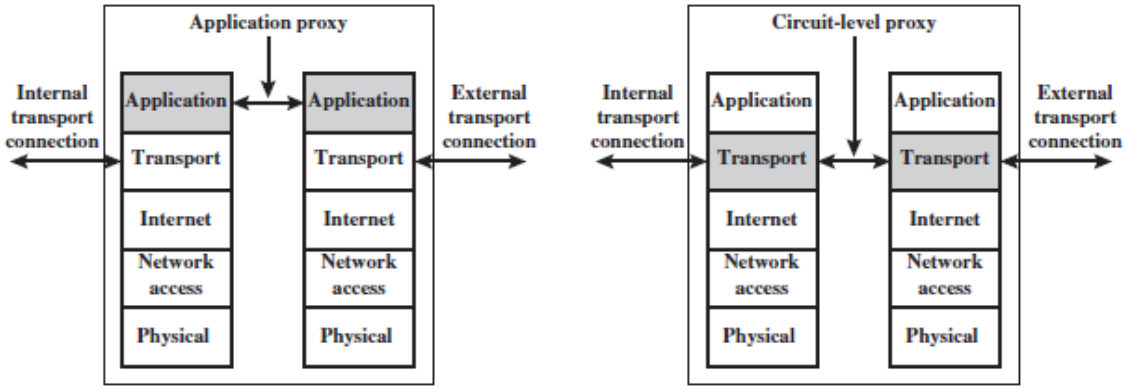
- A stateful packet inspection firewall reviews the same packet information as a packet filtering firewall, but also records information about TCP connections. Some stateful firewalls also keep track of TCP sequence numbers to prevent attacks that depend on the sequence number, such as session hijacking. Some even inspect limited amounts of application data for some well-known protocols like FTP, IM and SIPS commands, in order to identify and track related connections.

Application-Level Gateway:

- An application-level gateway, also called an **application proxy,** acts as a relay of application-level traffic. The user contacts the gateway using a TCP/IP application, such as Telnet or FTP, and the gateway asks the user for the name of the remote host to be accessed.

- When the user responds and provides a valid user ID and authentication information, the gateway contacts the application on the remote host and relays TCP segments containing the application data between the two endpoints. If the gateway does not implement the proxy code for a specific application, the service is not supported and cannot be forwarded across the firewall.

- A prime disadvantage of this type of gateway is the additional processing overhead on each connection. In effect, there are two spliced connections between the end users, with the gateway at the splice point, and the gateway must examine and forward all traffic in both directions.



(d) Application proxy firewall (e) Circuit-level proxy firewall

Figure 22.1 Types of Firewalls

Circuit-Level Gateway:

- A fourth type of firewall is the circuit-level gateway or **circuit-level proxy**. This can be a stand-alone system or it can be a specialized function performed by an application-level gateway for certain applications.

- As with an application gateway, a circuit-level gateway does not permit an end-to-end TCP connection; rather, the gateway sets up two TCP connections, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outside host.

- A typical use of circuit-level gateways is a situation in which the system administrator trusts the internal users. The gateway can be configured to support application- level or proxy service on inbound connections and circuit-level functions for outbound connections. In this configuration, the gateway can incur the processing overhead of examining incoming application data for forbidden functions but does not incur that overhead on outgoing data.

- An example of a circuit-level gateway implementation is the SOCKS package; version 5 of SOCKS is specified in RFC 1928.The RFC defines SOCKS in the following fashion:

  SOCKS consists of the following components:

  - The SOCKS server, which often runs on a UNIX-based firewall. SOCKS is also implemented on Windows systems.
  - The SOCKS client library, which runs on internal hosts protected by the firewall.
  - SOCKS-ified versions of several standard client programs such as FTP and TELNET.

FIREWALL DESIGN OR FIREWALL CONFIGURATION

Contents

• **Firewall Designs**
  ○ Screened Host Firewall (Single-homed Bastion Host)
  ○ Screened Host Firewall (Dual-homed Bastion Host)
  ○ Screened Subnet Firewall

Firewall design

- A security administrator must decide on the location and on the number of firewalls needed. to implement a firewall strategy. The primary step in designing a secure firewall is obviously to prevent the firewall devices from being compromised by threats.
- To provide a certain level of security, the three basic firewall designs are considered: a single-homed bastion host, a dual-homed bastion host and a screened subnet firewall. The first two options are for creating a screened host firewall, and the third option contains an additional packet-filtering router to achieve another level of security.

Screened Host Firewall (Single-homed Bastion Host)

- The first type of firewall is a **screened host** which uses a single-homed bastion host plus a packet-filtering router, as shown in Figure 10.4.
- Single-homed bastion hosts can be configured as either circuit-level or application-level gateways.
- NAT is essentially needed for developing an address scheme internally. It is a critical component of any firewall strategy. It translates the internal IP addresses to IANA

registered addresses to access the Internet. Hence, using NAT allows network administrators to use any internal IP address scheme.

- The screened host firewall is designed such that all incoming and outgoing information is passed through the bastion host.
- The external screening router is configured to route all incoming traffic directly to the bastion host as indicated in Figure 10.4.

- The screening router is also configured to route outgoing traffic only if it originates from the bastion host.

- **A single-homed implementation** may allow a hacker to modify the router not to forward packets to the bastion host. This action would bypass the bastion host and allow the hacker directly into the network.
- But such a bypass usually does not happen because a network using a single-homed bastion host is normally configured to send packets only to the bastion host, and ot directly to the Internet.

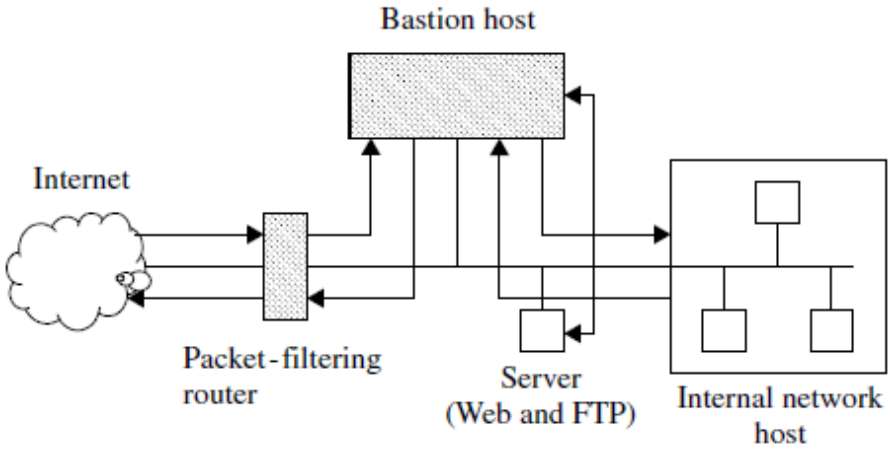


Figure 10.4 Screened host firewall system (single-homed bastion host).

Screened Host Firewall (Dual-homed Bastion Host)

- The configuration of the screened host firewall using a dual-homed bastion host adds significant security, compared with a single-homed bastion host. As shown in Figure 10.5, a dual-homed bastion host has two network interfaces.

- This firewall implementation is secure due to the fact that it creates a complete break between the internal network and the external Internet. As with the single-homed bastion, all external traffic is forwarded directly to the bastion host for processing.

- However, a hacker may try to subvert the bastion host and the router to bypass the firewall mechanisms. Even if a hacker could defeat either the screening router or the dual-homed bastion host, the hacker would still have to penetrate the other. Nevertheless, a dual-homed bastion host removes even this possibility. It is also possible to implement NAT for dual-homed bastion hosts.

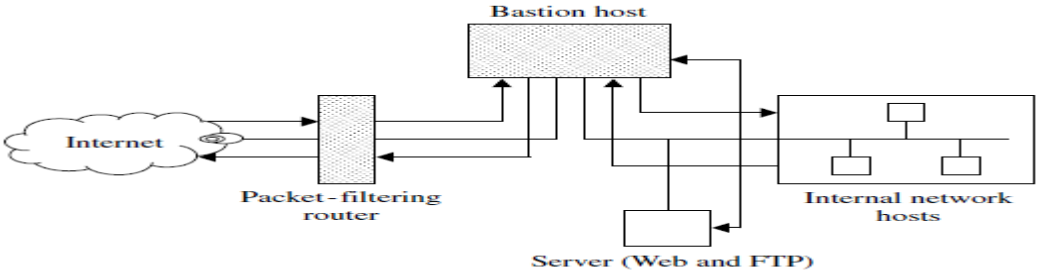


Figure 10.5 Screened host firewall system (dual-homed bastion host).

Screened Subnet Firewall

- The third implementation of a firewall is the screened subnet, which is also known as a DMZ. This firewall is the most secure one among the three implementations, simply because it uses a bastion host to support both circuit- and application-level gateways.
- As shown in Figure 10.6, all publicly accessible devices, including modem and server, are placed inside the DMZ.
- These DMZ then functions as a small isolated network positioned between the Internet and the internal network.
- The screened subnet firewall contains external and internal screening routers. Each is configured such that its traffic flows only to or from the bastion host. This arrangement prevents any traffic from directly traversing the DMZ subnetwork.
- The external screening router uses standard filtering to restrict external access to the bastion host, and rejects any traffic that does not come from the bastion host.
- The benefits of the screened subnet firewall are based on the following facts.
- First, a hacker must subvert three separate tri-homed interfaces when he or she wants to access the internal network. But it is almost infeasible.
- Second, the internal network is effectively invisible to the Internet because all inbound/outbound packets go directly through the DMZ.
- Third, internal users cannot access the Internet without going through the bastion host because the routing information is contained within the network.

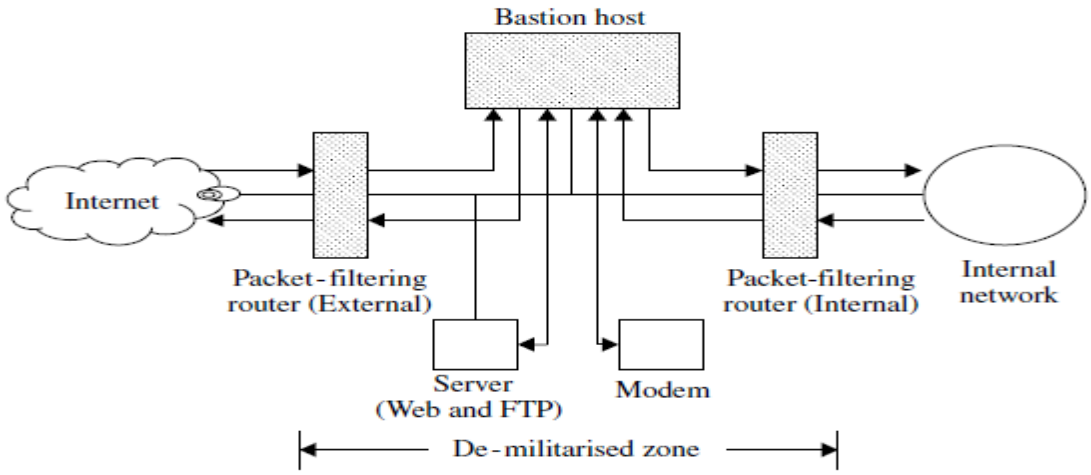


Figure 10.6 Screened subnet firewall system.

UNIT IV
MESSAGE AUTHENTICATION AND INTEGRITY

Authentication requirement – Authentication function – MAC – Hash function – Security of hash function and MAC – SHA –Digital signature and authentication protocols – DSS- Entity Authentication: Biometrics, Passwords, Challenge Response protocols- Authentication applications – Kerberos, X.509

4.1. AUTHENTICATION REQUIREMENT

- In the context of communications across a network, the following attacks can be identified.

  1. Disclosure: Release of message contents to any person or process not possessing the appropriate cryptographic key.

  2. Traffic analysis: Discovery of the pattern of traffic between parties. In a connection-oriented application, the frequency and duration of connections could be determined. In either a connection-oriented or connectionless environment, the number and length of messages between parties could be determined.

  3. Masquerade: Insertion of messages into the network from a fraudulent source. This includes the creation of messages by an opponent that are purported to come from an authorized entity.

  4. Content modification: Changes to the contents of a message, including insertion, deletion, transposition, and modification.

  5. Sequence modification: Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.

  6. Timing modification: Delay or replay of messages. In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed.

  7. Source repudiation: Denial of transmission of message by source.

  8. Destination repudiation: Denial of receipt of message by destination.
- **In summary**, message authentication is a procedure to verify that received messages come from the alleged source and have not been altered. Message authentication may also verify sequencing and timeliness.
- A digital signature is an authentication technique that also includes measures to counter repudiation by the source.

4.2. AUTHENTICATION FUNCTION

- Any message authentication or digital signature mechanism has two levels of functionality.
- At the lower level, there must be some sort of function that produces an authenticator: a value to be used to authenticate a message. This lower-level function is then used as a primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message.
- These may be grouped into three classes
  - ***Hash function***
  - ***Message Encryption***

- ***Message Authentication Code***

.

• Hash function: A function that maps a message of any length into a fixedlength hash value, which serves as the authenticator

• Message encryption: The ciphertext of the entire message serves as its authenticator

• Message authentication code (MAC): A function of the message and a secret key that produces a fixed-length value that serves as the authenticator

4.3. MAC - MESSAGE AUTHENTICATION CODE

- An alternative authentication technique involves the use of a secret key to generate a small fixed-size block of data, known as a cryptographic checksum or MAC, that is appended to the message. This technique assumes that two communicating parties, say A and B, share a common secret key K.
- When A has a message to send to B, it calculates the MAC as a function of the message and the key:

$$\text{MAC} = C(K, M)$$

where

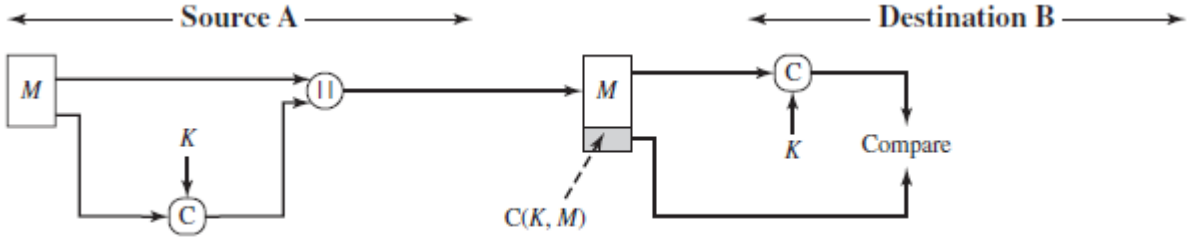
M = input message
C = MAC function
K = shared secret key
MAC = message authentication code

- If we assume that only the receiver and the sender know the identity of the secret key, and if the received MAC matches the calculated MAC, then

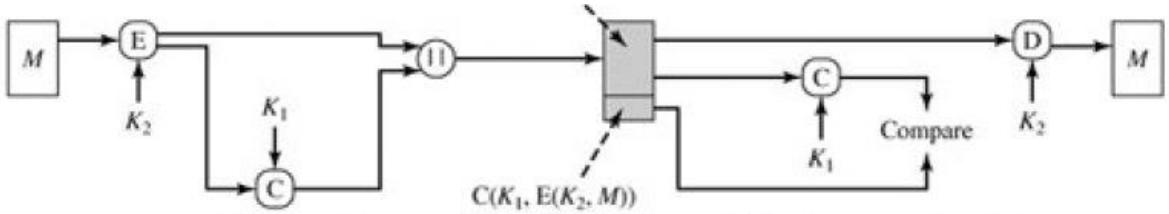
1. The receiver is assured that the message has not been altered.

2. The receiver is assured that the message is from the alleged sender.

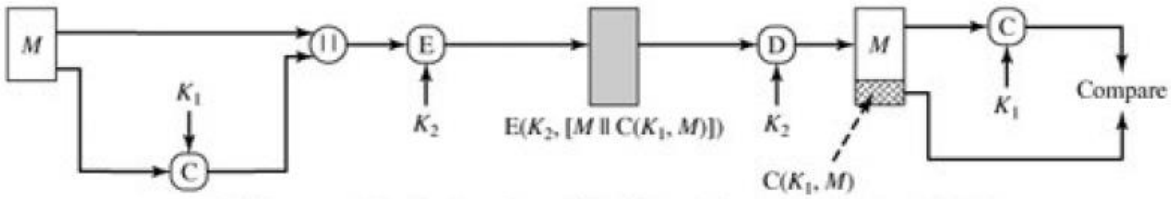
3. If the message includes a sequence number (such as is used with HDLC, X.25, and TCP), then the receiver can be assured of the proper sequence because an attacker cannot successfully alter the sequence number.



(a) Message authentication



(c) Message authentication and confidentiality; authentication tied to ciphertext



(b) Message authentication and confidentiality; authentication tied to plaintext

$E(K_2, M)$

- The process depicted in Figure a provides authentication but not confidentiality, because the message as a whole is transmitted in the clear.
- Confidentiality can be provided by performing message encryption either after (Figure .b) or before (Figure c) the MAC algorithm. In both these cases, two separate keys are needed, each of which is shared by the sender and the receiver.
- In the first case, the MAC is calculated with the message as input and is then concatenated to the message. The entire block is then encrypted. In the second case, the message is encrypted first.
- Then the MAC is calculated using the resulting ciphertext and is concatenated to the ciphertext to form the transmitted block. Typically, it is preferable to tie the authentication directly to the plaintext, so the method of Figure b is used.

Application of MAC

- Application in message is broadcast to a number of destinations.
- Authentication of a computer program in plain text is an attractive service

4.4. HASH FUNCTION

- A variation on the message authentication code is the one way hash function. As with MAC, a hash function accepts a variable size message M as input and produces affixed-size output, referred to as hash code h**=H(M).**

- Unlike a MAC, a hash code does not use a key but is a function only of the input message. The hash code is also referred to as a **message digest or hash value.**

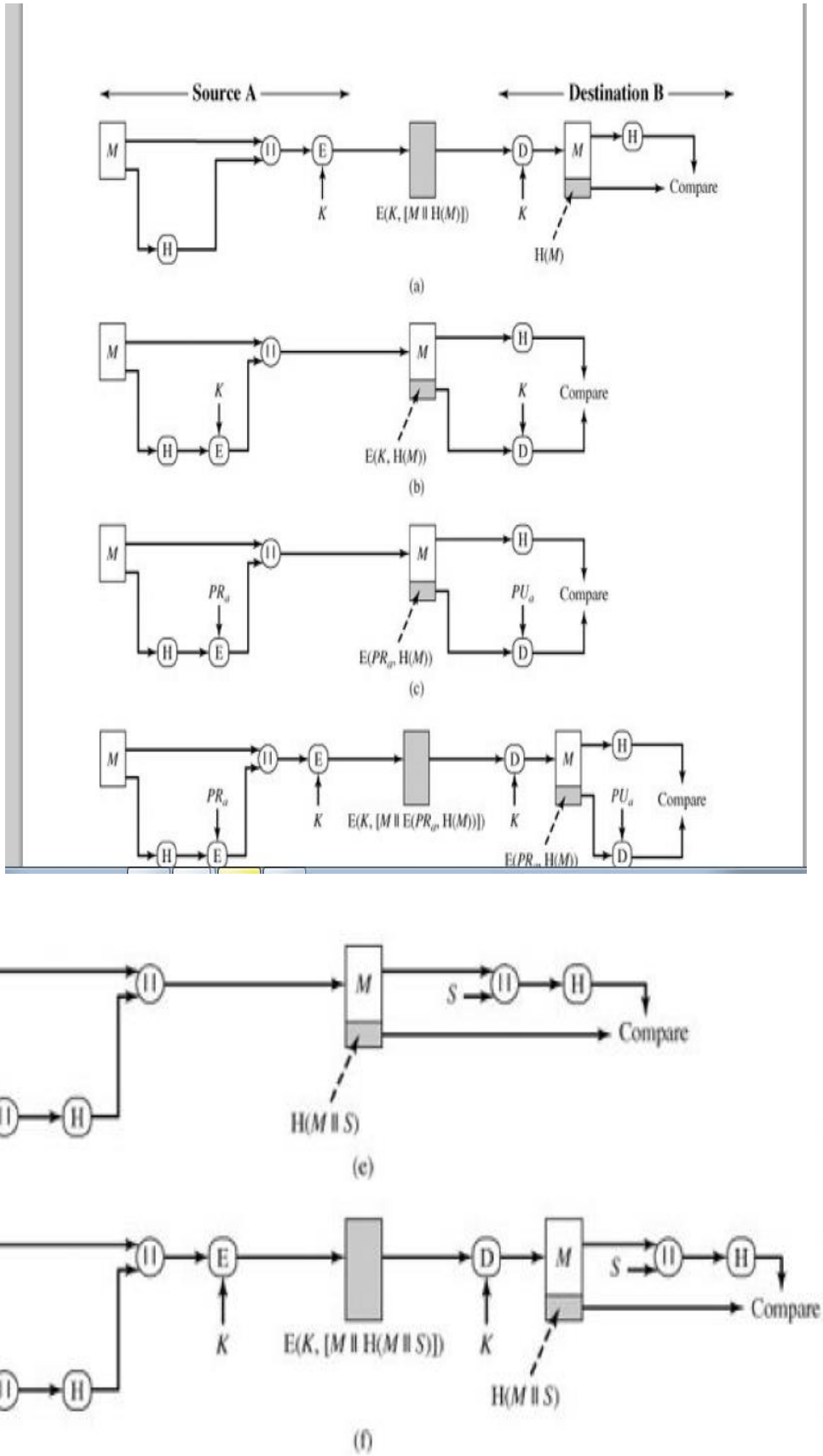


Fig .a. Encrypt message plus hash code

Sender:

- The sender creates a message using SHA to generate a 160 bit hashcode.
- The message and hashcode is concatenated and the result is encrypted using symmetric Encryption Algorithm.

Receiver:

- The receiver uses RSA (or) DSA algorithm to decrypt the message and recover hashcode.
- The receiver generates a new hashcode for the message and compare it with decrypted hashcode
- If two hashcodes are match ,the message is accepted,else it is rejected.

Fig .b. Encrypt hash code shared secret key

Sender:

- The sender creates a message using SHA to generate a 160 bit hashcode.
- Only the hash code is encrypted using Symmetric Encryption.
- The message and hash code encrypted and result is concatented

Receiver:

- The receiver uses RSA (or) DSA algorithm to decrypt the message and recover hashcode.
- The receiver generates a new hashcode for the message and compare it with decrypted hashcode
- If two hashcodes are match ,the message is accepted,else it is rejected.

Fig .c. Encrypt hash code sender's private key

Sender:

- The sender creates a message using SHA to generate a 160 bit hashcode.
- Only the hash code is encrypted using public key encryption and using the sender's private key.
- The message and hash code encrypted and result is concatented

Receiver:

- The receiver uses RSA (or) DSA algorithm to decrypt the message and recover hashcode.
- The receiver generates a new hashcode for the message and compare it with decrypted hashcode
- If two hashcodes are match ,the message is accepted,else it is rejected.

D) **Sender:**

- The sender creates a message using SHA to generate a 160 bit hashcode.
- Only the hash code is encrypted using public key encryption and using the sender's private key.
- The message and hash code encrypted and result is concatented

Receiver:

- The receiver uses RSA (or) DSA algorithm to decrypt the message and

recover hashcode.

- The receiver generates a new hashcode for the message and compare it with decrypted hashcode which uses the public key Encryption Algorithm of public key of sender.
- If two hashcodes are match ,the message is accepted,else it is rejected.

E) Sender:

- The sender creates a message M using SHA to generate a 160 bit hashcode.\
- This technique uses a hash fuction,but no encryption for message authentication
- This technique assumes that the two communicating parties share a common secret value 'S'.
- The source computes the hash value over the concatenation of M and S and appends the resulting hashvalue to M.

Receiver:

- The receiver uses RSA (or) DSA algorithm to decrypt the message and recover hashcode.
- The receiver generates a new hashcode for the message and compare it with decrypted hashcode which uses the public key Encryption Algorithm of public key of sender.
- The Message concatenated with hash value and 'S' is compared with receiver 's hash value.
- If two hashcodes are match ,the message is accepted,else it is rejected.

f) Confidentiality can be added to the previous approach by encrypting the entire message plus the hash code.

Requirements for a Hash Function

1. H can be applied to a block of data of any size.

2.H produces a fixed-length output.

3. H(x) is relatively easy to compute for any given x, making both hardware and Software implementations practical.

4. For any* given value h, it is computationally infeasible to find x such that H(x) =h. This is sometimes referred to in the literature as the one-way property.

5. For any given block x, it is computationally infeasible to find y x such that H(y) = H(x). This is

sometimes referred to as weak hash function.

6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y). This is sometimes referred to as **strong collision resistance**.

- ✓ The first three properties are requirements for the practical application of a hash function to message authentication.
- ✓ The fourth property, the one-way property, states that it is easy to generate a code given a message but virtually impossible to generate a message given a code.
- ✓ The fifth property guarantees that an alternative message hashing to the same value as a given message cannot be found.
- ✓ This prevents forgery when an encrypted hash code is used (Figures b and c).
- ✓ The sixth property refers to how resistant the hash function is to a type of attack known as the birthday attack, which we examine shortly.

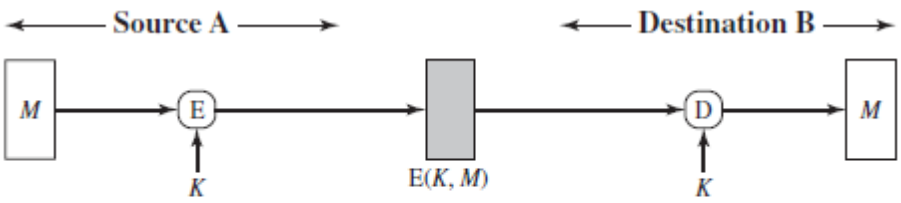
Message Encryption

- Message encryption by itself can provide a measure of authentication.
- The analysis differs for
  - **symmetric and**
  - **public-key encryption schemes**.

Symmetric Encryption

- Consider the straightforward use of symmetric encryption (Figure 12.1a). A message M transmitted from source A to destination B is encrypted using a secret key K shared by A and B. If no other party knows the key, then confidentiality is provided: No other party can recover the plaintext of the message.
- In addition, B is assured that the message was generated by A. Why? The message must have come from A, because A is the only other party that possesses K and therefore the only other party with the information necessary to construct ciphertext that can be decrypted with K.
- Furthermore, if M is recovered, B knows that none of the bits of M have been altered, because an opponent that does not know K would not know how to alter bits in the ciphertext to produce the desired changes in the plaintext.

Figure 12.1 Basic Uses of Message Encryption



(a) Symmetric encryption: confidentiality and authentication

- Thus, in general, we require that only a small subset of all possible bit patterns be considered legitimate plaintext. In that case, any spurious ciphertext is unlikely to produce legitimate plaintext.

- For example, suppose that only one bit pattern in 106 is legitimate plaintext. Then the probability that any randomly chosen bit pattern, treated as ciphertext, will produce a legitimate plaintext message is only 10-6. For a number of applications and encryption schemes, the desired conditions prevail as a matter of course.
- For example, suppose that we are transmitting Englishlanguage messages using a Caesar cipher with a shift of one (K = 1). A sends the following legitimate ciphertext:
    - `nbsftfbupbutboeepftfbupbutboemjuumfmbnctfbujwz`

  B decrypts to produce the following plaintext:

  `mareseatoatsanddoeseatoatsandlittlelambseativy`

- A simple frequency analysis confirms that this message has the profile of ordinary English. On the other hand, if an opponent generates the following random sequence of letters:

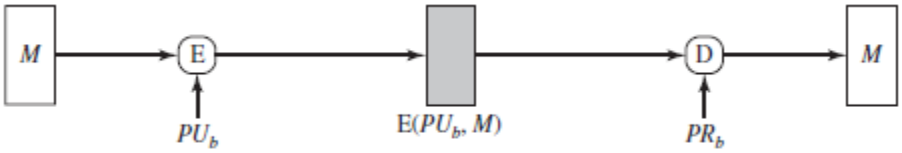
  `zuvrsoevgqxlzwigamdvnmhpmccxiuureosfbcebtqxsxq`

  this decrypts to

  `ytuqrndufpwkyvhfzlcumlgolbbwhttqdnreabdaspwrwp`

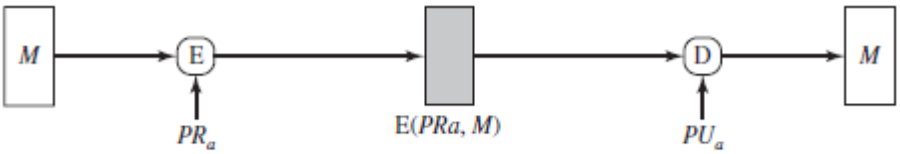
Public-Key Encryption

- The straightforward use of public-key encryption (Figure 12.1b) provides confidentiality but not authentication. The source (A) uses the public key PU_b of the destination (B) to encrypt M. Because only B has the corresponding private key PR_b, only B can decrypt the message. This scheme provides no authentication, because any opponent could also use B's public key to encrypt a message and claim to be A.

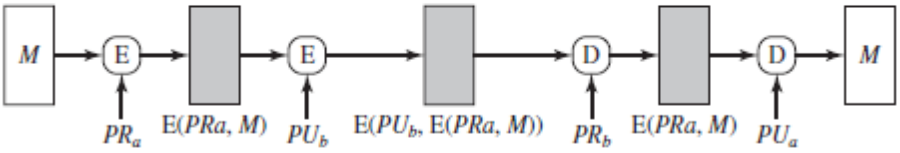


(b) Public-key encryption: confidentiality

- To provide authentication, A uses its private key to encrypt the message, and B uses A's public key to decrypt (Figure 12.1c). This provides authentication using the same type of reasoning as in the symmetric encryption case: The message must have come from A because A is the only party that possesses PR_a and therefore the only party with the information necessary to construct ciphertext that can be decrypted with PU_a.
- Again, the same reasoning as before applies: There must be some internal structure to the plaintext so that the receiver can distinguish between well-formed plaintext and random bits.



(c) Public-key encryption: authentication and signature



(d) Public-key encryption: confidentiality, authentication, and signature

4.5. SECURITY OF HASH FUNCTION AND MAC

Contents

• **Security of hash function and MAC**
• **Brute-Force Attacks**
  ○ ***Hash functions***
  ○ ***Message Authentication Code***.
• **Cryptanalysis**

Security of hash function and MAC

- We can group attacks on MACs into two categories: brute-force attacks and cryptanalysis.

Brute-Force Attacks

- A brute-force attack on a MAC is a more difficult undertaking than a brute-force attack on a hash function because it requires known message-tag pairs. Let us see why this is so. To attack a hash code, we can proceed in the following way.

Hash functions

- The strength of a hash function against brute-force attacks depends solely on the length of the hash code produced by the algorithm. Recall from our discussion of hash **functions that there are three desirable properties**:
  - **One-way**: For any given code *h*, it is computationally infeasible to find *x* such that H(*x*) = *h*.
  - **Weak collision resistance**: For any given block *x*, it is computationally infeasible to find *y x* with H(*y*) = H(*x*).
  - **Strong collision resistance**: It is computationally infeasible to find any pair (*x*, *y*) such that H (*x*) = H(*y*).
- For a hash code of length *n*, the level of effort required, as we have seen is proportional to the following

One way	2^n
Weak collision resistance	2^n
Strong collision resistance	$2^n/2$

Message Authentication Codes

A brute-force attack on a MAC is a more difficult undertaking because it requires known message-MAC pairs. Let us see why this is so.

To attack a hash code, we can proceed **in the following way**.

- Given a fixed message x with n-bit hash code $h = H(x)$, a brute-force method of finding a collision is to pick a random bit string y and check if $H(y) = H(x)$.
- The attacker can do this repeatedly off line. Whether an off-line attack can be used on a MAC algorithm depends on the relative size of the key and the MAC.
- To proceed, we need to state the desired security property of a MAC algorithm, which can be expressed as follows:

Computation resistance:

- Given one or more text-MAC pairs [xi, $C(K, xi)$], it is computationally infeasible to compute any text-MAC pair [x, $C(K, x)$] for any new input $x \neq xi$.
- The attacker would like to come up with the valid MAC code for a given message x.
- There are two lines of attack possible: Attack the key space and attack the MAC value
- If an attacker can determine the MAC key, then it is possible to generate a valid MAC value for any input x.
- Suppose the key size is k bits and that the attacker has one known text-MAC pair. Then the attacker can compute the n-bit MAC on the known text for all possible keys. At least one key is guaranteed to produce the correct MAC, namely, the valid key that was initially used to produce the known text-MAC pair. This phase of the attack takes a level of effort proportional to 2^k (that is, one operation for each of the 2^k possible key values).
- It can be shown that the level of effort drops off rapidly with each additional text-MAC pair and that the overall level of effort is roughly 2^k
- To summarize, the level of effort for brute-force attack on a MAC algorithm can be expressed as $\min(2^k, 2^n)$

Cryptanalysis

- The way to measure the resistance of a MAC algorithm to cryptanalysis is to compare its strength to the effort required for a bruteforce attack. That is, an ideal MAC algorithm will require a cryptanalytic effort greater than or equal to the brute-force effort.
- There is much more variety in the structure of MACs than in hash functions, so it is difficult to generalize about the cryptanalysis of MACs. Furthermore, far less work has been done on developing such attacks.

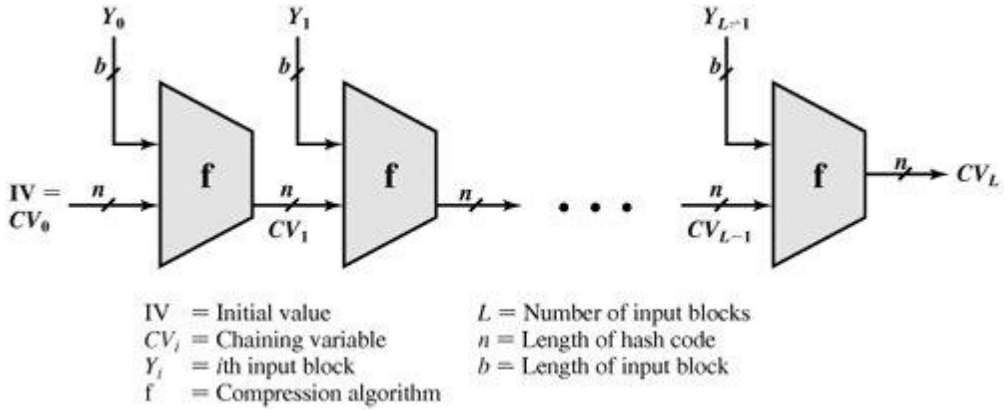


Figure 11.9. General Structure of Secure Hash Code

- The hash algorithm involves repeated use of a **compression function**, f, that takes two inputs (an *n*-bit input from the previous step, called the ***chaining variable***, and a *b*-bit block) and produces an *n*-bit output.
- At the start of hashing, the chaining variable has an initial value that is specified as part of the algorithm. The final value of the chaining variable is the hash value. Often, *b* > *n*; hence the term
- ***Compression***. The hash function can be summarized as follows:

$$CV_o = IV = \text{initial } n\text{-bit value}$$

$$CV_i = f(CV_{i1}, Y_{i1})\ 1 \leq i \leq L$$

$$H(M) = CV_L$$

- Where the input to the hash function is a message *M* consisting of the blocks *Y*o, *Y*1,..., *YL*1.

4.6. SHA

Contents
• **Secure Hash Algorithm(SHA)** ○ **SHA-512 Logic** ○ **SHA-512 Round Function**

Secure Hash Algorithm (SHA)

- SHA was developed by the Nation-bits foral Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993. When weaknesses were discovered in SHA, now known as **SHA-0**, a revised version was issued as FIPS 180-1 in 1995 and is referred to as **SHA-1**.

Table 11.3 Comparison of SHA Parameters

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message Digest Size	160	224	256	384	512
Message Size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

- The actual standards document is entitled "Secure Hash Standard." SHA is based on the hash function MD4, and its design closely models MD4. SHA-1 produces a hash value of 160 bits.
- NIST produced a revised version of the standard, FIPS 180-2, that defined three new versions of SHA, with hash value lengths of 256, 384, and 512 bits, known as SHA-256, SHA-384, and SHA-512, respectively.
- Collectively, these hash algorithms are known as **SHA-2**. These new versions have the same underlying structure and use the same types of modular arithmetic and logical binary operations as SHA-1. A revised document was issued as FIP PUB 180-3 in 2008, which added a 224-bit version (Table 11.3).

SHA-512 Logic

- The algorithm takes as input a message with a maximum length of less than 2128 bits and produces as output a 512-bit message digest. The input is processed in 1024-bit blocks.
- Figure 11.9 depicts the overall processing of a message to produce a digest. This follows the general structure depicted in Figure 11.8. The processing consists of the following steps.

  - **Step 1 Append padding bits.** The message is padded so that its length is congruent to 896 modulo 1024 [length K 896(mod 1024)]. Padding is always added, even if the message is already of the desired length. Thus, the number of padding bits is in the range of 1 to 1024. The padding consists of a single 1 bit followed by the necessary number of 0 bits.
  - **Step 2 Append length.** A block of 128 bits is appended to the message. This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the padding).

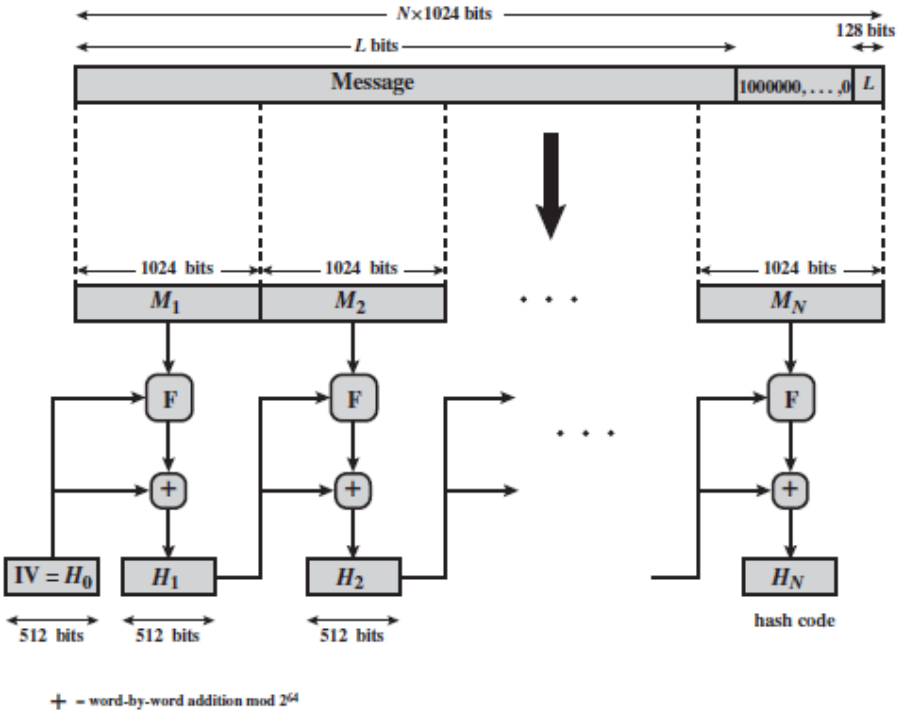


Figure 11.9 Message Digest Generation Using SHA-512

- The outcome of the first two steps yields a message that is an integer multiple of 1024 bits in length. In Figure 11.9, the expanded message is represented as the sequence of 1024-bit blocks *M*1, *M*2, c, *MN*, so that the total length of the expanded message is *N* * 1024 bits.

❖ **Step 3 Initialize hash buffer.** A 512-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as eight 64-bit registers (a, b, c, d, e, f, g, h). These registers are initialized to the following 64-bit integers (hexadecimal values):

```
a = 6A09E667F3BCC908    e = 510E527FADE682D1
b = BB67AE8584CAA73B    f = 9B05688C2B3E6C1F
c = 3C6EF372FE94F82B    g = 1F83D9ABFB41BD6B
d = A54FF53A5F1D36F1    h = 5BE0CD19137E2179
```

- These values are stored in **big-endian** format, which is the most significant byte of a word in the low-address (leftmost) byte position. These words were obtained by taking the first sixty-four bits of the fractional parts of the square roots of the first eight prime numbers.

❖ **Step 4 Process message in 1024-bit (128-word) blocks.** The heart of the algorithm is a module that consists of 80 rounds; this module is labeled F in Figure 11.9. The logic is illustrated in Figure 11.10.

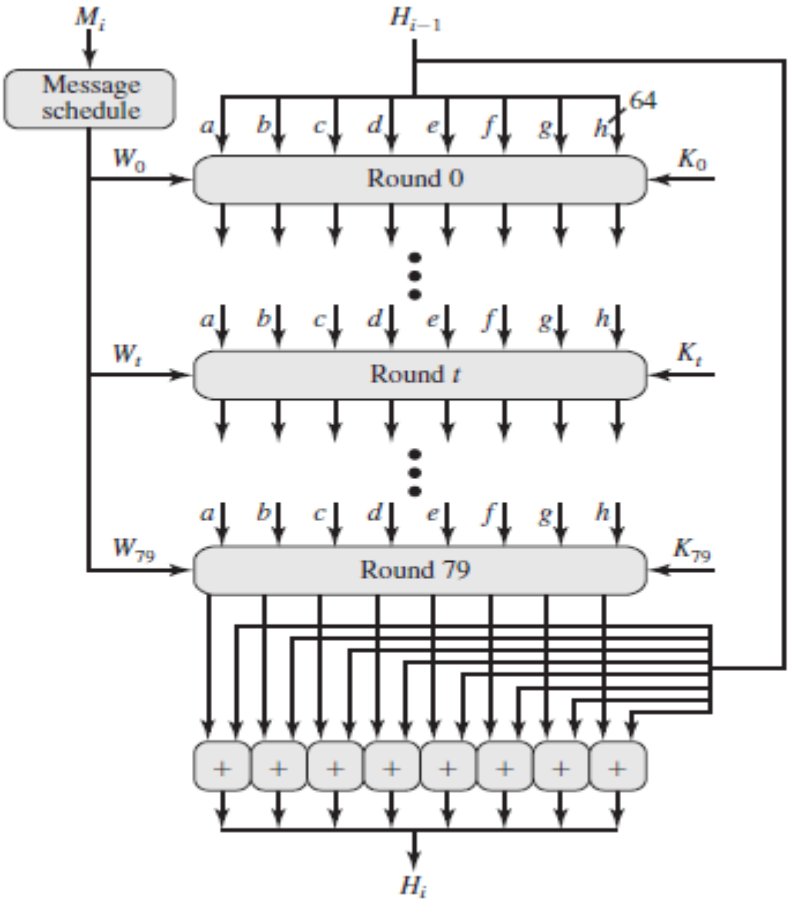


Figure 11.10 SHA-512 Processing of a Single 1024-Bit Block

- Each round takes as input the 512-bit buffer value, abcdefgh, and updates the contents of the buffer. At input to the first round, the buffer has the value of the intermediate hash value, *Hi*-1.
- Each round *t* makes use of a 64-bit value *Wt*, derived from the current 1024-bit block being processed (*Mi*). These values are derived using a message schedule described subsequently.Each round also makes use of an additive constant *Kt*, where 0 … *t* … 79 indicates one of the 80 rounds.

❖ **Step 5 Output.** After all *N* 1024-bit blocks have been processed, the output from the *N*th stage is the 512-bit message digest. We can summarize the behavior of SHA-512 as follows:

$$H_0 = \text{IV}$$
$$H_i = \text{SUM}_{64}(H_{i-1}, \text{abcdefgh}_i)$$
$$MD = H_N$$

where

IV----------→ initial value of the abcdefgh buffer, defined in step 3

abcdefgh_i --→ the output of the last round of processing of the *i*th message block

N ----------→ the number of blocks in the message (including padding and length fields)

SUM64 -----→addition modulo 264

MD ----------→final message digest value

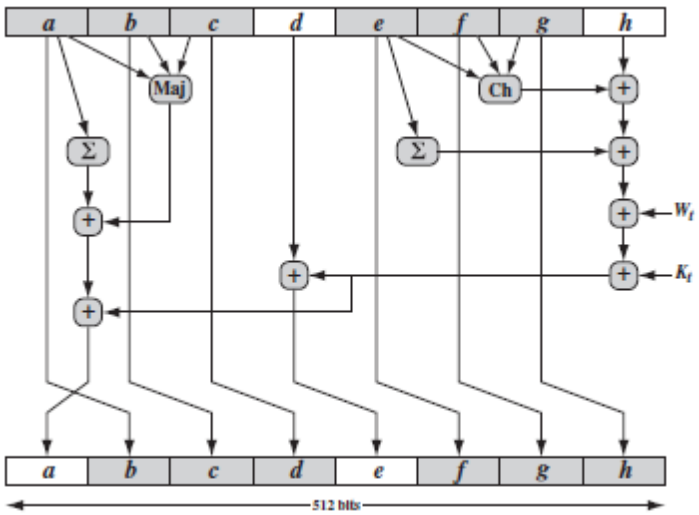


Figure 11.11 Elementary SHA-512 Operation (single round)

SHA-512 Round Function

Let us look in more detail at the logic in each of the 80 steps of the processing of one 512-bit block (Figure 11.11). Each round is defined by the following set of equations:

$$T_1 = h + \text{Ch}(e, f, g) + \left(\sum_1^{512} e\right) + W_t + K_t$$
$$T_2 = \left(\sum_0^{512} a\right) + \text{Maj}(a, b, c)$$
$$h = g$$
$$g = f$$
$$f = e$$
$$e = d + T_1$$
$$d = c$$
$$c = b$$
$$b = a$$
$$a = T_1 + T_2$$

Where

t = step number; $0 \leq t \leq 79$

$\text{Ch}(e, f, g)$ = $(e \text{ AND } f) \oplus (\text{NOT } e \text{ AND } g)$
the conditional function: If e then f else g

$\text{Maj}(a, b, c)$ = $(a \text{ AND } b) \oplus (a \text{ AND } c) \oplus (b \text{ AND } c)$
the function is true only of the majority (two or three) of the arguments are true

$\left(\sum_0^{512} a\right)$ = $\text{ROTR}^{28}(a) \oplus \text{ROTR}^{34}(a) \oplus \text{ROTR}^{39}(a)$

$\left(\sum_1^{512} e\right)$ = $\text{ROTR}^{14}(e) \oplus \text{ROTR}^{18}(e) \oplus \text{ROTR}^{41}(e)$

$\text{ROTR}^n(x)$ = circular right shift (rotation) of the 64-bit argument x by n bits

W_t = a 64-bit word derived from the current 1024-bit input block

K_t = a 64-bit additive constant

$+$ = addition modulo 2^{64}

Two observations can be made about the round function.

1. Six of the eight words of the output of the round function involve simply permutation (b, c, d, f , g, h) by means of rotation. This is indicated by shading in Figure 11.11.

2. Only two of the output words (a, e) are generated by substitution. Word e is a function of input variables (d, e, f, g, h), as well as the round word Wt and the constant Kt. Word a is a function of all of the input variables except d, as well as the round word Wt and the constant Kt. It remains to indicate how the 64-bit word values Wt are derived from the 1024-bit message. The remaining values are defined as

$$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$$

where

$$\sigma_0^{512}(x) = \text{ROTR}^1(x) \oplus \text{ROTR}^8(x) \oplus \text{SHR}^7(x)$$
$$\sigma_1^{512}(x) = \text{ROTR}^{19}(x) \oplus \text{ROTR}^{61}(x) \oplus \text{SHR}^6(x)$$
$$\text{ROTR}^n(x) = \text{circular right shift (rotation) of the 64-bit argument } x \text{ by } n \text{ bits}$$
$$\text{SHR}^n(x) = \text{left shift of the 64-bit argument } x \text{ by } n \text{ bits with padding by zeros on the right}$$
$$+ = \text{addition modulo } 2^{64}$$

- Thus, in the first 16 steps of processing, the value of Wt is equal to the corresponding word in the message block. For the remaining 64 steps, the value of Wt consists of the circular left shift by one bit of the XOR of four of the preceding values of Wt, with two of those values subjected to shift and rotate operations.
- This introduces a great deal of redundancy and interdependence into the message blocks that are compressed, which complicates the task of finding a different message block that maps to the same compression function output

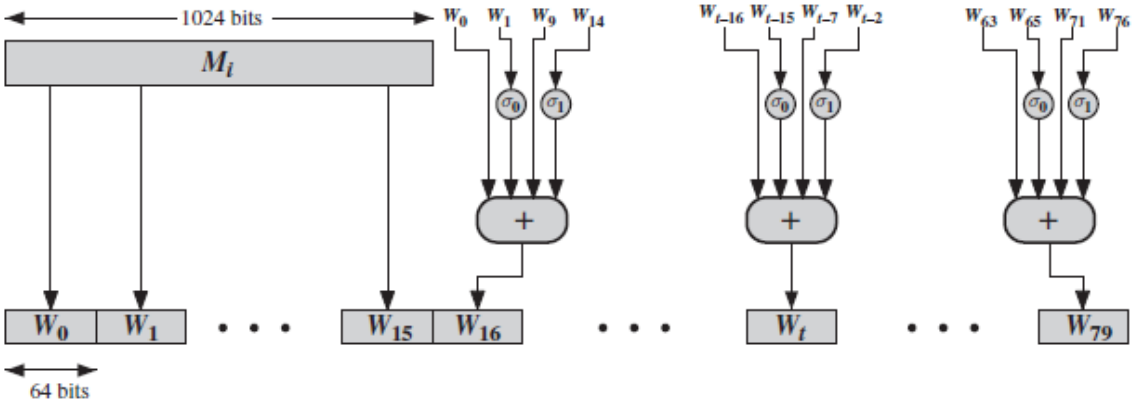


Figure 11.12 Creation of 80-word Input Sequence for SHA-512 Processing of Single Block

4.7. DIGITAL SIGNATURE

Contents
• **Digital Signatures** ○ Properties ○ Attacks and Forgeries ○ Digital Signature Requirements ○ Direct Digital Signature

Digital Signatures

- A digital signature is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature. The signature is formed by taking the hash of the message and encrypting the message with the creators private key

Properties

- Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each other. Several forms of dispute between the two are possible.
- For example, suppose that John sends an authenticated message to Mary, using one of the schemes of Figure 12.1. Consider the following disputes that could arise.

  1. Mary may forge a different message and claim that it came from John. Mary would simply have to create a message and append an authentication code using the key that John and Mary share.

  2. John can deny sending the message. Because it is possible for Mary to forge a message, there is no way to prove that John did in fact send the message.

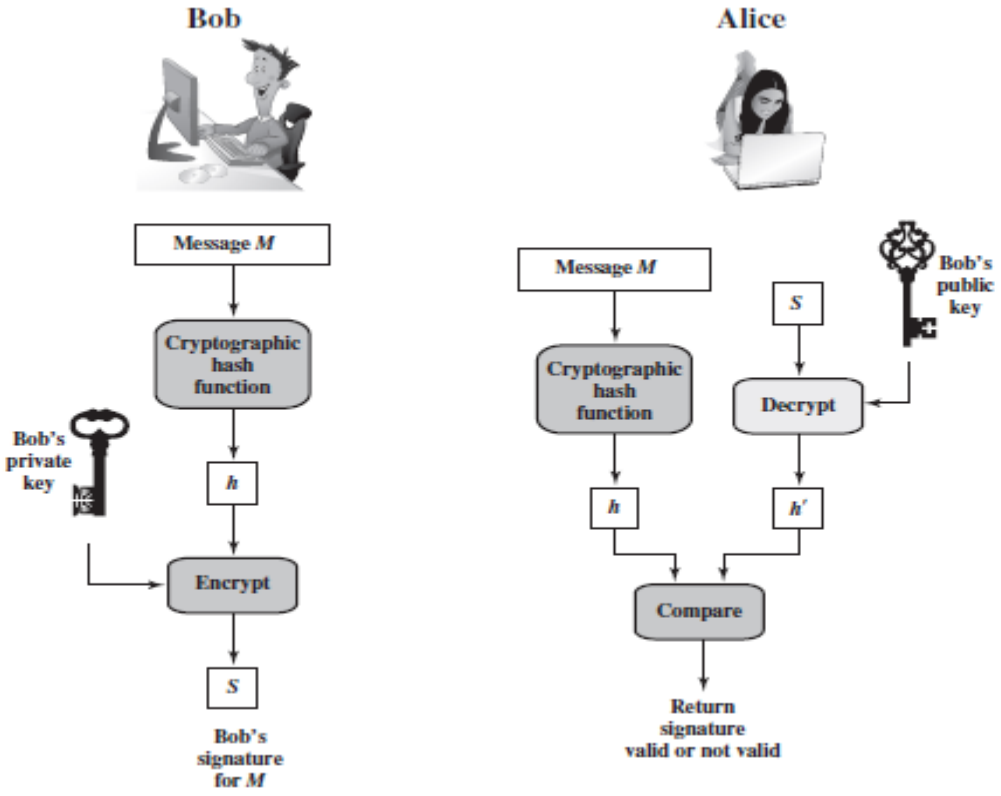


Figure 13.2 Simplified Depiction of Essential Elements of Digital Signature Process

- Both scenarios are of legitimate concern. Here is an example of the first scenario: An electronic funds transfer takes place, and the receiver increases the amount of funds transferred and claims that the larger amount had arrived from the sender.
- An example of the second scenario is that an electronic mail message contains instructions to a stockbroker for a transaction that subsequently turns out badly. The sender pretends that the message was never sent.
- In situations where there is not complete trust between sender and receiver, something more than authentication is needed. The most attractive solution to this problem is the digital signature.
- The digital signature must have the following properties:
  - It must verify the author and the date and time of the signature.
  - It must authenticate the contents at the time of the signature.
  - It must be verifiable by third parties, to resolve disputes.

Thus, the digital signature function includes the authentication function.

Attacks and Forgeries

- [GOLD88] lists the following types of attacks, in order of increasing severity. Here A denotes the user whose signature method is being attacked, and C denotes the attacker.
  • **Key-only attack:** C only knows A's public key.
  • **Known message attack:** C is given access to a set of messages and their signatures.
  • **Generic chosen message attack:** C chooses a list of messages before attempting to breaks A's signature scheme, independent of A's public key. C then obtains from A valid signatures for the chosen messages.
  • **Directed chosen message attack:** Similar to the generic attack, except that the list of messages to be signed is chosen after C knows A's public key but before any signatures are seen.
  • **Adaptive chosen message attack:** C is allowed to use A as an "oracle." This means that C may request from A signatures of messages that depend on previously obtained message-signature pairs.

- [GOLD88] then defines success at breaking a signature scheme as an outcome in which C can do any of the following with a non-negligible probability:
  • **Total break:** C determines A's private key.
  • **Universal forgery:** C finds an efficient signing algorithm that provides an equivalent way of constructing signatures on arbitrary messages.
  • **Selective forgery:** C forges a signature for a particular message chosen by C.
  • **Existential forgery:** C forges a signature for at least one message. C has no control over the message. Consequently, this forgery may only be a minor nuisance to A.

Digital Signature Requirements

- On the basis of the properties and attacks just discussed, we can formulate the following requirements for a digital signature.
  • The signature must be a bit pattern that depends on the message being signed.
  • The signature must use some information unique to the sender to prevent both forgery and denial.
  • It must be relatively easy to produce the digital signature.
  • It must be relatively easy to recognize and verify the digital signature.
  • It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
  • It must be practical to retain a copy of the digital signature in storage.
- **Two general schemes for digital signatures**
  - Direct Digital Signature
  - Arbitrated Digital Signature

Direct Digital Signature

- The term **direct digital signature** refers to a digital signature scheme that involves only the communicating parties (source, destination). It is assumed that the destination knows the public key of the source.
- Confidentiality can be provided by encrypting the entire message plus signature with a shared secret key (symmetric encryption). Note that it is important to perform the signature function first and then an outer confidentiality function.
- In case of dispute, some third party must view the message and its signature. If the signature is calculated on an encrypted message, then the third party also needs access to the decryption key to read the original message. However, if the signature is the inner

operation, then the recipient can store the plaintext message and its signature for later use in dispute resolution.

- The validity of the scheme just described depends on the security of the sender's private key. If a sender later wishes to deny sending a particular message, the sender can claim that the private key was lost or stolen and that someone else forged his or her signature.
- Administrative controls relating to the security of private keys can be employed to thwart or at least weaken this ploy, but the threat is still there, at least to some degree. One example is to require every signed message to include a **timestamp** (date and time) and to require prompt reporting of compromised keys to a central authority.
- Another threat is that some private key might actually be stolen from X at time T. The opponent can then send a message signed with X's signature and stamped with a time before or equal to T.

Arbitrated Digital Signature

- The problems associated with direct digital signatures can be addressed by using an arbiter.
- As with direct signature schemes, there is a variety of arbitrated signature schemes.

In general terms, they all operate as follows.

- Every signed message from a sender X to a receiver Y goes first to an arbiter A, who subjects the message and its signature to a number of tests to check its origin and content.
- The message is then dated and sent to Y with an indication that it has been verified to the satisfaction of the arbiter. The presence of A solves the problem faced by direct signature schemes: that X might disown the message.
- The arbiter plays a sensitive and crucial role in this sort of scheme, and all parties must have a great deal of trust that the arbitration mechanism is working properly Table 13.1, based on scenarios described in [AKL83] and [MITC92], gives several examples of arbitrated digital signatures.
- In the first, symmetric encryption is used. It is assumed that the sender X and the arbiter A share a secret key *Kxa* and that A and Y share secret key *Kay*. X constructs a message *M* and computes its hash value H(*M*).
- Then X transmits the message plus a signature to A. The signature consists of an identifier *IDX* of X plus the hash value, all encrypted using *Kxa*. A decrypts the signature and checks the hash value to validate the message.
- Then A transmits a message to Y, encrypted with *Kay*. The message includes *IDX*, the original message from X, the signature, and a timestamp. Y can decrypt this to recover the message and the signature. The timestamp informs Y that this message is timely and not a replay. Y can store *M* and the signature. In case of dispute, Y, who claims to have received *M* from X, sends the following message to A:

- The arbiter uses *Kay* to recover *IDX*, *M*, and the signature, and then uses *Kxa* to decrypt the signature and verify the hash code. In this scheme, Y cannot directly check X's signature; the signature is there solely to settle disputes. Y considers the message from X authentic because it comes through A.

In this scenario, both sides must have a high degree of trust in A:

  - X must trust A not to reveal *Kxa* and not to generate false signatures of the form E(*Kxa*, [*IDX*||H (*M*)]).
  - Y must trust A to send E(*Kay*, [*IDX*||*M*||E(*Kxa*, [*IDX*||H(*M*)])||*T*]) only if the hash value is correct and the signature was generated by X.

- Both sides must trust A to resolve disputes fairly.
- If the arbiter does live up to this trust, then X is assured that no one can forge his signature and Y is assured that X cannot disavow his signature.

4.8. AUTHENTICATION PROTOCOLS

Contents
• **Authentication Protocols** o Mutual Authentication o One-Way Authentication

Authentication Protocols

- Authentication Protocols are used to convince parties of each others identity and to exchange session keys.they may be **(mutual authentication and one-way authentication)**

Mutual Authentication

- **Symmetric Encryption Approaches**
- **Public-Key Encryption Approaches**

- An important application area is that of mutual authentication protocols. Such protocols enable communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys.
- Central to the problem of authenticated key exchange are two issues: **confidentiality and timeliness.**
- To prevent masquerade and to prevent compromise of session keys, essential identification and session key information must be communicated in encrypted form.
- This requires the prior existence of secret or public keys that can be used for this purpose. The second issue, timeliness, is important because of the threat of message replays. Such replays, at worst, could allow an opponent to compromise a session key or successfully impersonate another party.
- At minimum, a successful replay can disrupt operations by presenting parties with messages that appear genuine but are not.

Lists the following examples of replay attacks:

- **Simple replay:** The opponent simply copies a message and replays it later.
- **Repetition that can be logged:** An opponent can replay a timestamped message within the valid time window.
- **Repetition that cannot be detected:** This situation could arise because the original message could have been suppressed and thus did not arrive at its destination; only the replay message arrives.
- **Backward replay without modification:** This is a replay back to the message sender. This attack is possible if symmetric encryption is used and the sender cannot easily recognize the difference between messages sent and messages received on the basis of content.

- **The following two general approaches is used:**

  - **Timestamps:** Party A accepts a message as fresh only if the message contains a timestamp that, in A's judgment, is close enough to A's knowledge of current time. This approach requires that clocks among the various participants be synchronized.

- **Challenge/response:** Party A, expecting a fresh message from B, first sends B a nonce (challenge) and requires that the subsequent message (response) received from B contain the correct nonce value. It can be argued.

Symmetric Encryption Approaches

- A two-level hierarchy of symmetric encryption keys can be used to provide confidentiality for communication in a distributed environment. In general, this strategy involves the use of a trusted key distribution center (KDC).
- Each party in the network shares a secret key, known as a master key, with the KDC. The KDC is responsible for generating keys to be used for a short time over a connection between two parties, known as session keys, and for distributing those keys using the master keys to protect the distribution.

1. A $\longrightarrow$ KDC: $ID_A||ID_B||N_1$
2. KDC $\longrightarrow$ A: $E(K_a, [K_s||ID_B||N_1||E(K_b, [K_s||ID_A])])$
3. A $\longrightarrow$ B: $E(K_b, [K_s||ID_A])$
4. A $\longrightarrow$ A: $E(K_s, N_2)$
5. A $\longrightarrow$ B: $E(K_s, f(N_2))$

- In step 1.Secret keys *Ka* and *Kb* are shared between A and the KDC and B and the KDC, respectively. The purpose of the protocol is to distribute securely a session key *Ks* to A and B. A securely acquires a new session key in step 2.
- The message in step 3 can be decrypted, and hence understood, only by B. Step 4 reflects B's knowledge of *Ks*, and step 5 assures B of A's knowledge of *Ks* and assures B that this is a fresh message because of the use of the nonce *N*2 that the purpose of steps 4 and 5 is to prevent a certain type of replay attack.

Suppress-replay attacks

- One way to counter suppress-replay attacks is to enforce the requirement that parties regularly check their clocks against the KDC's clock. The other alternative, which avoids the need for clock synchronization, is to rely on handshaking protocols using nonces.
- This latter alternative is not vulnerable to a suppress-replay attack because the nonces the recipient will choose in the future are unpredictable to the sender. The Needham/Schroeder protocol relies on nonces only but, as we have seen, has other vulnerabilities.

The protocol is as follows:

1. A ⟶ B: $ID_A||N_a$

2. B ⟶ KDC: $ID_B||N_b||E(K_b, [ID_A||N_a||T_b])$

3. KDC ⟶ A: $E(K_a, [ID_B||N_a||K_s||T_b])||E(K_b,[ID_A||K_s||T_b])||N_b$

4. A ⟶ B: $E(K_b, [ID_A||K_s||T_b])||E(K_s, N_b)$

Let us follow this exchange step by step.

1. A initiates the authentication exchange by generating a nonce, *Na*, and sending that plus its identifier to B in plaintext. This nonce will be returned to A in an encrypted message that includes the session key, assuring A of its timeliness.

2. B alerts the KDC that a session key is needed. Its message to the KDC includes its identifier and a nonce, *Nb* This nonce will be returned to B in an encrypted message that includes the session key, assuring B of its timeliness. B's message to the KDC also includes a block encrypted with the secret key shared by B and the KDC. This block is used to instruct the KDC to issue credentials to A; the block specifies the intended recipient of the credentials, a suggested expiration time for the credentials, and the nonce received from A.

3. The KDC passes on to A B's nonce and a block encrypted with the secret key that B shares with the KDC. The block serves as a "ticket" that can be used by A for subsequent authentications, as will be seen. The KDC also sends to A block encrypted with the secret key shared by A and the KDC. This block verifies that B has received A's initial message (*IDB*) and that this is a timely message and not a replay (*Na*) and it provides A with a session key (*Ks*) and the time limit on its use (*Tb*).

4. A transmits the ticket to B, together with the B's nonce, the latter encrypted with the session key. The ticket provides B with the secret key that is used to decrypt E(*Ks*, *Nb*) to recover the nonce. The fact that B's nonce is encrypted with the session key authenticates that the message came from A and is not a replay.

Public-Key Encryption Approaches

- This protocol assumes that each of the two parties is in possession of the current public key of the other.

A protocol using timestamps is provided in

1. A ⟶ AS: $ID_A||ID_B$

2. AS ⟶ A: $E(PR_{as}, [ID_A||PU_a||T])||E(PR_{as}, [ID_B||PU_b||T])$

3. A ⟶ B: $E(PR_{as}, [ID_A||PU_a||T])||E(PR_{as}, [ID_B||PU_b||T])$
$||E(PU_b, E(PR_a, [K_s||T]))$

- In this case, the central system is referred to as an authentication server (AS), because it is not actually responsible for secret key distribution. Rather, the AS provides public-key certificates. The session key is chosen and encrypted by A; hence, there is no risk of exposure by the AS.

- The timestamps protect against replays of compromised keys. This protocol is compact but, as before, requires synchronization of clocks. Another approach, proposed by Woo and Lam [WOO92a], makes use of nonces.

The protocol consists of the following steps:

1. A $\longrightarrow$ KDC: $ID_A||ID_B$
2. KDC $\longrightarrow$ A: $E(PR_{auth}, [ID_B||PU_b])$
3. A $\longrightarrow$ B: $E(PU_b, [N_a||ID_A])$
4. B $\longrightarrow$ KDC: $ID_A||ID_B||E(PU_{auth}, N_a)$
5. KDC $\longrightarrow$ B: $E(PR_{auth}, [ID_A||PU_a])||E(PU_b, E(PR_{auth}, [N_a||K_s||ID_B]))$
6. B $\longrightarrow$ A: $E(PU_a, E(PR_{auth}, [(N_a||K_s||ID_B)||N_b]))$
7. A $\longrightarrow$ B: $E(K_s, N_b)$

- In step 1, A informs the KDC of its intention to establish a secure connection with B. The KDC returns to A. a copy of B's public-key certificate (step 2). Using B's public key, A informs B of its desire to communicate and sends a nonce *Na* (step 3). In step 4, B asks the KDC for A's public-key certificate and requests a session key; B includes A's nonce so that the KDC can stamp the session key with that nonce.
- The nonce is protected using the KDC's public key. In step 5, the KDC returns to B a copy of A's public-key certificate, plus the information {*Na*, *Ks*, *IDB*}. This information basically says that *Ks* is a secret key generated by the KDC on behalf of B and tied to *Na*; the binding of *Ks* and *Na* will assure A that *Ks* is fresh. This triple is encrypted, using the KDC's private key, to allow B to verify that the triple is in fact from the KDC.
- It is also encrypted using B's public key, so that no other entity may use the triple in an attempt to establish a fraudulent connection with A. In step 6, the triple {*Na, Ks, IDB*}, still encrypted with the KDC's private key, is relayed to A, together with a nonce *Nb* generated by B. All the foregoing are encrypted using A's public key.
- A retrieves the session key *Ks* and uses it to encrypt *Nb* and return it to B. This last message assures B of A's knowledge of the session key. This seems to be a secure protocol that takes into account the various attacks.
- However, the authors themselves spotted a flaw and submitted a revised version of the algorithm in [WOO92b]:

1. A $\longrightarrow$ KDC: $ID_A||ID_B$

2. KDC ⟶ A: $E(PR_{auth}, [ID_B||PU_b])$

3. A ⟶ B: $E(PU_b, [N_a||ID_A])$

4. B ⟶ KDC: $ID_A||ID_B||E(PU_{auth}, N_a)$

5. KDC ⟶ B: $E(PR_{auth}, [ID_A||PU_a])||E(PU_b, E(PR_{auth}, [N_a||K_s||ID_A||ID_B])$

6. B ⟶ A: $E(PU_a, E(PR_{auth}, [(N_a||K_s||ID_A||ID_B)||N_b]))$

7. A ⟶ B: $E(K_s, N_b)$

- The identifier of A, *IDA*, is added to the set of items encrypted with the KDC's private key in steps 5 and 6. This binds the session key *Ks* to the identities of the two parties that will be engaged in the session.
- This inclusion of *IDA* accounts for the fact that the nonce value *Na* is considered unique only among all nonces generated by A, not among all nonces generated by all parties. Thus, it is the pair {*IDA*, *Na*} that uniquely identifies the connection request of A.

One-Way Authentication

- One application for which encryption is growing in popularity is electronic mail (e-mail). The very nature of electronic mail, and its chief benefit, is that it is not necessary for the sender and receiver to be online at the same time. Instead, the e-mail message is forwarded to the receiver's electronic mailbox, where it is buffered until the receiver is available to read it.
- The "envelope" or header of the e-mail message must be in the clear, so that the message can be handled by the store-and-forward e-mail protocol, such as the Simple Mail Transfer Protocol (SMTP) or X.400. However, it is often desirable that the mail-handling protocol not require access to the plaintext form of the message, because that would require trusting the mail-handling mechanism. Accordingly, the e-mail message should be encrypted such that the mail-handling system is not in possession of the decryption key.
- A second requirement is that of authentication. Typically, the recipient wants some assurance that the message is from the alleged sender.

Symmetric Encryption Approach

- Using symmetric encryption, the decentralized key distribution scenario illustrated in Figure 7.11 is impractical. This scheme requires the sender to issue a request to the intended recipient, await a response that includes a session key, and only then send the message.
- With some refinement, the KDC strategy illustrated in Figure 7.9 is a candidate for encrypted electronic mail. Because we wish to avoid requiring that the recipient (B) be on line at the same time as the sender (A), steps 4 and 5 must be eliminated. For a message with content *M*, the sequence is as follows:

1. A ⟶ KDC: $ID_A||ID_B||N_1$

2. KDC ⟶ A: $E(K_a, [K_s||ID_B||N_1||E(K_b, [K_s||ID_A])])$

3. A ⟶ B: $E(K_b, [K_s||ID_A])||E(K_s, M)$

- This approach guarantees that only the intended recipient of a message will be able to read it. It also provides a level of authentication that the sender is A. As specified, the protocol does not protect against replays. Some measure of defense could be provided by including a timestamp with the message.
- However, because of the potential delays in the e-mail process, such timestamps may have limited usefulness.

Public-Key Encryption Approaches

- We have already presented public-key encryption approaches that are suited to electronic mail, including the straightforward encryption of the entire message for confidentiality, authentication or both
- These approaches require that either the sender know the recipient's public key (confidentiality) or the recipient know the sender's public key (authentication) or both (confidentiality plus authentication). In addition, the public-key algorithm must be applied once or twice to what may be a long message.
- If confidentiality is the primary concern, then the following may be more efficient:

$$A \longrightarrow B: E(PU_b, K_s)||E(K_s, M)$$

- In this case, the message is encrypted with a one-time secret key. A also encrypts this one-time key with B's public key. Only B will be able to use the corresponding private key to recover the one-time key and then use that key to decrypt the message. This scheme is more efficient than simply encrypting the entire message with B's public key.
- If authentication is the primary concern, then a digital signature may suffice, as was illustrated in

$$A \longrightarrow B: M||E(PR_a, H(M))$$

- This method guarantees that A cannot later deny having sent the message. However, this technique is open to another kind of fraud. Bob composes a message to his boss Alice that contains an idea that will save the company money. He appends his digital signature and sends it into the e-mail system.
- Eventually, the message will get delivered to Alice's mailbox. But suppose that Max has heard of Bob's idea and gains access to the mail queue before delivery. He finds Bob's message, strips off his signature, appends his, and requeues the message to be delivered to Alice. Max gets credit for Bob's idea.
- To counter such a scheme, both the message and signature can be encrypted with the recipient's public key:

$$A \longrightarrow B: E(PU_b, [M||E(PR_a, H(M))])$$

4.9. DIGITAL SIGNATURE STANDARD

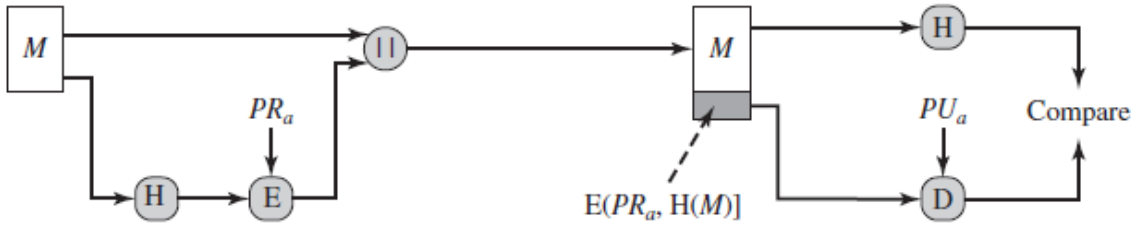
Contents
• **Digital Signatures Standard** ○ The DSS Approach ○ The Digital Signature Algorithm

Digital Signatures Standard

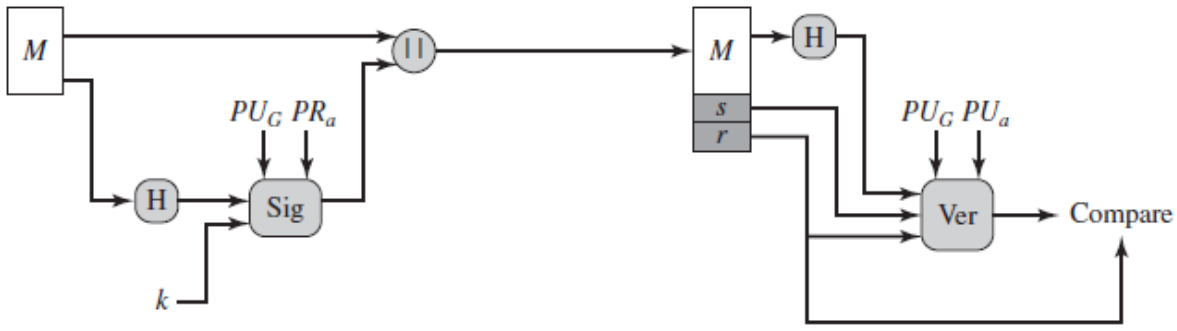
- The DSS makes use of the Secure Hash Algorithm (SHA) presents a new digital signature technique, the **Digital Signature Algorithm (DSA)**.

The DSS Approach

- The DSS uses an algorithm that is designed to provide only the digital signature function. Unlike RSA, it cannot be used for encryption or key exchange. Nevertheless, it is a public-key technique.
- Figure 13.3 contrasts the DSS approach for generating digital signatures to that used with RSA. In the RSA approach, the message to be signed is input to a hash function that produces a secure hash code of fixed length. This hash code is then encrypted using the sender's private key to form the signature.
- The DSS approach also makes use of a hash function. The hash code is provided as input to a signature function along with a random number generated for this particular signature.
- The signature function also depends on the sender's private key and a set of parameters known to a group of communicating principals. We can consider this set to constitute a global public key .1 The result is a signature consisting of two components, labeled *s* and *r*.



(a) RSA approach



(b) DSS approach

Figure 13.3 Two Approaches to Digital Signatures

- At the receiving end, the hash code of the incoming message is generated. This plus the signature is input to a verification function. The verification function also depends on the global public key as well as the sender's public key , which is paired with the sender's private key.
- The output of the verification function is a value that is equal to the signature component if the signature is valid. The signature function is such that only the sender, with knowledge of the private key, could have produced the valid signature. We turn now to the details of the algorithm.

The Digital Signature Algorithm

- Figure 13.4 summarizes the algorithm. There are three parameters that are public and can be common to a group of users. A 160-bit prime number is chosen. Next, a prime number is selected with a length between 512 and 1024 bits such that divides (*p* - 1).
- Finally, *g* is chosen to be of the form *h*(*p*-1)/*q*mod *p*, where *h* is an*p q q r* (*PUa*) 2In number-theoretic terms, g is of order q modp; see Chapter 8. integer between 1 and with the restriction that must be greater than 1.2
- Thus, the global public-key components of DSA have the same for as in the Schnorr signature scheme. With these numbers in hand, each user selects a private key and generates a public key. The private key must be a number from 1 to and should be chosen randomly or pseudorandomly.
- The public key is calculated from the private key as .The calculation of given is relatively straightforward. However, given the public key, it is believed to be computationally infeasible to determine , which is the discrete logarithm of *y* to the base *g*, mod p.

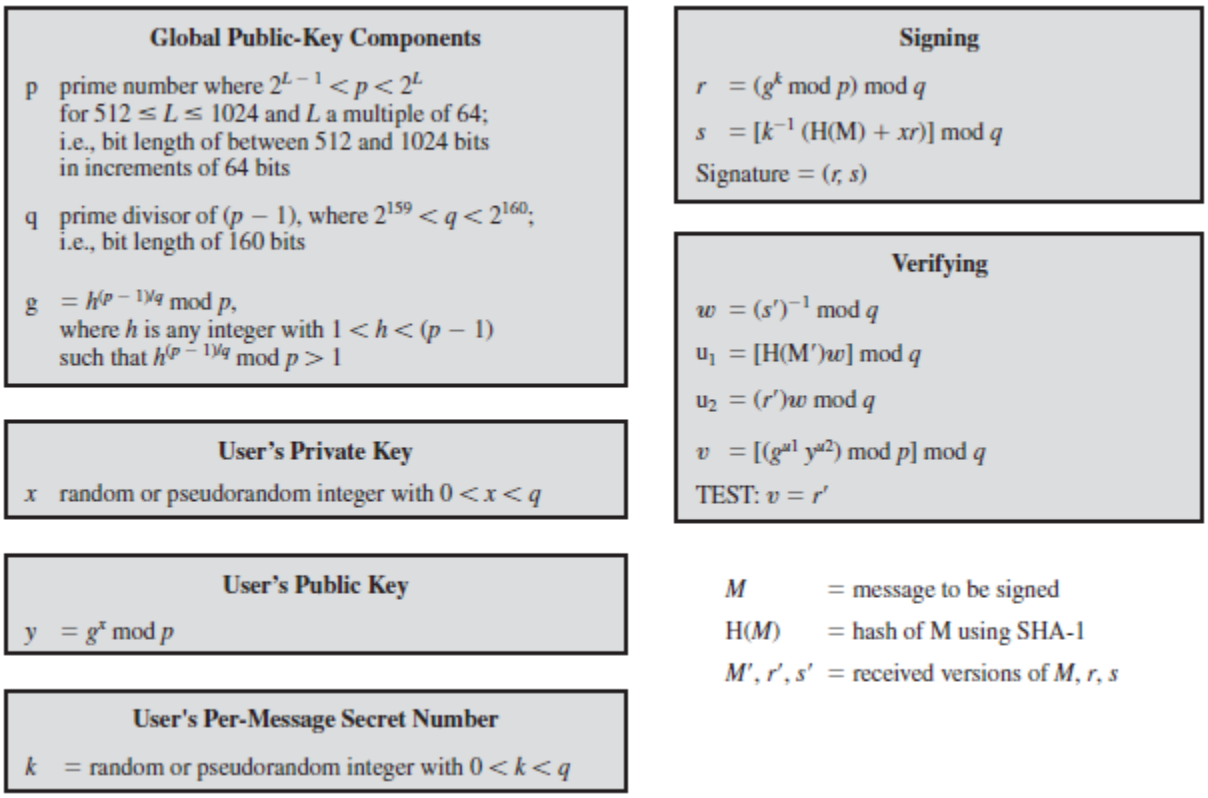
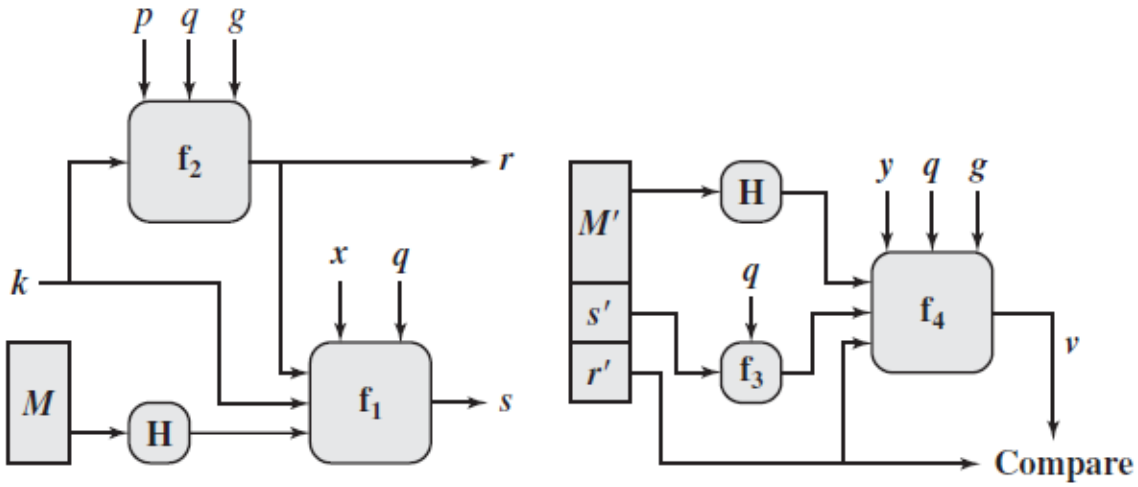


Figure 13.4 The Digital Signature Algorithm (DSA)

- To create a signature, a user calculates two quantities, and , that are functions of the public key components , the user's private key , the hash code of the message , and an additional integer that should be generated randomly or pseudorandomly and be unique for each signing.
- At the receiving end, verification is performed using the formulas shown in Figure 13.4.The receiver generates a quantity that is a function of the public key components, the sender's public key, and the hash code of the incoming message. If this quantity matches the component of the signature, then the signature is validated. Figure 13.5 depicts the functions of signing and verifying.

- The structure of the algorithm, as revealed in Figure 13.5, is quite interesting. Note that the test at the end is on the value, which does not depend on the message at all. Instead, is a function of and the three global public-key components.
- The multiplicative inverse of is passed to a function that also has as inputs the message hash code and the user's private key. The structure of this function is such that the receiver can recover using the incoming message and signature, the public key of the user, and the global public key. It is certainly not obvious from Figure 13.4 or Figure 13.5 that such a scheme would work. Because this value does not depend on the message to be signed, it can be computed ahead of time.
- Indeed, a user could precalculate a number of values of to be used to sign documents as needed. The only other somewhat demanding task is the determination of a multiplicative inverse, Again, a number of these values can be precalculated.



$s = f_1(H(M), k, x, r, q) = (k^{-1} (H(M) + xr)) \bmod q$

$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q$

(a) Signing

$w = f_3(s', q) = (s')^{-1} \bmod q$

$v = f_4(y, q, g, H(M'), w, r')$

$= ((g^{(H(M')w) \bmod q} y^{r'w \bmod q}) \bmod p) \bmod q$

(b) Verifying

Figure 13.5 DSS Signing and Verifying

4.10. ENTITY AUTHENTICATION

4.10.1. BIOMETRICS

4.10.2. PASSWORDS

4.11. CHALLENGE RESPONSE PROTOCOLS

4.12. AUTHENTICATION APPLICATIONS

- **Kerberos**
  - Motivation
  - Kerberos Version 4
  - Kerberos Version 5
- **X.509 Authentication Service**
  - Certificates
  - Authentication Procedures
  - X.509 Version 3

4.12.1. KERBEROS

Contents

Kerberos

- Kerberos4 is an authentication service developed as part of Project Athena at MIT. The problem that Kerberos addresses is this:
- Assume an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network.

In particular, the following three threats exist:

1. A user may gain access to a particular workstation and pretend to be another user operating from that workstation.
2. A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation.
3. A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations.

- In any of these cases, an unauthorized user may be able to gain access to services and data that he or she is not authorized to access.

- Rather than building in elaborate authentication protocols at each server, Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users.

Motivation

1. Rely on each individual client workstation to assure the identity of its user or users and rely on each server to enforce a security policy based on user identification (ID).

2. Require that client systems authenticate themselves to servers, but trust the client system concerning the identity of its user.
3. Require the user to prove his or her identity for each service invoked. Also require that servers prove their identity to clients.

Kerberos Version 4

- Version 4 of Kerberos makes use of DES, in a rather elaborate protocol, to provide the authentication service.
- Viewing the protocol as a whole, it is difficult to see the need for the many elements contained therein.

- Therefore, we adopt a strategy used by Bill Bryant of Project Athena [BRYA88] and build up to the full protocol by looking first at several hypothetical dialogues.
- Each successive dialogue adds additional complexity to counter security vulnerabilities revealed in the preceding dialogue.

A Simple Authentication Dialogue

In an unprotected network environment, any client can apply to any server for service. The obvious security risk is that of impersonation.

- An opponent can pretend to be another client and obtain unauthorized privileges on server machines.
- To counter this threat, servers must be able to confirm the identities of clients who request service.
- Each server can be required to undertake this task for each client/server interaction, but in an open environment, this places a substantial burden on each server.
- An alternative is to use an authentication server (AS) that knows the passwords of all users and stores these in a centralized database.
- In addition, the AS shares a unique secret key with each server. These keys have been distributed physically or in some other secure manner.
- Consider the following hypothetical dialogue:

(1) C ⟶ AS: $ID_C||P_C||ID_V$

(2) AS ⟶ C: $Ticket$

(3) C ⟶ V: $ID_C||Ticket$

$Ticket = E(K_V, [ID_C||AD_C||ID_V])$

Where

C = client
AS = authentication server
V =server
ID_C = identifier of user on C
ID_V = identifier of V
P_C = password of user on C
AD_C = network address of C
Kv = secret encryption key shared by AS and V

A More Secure Authentication Dialogue

- Although the foregoing scenario solves some of the problems of authentication in an open network environment, problems remain.

Once per user logon session:

(1) C ⟶ AS: $ID_C||ID_{tgs}$

(2) AS ⟶ C: $E(K_C, Ticket_{tgs})$

Once per type of service:

(3) C ⟶ TGS: $ID_C||ID_V||Ticket_{tgs}$

(4) TGS ⟶ C: $Ticket_V$

Once per service session:

(5) C ⟶ V: $ID_C||Ticket_V$

$Ticket_{tgs} = E(K_{tgs}, [ID_C||AD_C||ID_{tgs}||TS_1||Lifetime_1])$

$Ticket_V = E(K_V, [ID_C||AD_C||ID_V||TS_2||Lifetime_2])$

Let us look at the details of this scheme:

1.The client requests a ticket-granting ticket on behalf of the user by sending its user's ID and password to the AS, together with the TGS ID, indicating a request to use the TGS service.

2. The AS responds with a ticket that is encrypted with a key that is derived from the user's password. When this response arrives at the client, the client prompts the user for his or her password, generates the key, and attempts to decrypt the incoming message.

- If the correct password is supplied, the ticket is successfully recovered. Because only the correct user should know the password, only the correct user can recover the ticket.
- Thus, we have used the password to obtain credentials from Kerberos without having to transmit the password in plaintext.
- The ticket itself consists of the ID and network address of the user, and the ID of the TGS. This corresponds to the first scenario.

3. The client requests a service-granting ticket on behalf of the user. For this purpose, the client transmits a message to the TGS containing the user's ID, the ID of the desired service, and the ticket-granting ticket.

4. The TGS decrypts the incoming ticket and verifies the success of the decryption by the presence of its ID. It checks to make sure that the lifetime has not expired.

Then it compares the user ID and network address with the incoming information to authenticate the user.

- If the user is permitted access to the server V, the TGS issues a ticket to grant access to the requested.

- The service-granting ticket has the same structure as the ticket-granting ticket. Indeed, because the TGS is a server, we would expect that the same elements are needed to authenticate a client to the TGS and to authenticate a client to an application server.
- Again, the ticket contains a timestamp and lifetime. If the user wants access to the same service at a later time, the client can simply use the previously acquired service-granting ticket and need not bother the user for a password.
- Note that the ticket is encrypted with a secret key (*Kv*).

The Version 4 Authentication Dialogue:

Although the foregoing scenario enhances security compared to the first attempt, two additional problems remain.

- The heart of the first problem is the lifetime associated with the ticket-granting ticket.
- If this lifetime is very short (e.g., minutes), then the user will be repeatedly asked for a password. If the lifetime is long (e.g., hours), then an opponent has a greater opportunity for replay.
- An opponent could eavesdrop on the network and capture a copy of the ticket-granting ticket and then wait for the legitimate user to log out.
- Then the opponent could forge the legitimate user's network address and send the message of step (3) to the TGS.
- This would give the opponent unlimited access to the resources and files available to the legitimate user.

Summary of Kerberos Version 4 Message Exchanges

(a) Authentication Service Exchange: to obtain ticket-granting ticket
(1) C → AS: $ID_c \parallel ID_{tgs} \parallel TS_1$
(2) AS → C: $E_{K_c}\left[K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}\right]$
$Ticket_{tgs} = E_{K_{tgs}}\left[K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2\right]$
(b) Ticket-Granting Service Exchange: to obtain service-granting ticket
(3) C → TGS: $ID_v \parallel Ticket_{tgs} \parallel Authenticator_c$
(4) TGS → C: $E_{K_{c,tgs}}\left[K_{c,v} \parallel ID_v \parallel TS_4 \parallel Ticket_v\right]$
$Ticket_{tgs} = E_{K_{tgs}}\left[K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2\right]$
$Ticket_v = E_{K_v}\left[K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4\right]$
$Authenticator_c = E_{K_{tgs}}\left[ID_C \parallel AD_C \parallel TS_3\right]$
(c) Client/Server Authentication Exchange: to obtain service
(5) C → V: $Ticket_v \parallel Authenticator_c$
(6) V → C: $E_{K_{c,v}}\left[TS_5 + 1\right]$ (for mutual authentication)
$Ticket_v = E_{K_v}\left[K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4\right]$
$Authenticator_c = E_{K_{c,v}}\left[ID_C \parallel AD_C \parallel TS_5\right]$

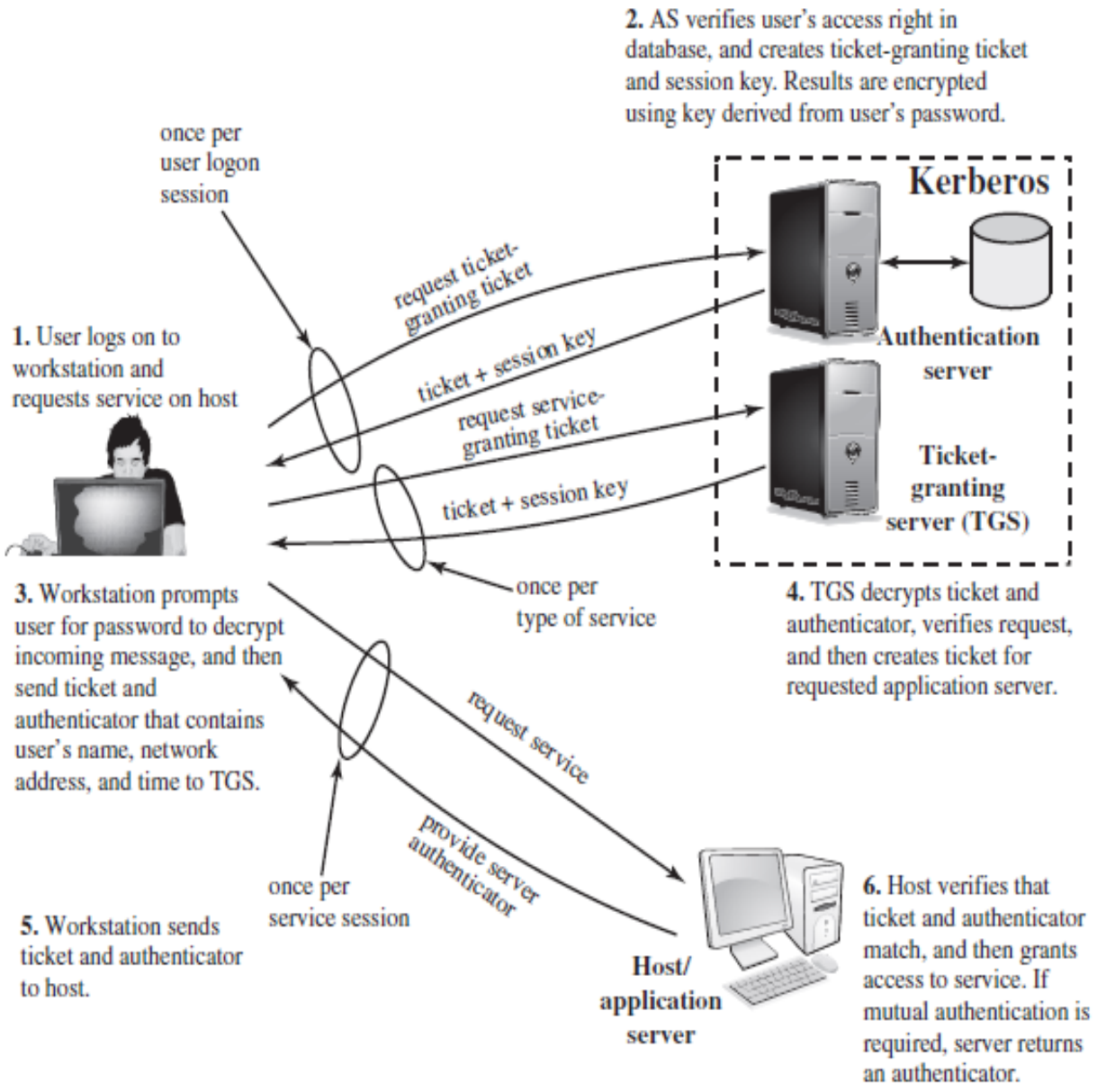
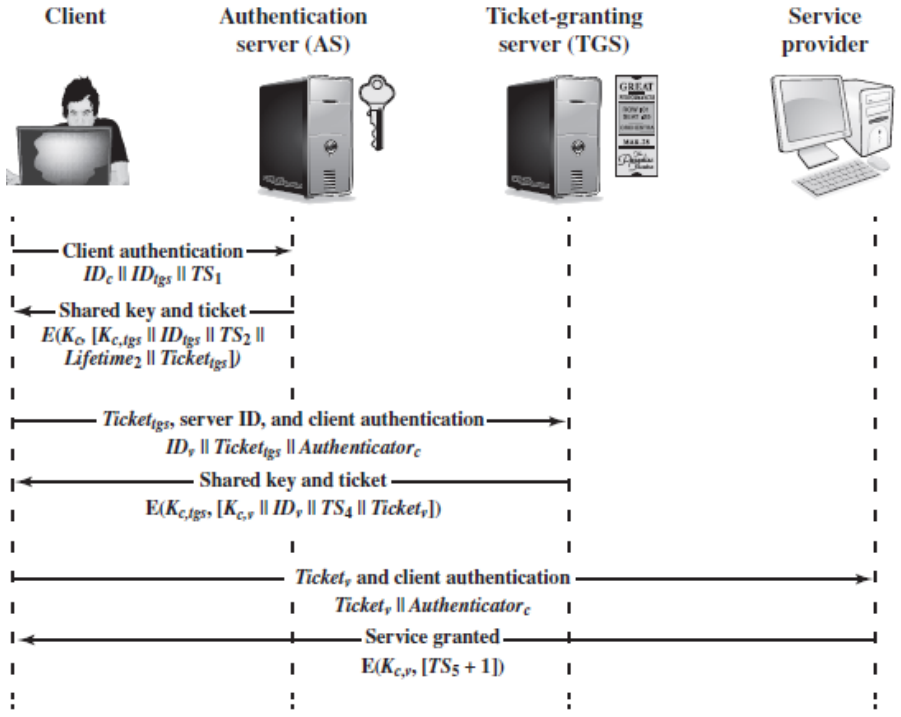


Figure 15.1 Overview of Kerberos

Figure 15.2 illustrates the Kerberos exchanges among the parties. Table 15.2 summarizes the justification for each of the elements in the Kerberos protocol.



ure 15.2 Kerberos Exchanges

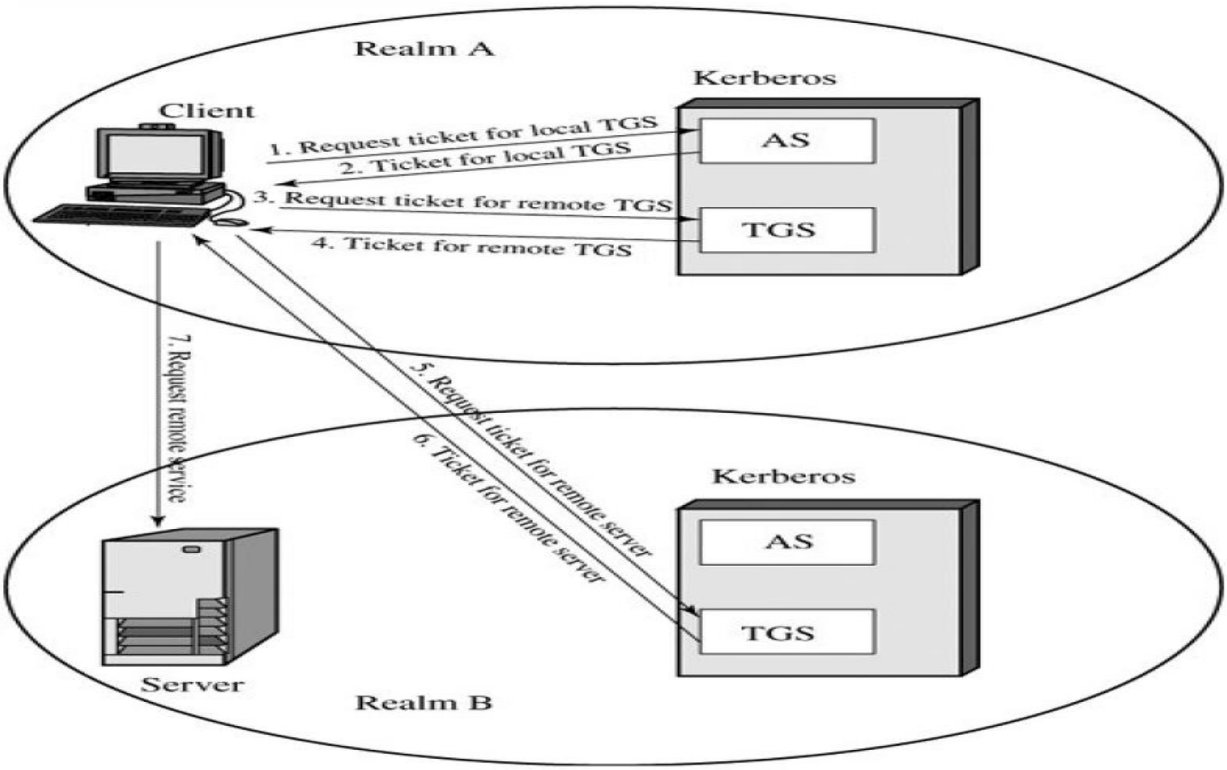
Kerberos Realms and Multiple Kerberi

- A full-service Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers requires the following:
  1. The Kerberos server must have the user ID and hashed passwords of all participating
     users in its database. All users are registered with the Kerberos server.
  2. The Kerberos server must share a secret key with each server. All servers are registered with the Kerberos server. Such an environment is referred to as a **Kerberos realm**.
- The concept of **realm** can be explained as follows. A Kerberos realm is a set of managed nodes that share the same Kerberos database.

- Changing or accessing the contents of a Kerberos database requires the Kerberos master password. A related concept is that of a **Kerberos principal**, which is a service or user that is known to the Kerberos system.
- Each Kerberos principal is identified by its principal name. Principal names consist of three parts: a service or user name, an instance name, and a realm name.

  3. The Kerberos server in each interoperating realm shares a secret key with the server in the other realm. The two Kerberos servers are registered with each other.

- The scheme requires that the Kerberos server in one realm trust the Kerberos server in the other realm to authenticate its users..



(1) C ⟶ AS: $ID_c||ID_{tgs}||TS_1$

(2) AS ⟶ C: $E(K_c, [K_{c,tgs}||ID_{tgs}||TS_2||Lifetime_2||Ticket_{tgs}])$

(3) C ⟶ TGS: $ID_{tgsrem}||Ticket_{tgs}||Authenticator_c$

(4) TGS ⟶ C: $E(K_{c,tgs}, [K_{c,tgsrem}||ID_{tgsrem}||TS_4||Ticket_{tgsrem}])$

(5) C ⟶ TGS_{rem}: $ID_{vrem}||Ticket_{tgsrem}||Authenticator_c$

(6) TGS_{rem} ⟶ C: $E(K_{c,tgsrem}, [K_{c,vrem}||ID_{vrem}||TS_6||Ticket_{vrem}])$

(7) C ⟶ V_{rem}: $Ticket_{vrem}||Authenticator_c$

Kerberos Version 5

- Kerberos Version 5 is specified in RFC 1510 and provides a number of improvements over version 4

Differences between Versions 4 and 5

- Version 5 is intended to address the limitations of version 4 in two areas: environmental shortcomings and technical deficiencies
- Kerberos Version 4 was developed for use within the Project Athena environment and, accordingly, did not fully address the need to be of general purpose.

The Version 5 Authentication Dialogue

(a) Authentication Service Exchange: to obtain ticket-granting ticket
(1) C → AS: Options ‖ ID_c ‖ $Realm_c$ ‖ ID_{tgs} ‖ $Times$ ‖ $Nonce_1$
(2) AS → C: $Realm_c \| ID_C \| Ticket_{tgs} \| E_{K_c}\left[K_{c,tgs} \| Times \| Nonce_1 \| Realm_{tgs} \| ID_{tgs}\right]$
$Ticket_{tgs} = E_{K_{tgs}}\left[Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times\right]$
(b) Ticket-Granting Service Exchange: to obtain service-granting ticket
(3) C → TGS: Options ‖ ID_v ‖ $Times$ ‖ ‖ $Nonce_2$ ‖ $Ticket_{tgs}$ ‖ $Authenticator_c$
(4) TGS → C: $Realm_c \| ID_C \| Ticket_v \| E_{K_{c,tgs}}\left[K_{c,v} \| Times \| Nonce_2 \| Realm_v \| ID_V\right]$
$Ticket_{tgs} = E_{K_{tgs}}\left[Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times\right]$
$Ticket_v = E_{K_v}\left[Flags \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times\right]$
$Authenticator_c = E_{K_{c,tgs}}\left[ID_C \| Realm_c \| TS_1\right]$
(c) Client/Server Authentication Exchange: to obtain service
(5) C → V: Options ‖ $Ticket_v$ ‖ $Authenticator_c$
(6) V → C: $E_{K_{c,V}}$ [TS_2 ‖ Subkey ‖ Seq#]
$Ticket_v = E_{K_v}\left[Flags \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times\right]$
$Authenticator_c = E_{K_{c,V}}\left[ID_C \| Realm_c \| TS_2 \| Subkey \| Seq\#\right]$

The following new elements are added:

- **Realm:** Indicates realm of user
- **Options:** Used to request that certain flags be set in the returned ticket
- **Times:** Used by the client to request the following time settings in the ticket:
- **Nonce:** A random value to be repeated in message (2) to assure that the response is fresh and has not been replaced by an opponent.
-

The authenticator includes several new fields as follows:

Subkey: The client's choice for an encryption key to be used to protect this specific application session. If this field is omitted, the session key from the ticket (Kc,v) is used.

Sequence number: An optional field that specifies the starting sequence number to be used by the server for messages sent to the client during this session. Messages may be sequence numbered to detect replays.

Ticket Flags*: The* flags field included in tickets in version 5 supports expanded functionality compared to that available in version 4.

4.12.2. X.509

Contents

• **X.509 Authentication Service**
  o Certificates
  o Authentication Procedures
  o X.509 Version 3

X.509 Authentication services:

- X.509 defines a framework for the provision of authentication services by the X.500 directory to its users. Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority.
- In addition, X.509 defines alternative authentication protocols based on the use of public-key certificates.
- X.509 is based on the use of public-key cryptography and digital signatures. The standard does not dictate the use of a specific algorithm but recommends RSA. The digital signature scheme is assumed to require the use of a hash function.

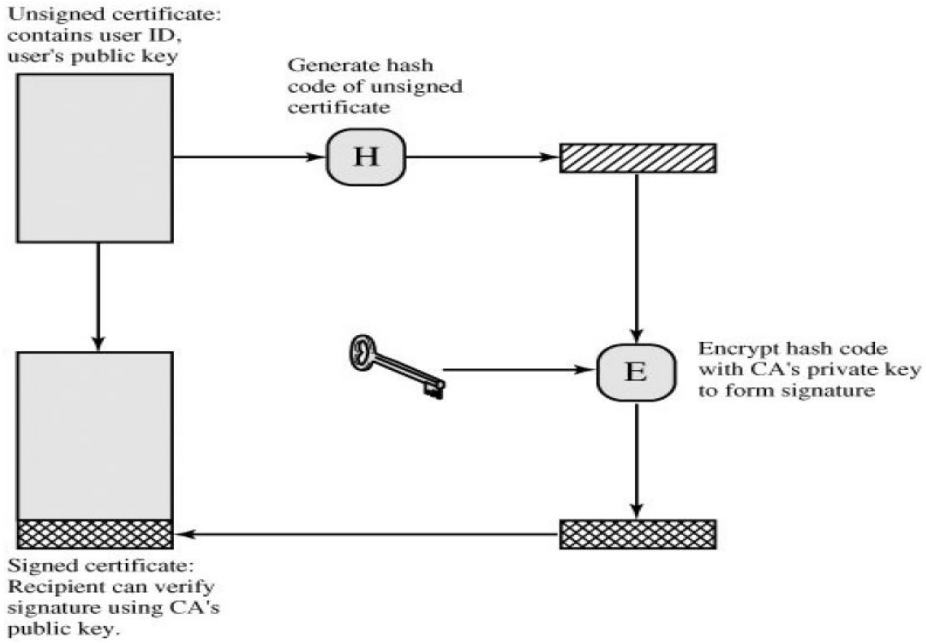


Figure 14.3. Public-Key Certificate Use

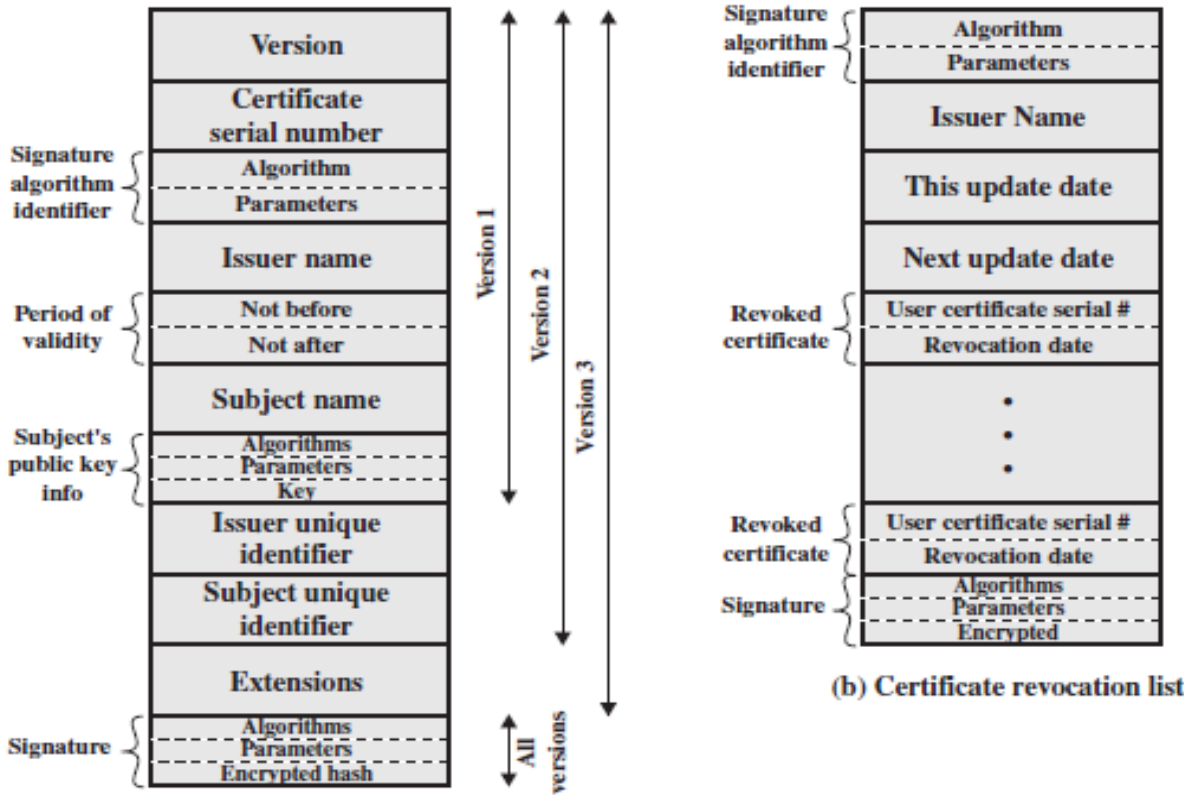
Certificates:

- The heart of the X.509 scheme is the public-key certificate associated with each user. These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user.

- The directory server itself is not responsible for the creation of public keys or for the certification function; it merely provides an easily accessible location for users to obtain certificates.

- Figure 14.15a shows the general format of a certificate, which includes the following elements.

    - **Version:** Differentiates among successive versions of the certificate format; the default is version 1. If the Issuer Unique Identifier or Subject Unique Identifier are present, the value must be version 2. If one or more extensions are present, the version must be version 3.
    - **Serial number:** An integer value, unique within the issuing CA, that is unambiguously associated with this certificate.
    - **Signature algorithm identifier:** The algorithm used to sign the certificate, together with any associated parameters. Because this information is repeated in the Signature field at the end of the certificate, this field has little, if any, utility.
    - **Issuer name:** X.500 name of the CA that created and signed this certificate.
    - **Period of validity:** Consists of two dates: the first and last on which the certificate is valid.
    - **Subject name:** The name of the user to whom this certificate refers. That is, this certificate certifies the public key of the subject who holds the corresponding private key.
    - **Subject's public-key information:** The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.
    - **Issuer unique identifier:** An optional bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.
    - **Subject unique identifier:** An optional bit string field used to identify uniquely the subject in the event the X.500 name has been reused for different entities.
    - **Extensions:** A set of one or more extension fields. Extensions were added in version 3 and are discussed later in this section.
    - **Signature:** Covers all of the other fields of the certificate; it contains the hash code of the other fields, encrypted with the CA's private key. This field includes the signature algorithm identifier.



(a) X.509 certificate

(b) Certificate revocation list

- The unique identifier fields were added in version 2 to handle the possible reuse of subject and/or issuer names over time. These fields are rarely used. The standard uses the following notation to define a certificate:

$$CA \ll A \gg = CA \{V, SN, AI, CA, UCA, A, UA, Ap, T^A\}$$

Where

V = version of the certificate
SN = serial number of the certificate
AI = identifier of the algorithm used to sign the certificate
CA = name of certificate authority
UCA = optional unique identifier of the CA
A = name of user A
UA = optional unique identifier of the user A
Ap = public key of user A
T^A = period of validity of the certificate

Obtaining a User's Certificate

- User certificates generated by a CA have the following characteristics:
- Any user with access to the public key of the CA can verify the user public key that was certified.
- No party other than the certification authority can modify the certificate without this being detected.

- The CA signs the certificate with its private key. If the corresponding public key is known to a user, then that user can verify that a certificate signed by the CA is valid.
- Figure 14.16, taken from X.509, is an example of such a hierarchy. The connected circles indicate the hierarchical relationship among the CAs; the associated boxes indicate certificates maintained in the directory for each CA entry.

- The directory entry for each **CA includes two types of certificates:**
  • **Forward certificates:** Certificates of X generated by other CAs
  • **Reverse certificates:** Certificates generated by X that are the certificates of other CAs

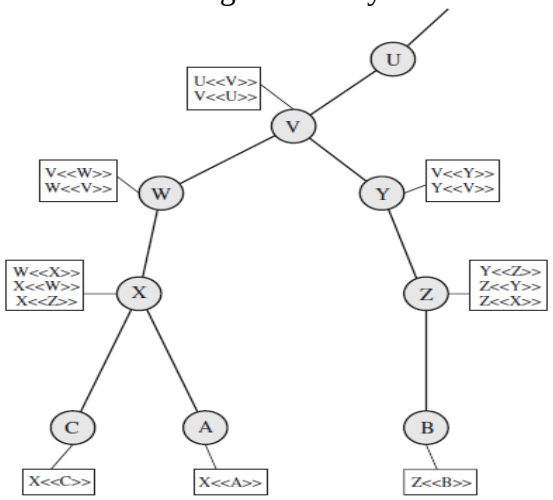


Figure 14.16 X.509 Hierarchy: A Hypothetical Example

- In this example, user A can acquire the following certificates from the directory to establish a certification path to B:

$$X \ll W \gg W \ll V \gg V \ll Y \gg Y \ll Z \gg Z \ll B \gg$$

- When A has obtained these certificates, it can unwrap the certification path in sequence to recover a trusted copy of B's public key. Using this public key, A can send encrypted messages to B. If A wishes to receive encrypted messages back from B, or to sign messages sent to B, then B will require A's public key, which can be obtained from the following certification path:

$$Z \ll Y \gg Y \ll V \gg V \ll W \gg W \ll X \gg X \ll A \gg$$

- B can obtain this set of certificates from the directory, or A can provide them as part of its initial message to B.

Revocation of Certificates

- Recall from Figure 14.4 that each certificate includes a period of validity, much like a credit card. Typically, a new certificate is issued just before the expiration of the old one.
- In addition, it may be desirable on occasion to revoke a certificate before it expires, for one of the following reasons:
  1. The user's private key is assumed to be compromised.
  2. The user is no longer certified by this CA.
  3. The CA's certificate is assumed to be compromised.

Authentication Procedures

- X.509 also includes three alternative authentication procedures that are intended for use across a variety of applications.
- All these procedures make use of public-key signatures. It is assumed that the two parties know each other's public key, either by obtaining each other's certificates from the directory or because the certificate is included in the initial message from each side.

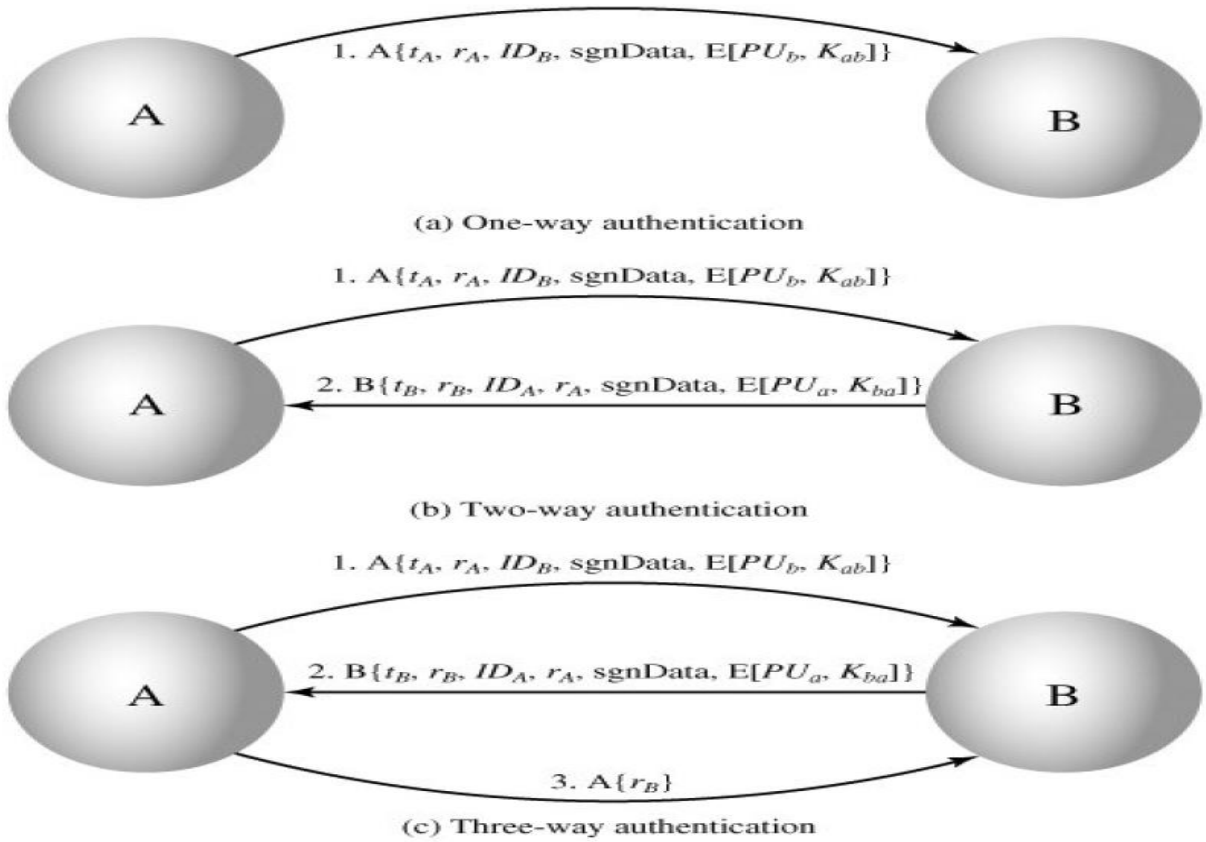
One-Way Authentication

- One way authentication involves a single transfer of information from one user (A) to another (B), and establishes the following:

  1. The identity of A and that the message was generated by A

  2. That the message was intended for B

  3. The integrity and originality (it has not been sent multiple times) of the message. At a minimum, the message includes a timestamp *tA*, a nonce *rA* and the identity of B and is signed with A's private key.



- **Figure 14.6. X.509 Strong Authentication Procedures**

- The timestamp consists of an optional generation time and an expiration time. This prevents delayed delivery of messages.
- The nonce can be used to detect replay attacks. The nonce value must be unique within the expiration time of the message.

- Thus, B can store the nonce until it expires and reject any new messages with the same nonce.

Two-Way Authentication

- In addition to the three elements just listed, two-way authentication establishes the following elements:

    4. The identity of B and that the reply message was generated by B

    5. That the message was intended for A

    6. The integrity and originality of the reply

- Two-way authentication thus permits both parties in a communication to verify the identity of the other.
- The reply message includes the nonce from A, to validate the reply. It also includes a timestamp and nonce generated by B. As before, the message may include signed additional information and a session key encrypted with A's public key.

Three-Way Authentication

- In three-way authentication, a final message from A to B is included, which contains a signed copy of the nonce *rB*.
- The intent of this design is that timestamps need not be checked: Because both nonces are echoed back by the other side, each side can check the returned nonce to detect replay attacks. This approach is needed when synchronized clocks are not available.

X.509 Version 3

- The X.509 version 2 format does not convey all of the information that recent design and implementation experience has shown to be needed.

Lists the following requirements not satisfied by version 2:

1. The Subject field is inadequate to convey the identity of a key owner to a public-key user. X.509 names may be relatively short and lacking in obvious identification details that may be needed by the user.

2. The Subject field is also inadequate for many applications, which typically recognize entities by an Internet e-mail address, URL, or some other Internet-related identification.

3. There is a need to indicate security policy information. This enables a security application or function, such as IPSec, to relate an X.509 certificate to a given policy.

4. There is a need to limit the damage that can result from a faulty or malicious CA by setting constraints on the applicability of a particular certificate.

4. It is important to be able to identify different keys used by the same owner at different times. This feature supports key life cycle management, in particular the ability to update key pairs for users and CAs on a regular basis or under exceptional circumstances.

Certificate Subject and Issuer Attributes

- These extensions support alternative names, in alternative formats, for a certificate subject or certificate issuer and can convey additional information about the certificate subject, to increase a certificate user's confidence that the certificate subject is a particular person or entity.
- For example, information such as postal address, position within a corporation, or picture image may be required.

The extension fields in this area include the following:

- **Subject alternative name:** Contains one or more alternative names, using any of a variety of forms. This field is important for supporting certain applications, such as electronic mail, EDI, and IPSec, which may employ their own name forms.
- **Issuer alternative name:** Contains one or more alternative names, using any of a variety of forms.
- **Subject directory attributes:** Conveys any desired X.500 directory attribute values for the subject of this certificate.
-

Certification Path Constraints

- These extensions allow constraint specifications to be included in certificates issued for CAs by other CAs.
- The constraints may restrict the types of certificates that can be issued by the subject CA or that may occur subsequently in a certification chain.

The extension fields in this area include the following:

Basic constraints: Indicates if the subject may act as a CA. If so, a certification path length constraint may be specified.

Name constraints: Indicates a name space within which all subject names in subsequent certificates in a certification path must be located.

Policy constraints: Specifies constraints that may require explicit certificate policy identification or inhibit policy mapping for the remainder of the certification path.

UNIT II
SYMMETRIC KEY CRYPTOGRAPHY

MATHEMATICS OF SYMMETRIC KEY CRYPTOGRAPHY: Algebraic structures – Modular arithmetic-Euclid‟s algorithm- Congruence and matrices – Groups, Rings, Fields-Finite fields- SYMMETRIC KEY CIPHERS: SDES – Block cipher Principles of DES – Strength of DES – Differential and linear cryptanalysis – Block cipher design principles – Block cipher mode of operation – Evaluation criteria for AES – Advanced Encryption Standard – RC4 – Key distribution.

MATHEMATICS OF SYMMETRIC KEY CRYPTOGRAPHY
2.2. MODULAR ARITHMETIC

Contents
• **The Modulus** • **Properties of Congruences** • **Modular Arithmetic Operations** • **Properties of Modular Arithmetic**

The Modulus

- If a is an integer and n is a positive integer, we define a mod n to be the remainder when a is divided by n. The integer n is called the modulus. Thus, for any integer a, we can rewrite Equation (4.1) as follows:

$$a = qn + r \qquad 0 \le r < n; q = \lfloor a/n \rfloor$$
$$a = \lfloor a/n \rfloor \times n + (a \bmod n)$$

$$11 \bmod 7 = 4; \qquad -11 \bmod 7 = 3$$

- Two integers a and b are said to be congruent modulo n, if (a mod n) = (b mod n). This is written as a K b (mod n).

$$73 \equiv 4 \pmod{23}; \qquad 21 \equiv -9 \pmod{10}$$

Note that if $a \equiv 0 \pmod{n}$, then $n \mid a$.

Properties of Congruences

Congruences have the following properties:

1. $a \equiv b \pmod{n}$ if $n \mid (a - b)$.
2. $a \equiv b \pmod{n}$ implies $b \equiv a \pmod{n}$.
3. $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$ imply $a \equiv c \pmod{n}$.

To demonstrate the first point, if n | (a - b), then (a - b) = k*n* for some k.

- So when b is divided by n) = (b mod n).we can write a = b + k*n*. Therefore, (a mod n) = (remainder when b + k*n* is divided by n) = remainder

$$23 \equiv 8 \pmod{5} \quad \text{because} \quad 23 - 8 = 15 = 5 \times 3$$
$$-11 \equiv 5 \pmod{8} \quad \text{because} \quad -11 - 5 = -16 = 8 \times (-2)$$
$$81 \equiv 0 \pmod{27} \quad \text{because} \quad 81 - 0 = 81 = 27 \times 3$$

The remaining points are as easily proved.

Modular Arithmetic Operations

❖ Note that, by definition (Figure 4.1), the (mod n) operator maps all integers into the set of integers {0, 1, c, (n - 1)}. this technique is known as modular arithmetic.

Modular arithmetic exhibits the following properties:

1. $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
2. $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
3. $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

Properties of Modular Arithmetic

Define the set Zn as the set of nonnegative integers less than n:

$$Z_n = \{0, 1, \ldots, (n - 1)\}$$

This is referred to as the set of residues, or residue classes (mod n). To be more precise, each integer in Zn represents a residue class. We can label the residue classes (mod n) as [0], [1], [2], c, [n - 1], where

The residue classes (mod 4) are

$[0] = \{\ldots, -16, -12, -8, -4, 0, 4, 8, 12, 16, \ldots\}$

$[1] = \{\ldots, -15, -11, -7, -3, 1, 5, 9, 13, 17, \ldots\}$

Table 4.3 Properties of Modular Arithmetic for Integers in Z_n

Property	Expression
Commutative Laws	$(w + x) \bmod n = (x + w) \bmod n$ $(w \times x) \bmod n = (x \times w) \bmod n$
Associative Laws	$[(w + x) + y] \bmod n = [w + (x + y)] \bmod n$ $[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$
Distributive Law	$[w \times (x + y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$
Identities	$(0 + w) \bmod n = w \bmod n$ $(1 \times w) \bmod n = w \bmod n$
Additive Inverse $(-w)$	For each $w \in Z_n$, there exists a z such that $w + z = 0 \bmod n$

2.4. EUCLID'S ALGORITHM

Contents
• **Introduction** • **Greatest Common Divisor** • **Finding the Greatest Common Divisor**

Introduction

❖ One of the basic techniques of number theory is the Euclidean algorithm, which is a simple procedure for determining the greatest common divisor of two positive integers. First, we need a simple definition: Two integers are **relatively prime** if their only common positive integer factor is 1.

Greatest Common Divisor:

- Recall that nonzero b is defined to be a divisor of a if $a = mb$ for some m, where a, b, and m are integers.
- We will use the notation gcd(a, b) to mean the **greatest common divisor** of a and b. The greatest common divisor of a and$b$ is the largest integer that divides both a and b.
- We also define gcd(0, 0) = 0.More formally, the positive integer c is said to be the greatest common divisor of a and b if
    1. c is a divisor of a and of b.
    2. Any divisor of a and b is a divisor of c.

An equivalent definition is the following:

$$\gcd(a, b) = \max[k, \text{such that } k|a \text{ and } k|b]$$

- Because we require that the greatest common divisor be positive, gcd(a, b) = gcd(a, −b) = gcd(−a, b) = gcd(−a,−b). In general, gcd(a, b) = gcd(|a| , |b|).

$$\gcd(60, 24) = \gcd(60, -24) = 12$$

Finding the Greatest Common Divisor

- Suppose we have integers a, b such that d = gcd(a, b). Because gcdgcd(|a| , |b|) = gcd(a, b), there is no harm in assuming a ≥ b > 0. Now dividing a by b and applying the division algorithm, we can state:

$$a = q_1 b + r_1 \qquad 0 \le r_1 < b \qquad \textbf{(4.2)}$$

- Let us now return to Equation (4.2) and assume that $r_1 \neq 0$. Because $b > r_1$, we can divide b by r_1 and apply the division algorithm to obtain:

$$b = q_2 r_1 + r_2 \qquad 0 \le r_2 < r_1$$

The result is the following system of equations:

$$\left.\begin{array}{ll} a = q_1 b + r_1 & 0 < r_1 < b \\ b = q_2 r_1 + r_2 & 0 < r_2 < r_1 \\ r_1 = q_3 r_2 + r_3 & 0 < r_3 < r_2 \\ \quad\vdots & \quad\vdots \\ r_{n-2} = q_n r_{n-1} + r_n & 0 < r_n < r_{n-1} \\ r_{n-1} = q_{n+1} r_n + 0 & \\ d = \gcd(a, b) = r_n & \end{array}\right\} \qquad \textbf{(4.3)}$$

Let us now look at an example with relatively large numbers to see the power of this algorithm:

To find d = gcd (a,b) = gcd (1160718174, 316258250)		
$a = q_1 b + r_1$	1160718174 = 3 × 316258250 + 211943424	d = gcd(316258250, 211943424)
$b = q_2 r_1 + r_2$	316258250 = 1 × 211943424 + 104314826	d = gcd(211943424, 104314826)
$r_1 = q_3 r_2 + r_3$	211943424 = 2 × 104314826 + 3313772	d = gcd(104314826, 3313772)
$r_2 = q_4 r_3 + r_4$	104314826 = 31 × 3313772 + 1587894	d = gcd(3313772, 1587894)
$r_3 = q_5 r_4 + r_5$	3313772 = 2 × 1587894 + 137984	d = gcd(1587894, 137984)
$r_4 = q_6 r_5 + r_6$	1587894 = 11 × 137984 + 70070	d = gcd(137984, 70070)
$r_5 = q_7 r_6 + r_7$	137984 = 1 × 70070 + 67914	d = gcd(70070, 67914)
$r_6 = q_8 r_7 + r_8$	70070 = 1 × 67914 + 2156	d = gcd(67914, 2156)
$r_7 = q_9 r_8 + r_9$	67914 = 31 × 2516 + 1078	d = gcd(2156, 1078)
$r_8 = q_{10} r_9 + r_{10}$	2156 = 2 × 1078 + 0	d = gcd(1078, 0) = 1078
Therefore, d = gcd(1160718174, 316258250) = 1078		

In this example, we begin by dividing 1160718174 by 316258250, which gives 3 with a remainder of 211943424. Next we take 316258250 and divide it by 211943424. The process continues until we get a remainder of 0, yielding a result of 1078.

2.5. CONGRUENCE AND MATRICES

2.6. GROUPS, RINGS, FIELDS

Contents

- **Groups**
  - A1- Closure
  - A2 - Associative
  - A3 - Identity
  - A4 - Inverse
  - A5 - Commutative
- **Rings**
  - M1- Closure under multiplication
  - M2 - Associativity of multiplication
  - M3 - Distributive law
  - M4 – Commutativity of multiplication
  - M5 – Multiplicative Identity
  - M6 – No zero divisors
- **Fields**
  - M7 – Multiplicative Inverse

Groups, rings, and fields are the fundamental elements of a branch of mathematicsknown as abstract algebra, or modern algebra.

Groups

- A **group** G, sometimes denoted by $\{G, \cdot\}$, is a set of elements with a binary operationdenoted by $\cdot$ that associates to each ordered pair (a, b) of elements in G an element$(a \cdot b)$ 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 $a \cdot a' = a' \cdot a = e$.

- If a group has a finite number of elements, it is referred to as a **finite group**, and the **order** of the group is equal to the number of elements in the group. Otherwise, the group is an **infinite group**.
- A group is said to be **abelian** if it satisfies the following additional condition:

(A5) Commutative:	$a \cdot b = b \cdot a$ for all a, b in G.

Rings

- A **ring** R, sometimes denoted by $\{R, +, *\}$, is a set of elements with two binary operations, called ***addition* and *multiplication***, such that for all a, b, c in R the following axioms are obeyed.

(A1–A5)	R is an abelian group with respect to addition; that is, R satisfies axioms A1 through A5. For the case of an additive group, we denote the identity element as 0 and the inverse of a as $-a$.
(M1) Closure under multiplication:	If a and b belong to R, then ab is also in R.
(M2) Associativity of multiplication:	$a(bc) = (ab)c$ for all a, b, c in R.
(M3) Distributive laws:	$a(b + c) = ab + ac$ for all a, b, c in R. $(a + b)c = ac + bc$ for all a, b, c in R.

- A ring is said to be **commutative** if it satisfies the following additional condition:

(M4) Commutativity of multiplication:	$ab = ba$ for all a, b in R.

- Next, we define an **integral domain**, which is a commutative ring that obeys the following axioms.

(M5) Multiplicative identity:	There is an element 1 in R such that $a1 = 1a = a$ for all a in R.
(M6) No zero divisors:	If a, b in R and $ab = 0$, then either $a = 0$ or $b = 0$.

Fields:

- A **field** F, sometimes denoted by $\{F, +, *\}$, is a set of elements with two binary operations, called *addition* and *multiplication*, such that for all a, b, c in F the following axioms are obeyed.

(A1–M6) F is an integral domain; that is, F satisfies axioms A1 through A5 and M1 through M6.

(M7) Multiplicative inverse: For each a in F, except 0, there is an element a^{-1} in F such that $aa^{-1} = (a^{-1})a = 1$.

2.7. FINITE FIELDS

SYMMETRIC KEY CIPHERS

2.8. SDES

Contents

Introduction

- Proposed by NIST in 1977.
- It is a block cipher and encrypts 64 bits data using 56 bit key.

DES Encryption:

- There are two inputs to the encryption function: the plaintext to be encrypted and the key. In this case, the plaintext must be **64 bits in length and key is 56** in length.
- Looking at the left-hand side of the figure, we can see that the processing of the plaintext proceeds in **three phases.**
- First, the 64-bit plaintext passes through an **initial permutation (IP)** that rearranges the bits to produce the **permuted input.**
- This is followed by a phase consisting of **sixteen rounds** of the same function, which involves both permutation and substitution functions.
- The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key.
- The left and right halves of the output are swapped to produce the **pre output.**
- Finally, the preoutput is passed through a permutation [IP -1] that is the inverse of the initial permutation function, to produce the 64-bit **ciphertext.** With the exception of the initial and final permutations, DES has the exact structure of a **Feistel Cipher.**

- The right-hand portion of Figure shows the way in which the **56-bit key is used.**
- Initially, the key is passed through a permutation function.
- Then, for each of the sixteen rounds, a **subkey (Ki)** is produced by the combination of a left circular shift and a permutation.
- The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits.

DES Decryption

- As with any **Feistel cipher**, decryption uses the same algorithm as encryption, except that the application of the subkeys is reversed. Additionally, the initial and final permutations are reversed.

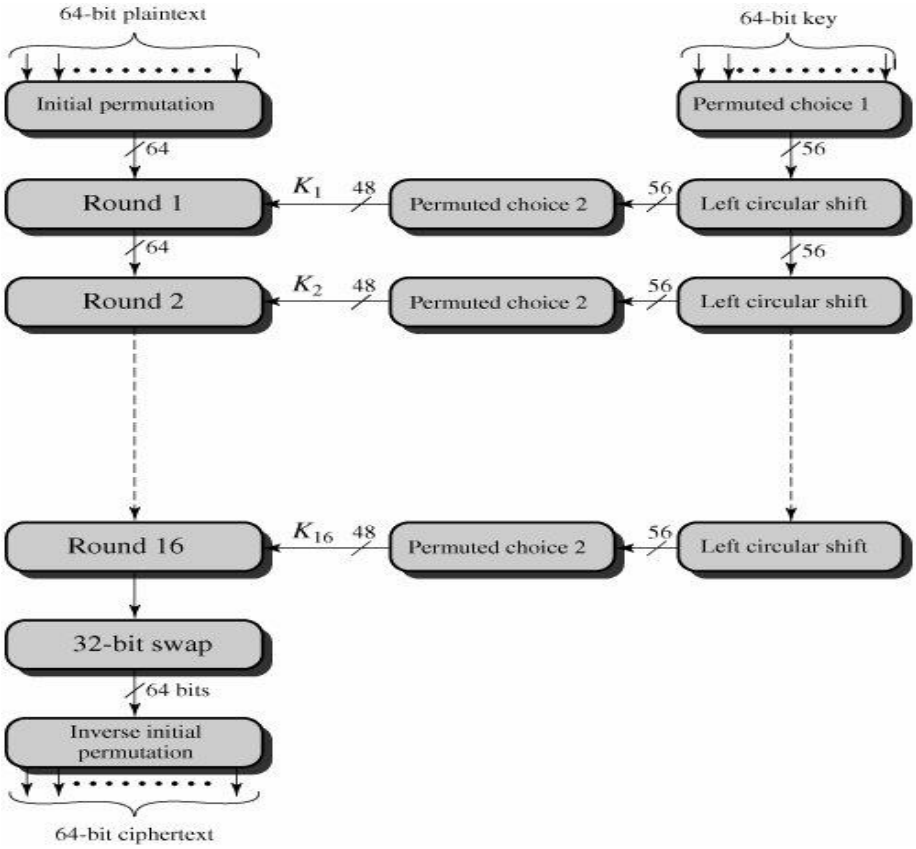


Fig .General Depiction of DES Encryption Algorithm

The Avalanche Effect

- A desirable property of any encryption algorithm is that a small change in either the plaintext or the key should produce a significant change in the cipher text.
- In particular, a change in one bit of the plaintext or one bit of the key should produce a change in many bits of the cipher text.
- This is referred to as the avalanche effect.

DES Round structure.

- ✓ Uses two 32 bit L & R halves.
- ✓ As in any classic Feistel cipher, the overall processing at each round can be summarized in the following formulas:

Li = Ri-1
Ri= Li 1 { F(Ri 1, Ki)

- ✓ The round key is 48 bits. The input is 32 bits.
- ✓ This input is first expanded to 48 bits by using a table that defines a permutation plus an expansion that involves duplication of 16 of the bits.
- ✓ The resulting 48 bits are XOR ed with. This 48-bit result passes through a substitution function that produces a 32-bit output, which is permuted as defined by table.
- ✓ The role of the S-boxes in the function F is illustrated in Figure.
- ✓ The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input and Produces 4 bits as output.

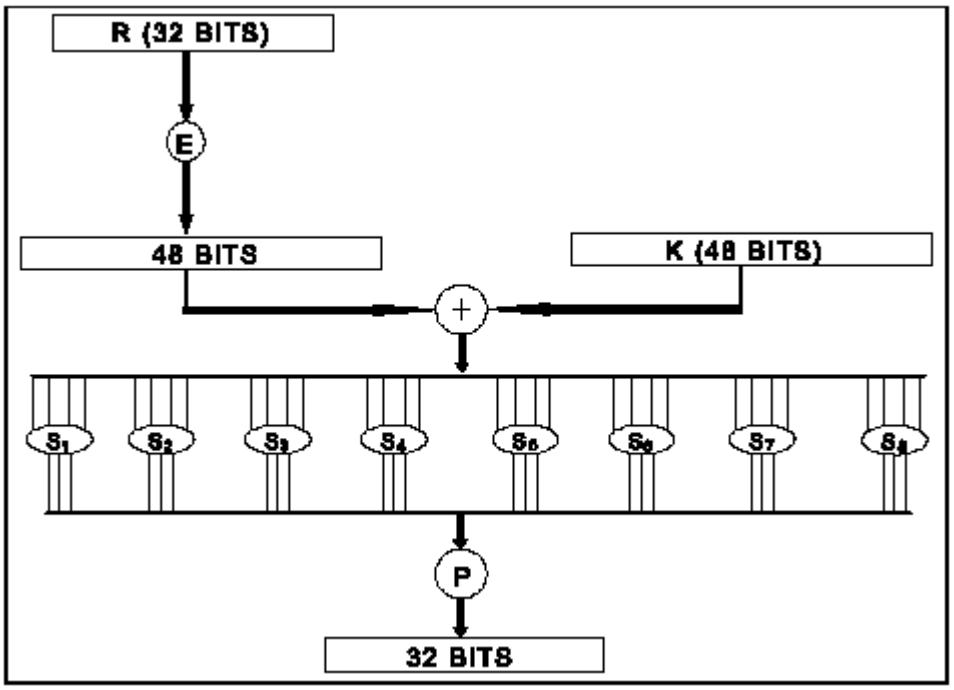


Fig: Calculation of F(R, K)

2.9. STRENGTH OF DES

- The Use of 56-Bit Keys
- The Nature of the DES Algorithm
- Timing Attacks

2.9. DIFFERENTIAL AND LINEAR CRYPTANALYSIS

2.10. BLOCK CIPHER DESIGN PRINCIPLES

– Block cipher Principles of DES

Contents
• Introduction • Number of rounds • Design of function • Key Schedule Algorithm

Introduction

- There are three critical aspects of block cipher design: the number of rounds, design of the function F, and key scheduling

Number of Rounds

- The greater the number of rounds, the more difficult it is to perform cryptanalysis, even for a relatively weak F.
- In general, the criterion should be that the number of rounds is chosen so that known cryptanalytic efforts require greater effort than a simple **brute-force key search attack.**
- The differential cryptanalysis attack requires 255.1 operations, whereas brute force requires 255.
- If DES had 15 or fewer rounds, differential cryptanalysis would require less effort than a brute-force key search.
- This criterion is attractive, because it makes it easy to judge the strength of an algorithm and to compare different algorithms.
- In the absence of a cryptanalytic breakthrough, the strength of any algorithm that satisfies the criterion can be judged solely on key length.

Design of Function F

- The heart of a **Feistel block cipher** is the function F, which provides the element of confusion in a Feistel cipher. Thus, it must be difficult to "**unscramble**" the substitution performed by F.
- One obvious criterion is that F be **nonlinear**. The more nonlinear F, the more difficult any type of cryptanalysis will be.
- The more difficult it is to approximate F by a set of linear equations, the more nonlinear F is. Several other criteria should be considered in designing F.
- We would like the algorithm to have good avalanche properties.
- A more stringent version of this is the **strict avalanche criterion (SAC)**, which states that any output bit *j* of an S-box should change with probability 1/2 when any single input bit *i* is inverted for all *i*, *j*.

- Although SAC is expressed in terms of S-boxes, a similar criterion could be applied to F as a whole. This is important when considering designs that do not include S-boxes.
- Another criterion proposed is the **bit independence criterion (BIC)**, which states that output bits *j* and *k* should change independently when any single input bit *i* is inverted for all *i*, *j*, and *k*. The SAC and BIC criteria appear to strengthen the effectiveness of the confusion function.

Key Schedule Algorithm

- With **any Feistel block cipher**, the key is used to generate one subkey for each round.
- In general, we would like to select subkeys to maximize the difficulty of deducing individual subkeys and the difficulty of working back to the main key.
- No general principles for this have yet been promulgated.
- At minimum, the key schedule should **guarantee key/ciphertext Strict Avalanche Criterion** and **Bit Independence Criterion.**

2.11. BLOCK CIPHER MODE OF OPERATION

Contents

- Electronic Code Book
- Cipher Block Chaining Mode
- Cipher Feedback Mode
- Output Feedback Mode
- Counter Mode

Electronic Code Book

- The simplest mode is the **Electronic codebook (ECB) mode**, in which plaintext is handled one block at a time and each block of plaintext is encrypted using the same key (Figure 6.3).
- The term codebook is used because, for a given key, there is a **unique ciphertext for every b-bit block of plaintext**.

Table 6.1 Block Cipher Modes of Operation

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of plaintext bits is encoded independently using the same key.	• Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next block of plaintext and the preceding block of ciphertext.	• General-purpose block-oriented transmission • Authentication
Cipher Feedback (CFB)	Input is processed s bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	• General-purpose stream-oriented transmission • Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding encryption output, and full blocks are used.	• Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	• General-purpose block-oriented transmission • Useful for high-speed requirements

- For a message longer than b bits, the procedure is simply to break the message into b-bit blocks, padding the last block if necessary.
- **Decryption is performed one block at a time**, always using the same key.
- We can define ECB mode as follows.

ECB	$C_j = E(K, P_j) \qquad j = 1, \ldots, N$	$P_j = D(K, C_j) \qquad j = 1, \ldots, N$

- The ECB method is ideal for a **short amount of data**, such as an encryption key.
- For lengthy messages, the ECB mode may not be secure. If the message is highly structured, it may be possible for a cryptanalyst to exploit these regularities.

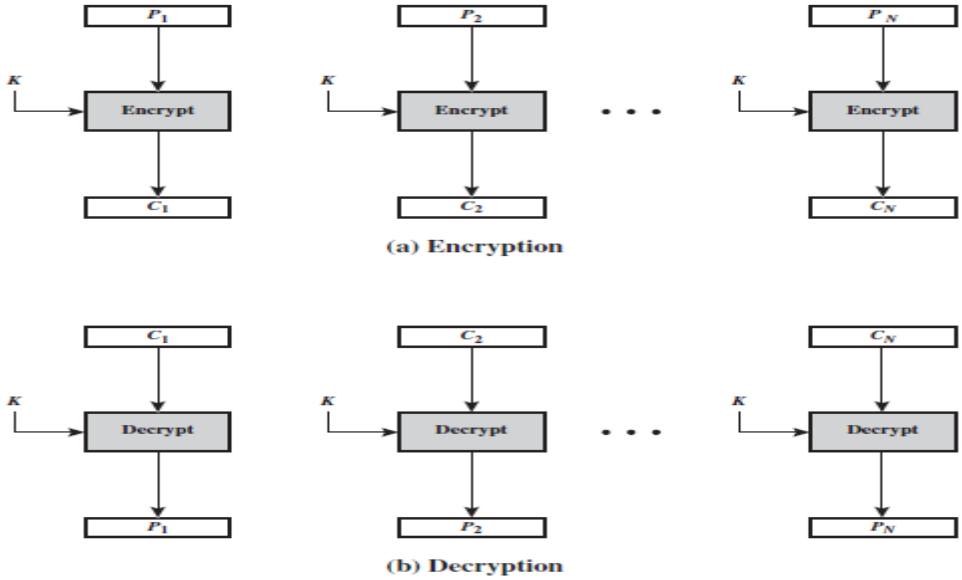


Figure 6.3 Electronic Codebook (ECB) Mode

Cipher Block Chaining Mode

- In this scheme, the input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext block; the same key is used for each block.
- In effect, we have chained together the processing of the sequence of plaintext blocks.
- The input to the encryption function for each plaintext block bears no fixed relationship to the plaintext block. Therefore, repeating patterns of b bits are not exposed.
- The result is XORed with the preceding ciphertext block to produce the plaintext block. To see that this works, we can write

$$C_j = \mathrm{E}(K, [C_{j-1} \oplus P_j])$$

Then

$$\mathrm{D}(K, C_j) = \mathrm{D}(K, E(K, [C_{j-1} \oplus P_j]))$$
$$\mathrm{D}(K, C_j) = C_{j-1} \oplus P_j$$
$$C_{j-1} \oplus \mathrm{D}(K, C_j) = C_{j-1} \oplus C_{j-1} \oplus P_j = P_j$$

- To produce the first block of ciphertext, an initialization vector (IV) is XORed with the first block of plaintext. On decryption, the IV is XORed with the output of the decryption algorithm to recover the first block of plaintext.
- We can define CBC mode as

CBC	$C_1 = \mathrm{E}(K, [P_1 \oplus \mathrm{IV}])$ $C_j = \mathrm{E}(K, [P_j \oplus C_{j-1}])\ j = 2, \ldots, N$	$P_1 = \mathrm{D}(K, C_1) \oplus \mathrm{IV}$ $P_j = \mathrm{D}(K, C_j) \oplus C_{j-1}\ j = 2, \ldots, N$

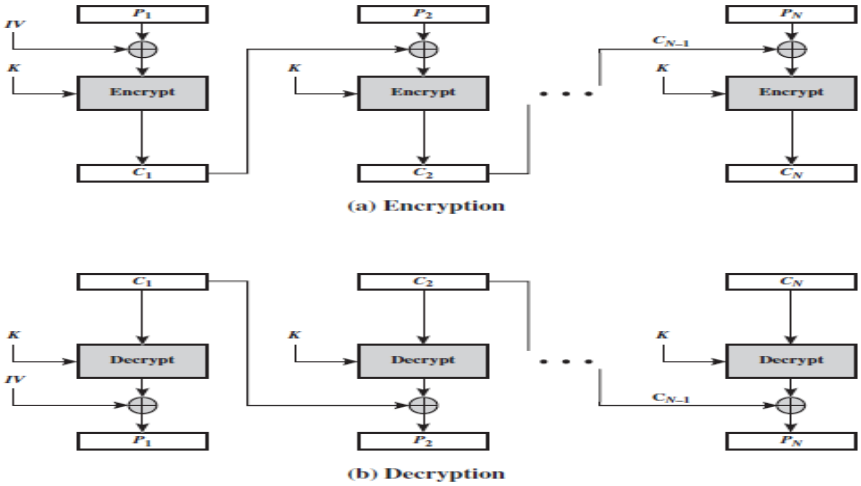


Figure 6.4 Cipher Block Chaining (CBC) Mode

Cipher Feedback Mode

- As with CBC, the units of plaintext are chained together, so that the cipher text of any plaintext unit is a function of all the preceding plaintext.
- In this case, rather than blocks of b bits, the plaintext is divided into segments of s bits.
- First, consider encryption. The input to the encryption function is a b-bit shift register that is initially set to some initialization vector (IV).
- **The leftmost (most significant) s** bits of the output of the encryption function are XORed with the first segment of plaintext P1 to produce the first unit of ciphertext C1, which is then transmitted.
- In addition, the contents of the shift register are shifted left by s bits, and C1 is placed in the rightmost (least significant) s bits of the shift register.
- This process continues until all plaintext units have been encrypted. For decryption, the same scheme is used, except that the received ciphertext unit is XORed with the output of the encryption function to produce the plaintext unit
- This is easily explained. Let MSBs(X) be defined as the **most significant s bits of X.** Then

$$C_1 = P_1 \oplus \text{MSB}_s[\text{E}(K, \text{IV})]$$

Therefore, by rearranging terms:

$$P_1 = C_1 \oplus \text{MSB}_s[\text{E}(K, \text{IV})]$$

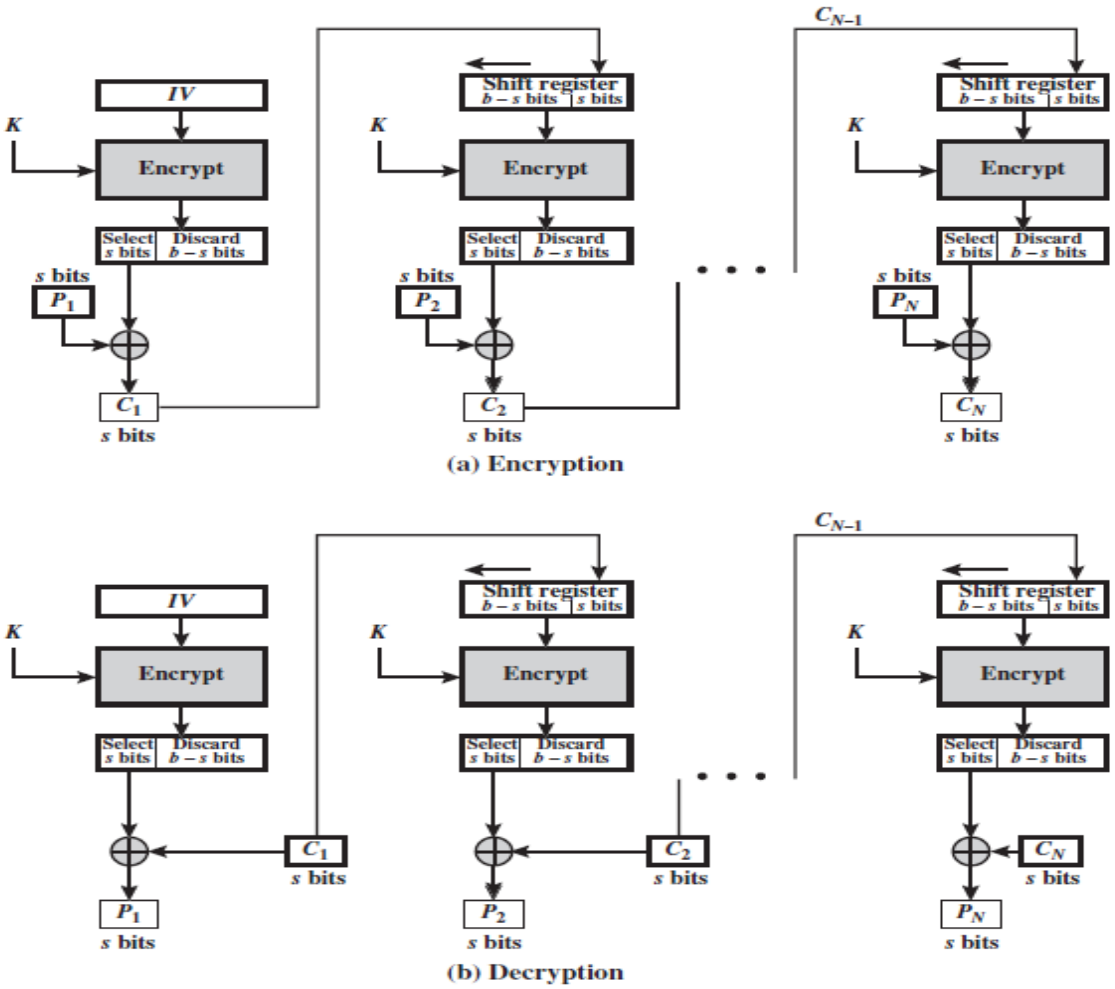


Figure 6.5 *s*-bit Cipher Feedback (CFB) Mode

We can define CFB mode as follows.

CFB	$I_1 = IV$ $I_j = \text{LSB}_{b-s}(I_{j-1}) \| C_{j-1}$ $j = 2, \ldots, N$ $O_j = \text{E}(K, I_j)$ $j = 1, \ldots, N$ $C_j = P_j \oplus \text{MSB}_s(O_j)$ $j = 1, \ldots, N$	$I_1 = IV$ $I_j = \text{LSB}_{b-s}(I_{j-1}) \| C_{j-1}$ $j = 2, \ldots, N$ $O_j = \text{E}(K, I_j)$ $j = 1, \ldots, N$ $P_j = C_j \oplus \text{MSB}_s(O_j)$ $j = 1, \ldots, N$

Output feedback (OFB) mode

- The output feedback (OFB) mode is similar in structure to that of CFB. For OFB, the output of the encryption function is feed back to become the input for encrypting the next block of plaintext (Figure 6.6).
- The other difference is that the OFB mode operates on full blocks of plaintext and ciphertext, whereas CFB operates on an s-bit subset. OFB encryption can be expressed as

$$C_j = P_j \oplus \mathrm{E}(K, O_{j-1})$$

where

$$O_{j-1} = \mathrm{E}(K, O_{j-2})$$

Some thought should convince you that we can rewrite the encryption expression as:

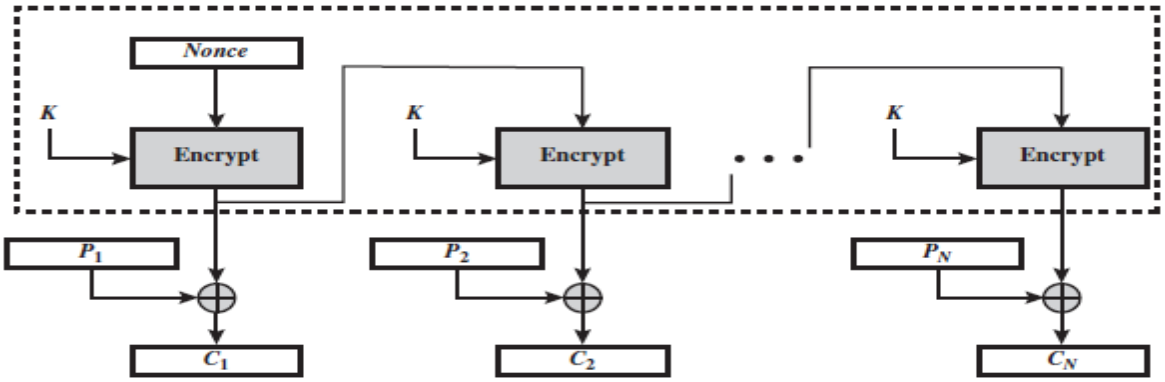
$$C_j = P_j \oplus \mathrm{E}(K, [C_{j-1} \oplus P_{j-1}])$$

By rearranging terms, we can demonstrate that decryption works.

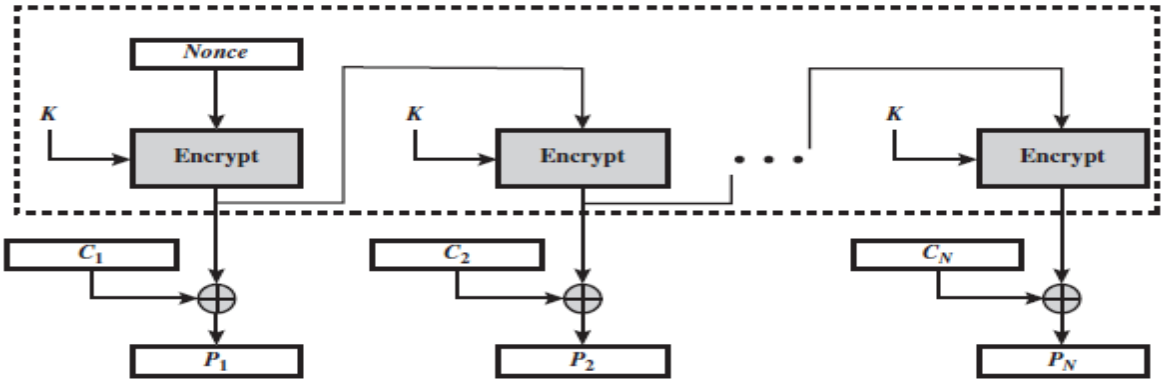
$$P_j = C_j \oplus \mathrm{E}(K, [C_{j-1} \oplus P_{j-1}])$$

We can define OFB mode as follows.

OFB	$I_1 = Nonce$ $I_j = O_{j-1}$ $\quad j = 2, \ldots, N$ $O_j = \mathrm{E}(K, I_j)$ $\quad j = 1, \ldots, N$ $C_j = P_j \oplus O_j$ $\quad j = 1, \ldots, N-1$ $C_N^* = P_N^* \oplus \mathrm{MSB}_u(O_N)$	$I_1 = Nonce$ $I_j = O_{j-1}$ $\quad j = 2, \ldots, N$ $O_j = \mathrm{E}(K, I_j)$ $\quad j = 1, \ldots, N$ $P_j = C_j \oplus O_j$ $\quad j = 1, \ldots, N-1$ $P_N^* = C_N^* \oplus \mathrm{MSB}_u(O_N)$



(a) Encryption



(b) Decryption

Figure 6.6 Output Feedback (OFB) Mode

Counter Mode

- Although interest in the **counter (CTR**) mode has increased recently with applications to ATM **(asynchronous transfer mode) network security and IP sec (IP security**), this mode was proposed early on (e.g., [DIFF79]).
- Figure 6.7 depicts the CTR mode. A counter equal to the plaintext block size is used.
- Typically, the counter is initialized to some value and then incremented by 1 for each subsequent block (modulo 2b, where b is the block size).
- For encryption, the counter is encrypted and then XORed with the plaintext block to produce the ciphertext block; there is no chaining.
- For decryption, the same sequence of counter values is used, with each encrypted counter XORed with a ciphertext block to recover the corresponding plaintext block. Thus, the initial counter value must be made available for decryption. Given a sequence of counters T1, T2, c, TN, we can define CTR mode as follows.

CTR	$C_j = P_j \oplus \mathrm{E}(K, T_j) \quad j = 1, \ldots, N-1$ $C_N^* = P_N^* \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)]$	$P_j = C_j \oplus \mathrm{E}(K, T_j) \quad j = 1, \ldots, N-1$ $P_N^* = C_N^* \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)]$

2.12. EVALUATION CRITERIA FOR AES

2.13. ADVANCED ENCRYPTION STANDARD

Finite Field Arithmetic

- In AES, all operations are performed on 8-bit bytes. In particular, the arithmetic operations of addition, multiplication, and division are performed over the finite field.
- In essence, a field is a set in which we can do **addition, subtraction, multiplication, and division** without leaving the set.
- Division is defined with the following rule: ***a*/*b* = *a*(*b*-1).**

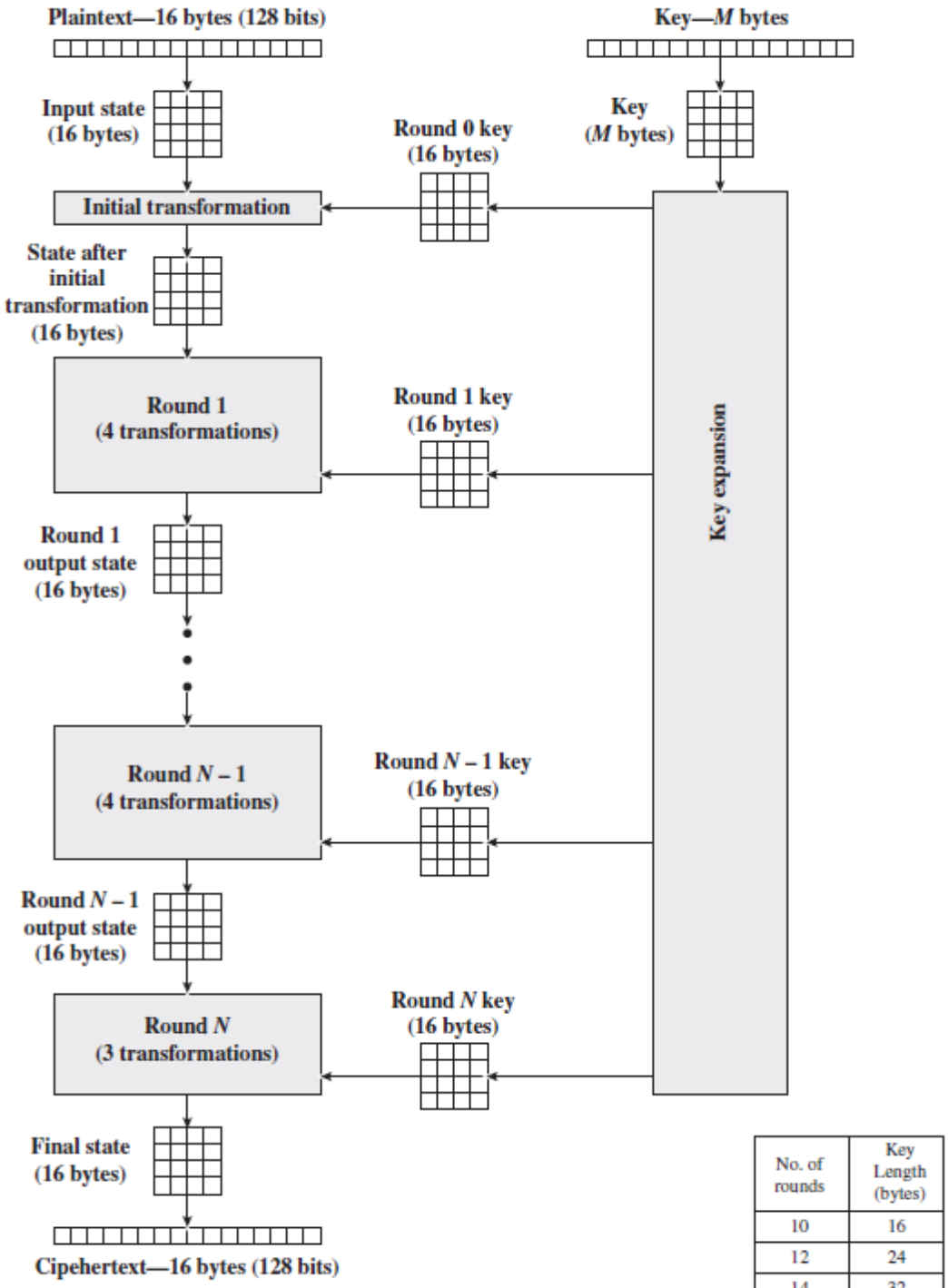
AES Structure

- General Structure
- Detailed Structure

General Structure

- Figure(5.1). shows the overall structure of the **AES encryption process**.
- The cipher takes a plaintext block size of **128 bits, or 16 bytes**.
- The key length can be 16, 24, or 32 bytes (128, 192, or 256 bits). The algorithm is referred to as AES-128, AES-192, or AES-256, depending on the key length.

- The input to the encryption and decryption algorithms is a single 128-bit block. In FIPS PUB 197, this block is depicted as a 4 * 4 square matrix of bytes. This block is copied into the **State** array, which is modified at each stage of **encryption or decryption**.
- After the final stage, **State** is copied to an output matrix.
- This key is then expanded into an array of key schedule words.
- Each word is four bytes, and the total key schedule is 44 words for the 128-bit key
- The **cipher** consists of *N* rounds, where the number of rounds depends on the key length: 10 rounds for a 16-byte key, 12 rounds for a 24-byte key, and 14 rounds for a 32-byte key.
- The first *N* - 1 rounds consist of four distinct transformation functions: **SubBytes, ShiftRows, MixColumns, and AddRoundKey**, which are described subsequently.
- The final round contains only three transformations, and there is a initial single transformation **(AddRoundKey)** before the first round, which can be considered Round 0. Each transformation takes one or more 4 * 4 matrices as input and produces a 4 * 4 matrix as output.



No. of rounds	Key Length (bytes)
10	16
12	24
14	32

Fig : AES Encryption Process

Table 5.1 AES Parameters

Key Size (words/bytes/bits)
Plaintext Block Size (words/bytes/bits)
Number of Rounds
Round Key Size (words/bytes/bits)
Expanded Key Size (words/bytes)

Detailed Structure

Figure (5.1) shows the AES cipher in more detail, indicating the sequence of transformations in each round and showing the corresponding decryption function.

We can make several comments about the overall AES structure.

1. One noteworthy feature of this structure is that it **is not a Feistel structure**
2. The key that is provided as input is expanded into an array of forty-four 32-bit words, **w**[*i*]. Four
   distinct words (128 bits) serve as a round key for each round; these are indicated in Figure 5.**3**

3.Four different stages are used, one of permutation and three of substitution:

• **Substitute bytes:** Uses an S-box to perform a byte-by-byte substitution of the block

• **ShiftRows:** A simple permutation

• **MixColumns:** A substitution that makes use of arithmetic over GF(28)

• **AddRoundKey:** A simple bitwise XOR of the current block with a portion of the expanded Key.

4. The structure is **quite simple**.

5. Only the **AddRoundKey** stage makes use of the key. For this reason, the cipher begins and ends with an AddRoundKey stage.

6. The AddRoundKey stage is, in effect, a form of Vernam cipher and by itself would not be formidable. The other three stages together provide confusion, diffusion, and nonlinearity, but by themselves would provide no security because they do not use the key.

7. Each stage is easily reversible.

8. Once it is established that all four stages are reversible, it is easy to verify that decryption does recover the plaintext.

9. The final round of both encryption and decryption consists of only three stages.Again, this is a consequence of the particular structure of AES and is required to make the cipher reversible

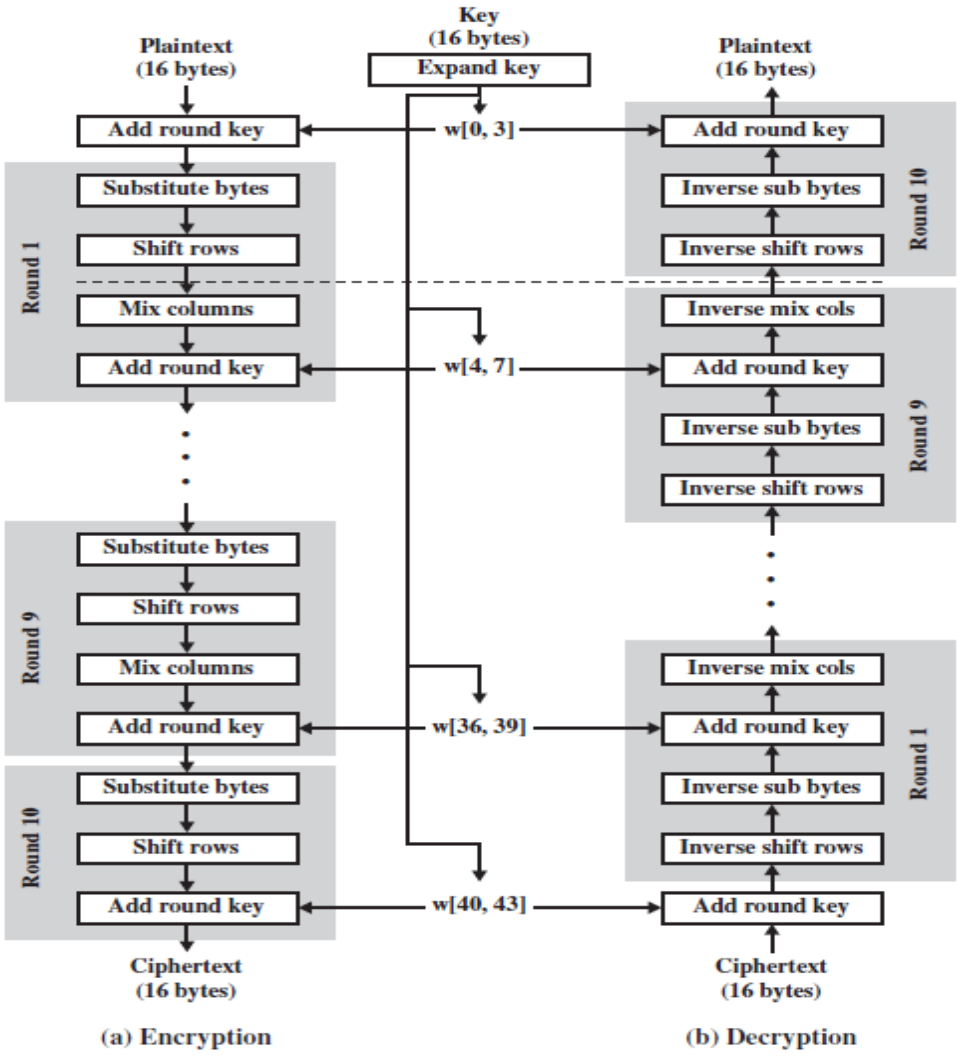


Fig (5.3): AES Encryption and Decryption

AES Transformation Functions

The four transformations used in AES. For each stage, we describe the forward (encryption) algorithm, the inverse (decryption) algorithm, and the rationale for the stage.

- Substitute Bytes Transformation
- Shift Rows Transformation
- Mix Columns Transformation
- AddRoundKey Transformation

Substitute Bytes Transformation

- The **forward substitute byte transformation**, called SubBytes, is a simple table lookup. AES defines a 16 * 16 matrix of byte values, called **an S-box**, that contains a permutation of all possible 256 8-bit values.
- Each individual byte of **State is mapped into a new byte in the following way:**

- The leftmost 4 bits of the byte are used as a row value and the rightmost 4 bits are used as a column value.
- These row and column values serve as indexes into the S-box to select a unique 8-bit output value.

Shift Rows Transformation

- The first row of State is not altered.
- For the second row, a 1-byte circular left shift is performed.
- For the third row, a 2-byte circular left shift is performed.
- For the fourth row, a 3-byte circular left shift is performed.
- The inverse shift row transformation, called InvShiftRows, performs the circular shifts in the opposite direction for each of the last three rows, with a 1-byte circular right shift for the second row, and so on.

MixColumns Transformation

- MixColumns, operates on each column individually.
- Each byte of a column is mapped into a new value that is a function of all four bytes in that column.

AddRoundKey Transformation

- AddRoundKey, the 128 bits of State are bitwise XORed with the 128 bits of the round key.

2.14. RC4

Contents
• **Characteristics** • **RC5 Parameters** • **Key Expansion** • **Encryption** • **Decryption** • **RC5 Modes**

RC5 is a symmetric encryption algorithm developed by Ron Rivest. RC5 was designed to have the **following characteristics:**

- **Suitable for hardware or software**
- **Fast**
- **Adaptable to processors of different word lengths**
- **Variable number of rounds:**
- **Variable-length key**

- **Simple** .
- **Low memory requirement**
- **High security**

❖ RC5 has been incorporated into RSA Data Security, Inc-'s major products, including BSAFE, JSAFE, and S/MAIL.

RC5 Parameters

RC5 is actually a family of encryption algorithms determined by three parameters, as follows:

Parameter	Definition	Allowable Values
w	Word size in bits. RC5 encrypts 2-word blocks	16, 32, 64
r	Number of rounds	0, 1, ..., 255
b	Number of 8-bit bytes (octets) in the secret key K	0, 1, ..., 255

Key Expansion

❖ RC5 performs a complex set of operations on the secret key to produce a total of *t* subkeys. Two subkeys are used in each round, and two subkeys are used on an additional operation that is not part of any round, so $t = 2r + 2$. Each subkey is one Word (w bits) in length.

❖ Figure 4-11 illustrates the technique used to generate subkeys; The subkeys are stored in a *t*-word array labeled S[0], S[1], ….,' S[t-1]. Using the parameters *r* and *w* as inputs, this array is initialized to a particular fixed pseudorandom bit pattern.

❖ Then the *b*-byte key, K[0…. *b* - 1], is converted into a *c*-word array L[0…. *c -1*]. On a little endian machine, this is accomplished by zeroing out the array *L* and copying the string *K* directly into the memory positions represented by L.

❖ If *b* is not an integer multiple of w, then a portion of L at the right end remains zero-Finally, a mixing operation is performed that applies the contents of *L* to the initialized value of S to produce a final value for the array S.

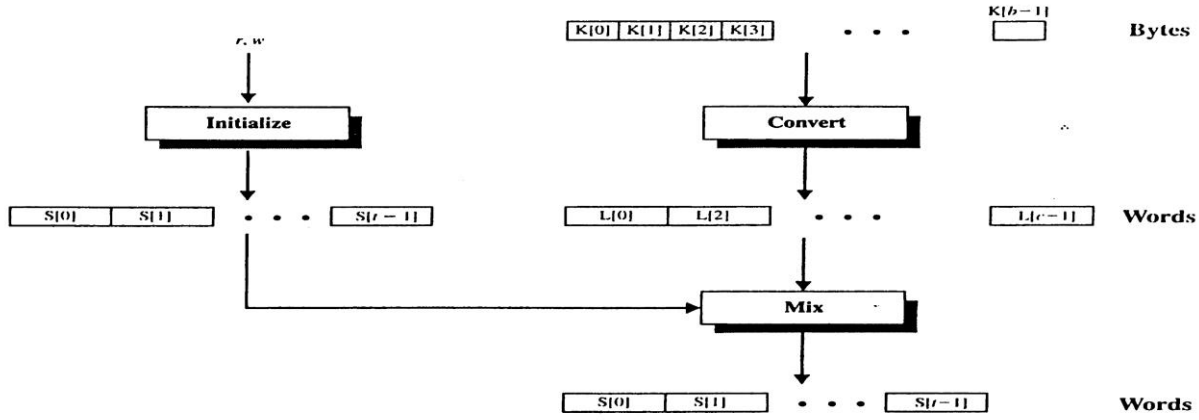


Figure 4.11 RC5 Key Expansion.

Let us look at this operations in detail. The initialize operation makes use of two word-length constants defined as follows,

$$P_w = \text{Odd}[(e-2)2^w]$$
$$Q_w = \text{Odd}[(\phi-1)2^w]$$

Where

$e = 2.718281828459 \ldots$ (base of natural logarithms)

$\phi = 1.618033988749 \ldots$ (golden ratio[3]) $= \left(\frac{1+\sqrt{5}}{2}\right)$

Encryption:

- RC5 uses three primitive operations (and their inverses):
  - **Addition:** Addition of words, denoted by +, is performed modulo 2^w. The inverse operation, denoted by -, is subtraction modulo 2^w.
  - **Bitwise exclusive-OR:** This operation is denoted by $\oplus$.
  - **Left circular rotation:** The cyclic rotation of word x left by y bits is denoted by $x <<< y$. The inverse is the right circular rotation of word x by y bits, denoted by $x >>> y$.

Figure 4-12a depicts the encryption operation. Note that this is not a classic Feistel structure. The plaintext is assumed to initially reside in the two w-bit registers A and B.

We use the variables LE_i and RE_i to refer to the left and right half of the data after round i has completed.

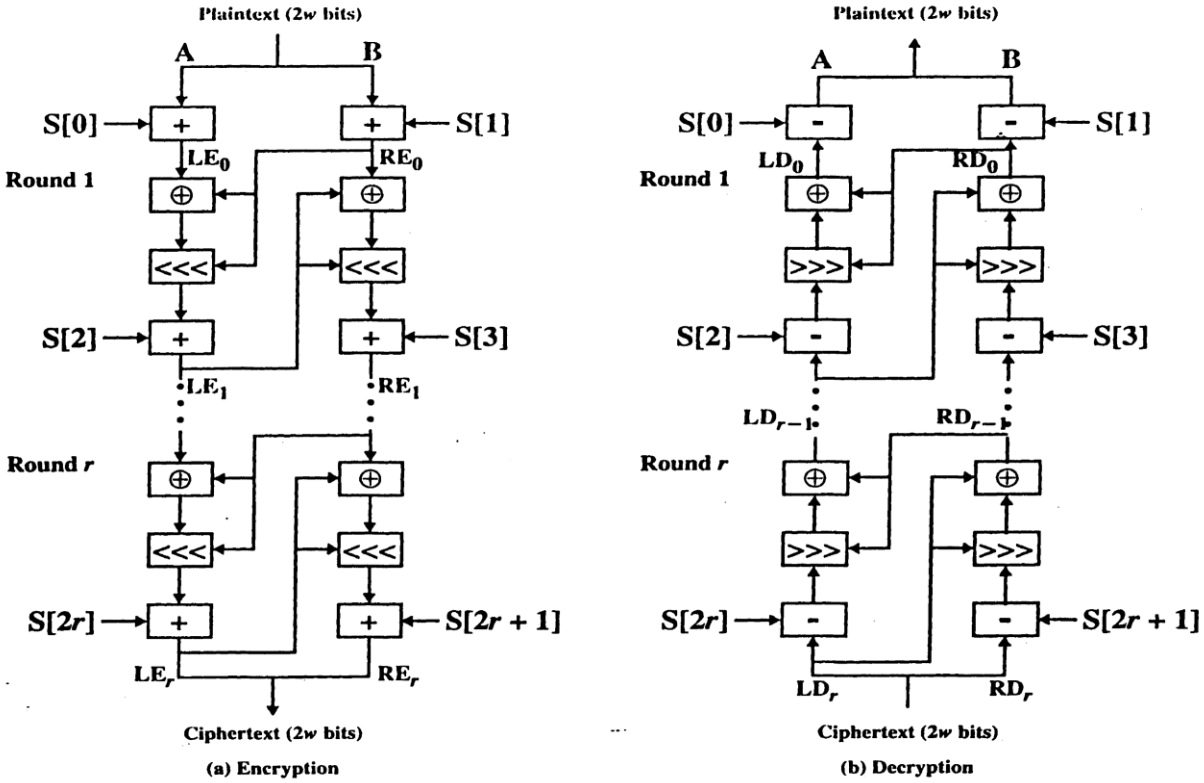


Figure 4.12 RC5 Encryption and Decryption.

Decryption

- Decryption, shown in Figure 4-12b, is easily derived from the encryption algorithm. In this case, the 2*w* bits of ciphertext are initially assigned to the two one-word variables LD_r, and RD_r.
- We use the variables LD_i and RD_i to refer to the left and right half of the data before round *i* has begun, where the rounds are numbered from *r* down to 1.

RC5 Modes:

To enhance the effectiveness of RC5 in interoperable implementations, RFC 2040 defines four different modes of operation:

- **RC5 block cipher:** This is the raw encryption algorithm that takes a fixed—size input block (2*w* bits) and produces a ciphertext block of the same length using a transformation that depends on a key.
- **RCS-CBC:** This is the cipher block chaining mode for RC5- CBC. CBC processes messages whose length is a multiple of the RC5 block size (multiples of 2w bits. CBC provides enhanced security compared to ECB because repeated blocks of plaintext produce different blocks of ciphertext.

- **RCS-CBC-Pad:** This is a CBC style of algorithm that handles plaintext of any length- The ciphertext will be longer than the plaintext by at most the size of a single RC5 block.
- **RCS-CTS:** This is the ciphertext stealing mode, which is also a CBC style of algorithm- This mode handles plaintext of any length and produces ciphertext of equal length.

The encryption sequence is as follows:

1. Encrypt the first $(N - 2)$ blocks using the traditional CBC technique.
2. Exclusive-OR P_{N-1} with the previous ciphertext block C_{N-2} to create Y_{N-1}.
3. Encrypt Y_{N-1} to create E_{N-1}.
4. Select the first L bytes of E_{N-1} to create C_N.
5. Pad P_N with zeros at the end and exclusive-OR with E_{N-1} to create Y_N.
6. Encrypt Y_N to create C_{N-1}.

2.15. KEY DISTRIBUTION

CS8792 CRYPTOGRAPHY AND NETWORK SECURITY

UNIT I
INTRODUCTION

Security trends – Legal, Ethical and Professional Aspects of Security, Need for Security at Multiple levels, Security Policies – Model of network security – Security attacks, services and mechanisms – OSI security architecture – Classical encryption techniques: substitution techniques, transposition techniques, steganography- Foundations of modern cryptography: perfect security – information theory – product cryptosystem – cryptanalysis.

1.1. SECURITY TRENDS

Internet Architecture Board (IAB) has issued report entitled "Security in the Internet Architecture" where they have identified key areas for security mechanisms. Among these were 1. need to secure network and 2. need to secure end to end transmission.

These concerns are fully justified. As confirmation, consider the trends reported by the Computer Emergency Response Team (CERT) Coordination Center (CERT/CC).

Figure 1.1 a shows the trend in Internet-related vulnerabilities reported to CERT over a 10-year period. These include security weaknesses in the operating systems of attached computers (e.g., Windows, Linux) as well as vulnerabilities in Internet routers and other network devices.

Figure 1.1 b shows the number of security-related incidents reported to CERT. These include denial of service attacks; IP spoofing, in which intruders create packets with false IP addresses and exploit applications that use authentication based on IP; and various forms of eavesdropping and packet sniffing, in which attackers read transmitted information, including logon information and database contents.

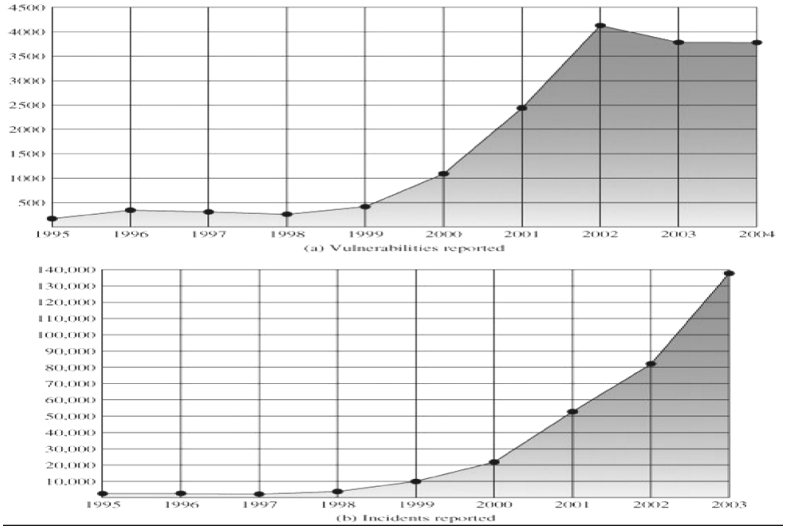


Figure 1.1 a - Trend in Internet-related vulnerabilities reported to CERT
1.1.b - Number of security-related incidents reported to CERT

Over time, the attacks on the Internet and Internet-attached systems have grown more sophisticated while the amount of skill and knowledge required to mount an attack has declined (Figure 1.2).

Attacks have become more automated and can cause greater amounts of damage. This increase in attacks coincides with an increased use of the Internet and with increases in the complexity of protocols, applications, and the Internet itself. Critical infrastructures increasingly rely on the Internet for operations. Individual users rely on the security of the Internet, email, the Web, and Web-based applications to a greater extent than ever. Thus, a wide range of technologies and tools are needed to counter the growing threat.

At a basic level, cryptographic algorithms for confidentiality and authentication assume greater importance. As well, designers need to focus on Internet-based protocols and the vulnerabilities of attached operating systems and applications.

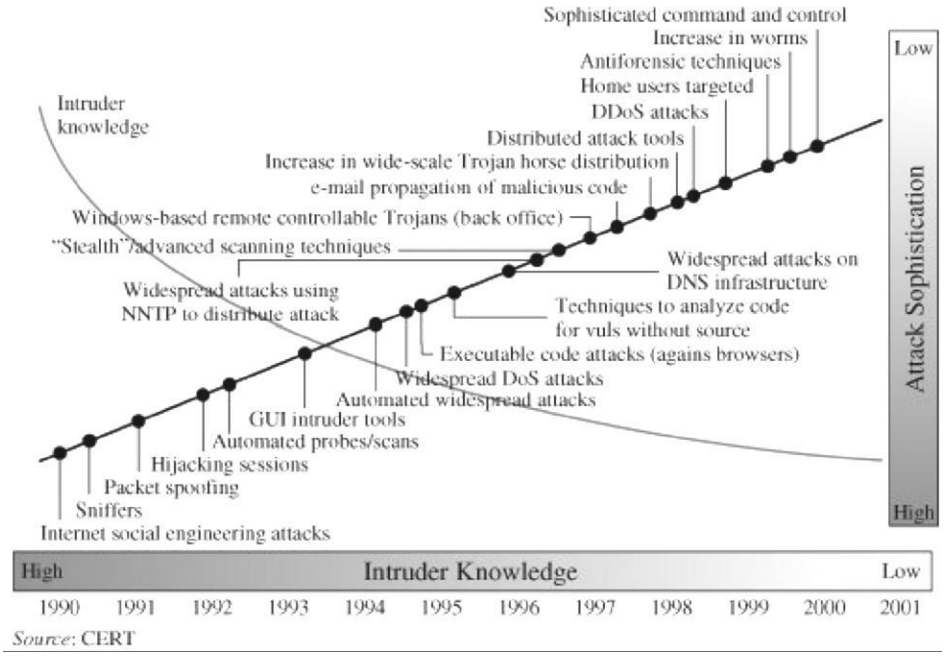


Figure 1.2 - CERT Statistics

1.2. LEGAL, ETHICAL AND PROFESSIONAL ASPECTS OF SECURITY

1.3. NEED FOR SECURITY AT MULTIPLE LEVELS

Multilevel security or multiple levels of security (MLS) is the application of a computer system to process information with incompatible classifications (i.e., at different security levels), permit access by users with different security clearances and needs-to-know, and prevent users from obtaining access to information for which they lack authorization. There are two contexts for the use of multilevel security. One is to refer to a system that is adequate to protect itself from subversion and has robust mechanisms to separate information domains, that is, trustworthy. Another context is to refer to an application of a computer that will require the computer to be strong enough to protect itself from subversion and possess adequate mechanisms to separate information domains, that is, a system we must trust. This distinction is important because systems that need to be trusted are not necessarily trustworthy.

1.4. SECURITY POLICIES

1.5. MODEL OF NETWORK SECURITY

- A message is to be transferred from one party to another across some sort of Internet service.

- A logical information channel is established by defining a route through the Internet from source to destination and by the cooperative use of communication protocols (e.g., TCP/IP) by the two principals.
- Security aspects come into play when it is necessary or desirable to protect the information transmission from an opponent who may present a threat to confidentiality, authenticity, and so on. All the techniques for providing security have two components:
    - A ***security-related transformation*** on the information to be sent.
    - Some ***secret information shared by the two principals*** and, it is hoped, unknown to the opponent. A trusted third party may be needed to achieve secure transmission.

This general model shows that there are four basic tasks in designing a particular security service:

1. Design an algorithm for performing the security-related transformation. The algorithm should be such that an opponent cannot defeat its purpose.

2. Generate the secret information to be used with the algorithm.

3. Develop methods for the distribution and sharing of the secret information.

4. Specify a protocol to be used by the two principals that makes use of the security algorithm and the secret information to achieve a particular security service.

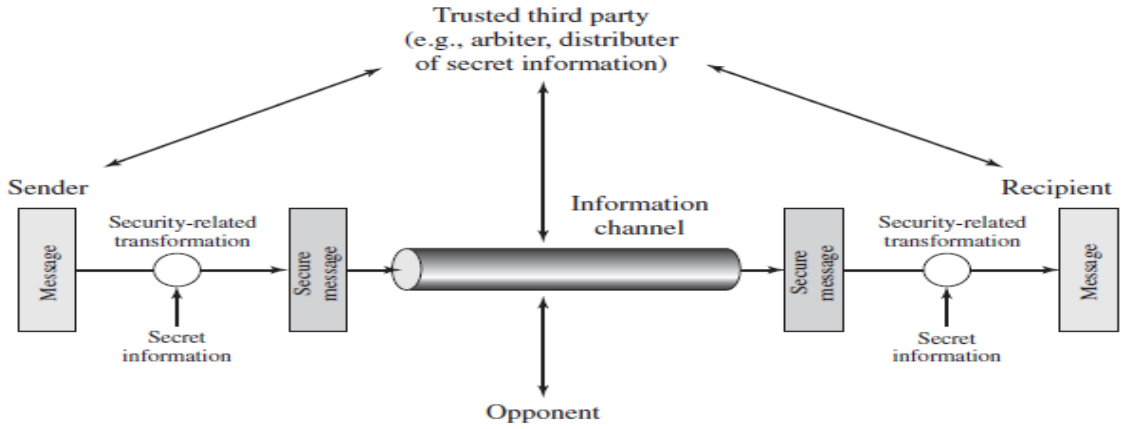


Figure 1.2 Model for Network Security

- A general model of these other situations is illustrated in Figure 1.3, which reflects a concern for protecting an information system from unwanted access.
- The hacker can be someone who, with no malign intent, simply gets satisfaction from breaking and entering a computer system.

Programs can present two kinds of threats:
Information access threats: Intercept or modify data on behalf of users who should not have access to that data.
Service threats: Exploit service flaws in computers to inhibit use by legitimate users.

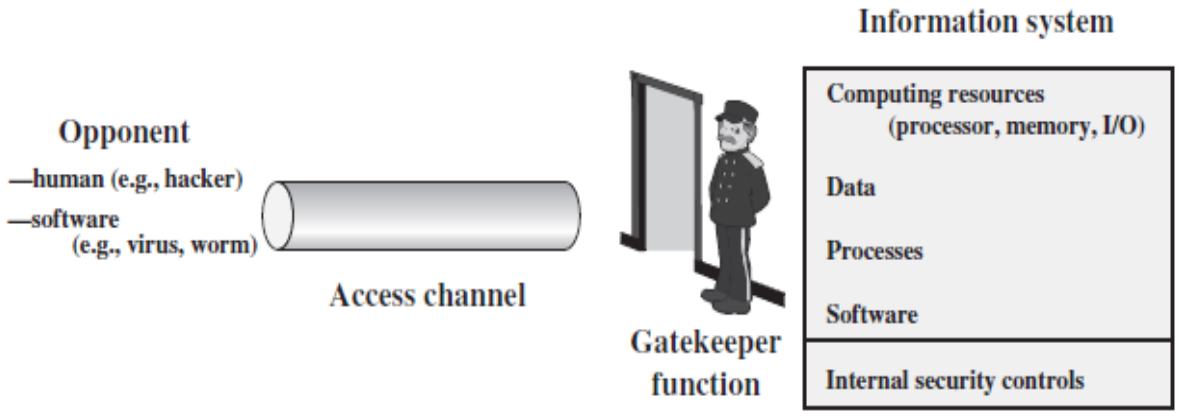


Figure 1.3 Network Access Security Model

- ***Viruses* and *worms*** are two examples of software attacks. They can also be inserted into a system across a network.

- The security mechanisms needed to cope with unwanted access fall into two broad categories (see Figure 1.3).
  - The first category might be termed **a gatekeeper function.** It includes password-based login procedures and screening logic that is designed to detect and reject worms, viruses.
  - The second line of defense consists of a **variety of internal controls** that monitor activity and analyze stored information in an attempt to detect the presence of unwanted intruders.

1.6. SECURITY ATTACKS

Contents

• **Introduction**
• **Passive Attacks**
  ○ the release of message contents and
  ○ traffic analysis.
• **Active Attacks**
  ○ masquerade,
  ○ replay,
  ○ modification of messages, and
  ○ denial of service.

Introduction

- security attacks, uses both in X.800 and RFC 2828, is in terms of ***passive attacks* and *active attacks.***
- A passive attack attempts to learn or make use of **information from the system but does not affect system resources.**
- An active attack attempts to alter **system resources or affect their operation.**

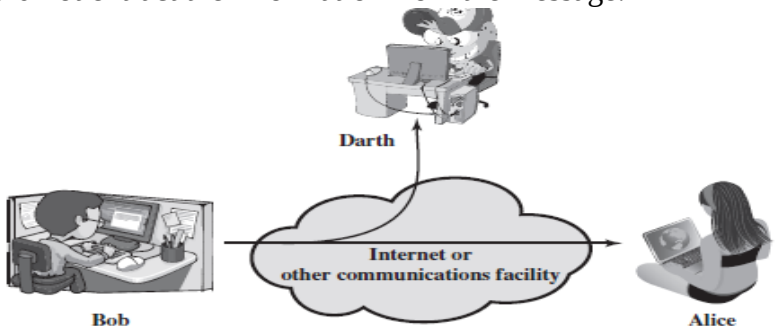
Passive Attacks

- Passive attacks (Figure 1.1) are in the nature of eavesdropping on, or monitoring of, transmissions.
- The goal of the opponent is to obtain information that is being transmitted.
- Two types of passive attacks are :

- the release of message contents and
- traffic analysis.

❖ The **release of message contents** is easily understood. A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information.

❖ A **traffic analysis**, is subtler. Suppose that we had a way of masking the contents of messages or other information traffic so that opponents, even if they captured the message, could not extract the information from the message.



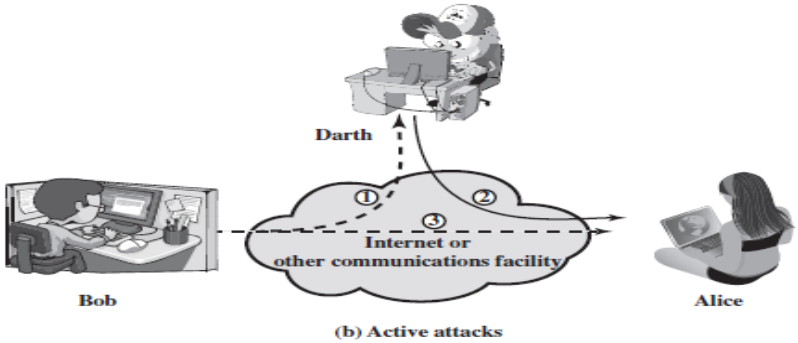
(a) Passive attacks

Active Attacks

❖ Active attacks (Figure 1.1b) involve some modification of the data stream or the creation of a false stream and can be subdivided into **four categories:**

- masquerade,
- replay,
- modification of messages, and
- denial of service.

- A **masquerade -** A masquerade attack usually includes one of the other forms of active attack.
- **Replay** involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect.
- **Modification of messages** simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect.
- The **denial of service** prevents or inhibits the normal use or management of communications facilities.



(b) Active attacks

Figure 1.1 Security Attacks

1.7. SECURITY SERVICES

Contents
1. Introduction

2. **Authentication**
   - Peer Entity Authentication
   - Data Origin Authentication
3. **Access Control**
4. **Data Confidentiality**
   - Connection Confidentiality
   - Connectionless Confidentiality
   - Selective-Field Confidentiality
   - Traffic Flow Confidentiality
5. **Data Integrity**
   - Connection Integrity with Recovery
   - Connection Integrity without Recovery
   - Selective-Field Connection Integrity
   - Connectionless Integrity
   - Selective-Field Connectionless Integrity
6. **Nonrepudiation**
   - Nonrepudiation, Origin
   - Nonrepudiation, Destination
7. **Availability Service**

Introduction

- Security services is defined as a processing or communication service that is provided by a system to give a specific kind of protection to system resources.
- X.800 divides these services into five categories and fourteen specific services.

1. Authentication

- The assurance that the communicating entity is the one that it claims to be.

Two specific authentication services are defined in X.800:

- **Peer Entity Authentication**
  Used in association with a logical connection to provide confidence in the identity of the entities connected.
- **Data Origin Authentication**
  In a connectionless transfer, provides assurance that the source of received data is as claimed.

2. Access Control

- The prevention of unauthorized use of a resource.

3. Data Confidentiality:

- The protection of data from unauthorized disclosure.
  - **Connection Confidentiality**
    The protection of all user data on a connection.
  - **Connectionless Confidentiality**
    The protection of all user data in a single data block
  - **Selective-Field Confidentiality**
    The confidentiality of selected fields within the user data on a connection or in a single data block.
  - **Traffic Flow Confidentiality**
    The protection of the information that might be derived from observation of traffic flows.

4. **Data Integrity:**
   - ❖ The assurance that data received are exactly as sent by an authorized entity (i.e.,contain no modification, insertion, deletion, or replay).
     - o **Connection Integrity with Recovery**

     Provides for the integrity of all user data on a connection and detects any modification, with recovery attempted.
     - o **Connection Integrity without Recovery**

     As above, but provides only detection without recovery.
     - o **Selective-Field Connection Integrity**

     Provides for the integrity of selected fields within the user data of a data block transferred over a connection.
     - o **Connectionless Integrity**

     Provides for the integrity of a single connectionless data block.
     - o **Selective-Field Connectionless Integrity**

     Provides for the integrity of selected fields within a single connectionless data block;
5. **Nonrepudiation**
   - ❖ Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication.
     - o **Nonrepudiation, Origin**

       Proof that the message was sent by the specified party.
     - o **Nonrepudiation, Destination**

       Proof that the message was received by the specified party.

1.8. SECURITY MECHANISMS

Contents
• **Introduction** • **Encipherment** • **Digital Signature** • **Access Control** • **Data Integrity** • **Authentication Exchange** • **Traffic Padding** • **Routing Control** • **Notarization** • **Pervasive Security Mechanisms** • **Trusted Functionality** • **Security Label** • **Event Detection** • **Security Audit Trail**

Introduction

- ❖ The mechanisms are divided into those that are implemented in a specific protocol layer, such as TCP or an application-layer protocol, and those that are not specific to any particular protocol layer or security service.

Encipherment

- ❖ The use of mathematical algorithms to transform data into a form that is not readily intelligible.

Digital Signature

- Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery (e.g., by the recipient).

Access Control

- A variety of mechanisms that enforce access rights to resources.

Data Integrity

- A variety of mechanisms used to assure the integrity of a data unit or stream of data units.

Authentication Exchange

- A mechanism intended to ensure the identity of an entity by means of information exchange.

Traffic Padding

- The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.

Routing Control

- Enables selection of particular physically secure routes for certain data and allows routing changes, especially when a breach of security is suspected.

Notarization

- The use of a trusted third party to assure certain properties of a data exchange.

Pervasive Security Mechanisms

- Mechanisms those are not specific to any particular OSI security service or protocol layer.

Trusted Functionality

- That which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).

Security Label

- The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.

Event Detection

- Detection of security-relevant events.

Security Audit Trail

- Data collected and potentially used to facilitate a security audit, which is an independent review and examination of system records and activities.
- A reversible encipherment mechanism is simply an encryption algorithm that allows data to be encrypted and subsequently decrypted.
- Irreversible encipherment mechanisms include hash algorithms and message authentication codes, which are used in digital signature and message authentication applications.

1.9. OSI SECURITY ARCHITECTURE

To assess effectively the security needs of an organization and to evaluate and choose various security products and policies, the manager responsible for security needs some systematic way of defining the requirements for security and characterizing the approaches to satisfying those requirements. This is difficult enough in a centralized data processing environment; with the use of local and wide area networks, the problems are compounded.

ITU-T3 Recommendation X.800, *Security Architecture for OSI*, defines such a systematic approach.4 The OSI security architecture is useful to managers as a way of organizing the task of providing security. Furthermore, because this architecture was developed as an international standard, computer and communications vendors have developed security

features for their products and services that relate to this structured definition of services and mechanisms.

For our purposes, the OSI security architecture provides a useful, if abstract, overview of many of the concepts that this book deals with. The OSI security architecture focuses on security attacks, mechanisms, and services. These can be defined briefly as

■ **Security attack:**

Any action that compromises the security of information owned by an organization.

■ **Security mechanism:**

A process (or a device incorporating such a process) that is designed to detect, prevent, or recover from a security attack.

■ **Security service:**

A processing or communication service that enhances the security of the data processing systems and the information transfers of an organization. The services are intended to counter security attacks, and they make use of one or more security mechanisms to provide the service.

1.10. CLASSICAL ENCRYPTION TECHNIQUES

Symmetric encryption, also referred to as conventional encryption or single-key encryption, was the only type of encryption in use prior to the development of public key encryption in the 1970s. It remains by far the most widely used of the two types of encryption.

Terminologies:

- An original message is known as the **plaintext**, while the coded message is called the **ciphertext**.
- The process of converting from plaintext to ciphertext is known as **enciphering** or **encryption**;
- Restoring the plaintext from the ciphertext is **deciphering** or **decryption**.
- The many schemes used for encryption constitute the area of study known as **cryptography**. Such a scheme is known as a **cryptographic system** or a **cipher**.
- Techniques used for deciphering a message without any knowledge of the enciphering details fall into the area of **cryptanalysis**.
- Cryptanalysis is what the layperson calls "breaking the code."
- The areas of cryptography and cryptanalysis together are called **cryptology**.

1.10.1. SUBSTITUTION TECHNIQUES

Contents
• **Introduction** • **Caesar Cipher** • **Monoalphabetic Ciphers** • **Playfair Cipher** • **Hill Cipher** • **Polyalphabetic Ciphers** • **One-Time Pad**

Introduction

- ❖ The two basic building blocks of all encryption techniques are ***substitution*** and ***transposition***.
- ❖ A **substitution technique** is one in which the ***letters of plaintext are replaced by other letters*** or by numbers or symbols.

Caesar Cipher

- The Caesar cipher involves replacing each letter of the alphabet with the letter standing three places further down the alphabet. **For example**,

```
plain:  meet me after the toga party
cipher: PHHW PH DIWHU WKH WRJD SDUWB
```

- The algorithm can be expressed as follows.

$$C = \mathrm{E}(k, p) = (p + k) \bmod 26 \qquad \textbf{(2.1)}$$

- The decryption algorithm is simply

$$p = \mathrm{D}(k, C) = (C - k) \bmod 26 \qquad \textbf{(2.2)}$$

- If it is known that a given ciphertext is a Caesar cipher, then a brute-force cryptanalysis is easily performed: simply try all the 25 possible keys.

MonoalphabeticCiphers

- With only 25 possible keys, the Caesar cipher is far from secure.
- Before proceeding, we define the term *permutation*. A **permutation** of a finite set of elements *S* is an ordered sequence of all the elements of *S*, with each element appearing exactly once.
- **For example, if** *S* = {a, b, c}, there are six permutations of *S*:
  abc, acb, bac, bca, cab, cba
- If, instead, the "cipher" line can be any permutation of the 26 alphabetic characters, then there are 26! or greater than 4 * 1026 possible keys.

Playfair Cipher

- The best-known ***multiple-letter encryption*** cipher is the Playfair.
- The Playfair algorithm is based on the use of a 5 * 5 matrix of letters constructed using a keyword. Here is an example,

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

- In this case, the keyword is *monarchy*.
- The matrix is constructed by filling in the letters of the keyword (minus duplicates) from left to right and from top to bottom, and then filling in the remainder of the matrix with the remaining letters in alphabetic order. The letters I and J count as one letter.
- **The rules to be followed are:**
  - Repeating plaintext letters that come in the same pair are separated with a filer letter, such as x.
  - Plaintext letters that fall in the same row are replaced by the letter to the right, with the first element of the row circularly following the first.
  - Plaintext letters that fall in the same column are replaced by the letter beneath, with the top element circularly following the last.
  - Otherwise each letter is replaced by the letter that lies in its own row and the column occupied by the other plaintext.

Hill Cipher

- ***The Hill Algorithm*** This encryption algorithm takes *m* successive plaintext letters and substitutes for them *m* ciphertext letters.

In general terms, the Hill system can be expressed as

$$\mathbf{C} = \mathrm{E}(\mathbf{K}, \mathbf{P}) = \mathbf{PK} \bmod 26$$
$$\mathbf{P} = \mathrm{D}(\mathbf{K}, \mathbf{C}) = \mathbf{CK}^{-1} \bmod 26 = \mathbf{PKK}^{-1} = \mathbf{P}$$

Polyalphabetic Ciphers

- Another way to improve on the simple monoalphabetic technique is to use different monoalphabetic substitutions as one proceeds through the plaintext message.
- The general name for this approach is **polyalphabetic substitution cipher**.
- A general equation of the ***encryption process*** is

$$C_i = (p_i + k_{i \bmod m}) \bmod 26 \qquad \textbf{(2.3)}$$

- Similarly, ***decryption is***

$$p_i = (C_i - k_{i \bmod m}) \bmod 26 \qquad \textbf{(2.4)}$$

- To encrypt a message, a key is needed that is as long as the message.
- For example, if the keyword is **deceptive**, the message "**we are discovered save yourself"** is encrypted as

```
key:            deceptivedeceptivedeceptive
plaintext:      wearediscoveredsaveyourself
ciphertext:     ZICVTWQNGRZGVTWAVZHCQYGLMGJ
```

Vignere cipher

- Simplest polyalphabetic substitution cipher is the Vigenère Cipher.

- effectively multiple caesar ciphers
- key is multiple letters long K = k1 k2 ... kd
- i^{th} letter specifies i^{th} alphabet to use
- use each alphabet in turn
- repeat from start after d letters in message
- decryption simply works in reverse

Example:

- write the plaintext out
- write the keyword repeated above it
- use each key letter as a caesar cipher key
- encrypt the corresponding plaintext letter
- eg using keyword *deceptive*

key: deceptivedeceptivedeceptive

plaintext: wearediscoveredsaveyourself

ciphertext:ZICVTWQNGRZGVTWAVZHCQYGLMGJ

Security:

- have multiple ciphertext letters for each plaintext letter
- hence letter frequencies are obscured
- but not totally lost
- start with letter frequencies
  - see if look monoalphabetic or not
- if not, then need to determine the 'number of alphabets' in the key string (aka. the *period* of the key), since then can attach each

Kasisky Method:

- method developed by Babbage / Kasiski
- repetitions in ciphertext give clues to period
- so find same plaintext an exact period apart
- which results in the same ciphertext
- e.g., repeated "VTW" in previous example
- suggests size of 3 or 9
- then attack each monoalphabetic cipher individually using same techniques as before

Autokey cipher

- ideally want a key as long as the message
- Vigenère proposed the autokey cipher
- with keyword is prefixed to message as key
- knowing keyword can recover the first few letters
- use these in turn on the rest of the message
- but still have frequency characteristics to attack
- e.g., given key '*deceptive'*

key: deceptivewearediscoveredsav

plaintext: wearediscoveredsaveyourself

ciphertext:ZICVTWQNGKZEIIGASXSTSLVVWLA

One-Time Pad

- using a random key that is as long as the message, so that the key **need not be repeated**.
- In addition, the key is to be used to encrypt and decrypt a single message, and then is **discarded**.
- Each new message requires a new key of the same length as the new message. Such a scheme, known as a **one-time pad**, is **unbreakable.**

1.10.2. TRANSPOSITION TECHNIQUES

- A very different kind of mapping is achieved by performing **some sort of permutation** on the plaintext letters. This technique is referred to as a transposition cipher.
- The simplest such cipher is the **rail fence** technique, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows.
- For example, to encipher the message **"meet me after the toga party**" with a rail fence of depth 2, we write the following:

```
m e m a t r h t g p r y
 e t e f e t e o a a t
```

The encrypted message is

MEMATRHTGPRYETEFETEOAAT

1.10.3. STEGANOGRAPHY

Contents
• **Techniques** ✓ **Character marking** ✓ **Invisible ink** ✓ **Pin punctures** ✓ **Typewriter correction ribbon**

Techniques

- The methods of steganography **conceal the existence of the message**.
- **For example**, the sequence of first letters of each word of the overall message spells out the hidden message.
- **Various other techniques have been used historically; some examples are the following:**

  • **Character marking:** Selected letters of printed or typewritten text are overwritten in pencil. The marks are ordinarily not visible unless the paper is held at an angle to bright light.

  • **Invisible ink:** A number of substances can be used for writing but leave no visible trace until heat or some chemical is applied to the paper.

  • **Pin punctures:** Small pin punctures on selected letters are ordinarily not visible unless the paper is held up in front of a light.

  • **Typewriter correction ribbon:** Used between lines typed with a black ribbon, the results of typing with the correction tape are visible only under a strong light.

Advantages:

- **The advantage of steganography** is that it can be employed by parties who have something to lose should the fact of their secret communication (not necessarily the content) be discovered.
- Encryption flags traffic as important or secret or may identify the sender or receiver as someone with something to hide. Steganography has a number of drawbacks when compared to encryption.
- It requires a lot of overhead to hide a relatively few bits of information, although using a scheme like that proposed in the preceding paragraph may make it more effective.

1.11. FOUNDATIONS OF MODERN CRYPTOGRAPHY

Modern cryptography is the cornerstone of computer and communications security. Its foundation is based on various concepts of mathematics such as number theory, computational-complexity theory, and probability theory.

Characteristics of Modern Cryptography

There are three major characteristics that separate modern cryptography from the classical approach.

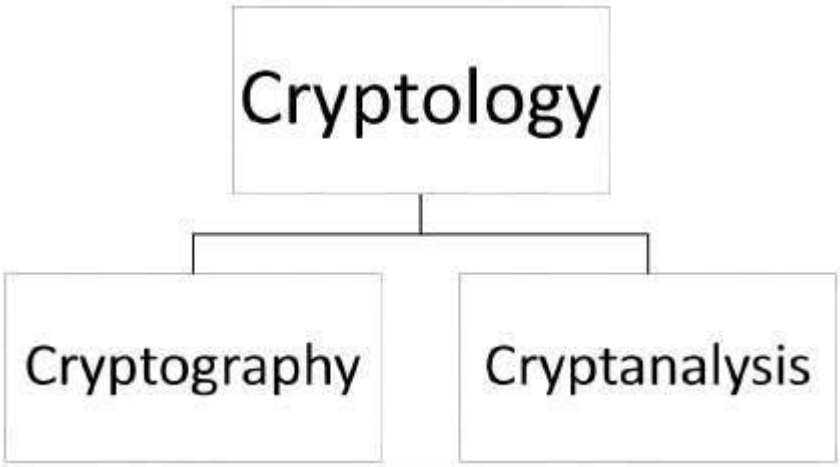
Classic Cryptography	Modern Cryptograph
It manipulates traditional characters, i.e., letters and digits directly.	It operates on binary bit sequences.
It is mainly based on 'security through obscurity'. The techniques employed for coding were kept secret and only the parties involved in communication knew about them.	It relies on publicly know mathematical algorithms for coding the information. Secrecy is obtained through a secrete key which is used as the seed fo the algorithms The computationa difficulty of algorithms, absence of secret key, etc., make it impossible for an attacker to obtain the original information

	even if he knows the algorithm use for coding.
It requires the entire cryptosystem for communicating confidentially.	Modern cryptography requires parties interested in secure communicatio to possess th secret key only.

Context of Cryptography

Cryptology, the study of cryptosystems, can be subdivided into two branches −

- Cryptography
- Cryptanalysis



What is Cryptography?

Cryptography is the art and science of making a cryptosystem that is capable of providing information security.

Cryptography deals with the actual securing of digital data. It refers to the design of mechanisms based on mathematical algorithms that provide fundamental information security services. You can think of cryptography as the establishment of a large toolkit containing different techniques in security applications.

What is Cryptanalysis?

The art and science of breaking the cipher text is known as cryptanalysis.

Cryptanalysis is the sister branch of cryptography and they both co-exist. The cryptographic process results in the cipher text for transmission or storage. It involves the study of cryptographic mechanism with the intention to break them. Cryptanalysis is also used during the design of the new cryptographic techniques to test their security strengths.

Note − Cryptography concerns with the design of cryptosystems, while cryptanalysis studies the breaking of cryptosystems.

1.11.1. PERFECT SECURITY

1.11.2. INFORMATION THEORY

1.11.3. PRODUCT CRYPTOSYSTEM

Another innovation introduced by Shannon in his 1949 paper was the idea of combining cryptosystems by forming their "product." In cryptography, a product cipher combines two or more transformations in a manner intending that the resulting cipher is more secure than the individual components to make it resistant to cryptanalysis. The product cipher combines a sequence of simple transformations such as substitution (S-box), permutation (P-box), and modular arithmetic. The concept of product ciphers is due to Claude Shannon.

This idea has been of fundamental importance in the design of present-day cryptosystems such as the Data Encryption Standard,

For simplicity, we will confine our attention in this section to cryptosystems in which $\mathcal{C} = \mathcal{P}$: cryptosystems of this type are called *endomorphic*.

Suppose $\mathbf{S}_1 = (\mathcal{P}, \mathcal{P}, \mathcal{K}_1, \mathcal{E}_1, \mathcal{D}_1)$ and $\mathbf{S}_2 = (\mathcal{P}, \mathcal{P}, \mathcal{K}_2, \mathcal{E}_2, \mathcal{D}_2)$ are two endomorphic cryptosystems which have the same plaintext (and ciphertext) spaces. Then the product of S_1 and S_2, denoted by $S_1 \times S_2$, is defined to be the cryptosystem

$$(\mathcal{P}, \mathcal{P}, \mathcal{K}_1 \times \mathcal{K}_2, \mathcal{E}, \mathcal{D}).$$

A key of the product cryptosystem has the form $K = (K_1, K_2)$,

where $K_1 \in \mathcal{K}_1$ and $K_2 \in \mathcal{K}_2$. The encryption and decryption rules of the product cryptosystem are defined as follows: For each $K = (K_1, K_2)$, we have an encryption rule e_K defined by the formula

$$e_{(K_1,K_2)}(x) = e_{K_2}(e_{K_1}(x)),$$

and a decryption rule defined by the formula

$$d_{(K_1,K_2)}(y) = d_{K_1}(d_{K_2}(y)).$$

That is, we first encrypt x with e_{K_1}, and then "re-encrypt" the resulting ciphertext with e_{K_2}. Decrypting is similar, but it must be done in the reverse order:

$$\begin{aligned} d_{(K_1,K_2)}(e_{(K_1,K_2)}(x)) &= d_{(K_1,K_2)}(e_{K_2}(e_{K_1}(x))) \\ &= d_{K_1}(d_{K_2}(e_{K_2}(e_{K_1}(x)))) \\ &= d_{K_1}(e_{K_1}(x)) \\ &= x. \end{aligned}$$

Recall also that cryptosystems have probability distributions associated with their keyspaces. Thus we need to define the probability distribution for the keyspace $\mathcal{K}$ of the product cryptosystem. We do this in a very natural way:

$$p_{\mathcal{K}}(K_1, K_2) = p_{\mathcal{K}_1}(K_1) \times p_{\mathcal{K}_2}(K_2).$$

1.11.4. CRYPTANALYSIS