CHAPTER 10

Symmetric-Key Cryptography

(Solution to Odd-Numbered Problems)

Review Questions

1. Symmetric-key cryptography is based on sharing secrecy; asymmetric-key cryptography is based on personal secrecy.

3. In cryptography, a trapdoor is a secret with which Bob can use a feasible algorithm to decrypt the ciphertext. If Eve does not know the trapdoor, she needs to use an algorithm which is normally infeasible.

5. RSA uses two exponents, e and d, where e is public and d is private. Alice calculates $C = P^e \bmod n$ to create ciphertext C from plaintext P; Bob uses $P = C^d \bmod n$ to retrieve the plaintext sent by Alice.

- **a.** The one-way function is the $C = P^e \bmod n$. Given P and e, it is easy to calculate C; given C and e, it difficult to calculate P if n is very large.
- **b.** The trapdoor in this system is the value of d, which enables Bob to use $P = C^d \bmod n$.
- **c.** The public key is the tuple (e, n). The private key is d.
- **d.** The security of RSA mainly depend on the factorization of n. If n is very large, and the value of e and d are chosen properly, the system is secure.

7. ElGamal is based on discrete logarithm problem. The plaintext is masked with $e_1{}^{rd}$ to create the ciphertext. Part of the mask is created by Bob and become public; the other part is created by Alice.

- **a.** The one-way function is C = mask (P). Given P and the mask, it is easy to calculate the C; given C is difficult to unmask P.
- **b.** The trapdoor is the value of d that enables Bob to unmask C.
- **c.** The public key is (e_1, e_2 and n). The private key is d.
- **d.** The security of ElGamal depends on two points; p should be very large and Alice needs to select a new r for each encryption.

9. An elliptic curve is an equation in two variables similar to the equations used to calculate the length of a curve in the circumference of an ellipse. Elliptic curves with points belonging to the groups GF(p) or GF(2^n) are used in cryptography.

Exercises

11. We use the 7-bit representation of character "a" as the plaintext.

Key Generation:

$t = [287, 451, 943, 762, 564, 86, 623] \rightarrow a = [623, 86, 564, 287, 451, 943, 762]$

Encryption:

Plaintext: $x = [1, 1, 0, 0, 0, 0, 1] \quad \rightarrow \quad$ Ciphertext: $s = 1471$

Decryption:

$$\mathbf{s' = (41^{-1}, 1471) \bmod 1001 = 293 \times 1471 \bmod 1001 = 573}$$

$$\mathbf{x' = [0, 0, 0, 1, 0, 1, 1] \rightarrow Plaintext\ x = [1, 1, 0, 0, 0, 0, 1]}$$

13. $n = 187 = 17 \times 11 \rightarrow \phi(n) = 17 \times 11 = 160 \rightarrow d = e^{-1} \bmod \phi(n) = 113$. This proves that the value of n in RSA must be very large.We could find d because we could factor n. The modulus must be large enough to make the factorization infeasible.

15. In a real situation, the value of n is so large that it is impossible for Alice to check if n and e are chosen properly. In this toy problem, it is easy to see that n is not properly selected because $n = 100$ cannot be factored into two primes ($n = p \times q$). Although we can still encrypt the message using $e = 13$ and $n = 100$, the encrypted message cannot be decrypted. The problem prove that Bob needs to first select p and q and be sure that they are primes before calculating $n = p \times q$. After the correct selection of n, then e needs to be selected in such a way that $\phi(n)$ and e be coprimes. **This problem has no solution.**

17.

a. If $e = 1$, there is no encryption: C = P. If Eve intercepts the ciphertext, she has the plaintext.

b. If $e = 2$, the cipher is actually Rabin, not RSA.

19. Eve has intercepted C = 57

a. Eve chooses X = 17 (which is in $\mathbf{Z}_{143}^*$).

b. Eve calculates $Y = C \times 17^7 \bmod 143 = 57 \times 17^7 \bmod 143 = 137$

c. Eve sends 137 to Bob and ask to decrypt it. The response is Z = 136 (We know how to calculate this because we can easily find $d = 103$, but we assume that Eve cannot; she needs to send the ciphertext to Bob and receive the plaintext.

d. $P = (Z \times X^{-1}) \bmod 143 = (136 \times 17^{-1}) \bmod 143 = 8 \bmod 143$. Which is the plaintext. This shows that RSA is very vulnerable to chosen-ciphertext attack.

21. One solution is to use different padding with each plaintext. As we discussed in the text, using OAEP encryption, each time with different random *r*, removes the relationships between different plaintexts that are sent by Alice to Bob.

23.

a. We choose $e_1 = 3$ (a primitive root of $p = 31$) and $d = 10$. Then we have $e_2 = 3^{10} \bmod 31 = 25$.

b. The common factor for calculation of C_2's is ${e_2}^7 \bmod 31 = 25^7 \bmod 31 = 25$.

$$
\begin{array}{llll}
\text{P} = \text{"H"} = 07 & C_1 = 3^7 \bmod 31 = 17 & C_2 = 07 \times 25 \bmod 31 = 20 & \rightarrow \quad \text{C} = (17, 20) \\
\text{P} = \text{"E"} = 04 & C_1 = 3^7 \bmod 31 = 17 & C_2 = 04 \times 25 \bmod 31 = 07 & \rightarrow \quad \text{C} = (17, 07) \\
\text{P} = \text{"L"} = 11 & C_1 = 3^7 \bmod 31 = 17 & C_2 = 11 \times 25 \bmod 31 = 27 & \rightarrow \quad \text{C} = (17, 27) \\
\text{P} = \text{"L"} = 11 & C_1 = 3^7 \bmod 31 = 17 & C_2 = 11 \times 25 \bmod 31 = 27 & \rightarrow \quad \text{C} = (17, 27) \\
\text{P} = \text{"O"} = 14 & C_1 = 3^7 \bmod 31 = 17 & C_2 = 14 \times 25 \bmod 31 = 09 & \rightarrow \quad \text{C} = (17, 09)
\end{array}
$$

c.

$$
\begin{array}{lllll}
\text{C} = (17, 20) & \rightarrow & \text{P} = 20 \times (17^{10)-1} \bmod 31 = 07 & \rightarrow & \text{"H"} \\
\text{C} = (17, 07) & \rightarrow & \text{P} = 07 \times (17^{10)-1} \bmod 31 = 04 & \rightarrow & \text{"E"} \\
\text{C} = (17, 27) & \rightarrow & \text{P} = 27 \times (17^{10)-1} \bmod 31 = 11 & \rightarrow & \text{"L"} \\
\text{C} = (17, 27) & \rightarrow & \text{P} = 27 \times (17^{10)-1} \bmod 31 = 11 & \rightarrow & \text{"L"} \\
\text{C} = (17, 09) & \rightarrow & \text{P} = 09 \times (17^{10)-1} \bmod 31 = 14 & \rightarrow & \text{"O"}
\end{array}
$$

25. Although the value of *p* (the modulus) and *d* are not given, we can assume a modulus which is greater that 17 and 37 and its primitive root is 2. We have chosen $p = 53$ and $d = 3$. In this case, we have $p = 53$, $d = 3$, $e_1 = 2$, and ${e_2} = {e_1}^3 \bmod 53 = 8$.

a. Alice uses $r = 9$ to encrypt two messages, 17 and 37. The values of ciphertexts are: $C_1 = 35$, $C_2 = 19$, $C_1' = 35$ and $C_2' = 32$.

b. Eve intercepts $C_1 = 35$, $C_2 = 19$, $C_1' = 35$ and $C_2' = 32$ and she know P = 17. Eve can use the known-plaintext attack to find P′.

$$\text{P'} = C_2' \times (e_2{}^r)^{-1} \bmod p = C_2' \times (C_2 \times \text{P}^{-1})^{-1} \bmod p = C_2' \times C_2{}^{-1} \times \text{P} \bmod p$$

$$\text{P'} = 32 \times 19^{-1} \times 17 \mod 53 = 32 \times 14 \times 17 \bmod 53 = 37$$

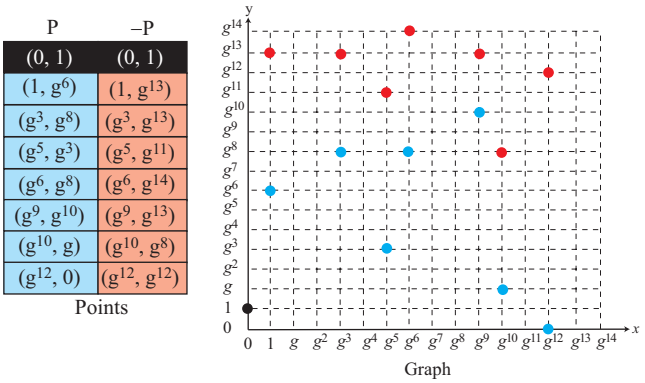
27.

a. $E(g^4, 1)$ means that $a = g^4$ and $b = 1$. The equation of the curve is

$$y^2 + xy = x^3 + g^4 x^2 + 1$$

b. Assume that the irreducible polynomial is $x^4 + x + 1$ (See Appendix G). We use the process in Example 10.16 in the textbook Page 326 to find the points on the curve as shown in Figure S10.27.

Figure S10.27 *Part of the solution to Exercise 27*



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

Points

Graph

c. Bob chooses $e_1 = (g^3, g^8)$ and $d = 2$. Then $e_2 = d \times e_1 = (g^5, g^3)$. The public key is the combination of e_1, e_2, and E(g^4, 1). The private key is d.

d. Alice chooses P = (g^{10}, g) and $r = 3$.

e. Alice calculates $C_1 = r \times e_1 =$ (g^9, g^{10}) and $C_2 = P + r \times e_2 = (1, g^6)$.

f. Bob decrypt the message $P = C_2 - (d \times C_1) = (g^{10}, g) = P$

29.

a. The following shows the encrypting algorithm. Alice uses the original message (m), a random number (r) and two functions (G and H).

```
RSA_OAEP_Encrypt (m, e, n, r)
{
        M ← pad (m)
        P1 ← M ⊕ G(r)
        P2 ← H(P1) ⊕ r
        C ← Fast_Exponentiation (P1 | P2, e, n)
        return C
}
```

b. The following shows the decrypting algorithm. Bob uses the ciphertext (C), the private key (d), modulus (n), and two functions G and H.

```
RSA_OAEP_Decrypt (C, d, n)
{
        P ← Fast_Exponentiation (C, d, n)
        P1, P2 ← Split_{m,k} (P)
        r ← H(P1) ⊕ P2
        M ← P1 ⊕ G(r)
        m ← Extract (M)
        return m
}
```

31. The following shows the AddPoint algorithm. P(x) and P(y) are x and y components of P. Calculations are done in GF(p) .

```
AddPoint (p, a, b, P1, P2)
{
        if (P1(x) ≠ P2(x))
                λ ← ( (P2(y) − P1(y) ) / (P2(x) − P1(x) ) )  mod p
        else
                λ ← ( (3 × (P1(x))^2 + a  ) / (2 × P1(y) ) )  mod p
        P3(x) ← (λ^2 − P1(x) − P2(x)) mod p
        P3(y)← (λ × (P1(x) − P3(x)) − P1(y)) mod p
        return P3
}
```