Problems

6.1. As we have seen in this chapter, public-key cryptography can be used for encryption and key exchange. Furthermore, it has some properties (such as nonrepudiation) which are not offered by secret key cryptography.

So why do we still use symmetric cryptography in current applications?

6.2. In this problem, we want to compare the computational performance of symmetric and asymmetric algorithms. Assume a fast public-key library such as OpenSSL [132] that can decrypt data at a rate of 100 Kbit/sec using the RSA algorithm on a modern PC. On the same machine, AES can decrypt at a rate of 17 Mbit/sec. Assume we want to decrypt a movie stored on a DVD. The movie requires 1 GByte of storage. How long does decryption take with either algorithm?

6.3. Assume a (small) company with 120 employees. A new security policy demands encrypted message exchange with a symmetric cipher. How many keys are required, if you are to ensure a secret communication for every possible pair of communicating parties?

6.4. The level of security in terms of the corresponding bit length directly influences the performance of the respective algorithm. We now analyze the influence of increasing the security level on the runtime.

Assume that a commercial Web server for an online shop can use either RSA or ECC for signature generation. Furthermore, assume that signature generation for RSA-1024 and ECC-160 takes 15.7 ms and 1.3 ms, respectively.

1. Determine the increase in runtime for signature generation if the security level from RSA is increased from 1024 bit to 3072 bit.
2. How does the runtime increase from 1024 bit to 15,360 bit?
3. Determine these numbers for the respective security levels of ECC.
4. Describe the difference between RSA and ECC when increasing the security level.

Hint: Recall that the computational complexity of both RSA and ECC grows with the cube of bit length. You may want to use Table 6.1 to determine the adequate bit length for ECC, given the security level of RSA.

6.5. Using the basic form of Euclid's algorithm, compute the greatest common divisor of

1. 7469 and 2464
2. 2689 and 4001

For this problem use only a pocket calculator. Show every iteration step of Euclid's algorithm, i.e., don't write just the answer, which is only a number. Also, for every gcd, provide the chain of gcd computations, i.e.,

$$\gcd(r_0, r_1) = \gcd(r_1, r_2) = \cdots .$$

6.6. Using the extended Euclidean algorithm, compute the greatest common divisor and the parameters s,t of

1. 198 and 243
2. 1819 and 3587

For every problem check if $s\,r_0 + t\,r_1 = \gcd(r_0, r_1)$ is actually fulfilled. The rules are the same as above: use a pocket calculator and show what happens in every iteration step.

6.7. With the Euclidean algorithm we finally have an efficient algorithm for finding the multiplicative inverse in Z_m that is much better than exhaustive search. Find the inverses in Z_m of the following elements a modulo m:

1. $a = 7$, $m = 26$ (affine cipher)
2. $a = 19$, $m = 999$

Note that the inverses must again be elements in Z_m and that you can easily verify your answers.

6.8. Determine $\phi(m)$, for $m = 12, 15, 26$, according to the definition: Check for each positive integer n smaller m whether $\gcd(n,m) = 1$. (You do *not* have to apply Euclid's algorithm.)

6.9. Develop formulae for $\phi(m)$ for the special cases when

1. m is a prime
2. $m = p \cdot q$, where p and q are primes. This case is of great importance for the RSA cryptosystem. Verify your formula for $m = 15, 26$ with the results from the previous problem.

6.10. Compute the inverse a^{-1} mod n with Fermat's Theorem (if applicable) or Euler's Theorem:

- $a = 4$, $n = 7$
- $a = 5$, $n = 12$
- $a = 6$, $n = 13$

6.11. Verify that Euler's Theorem holds in Z_m, $m = 6, 9$, for all elements a for which $\gcd(a,m) = 1$. Also verify that the theorem does not hold for elements a for which $\gcd(a,m) \neq 1$.

6.12. For the affine cipher in Chapter 1 the multiplicative inverse of an element modulo 26 can be found as

$$a^{-1} \equiv a^{11} \bmod 26.$$

Derive this relationship by using Euler's Theorem.

6.13. The extended Euclidean algorithm has the initial conditions $s_0 = 1, s_1 = 0, t_0 = 0, t_1 = 1$. *Derive* these conditions. It is helpful to look at how the general iteration formula for the Euclidean algorithm was derived in this chapter.