Problems

7.1. Let the two primes $p = 41$ and $q = 17$ be given as set-up parameters for RSA.

1. Which of the parameters $e_1 = 32, e_2 = 49$ is a valid RSA exponent? Justify your choice.
2. Compute the corresponding private key $K_{pr} = (p, q, d)$. Use the extended Euclidean algorithm for the inversion and point out every calculation step.

7.2. Computing modular exponentiation efficiently is inevitable for the practicability of RSA. Compute the following exponentiations x^e mod m applying the square-and-multiply algorithm:

1. $x = 2$, $e = 79$, $m = 101$
2. $x = 3$, $e = 197$, $m = 101$

After every iteration step, show the exponent of the intermediate result in binary notation.

7.3. Encrypt and decrypt by means of the RSA algorithm with the following system parameters:

1. $p = 3$, $q = 11$, $d = 7$, $x = 5$
2. $p = 5$, $q = 11$, $e = 3$, $x = 9$

Only use a pocket calculator at this stage.

7.4. One major drawback of public-key algorithms is that they are relatively slow. In Sect. 7.5.1 we learned that an acceleration technique is to use short exponents e. Now we study short exponents in this problem in more detail.

1. Assume that in an implementation of the RSA cryptosystem one modular squaring takes 75% of the time of a modular multiplication. How much quicker is one encryption on average if instead of a 2048-bit public key the short exponent $e = 2^{16} + 1$ is used? Assume that the square-and-multiply algorithm is being used in both cases.
2. Most short exponents are of the form $e = 2^n + 1$. Would it be advantageous to use exponents of the form $2^n - 1$? Justify your answer.
3. Compute the exponentiation x^e mod 29 of $x = 5$ with both variants of e from above for $n = 4$. Use the square-and-multiply algorithm and show each step of your computation.

7.5. In practice the short exponents $e = 3$, 17 and $2^{16} + 1$ are widely used.

1. Why can't we use these three short exponents as values for the exponent d in applications where we want to accelerate decryption?
2. Suggest a minimum bit length for the exponent d and explain your answer.

7.6. Verify the RSA with CRT example in the chapter by computing $y^d = 15^{103}$ mod 143 using the square-and-multiply algorithm.

7.7. An RSA encryption scheme has the set-up parameters $p = 31$ and $q = 37$. The public key is $e = 17$.

1. Decrypt the ciphertext $y = 2$ using the CRT.
2. Verify your result by encrypting the plaintext without using the CRT.

7.8. Popular RSA modulus sizes are 1024, 2048, 3072 and 4092 bit.

1. How many random odd integers do we have to test on average until we expect to find one that is a prime?
2. Derive a simple formula for any arbitrary RSA modulus size.

7.9. One of the most attractive applications of public-key algorithms is the establishment of a secure session key for a private-key algorithm such as AES over an insecure channel.

Assume Bob has a pair of public/private keys for the RSA cryptosystem. Develop a simple protocol using RSA which allows the two parties Alice and Bob to agree on a shared secret key. Who determines the key in this protocol, Alice, Bob, or both?

7.10. In practice, it is sometimes desirable that both communication parties influence the selection of the session key. For instance, this prevents the other party from choosing a key which is a *weak key* for a symmetric algorithm. Many block ciphers such as DES and IDEA have weak keys. Messages encrypted with weak keys can be recovered relatively easily from the ciphertext.

Develop a protocol similar to the one above in which both parties influence the key. Assume that both Alice and Bob have a pair of public/private keys for the RSA cryptosystem. Please note that there are several valid approaches to this problem. Show just one.

7.11. In this exercise, you are asked to attack an RSA encrypted message. Imagine being the attacker: You obtain the ciphertext $y = 1141$ by eavesdropping on a certain connection. The public key is $k_{pub} = (n, e) = (2623, 2111)$.

1. Consider the encryption formula. All variables except the plaintext x are known. Why can't you simply solve the equation for x?
2. In order to determine the private key d, you have to calculate $d \equiv e^{-1} \bmod \Phi(n)$. There is an efficient expression for calculating $\Phi(n)$. Can we use this formula here?
3. Calculate the plaintext x by computing the private key d through factoring $n = p \cdot q$. Does this approach remain suitable for numbers with a length of 1024 bit or more?

7.12. We now show how an attack with chosen ciphertext can be used to break an RSA encryption.

1. Show that the *multiplicative property* holds for RSA, i.e., show that the product of two ciphertexts is equal to the encryption of the product of the two respective plaintexts.

2. This property can under certain circumstances lead to an attack. Assume that Bob first receives an encrypted message y_1 from Alice which Oscar obtains by eavesdropping. At a later point in time, we assume that Oscar can send an innocent looking ciphertext y_2 to Bob, and that Oscar can obtain the decryption of y_2. In practice this could, for instance, happen if Oscar manages to hack into Bob's system such that he can get access to decrypted plaintext for a limited period of time.

7.13. In this exercise, we illustrate the problem of using nonprobabilistic cryptosystems, such as schoolbook RSA, imprudently. Nonprobabilistic means that the same sequence of plaintext letters maps to the same ciphertext. This allows traffic analysis (i.e., to draw some conclusion about the cleartext by merely observing the ciphertext) and in some cases even to the total break of the cryptoystem. The latter holds especially if the number of possible plaintexts is small. Suppose the following situation:

Alice wants to send a message to Bob encrypted with his public key pair (n,e). Therefore, she decides to use the ASCII table to assign a number to each character (*Space* $\rightarrow$ 32, ! $\rightarrow$ 33, ..., $A \rightarrow 65$, $B \rightarrow 66$, ..., $\sim\rightarrow 126$) and to encrypt them separately.

1. Oscar eavesdrops on the transferred ciphertext. Describe how he can successfully decrypt the message by exploiting the nonprobabilistic property of RSA.
2. Bob's RSA public key is $(n,e) = (3763, 11)$. Decrypt the ciphertext

$$y = 2514, 1125, 333, 3696, 2514, 2929, 3368, 2514$$

   with the attack proposed in 1. For simplification, assume that Alice only chose capital letters A–Z during the encryption.
3. Is the attack still possible if we use the OAEP padding? Exactly explain your answer.

7.14. The modulus of RSA has been enlarged over the years in order to thwart improved attacks. As one would assume, public-key algorithms become slower as the modulus length increases. We study the relation between modulus length and performance in this problem. The performance of RSA, and of almost any other public-key algorithm, is dependent on how fast modulo exponentiation with large numbers can be performed.

1. Assume that one modulo multiplication or squaring with k-bit numbers takes $c \cdot k^2$ clock cycles, where c is a constant. How much slower is RSA encryption/decryption with 1024 bits compared to RSA with 512 bits on average? Only consider the encryption/decryption itself with an exponent of full length and the square-and-multiply algorithm.
2. In practice, the Karatsuba algorithm, which has an asymptotical complexity that is proportional to $k^{\log_2 3}$, is often used for long number multiplication in cryptography. Assume that this more advanced technique requires $c' \cdot k^{\log_2 3} = c' \cdot k^{1.585}$ clock cycles for multiplication or squaring where c' is a constant. What is the

ratio between RSA encryption with 1024 bit and RSA with 512 bit if the Karatsuba algorithm is used in both cases? Again, assume that full-length exponents are being used.

7.15. (Advanced problem!) There are ways to improve the square-and-multiply algorithm, that is, to reduce the number of operations required. Although the number of squarings is fixed, the number of multiplications can be reduced. Your task is to come up with a modified version of the square-and-multiply algorithm which requires fewer multiplications. Give a detailed description of how the new algorithm works and what the complexity is (number of operations).
Hint: Try to develop a generalization of the square-and-multiply algorithm which processes more than one bit at a time. The basic idea is to handle k (e.g., $k = 3$) exponent bit per iteration rather than one bit in the original square-and-multiply algorithm.

7.16. Let us now investigate side-channel attacks against RSA. In a simple implementation of RSA without any countermeasures against side-channel leakage, the analysis of the current consumption of the microcontroller in the decryption part directly yields the private exponent. Figure 7.5 shows the power consumption of an implementation of the square-and-multiply algorithm. If the microcontroller computes a squaring or a multiplication, the power consumption increases. Due to the small intervals in between the loops, every iteration can be identified. Furthermore, for each round we can identify whether a single squaring (short duration) or a squaring followed by a multiplication (long duration) is being computed.

1. Identify the respective rounds in the figure and mark these with S for squaring or SM for squaring and multiplication.
2. Assume the square-and-multiply algorithm has been implemented such that the exponent is being scanned from left to right. Furthermore, assume that the starting values have been initialized. What is the private exponent d?
3. This key belongs to the RSA setup with the primes $p = 67$ and $q = 103$ and $e = 257$. Verify your result. (Note that in practice an attacker wouldn't know the values of p and q.)

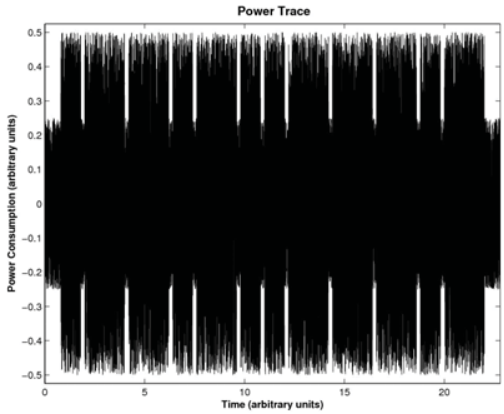

Fig. 7.5 Power consumption of an RSA decryption