Exercises

3.1 Prove Proposition 3.6.

3.2 The best algorithm known today for finding the prime factors of an n-bit number runs in time $2^{c \cdot n^{\frac{1}{3}} (\log n)^{\frac{2}{3}}}$. Assuming 4Ghz computers and $c = 1$ (and that the units of the given expression are clock cycles), estimate the size of numbers that cannot be factored for the next 100 years.

3.3 Prove that Definition 3.8 cannot be satisfied if Π can encrypt arbitrary-length messages and the adversary is *not* restricted to output equal-length messages in experiment $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$.

Hint: Let $q(n)$ be a polynomial upper-bound on the length of the ciphertext when Π is used to encrypt a single bit. Then consider an adversary who outputs $m_0 \in \{0,1\}$ and a random $m_1 \in \{0,1\}^{q(n)+2}$.

3.4 Say $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is such that for $k \in \{0,1\}^n$, algorithm Enc_k is only defined for messages of length at most $\ell(n)$ (for some polynomial ℓ). Construct a scheme satisfying Definition 3.8 even when the adversary is *not* restricted to output equal-length messages in experiment $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$.

3.5 Prove the equivalence of Definition 3.8 and Definition 3.9.

3.6 Let G be a pseudorandom generator where $|G(s)| > 2 \cdot |s|$.

(a) Define $G'(s) \stackrel{\text{def}}{=} G(s0^{|s|})$. Is G' necessarily a pseudorandom generator?

(b) Define $G'(s) \stackrel{\text{def}}{=} G(s_1 \cdots s_{n/2})$, where $s = s_1 \cdots s_n$. Is G' necessarily a pseudorandom generator?

3.7 Assuming the existence of a pseudorandom function, prove that there exists an encryption scheme that has indistinguishable multiple encryptions in the presence of an eavesdropper (i.e., is secure with respect to Definition 3.18), but is not CPA-secure (i.e., is not secure with respect to Definition 3.21).

Hint: The scheme need not be "natural". You will need to use the fact that in a chosen-plaintext attack the adversary can choose its queries to the encryption oracle *adaptively*.

3.8 Prove *unconditionally* the existence of an efficient pseudorandom function $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ where the input length is *logarithmic* in the security parameter (and the key has length polynomial in the security parameter).

Hint: Implement a random function with logarithmic input length.

3.9 Present a construction of a variable output-length pseudorandom generator from any pseudorandom function. Prove that your construction satisfies Definition 3.17.

3.10 Let G be a pseudorandom generator and define $G'(s)$ to be the output of G truncated to n bits (where $|s| = n$). Prove that the function $F_k(x) = G'(k) \oplus x$ is not pseudorandom.

3.11 Prove Proposition 3.27 (i.e., prove that any pseudorandom permutation is also a pseudorandom function).

Hint: Show that in polynomial time, a random permutation cannot be distinguished from a random function (use the results of Appendix A.4).

3.12 Define a notion of perfect secrecy against a chosen-plaintext attack via the natural adaptation of Definition 3.21. Show that the definition cannot be achieved.

3.13 Assume that F is a pseudorandom permutation. Show that there exists a function F' that is a pseudorandom permutation but is *not* a strong pseudorandom permutation.

Hint: Construct F' such that $F'_k(k) = 0^{|k|}$.

3.14 Let F be a pseudorandom permutation, and define a fixed-length encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ as follows: On input $m \in \{0,1\}^{n/2}$ and key $k \in \{0,1\}^n$, algorithm Enc chooses a random string $r \leftarrow \{0,1\}^{n/2}$ of length $n/2$ and computes $c := F_k(r \| m)$.

Show how to decrypt, and prove that this scheme is CPA-secure for messages of length $n/2$. (If you are looking for a real challenge, prove that this scheme is CCA-secure if F is a *strong* pseudorandom permutation.) What are the advantages and disadvantages of this construction as compared to Construction 3.24?

3.15 Let F be a pseudorandom function, and G a pseudorandom generator with expansion factor $\ell(n) = n+1$. For each of the following encryption schemes, state whether the scheme has indistinguishable encryptions in the presence of an eavesdropper and whether it is CPA-secure. In each case, the shared key is a random $k \in \{0,1\}^n$.

(a) To encrypt $m \in \{0,1\}^{2n+2}$, parse m as $m_1 \| m_2$ with $|m_1| = |m_2|$ and send $\langle G(k) \oplus m_1,\ G(k+1) \oplus m_2 \rangle$.

(b) To encrypt $m \in \{0,1\}^{n+1}$, choose a random $r \leftarrow \{0,1\}^n$ and send $\langle r, G(r) \oplus m \rangle$.

(c) To encrypt $m \in \{0,1\}^n$, send $m \oplus F_k(0^n)$.

(d) To encrypt $m \in \{0,1\}^{2n}$, parse m as $m_1 \| m_2$ with $|m_1| = |m_2|$, then choose $r \leftarrow \{0,1\}^n$ at random, and send $\langle r,\ m_1 \oplus F_k(r),\ m_2 \oplus F_k(r+1) \rangle$.

3.16 Consider a variant of CBC-mode encryption where the sender simply increments the IV by 1 each time a message is encrypted (rather than choosing IV at random each time). Show that the resulting scheme is *not* CPA-secure.

3.17 Present formulas for decryption of all the different modes of encryption we have seen. For which modes can decryption be parallelized?

3.18 Complete the proof of Theorem 3.29.

3.19 Let F be a pseudorandom function such that for $k \in \{0,1\}^n$ the function F_k maps $\ell_{in}(n)$-bit inputs to $\ell_{out}(n)$-bit outputs. (Throughout this chapter, we have assumed $\ell_{in}(n) = \ell_{out}(n) = n$.)

(a) Consider implementing counter mode encryption using an F of this form. For which functions ℓ_{in}, ℓ_{out} is the resulting encryption scheme CPA-secure?

(b) Consider implementing counter mode encryption using an F as above, but only for *fixed-length* messages of length $\ell(n)$ (which is always an integer multiple of $\ell_{out}(n)$). For which $\ell_{in}, \ell_{out}, \ell$ is the scheme CPA-secure? For which $\ell_{in}, \ell_{out}, \ell$ does the scheme have indistinguishable encryptions in the presence of an eavesdropper?

3.20 For a function $g : \{0,1\}^n \rightarrow \{0,1\}^n$, let $g^{\$}(\cdot)$ be an oracle that, on input 1^n, chooses $r \leftarrow \{0,1\}^n$ uniformly at random and returns $(r, g(r))$. We say a keyed function F is a weak pseudorandom function if for all PPT algorithms D, there exists a negligible function negl such that:

$$\left| \Pr[D^{F_k^{\$}(\cdot)}(1^n) = 1] - \Pr[D^{f^{\$}(\cdot)}(1^n) = 1] \right| \leq \mathsf{negl}(n),$$

where $k \leftarrow \{0,1\}^n$ and $f \leftarrow \mathsf{Func}_n$ are chosen uniformly at random.

(a) Prove that if F is pseudorandom then it is weakly pseudorandom.

(b) Let F' be a pseudorandom function, and define

$$F_k(x) = \begin{cases} F'_k(x) & \text{if } x \text{ is even} \\ F'_k(x+1) & \text{if } x \text{ is odd} \end{cases}.$$

Prove that F is weakly pseudorandom, but *not* pseudorandom.

(c) Is counter-mode encryption instantiated using a weak pseudorandom function F necessarily CPA-secure? Does it necessarily have indistinguishable encryptions in the presence of an eavesdropper? Prove your answers.

(d) Construct a CPA-secure encryption scheme based on a weak pseudorandom function.

Hint: One of the constructions in this chapter will work.

3.21 Let $\Pi_1 = (\mathsf{Gen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ and $\Pi_2 = (\mathsf{Gen}_2, \mathsf{Enc}_2, \mathsf{Dec}_2)$ be two encryption schemes for which it is known that at least one is CPA-secure. The problem is that you don't know which one is CPA-secure and which one may not be. Show how to construct an encryption scheme Π that is guaranteed to be CPA-secure as long as at least one of Π_1 or Π_2 is CPA-secure. Try to provide a full proof of your answer.

Hint: Generate two plaintext messages from the original plaintext so that knowledge of either one of the parts reveals nothing about the plaintext, but knowledge of both does yield the original plaintext.

3.22 Show that the CBC, OFB, and counter modes of encryption do not yield CCA-secure encryption schemes (regardless of F).

We have already proved that the above combination is CCA-secure in Theorem 4.20, and leave as an exercise the proof that it also provides authenticated communication.

Secure message transmission vs. CCA-security. Although we use the same construction for achieving CCA-security and secure message transmission, the security goals in each case are different. In the setting of CCA-security we are not necessarily interested in obtaining message authentication; rather, we wish to ensure privacy even against a strong adversary who is able to make decryption queries. When considering secure message transmission, in contrast, we are interested in the twin goals of CCA-security and integrity. Clearly, as we have defined it, secure message transmission implies CCA-security. The opposite direction is not necessarily true.

The need for independent keys. We conclude by stressing a basic principle of security and cryptography: *different security goals should always use different keys.* That is, if an encryption scheme and a message authentication code are both needed, then independent keys should be used for each one. In order to illustrate this here, consider what can happen to the encrypt-then-authenticate methodology when the same key k is used for both encryption and authentication. Let F be a strong pseudorandom permutation. It follows that both F and F^{-1} are strong pseudorandom permutations. Define $\mathsf{Enc}_k(m) = F_k(m\|r)$ for $m \in \{0,1\}^{n/2}$ and a random $r \leftarrow \{0,1\}^{n/2}$, and define $\mathsf{Mac}_k(c) = F_k^{-1}(c)$. It can be shown that the given encryption scheme is CPA-secure (in fact, it is even CCA-secure), and we know that the given message authentication code is a secure MAC. However, the encrypt-then-authenticate combination applied to the message m with the same key k yields:

$$\mathsf{Enc}_k(m), \mathsf{Mac}_k(\mathsf{Enc}_k(m)) = F_k(m\|r), F_k^{-1}(F_k(m\|r)) = F_k(m\|r), m\|r,$$

and the message m is revealed in the clear!

References and Additional Reading

The definition of security for message authentication codes was adapted by Bellare et al. [12] from the definition of security for digital signatures given by Goldwasser et al. [70] (see Chapter 12). The basic paradigm of using pseudorandom functions for message authentication (as in Construction 4.3) was introduced by Goldreich et al. [66], and Construction 4.5 for extending a fixed-length MAC to a variable-length MAC is due to Goldreich [65]. An alternate approach for extending a fixed-length MAC to a variable-length MAC using collision-resistant hash functions is presented in the context of digital signatures in Section 12.4 (and, as we have mentioned, is the same idea underlying NMAC).

CBC-MAC was standardized in the early '80s [148, 80] and was later formally analyzed and proven secure by Bellare et al. [12] (the proofs include both the fixed-length case and the secure extensions to variable-length messages). The NMAC and HMAC constructions were introduced by Bellare et al. [8] and later standardized in [110].

Collision-resistant hash functions were first formally defined by Damgård [43], and the Merkle-Damgård transform was introduced independently by Damgård and Merkle [44, 102]. Additional discussion regarding notions of security for hash functions besides collision resistance can be found in [99, 123]. For further information about SHA-1 and MD5, see, e.g., the textbook by Kaufman et al. [87]. Note, however, that their treatment pre-dates the recent attacks by Wang et al. [144, 143]. For other interesting applications of collision-resistant hash functions in computer security, see Kaufman et al. [87] (but beware that in many cases the security arguments they give are purely heuristic). There are many hash functions that appear in the literature; many have been broken and some have not. Up-to-date information is maintained at the "Hash Function Lounge" [7].

The method of encrypting and then applying a MAC in order to achieve CCA-security was described by Dolev et al. [50]. Bellare and Namprempre [13] and Krawczyk [90] analyze different methods for simultaneously achieving privacy and authentication, and Krawczyk also analyzes the authenticate-then-encrypt approach used within SSL. Other notions of secure encryption that incorporate integrity are discussed in [85, 13]. A solution to Exercise 4.5 is given in [11].

Exercises

4.1 Say $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ is a secure MAC, and for $k \in \{0,1\}^n$ the tag-generation algorithm Mac_k always outputs tags of length $t(n)$. Prove that t must be super-logarithmic or, equivalently, that if $t(n) = \mathcal{O}(\log n)$ then Π cannot be a secure MAC.

Hint: Consider the probability of randomly guessing a valid tag.

4.2 Consider the following fixed-length MAC for messages of length $\ell(n) = 2n - 2$ using a pseudorandom function F: On input a message $m_0\|m_1$ (with $|m_0| = |m_1| = n-1$) and key $k \in \{0,1\}^n$, algorithm Mac outputs $t = F_k(0\|m_0)\|F_k(1\|m_1)$. Algorithm Vrfy is defined in the natural way. Is $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ existentially unforgeable under a chosen-message attack? Prove your answer.

4.3 Let F be a pseudorandom function. Show that the following MAC for messages of length $2n$ is insecure: The shared key is a random $k \in \{0,1\}^n$. To authenticate a message $m_1\|m_2$ with $|m_1| = |m_2| = n$, compute the tag $\langle F_k(m_1), F_k(F_k(m_2))\rangle$.

4.4 Let F be a pseudorandom function. Show that each of the following message authentication codes is insecure. (In each case the shared key is a random $k \in \{0,1\}^n$.)

(a) To authenticate a message $m = m_1\|\cdots\|m_\ell$, where $m_i \in \{0,1\}^n$, compute $t := F_k(m_1) \oplus \cdots \oplus F_k(m_\ell)$.

(b) To authenticate a message $m = m_1\|\cdots\|m_\ell$, where $m_i \in \{0,1\}^n$, choose $r \leftarrow \{0,1\}^n$ at random, compute $t := F_k(r) \oplus F_k(m_1) \oplus \cdots \oplus F_k(m_\ell)$, and send $\langle r, t\rangle$.

(c) To authenticate a message $m = m_1\|\cdots\|m_\ell$, where $m_i \in \{0,1\}^{n/2}$, choose $r \leftarrow \{0,1\}^n$ at random, compute

$$t := F_k(r) \oplus F_k(\langle 1\rangle\|m_1) \oplus \cdots \oplus F_k(\langle \ell\rangle\|m_\ell)$$

(where $\langle i\rangle$ is an $n/2$-bit encoding of the integer i), and send $\langle r, t\rangle$.

4.5 Consider an extension of the definition of secure message authentication where the adversary is provided with both a Mac and a Vrfy oracle.

(a) Provide a formal definition of security in this case, and explain what real-world adversarial actions are modeled by providing the adversary with a Vrfy oracle.

(b) Show that if Π has unique tags (cf. Section 4.8), then Π satisfies your definition if it satisfies Definition 4.2.

(c) Show that if Π does *not* have unique tags, then Π may satisfy Definition 4.2 but not your definition.

4.6 Is Construction 4.3 necessarily secure when instantiated using a weak pseudorandom function (cf. Exercise 3.20)? Explain.

4.7 Prove that Construction 4.5 is secure if it is changed as follows: Instead of including ℓ in every block, set $t_i := F_k(r\|b\|i\|m_i)$ where b is a single bit such that $b = 0$ in all blocks but the last one, and $b = 1$ in the last block. What is the advantage of this modification?

4.8 Show that the basic CBC-MAC construction is *not* secure when used to authenticate messages of different lengths.

4.9 Prove that the following modifications of CBC-MAC do not yield a secure fixed-length MAC:

(a) Modify CBC-MAC so that a random IV is used each time a tag is computed (and the IV is output along with t_ℓ). I.e., $t_0 \leftarrow \{0,1\}^n$ is chosen uniformly at random rather than being fixed to 0^n, and the tag is t_0, t_ℓ.

(b) Modify CBC-MAC so that all blocks $t_1, \ldots, t_\ell$ are output (rather than just t_ℓ).

4.10 Provide formal definitions for second pre-image resistance and pre-image resistance. Formally prove that any hash function that is collision resistant is second pre-image resistant, and that any hash function that is second pre-image resistant is pre-image resistant.

4.11 Let (Gen_1, H_1) and (Gen_2, H_2) be two hash functions. Define (Gen, H) so that Gen runs Gen_1 and Gen_2 to obtain keys s_1 and s_2, respectively. Then define $H^{s_1,s_2}(x) = H^{s_1}(x)\|H^{s_2}(x)$.

(a) Prove that if at least one of (Gen_1, H_1) and (Gen_2, H_2) is collision resistant, then (Gen, H) is collision resistant.

(b) Determine whether an analogous claim holds for second pre-image resistance and pre-image resistance, respectively. Prove your answer in each case.

4.12 Let (Gen, H) be a collision-resistant hash function. Is $(\mathsf{Gen}, \hat{H})$ defined by $\hat{H}^s(x) \overset{\text{def}}{=} H^s(H^s(x))$ necessarily collision resistant?

4.13 Provide a formal proof of Theorem 4.14 (i.e., describe the formal reduction).

4.14 Generalize the Merkle-Damgård construction for any compression function that compresses by at least one bit. You should refer to a general input length ℓ' and general output length ℓ (with $\ell' > \ell$).

4.15 For each of the following modifications to the Merkle-Damgård transform, determine whether the result is collision resistant or not. If yes, provide a proof; if not, demonstrate an attack.

(a) Modify the construction so that the input length is not included at all (i.e., output z_B and not $z_{B+1} = h^s(z_B\|L)$).

(b) Modify the construction so that instead of outputting $z = h^s(z_B\|L)$, the algorithm outputs $z_B\|L$.

(c) Instead of using a fixed IV, choose $IV \leftarrow \{0,1\}^n$ and define $z_0 := IV$. Then, set the output to be $IV \,\|\, h^s(z_B\|L)$.

(d) Instead of using an IV, just start the computation from x_1. That is, define $z_1 := x_1$ and then compute $z_i := h^s(z_{i-1}\|x_i)$ for $i = 2, \ldots, B+1$ and output z_{B+1} as before.

(e) Instead of using a fixed IV, set $z_0 := L$ and then compute $z_i := h^s(z_{i-1}\|x_i)$ for $i = 1, \ldots, B$ and output z_B.

4.16 Provide a full and detailed specification of HMAC when the underlying compression function has input length ℓ' and output length ℓ (with $\ell' > \ell$). Describe the instantiation of HMAC with SHA-1.

4.17 Before HMAC was invented, it was quite common to define a MAC by $\mathsf{Mac}_k(m) = H^s(k\|m)$ where H is a collision-resistant hash function. Show that this is not a secure MAC when H is constructed via the Merkle-Damgård transform.

4.18 Show that Construction 4.19 is CCA-secure even when the MAC of Construction 4.5 is used (this MAC does not have unique tags).

4.19 Show that if any message authentication code having unique tags is used in the encrypt-and-authenticate approach, the resulting combination is not CPA-secure.

4.20 Show an encryption scheme that is CCA-secure but is not a secure message transmission scheme.

4.21 Show a message transmission scheme that achieves authenticated communication but is not a secure message transmission scheme.

4.22 Prove Theorem 4.25.

Chapter 5

Practical Constructions of Pseudorandom Permutations (Block Ciphers)

In previous chapters, we have studied how pseudorandom permutations can be used to construct secure encryption schemes and message authentication codes. However, one question of prime importance that we have not yet studied is how pseudorandom permutations are constructed in the first place, or even whether they exist at all! In the next chapter we will study these questions from a theoretical vantage point, and show constructions of pseudorandom permutations that can be proven secure based on quite weak assumptions. In this chapter, our focus will be on comparatively heuristic, but far more efficient, constructions of pseudorandom permutations — known as *block ciphers* — that are used in practice.

As just mentioned, the constructions of block ciphers that we will explore in this chapter are (for the most part) heuristic, at least in the sense that they have no known proof of security based on any weaker assumption. Nevertheless, a number of the block ciphers that are used in practice have withstood many years of public scrutiny and attempted cryptanalysis, and given this fact it is quite reasonable to assume that these block ciphers are indeed (strong) pseudorandom permutations, subject to the technical issues discussed below.

Of course, in some sense there is no fundamental difference between assuming, say, that factoring is hard and assuming that DES (a block cipher we will study in detail later in this chapter) is a pseudorandom permutation. There is, however, a significant *qualitative* difference between these assumptions.[1] The primary difference is that the former assumption is of a weaker type: That is, the requirement that a certain problem (i.e., factoring) be hard to solve seems "easier to satisfy" than the requirement that a given keyed function be indistinguishable from a random function. Less important but still relevant differences between the assumptions are that the problem of factoring has been studied much longer than the problem of distinguishing DES from a random function, and the fact that factoring was recognized as a hard mathematical problem well before the advent of cryptographic schemes based

[1] It should be clear that the discussion in this paragraph is informal, as we cannot formally argue about any of this when we cannot even prove that factoring is hard in the first place!

Differential cryptanalysis. This technique was first presented in the late '80s by Biham and Shamir, who used it to attack DES in 1993. The basic idea behind the attack is to tabulate *specific differences in the input* that lead to *specific differences in the output* with probability greater than would be expected for a random permutation. Specifically, say a block cipher has block length n and let $\Delta_x, \Delta_y \in \{0,1\}^n$. We say that *the differential* (Δ_x, Δ_y) *appears with probability* p if for random inputs x_1 and x_2 satisfying $x_1 \oplus x_2 = \Delta_x$ and random choice of key k, the probability that $F_k(x_1) \oplus F_k(x_2) = \Delta_y$ is p. It is clear that for a random function, no differential should appear with probability much higher than 2^{-n}. In a weak block cipher, however, there may be differentials that appear with significantly higher probability.

One can find a differential that occurs with high probability either through a brute-force search (done once-and-for-all for the block cipher, independent of any particular key) or via a careful analysis of the block cipher itself. If a differential exists with probability $p \gg 2^{-n}$ then the block cipher already no longer qualifies as a pseudorandom permutation. The point of differential cryptanalysis is to use many differentials, that may each be only slightly larger than 2^{-n}, to recover the secret key (and thus break the cipher entirely) using a chosen-plaintext attack. We will not discuss the details here, though we mention that applying the block cipher to random pairs of inputs that have the given differential enables a cryptanalyst to isolate portions of the secret key and verify guesses for those portions. As we discussed regarding the attack on a 2-round substitution-permutation network, the ability to isolate parts of a key enables an attacker to obtain the key in time less than a brute force search. Note, however, that the fact that chosen plaintexts are required makes the attack of somewhat limited practicality.

Although differential cryptanalysis does not appear to lead to any practical attacks on DES and AES (since the number of chosen plaintexts required to carry out the attack is huge), differential cryptanalysis has been used successfully to attack other block ciphers. One important example is FEAL-8, which was completely broken using differential cryptanalysis.

Linear cryptanalysis. Linear cryptanalysis was developed by Matsui in the early '90s. This method considers linear relationships between the input and output. Say that bit positions $i_1, \dots, i_\ell$ and $i'_1, \dots, i'_\ell$ have *bias* p if, for randomly-chosen input x and key k, it holds that

$$\Pr[x_{i_1} \oplus \cdots \oplus x_{i_\ell} \oplus y_{i'_1} \oplus \cdots \oplus y_{i'_\ell} = 0] = p,$$

where $y = F_k(x)$ and x_i, y_i represent the bits of x and y. For a truly random function, we expect the bias to be close to 0.5. Matsui showed how to use a large enough bias in a given cipher F to completely break the cipher by finding the secret key. An important feature of this attack is that it does not require *chosen* plaintexts, and *known* plaintexts are sufficient. Even so, if a very large number of plaintext/ciphertext pairs are needed the attack becomes impractical in most settings.

Additional Reading and References

The confusion-diffusion paradigm and substitution-permutation networks were introduced by Shannon [127] and Feistel [53]. See the thesis of Heys [78] for further information regarding the design of substitution-permutation networks. Feistel networks were first described in [53]. A theoretical analysis of them was given by Luby and Rackoff [97], and will be discussed in Chapter 6.

The DES standard can be found at [109], and a more reader-friendly description can be found in the textbook by Kaufman et al. [87]. Details of the competition leading to the selection of Rijndael as the AES can be found at `http://csrc.nist.gov/CryptoToolkit/aes/index.html`. A comprehensive description of AES can be found in [87] as well as the book written by its designers, Daemen and Rijmen [41]. Cid et al. show an approach that may lead to cryptanalytic attacks on AES [33]. There are a large number of other good (and less good) block ciphers in the literature. For a broad but somewhat outdated overview of other ciphers, see [99, Chapter 7].

The meet-in-the-middle attack on double encryption is due to Diffie and Hellman [48]. The attack on two-key triple encryption mentioned in the text (and explored in Exercise 5.12) is by Merkle and Hellman [103]. Theoretical analysis of the security of double and triple encryption can be found in [3, 17].

DESX is another technique for increasing the effective key length of DES, without using additional invocations of DES. The secret key consists of the values $k_i, k_o \in \{0,1\}^{64}$ and $k \in \{0,1\}^{56}$, and the cipher is defined by

$$DESX_{k_i,k,k_o}(x) = k_o \oplus DES_k(x \oplus k_i).$$

This methodology was first studied by Even and Mansour [52]. Its concrete application to DES was proposed by Rivest, and its security was later analyzed by Kilian and Rogaway [88].

Differential cryptanalysis was introduced by Biham and Shamir [18] and its application to DES is described in [19]. Coppersmith [34] describes the DES design in light of the public discovery of differential cryptanalysis. Linear cryptanalysis was discovered by Matsui [98]. Langford's thesis [93] contains further improvements of differential and linear cryptanalysis, and also surveys known attacks on DES (and reduced-round variants) as of 1995. For more information on these advanced cryptanalytic techniques, we refer the reader to the excellent tutorial on differential and linear cryptanalysis by Heys [77]. A more concise presentation can be found in the textbook by Stinson [138].

Exercises

5.1 Say a block cipher F has the property that, given oracle access to F_k for randomly-chosen key k, it is possible to determine k using only 100 queries to the oracle (and minimal computation time). Show formally that F cannot be a pseudorandom permutation.

5.2 In our attack on a two-round substitution-permutation network, we considered a block length of 64 bits and a network with 16 S-boxes that each take a 4-bit input. Repeat the analysis for the case of 8 S-boxes, each taking an 8-bit input. What is the complexity of the attack now? Repeat the analysis again with a 128-bit block length and 16 S-boxes that each take an 8-bit input. Does the block length make any difference?

5.3 Our attack on a three-round substitution-permutation network does not recover the key but only shows how to distinguish the cipher from a random permutation. Thus it is not a "complete break". Despite this, show that using a three-round substitution-permutation network in the *counter-mode* encryption scheme (see Section 3.6.4) can have disastrous effects on the security of encryption.

5.4 Consider a modified substitution-permutation network where instead of carrying out the key-mixing, substitution, and permutation steps in alternating order for r rounds, the cipher instead first applies r rounds of key-mixing, then carries out r rounds of substitution, and finally applies r permutations. Analyze the security of this construction.

5.5 What is the output of an r-round Feistel network when the input is (L_0, R_0) in each of the following two cases:

(a) Each round function outputs all 0s, regardless of the input.

(b) Each round function is the identity function.

5.6 Show that DES has the property that $DES_k(x) = \overline{DES_{\overline{k}}(\overline{x})}$ for every key k and input x (where $\overline{z}$ denotes the bitwise complement of z). This is called the *complementarity property* of DES. (The description of DES given in this chapter is sufficient for this exercise.)

5.7 Use the previous exercise to show how it is possible to find the secret key in DES (with probability 1) in time 2^{55}.

Hint: Use a chosen-plaintext attack with two carefully chosen plaintexts.

5.8 In the actual construction of DES, the two halves of the output of the final round of the Feistel network are swapped. That is, if the output of the final round is (L_{16}, R_{16}) then the output of the cipher is in fact (R_{16}, L_{16}). Show that the only difference between the computation of DES_k and DES_k^{-1} (given the swapping of halves) is the order of sub-keys.

5.9 (This exercise assumes the results of the previous exercise.)

(a) Show that for $k = 0^{56}$ it holds that $DES_k(DES_k(x)) = x$. Why does the use of such a key pose a security threat?

(b) Find three other DES keys with the same property. These keys are known as *weak keys* for DES.

(c) Does the existence of these 4 weak keys represent a serious vulnerability in DES? Explain your answer.

5.10 Describe attacks on the following modifications to DES:

(a) Each round sub-key is 32 bits long, and the mangler function simply XORs the round sub-key with the input to the round (i.e., $\hat{f}(k, R) = k_i \oplus R$). For this example, the key schedule is unimportant and you can treat the k_i as independent keys.

(b) Instead of using different sub-keys in every round, the same 48-bit sub-key is used in every round. Show how to distinguish the cipher from a random permutation without a 2^{48}-time brute-force search.

Hint: Exercises 5.8 and 5.9 may help...

5.11 Show an improvement to the attack on three-round DES that recovers the key using two input/output pairs but runs in time $2 \cdot 2^{28} + 2 \cdot 2^{12}$.

5.12 This question illustrates an attack on two-key triple encryption. Let F be a block cipher with n-bit block length and key length, and set $F'_{k_1,k_2}(x) = F_{k_1}(F^{-1}_{k_2}(F_{k_1}(x)))$.

(a) Assume that given a pair (m_1, m_2) it is possible to find in *constant* time all keys k_2 such that $m_2 = F^{-1}_{k_2}(m_1)$. Show how to recover the entire key for F' (with high probability) in time roughly 2^n using three known input/output pairs.

(b) In general, it will *not* be possible to find k_2 as above in constant time. However, show that by using a pre-processing step taking 2^n time it is possible, given m_2, to find in (essentially) constant time all keys k_2 such that $m_2 = F^{-1}_{k_2}(0^n)$.

(c) Assume k_1 is known and that the pre-processing step above has already been run. Show how to use a single pair (x, y) for a *chosen* input value x to determine k_2 in constant time.

(d) Put the above components together to devise an attack that recovers the entire key by running in roughly 2^n time and requesting the encryption of roughly 2^n chosen inputs.

5.13 Show how all matches can be found in the meet-in-the-middle attack on double encryption in time *linear* in the number of matches and the lengths of the two sorted lists.

5.14 Say the key schedule of DES is modified as follows: the left half of the master key is used to derive all the sub-keys in rounds 1–8, while the right half of the master key is used to derive all the sub-keys in rounds 9–16. Show an attack on this modified scheme that recovers the entire key in time roughly 2^{28}.

5.15 Consider using DES as a fixed-length collision-resistant hash function in the following way: Define $h : \{0,1\}^{112} \rightarrow \{0,1\}^{64}$ as $h(x_1 \| x_2) \overset{\text{def}}{=} DES_{x_1}(DES_{x_2}(0^{64}))$ where $|x_1| = |x_2| = 56$.

(a) Write down an explicit collision in h.

Hint: Use Exercise 5.9.

(b) Show how to find a pre-image of a given value y (that is, x_1, x_2 such that $h(x_1 \| x_2) = y$) in roughly 2^{56} time.

(c) Show a more clever pre-image attack that runs in roughly 2^{32} time and succeeds with high probability.

Hint: Rely on the results of Appendix A.4.

Chapter 6

* Theoretical Constructions of Pseudorandom Objects

In Chapter 3 we introduced the notion of pseudorandomness, and defined the basic cryptographic primitives of pseudorandom generators, functions, and permutations. We showed in Chapters 3 and 4 that these primitives serve as the basic building blocks for all of private-key cryptography. As such, it is of great importance to understand these primitives from a theoretical point of view. In this chapter we formally introduce the concept of *one-way functions* — functions that are, informally speaking, easy to compute but hard to invert — and show how pseudorandom generators and pseudorandom permutations can be constructed under the sole assumption that one-way functions exist.[1] Moreover, we will see that one-way functions are a necessary assumption for essentially any non-trivial cryptographic task in the private-key setting. Tying everything together, this means that *the existence of one-way functions is equivalent to the existence of all (non-trivial) private-key cryptography.* This result constitutes one of the major contributions of modern cryptography.

The constructions of, say, pseudorandom permutations based on one-way functions that we show in this chapter should be viewed as complementary to the constructions of block ciphers given in the previous chapter. The focus of the previous chapter was on how pseudorandom permutations are currently realized in practice, and the intent of that chapter was to introduce some basic approaches and design principles that are used in their construction. Somewhat disappointing, however, was the fact that none of the constructions we showed could be *proven* secure based on any weaker (i.e., more reasonable) assumptions. In contrast, in the present chapter we will prove that it is possible to construct pseudorandom permutations starting from the very mild assumption that one-way functions exist. This assumption is more reasonable than assuming, say, that DES is a pseudorandom permutation since we have a number of candidate one-way functions that have been studied for many years, even before the advent of cryptography. (See the very beginning of Chapter 5 for further discussion of this point.) The downside, though, is that the constructions we show here are all far less efficient than those of Chapter 5, and thus are not actually used. It remains an important challenge for cryp-

[1] Actually, this is not quite true since we are for the most part going to rely on one-way *permutations* in this chapter. Nevertheless, it is known that one-way functions suffice.

Exercises

7.1 Let $\mathbb{G}$ be an abelian group. Prove that there is a *unique* identity in $\mathbb{G}$, and that every element $g \in \mathbb{G}$ has a *unique* inverse.

7.2 Show that Proposition 7.36 does not necessarily hold when $\mathbb{G}$ is infinite.

Hint: Consider the set $\{1\} \cup \{2, 4, 6, 8, \ldots\} \subset \mathbb{R}$.

7.3 Let $\mathbb{G}$ be a finite group, and $g \in \mathbb{G}$. Show that $\langle g \rangle$ is a subgroup of $\mathbb{G}$. Is the set $\{g^0, g^1, \ldots\}$ necessarily a subgroup of $\mathbb{G}$ when $\mathbb{G}$ is infinite?

7.4 This question concerns the Euler phi function.

(a) Let p be a prime and $e \geq 1$ an integer. Show that

$$\phi(p^e) = p^{e-1}(p-1).$$

(b) Let p, q be relatively prime. Show that $\phi(pq) = \phi(p) \cdot \phi(q)$. (You may not use the Chinese remainder theorem.)

(c) Prove Theorem 7.19.

7.5 Compute the final two (decimal) digits of 3^{1000} (by hand).

Hint: The answer is $[3^{1000} \bmod 100]$.

7.6 Compute $[101^{4,800,000,023} \bmod 35]$ (by hand).

7.7 Prove that if $\mathbb{G}, \mathbb{H}$ are groups, then $\mathbb{G} \times \mathbb{H}$ is a group.

7.8 Let p, N be integers with $p \mid N$. Prove that for any integer X,

$$[[X \bmod N] \bmod p] = [X \bmod p].$$

Show that, in contrast, $[[X \bmod p] \bmod N]$ need not equal $[X \bmod N]$.

7.9 Complete the details of the proof of the Chinese remainder theorem, showing that $\mathbb{Z}_N^*$ is isomorphic to $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$.

7.10 Corollary 7.21 shows that if $N = pq$ and $ed = 1 \bmod \phi(N)$ then for all $x \in \mathbb{Z}_N^*$ we have $(x^e)^d = x \bmod N$. Show that this holds for all $x \in \mathbb{Z}_N$.

Hint: Use the Chinese remainder theorem.

7.11 This exercise develops an efficient algorithm for testing whether an integer is a perfect power.

(a) Show that if $N = \hat{N}^e$ for some integers $\hat{N}, e > 1$ then $e \leq \|N\| + 1$.

(b) Given N and e with $2 \leq e \leq \|N\| + 1$, show how to determine in $\mathsf{poly}(\|N\|)$ time whether there exists an integer $\hat{N}$ with $\hat{N}^e = N$.

Hint: Use binary search.

(c) Given N, show how to test in $\mathsf{poly}(\|N\|)$ time whether N is a perfect power.

7.12 Given N and $a \in \mathbb{Z}_N^*$, show how to test in polynomial time whether a is a strong witness that N is composite.

7.13 Let $N = pq$ be a product of two distinct primes. Show that if $\phi(N)$ and N are known, then it is possible to compute p and q in polynomial time.

Hint: Derive a quadratic equation (over the integers) in the unknown p.

7.14 Let $N = pq$ be a product of two distinct primes. Show that if N and an integer d such that $3 \cdot d = 1 \bmod \phi(N)$ are known, then it is possible to compute p and q in polynomial time.

Hint: Obtain a small list of possibilities for $\phi(N)$ and then use the previous exercise.

7.15 Prove formally that the hardness of the CDH problem relative to $\mathcal{G}$ implies the hardness of the discrete logarithm problem relative to $\mathcal{G}$.

7.16 Prove formally that the hardness of the DDH problem relative to $\mathcal{G}$ implies the hardness of the CDH problem relative to $\mathcal{G}$.

7.17 Prove the third statement in Proposition 7.65.

7.18 Determine whether or not the following problem is hard. Let p be prime, and fix $x \in \mathbb{Z}_{p-1}^*$. Given p, x, and $y := [g^x \bmod p]$ (where g is a random value between 1 and $p-1$), find g; i.e., compute $y^{1/x} \bmod p$. If you claim the problem is hard, show a reduction to one of the assumptions introduced in this chapter. If you claim the problem is easy, present an algorithm, justify its correctness, and analyze its complexity.

7.19 Let $\mathcal{G}$ be an algorithm that, on input 1^n, outputs p, q, g where $p = 2q+1$ is a strong prime and g is a generator of the subgroup of quadratic residues modulo p. (See Section 7.3.3.) Show how to obtain compression in Construction 7.72 for p large enough.

7.20 Let $\mathcal{G}_1$ be as in Section 7.3.3. Show that hardness of the discrete logarithm problem relative to $\mathcal{G}_1$ implies the existence of a family of one-way permutations.

Hint: Define a permutation on elements of $\mathbb{Z}_p^*$.

7.21 Let GenRSA be as in Section 7.2.4. Prove that if the RSA problem is hard relative to GenRSA then Construction 7.75 shown below is a fixed-length collision-resistant hash function.

> ***CONSTRUCTION 7.75***
>
> Define (Gen, H) as follows:
>
> - Gen: on input 1^n, run GenRSA(1^n) to obtain N, e, d, and select $y \leftarrow \mathbb{Z}_N^*$. The key is $s := \langle N, e, y \rangle$.
> - H: if $s = \langle N, e, y \rangle$, then H^s maps inputs in $\{0,1\}^{3n}$ to outputs in $\mathbb{Z}_N^*$. Let $f_0^s(x) \stackrel{\text{def}}{=} [x^e \bmod N]$ and $f_1^s(x) \stackrel{\text{def}}{=} [y \cdot x^e \bmod N]$. For a $3n$-bit long string $x = x_1 \cdots x_{3n}$, define
>
> $$H^s(x) \stackrel{\text{def}}{=} f_{x_1}^s\left(f_{x_2}^s\left(\cdots\left(1\right)\cdots\right)\right).$$

Hint: Show that the eth root of y can be computed from any collision.

7.22 Consider the generalization of Construction 7.72 shown below.

> ***CONSTRUCTION 7.76***
>
> Define a fixed-length hash function (Gen, H) as follows:
>
> - Gen: on input 1^n, run $\mathcal{G}(1^n)$ to obtain $(\mathbb{G}, q, h_1)$ and then select $h_2, \ldots, h_t \leftarrow \mathbb{G}$. Output $s := \langle \mathbb{G}, q, (h_1, \ldots, h_t) \rangle$ as the key.
> - H: given a key $s = \langle \mathbb{G}, q, (h_1, \ldots, h_t) \rangle$ and input $(x_1, \ldots, x_t)$ with $x_i \in \mathbb{Z}_q$, output $H^s(x_1, \ldots, x_t) := \prod_i h_i^{x_i}$.

- Prove that if the discrete logarithm problem is hard relative to $\mathcal{G}$, and q is prime, then the construction is a fixed-length collision-resistant hash function for any $t = \mathsf{poly}(n)$.
- Discuss how this construction can be used to obtain *compression* regardless of the number of bits needed to represent elements of $\mathbb{G}$ (as long as it is polynomial in n).

Chapter 8

* Algorithms for Factoring and Computing Discrete Logarithms

As discussed in Chapter 7, there are currently no known *polynomial-time* algorithms for factoring or for computing discrete logarithms in certain groups. But this does not mean that brute-force search is the best available approach for attacking these problems! Here, we survey some more efficient algorithms for these problems. These algorithms are interesting in their own right, and serve as a nice application of some of the number theory we have already learned. Moreover, understanding the effectiveness of these algorithms is crucial for choosing cryptographic parameters in practice. If a cryptographic scheme based on factoring is supposed to withstand adversaries mounting a dedicated attack for 15 years, then — at a minimum! — the modulus N used by the scheme needs to be long enough so that the best-known factoring algorithm will take (at least) 15 years to successfully factor N.

Algorithms for other problems. We focus here on algorithms for factoring and computing discrete logarithms, not on algorithms for, say, solving the RSA or decisional Diffie-Hellman problems. This may seem somewhat misguided since the latter are used much more often than the former when constructing cryptographic schemes and, as noted in the previous chapter, the latter are potentially easier to solve than the former. (That is, solving the RSA problem is potentially easier than factoring, and solving the decisional Diffie-Hellman problem is possibly easier than computing discrete logarithms.) Our focus is justified by the fact that, currently, the best known algorithm for solving RSA is to first factor the modulus, and (in appropriate groups as discussed in Sections 7.3.3 and 7.3.4) the best known algorithm for solving the decisional Diffie-Hellman problem is to compute discrete logarithms.

8.1 Algorithms for Factoring

Throughout this section, we assume that $N = pq$ is a product of two distinct primes. We will also be most interested in the case that p and q are each of

This can be achieved by mapping group elements to strings in some way that "preserves uniformity". Efficient techniques for doing so exist, and these are called "randomness extractors". These are by now relatively standard in computer science, but are beyond the scope of this book.

Active adversaries. So far we have considered only the case of an eavesdropping adversary. Although eavesdropping attacks are by far the most common (as they are so easy to carry out), they are by no means the only possible attack. *Active* attacks, in which the adversary sends messages of its own to one or both of the parties are also a concern, and any protocol used in practice must be resilient to active attacks as well. When considering active attacks, it is useful to distinguish, informally, between *impersonation* attacks where only one of the honest parties is executing the protocol and the adversary impersonates the other party, and *man-in-the-middle* attacks where both honest parties are executing the protocol and the adversary is intercepting and modifying messages being sent from one party to the other.

We will not define security against either class of attacks, as such a definition is rather involved and also cannot be achieved without the parties sharing *some* information in advance. Nevertheless, it is worth remarking that the Diffie-Hellman protocol is *completely insecure* against man-in-the-middle attacks. In fact, a man-in-the-middle adversary can act in such a way that Alice and Bob terminate the protocol with different keys k_A and k_B that are both known to the adversary, yet neither Alice nor Bob can detect that any attack was carried out. We leave the details of this attack as an exercise.

The fact that the Diffie-Hellman protocol is not resilient to man-in-the-middle attacks does not detract in any way from its importance. The Diffie-Hellman protocol served as the first demonstration that asymmetric techniques (and number-theoretic problems) could be used to alleviate the problems of key distribution in cryptography. Furthermore, extensions of the Diffie-Hellman protocol can be shown to prevent man-in-the-middle attacks, and such protocols are widely used today.

The Diffie-Hellman key-exchange protocol in practice. The Diffie-Hellman protocol in its basic form is typically not used in practice due to its insecurity against man-in-the-middle attacks, as discussed above. However, it does form the nucleus of other key-exchange protocols that are resilient to man-in-the-middle attacks and are in wide use today. One notable example of a standardized protocol that relies on Diffie-Hellman key exchange is IPsec.

References and Additional Reading

We have only briefly discussed the problems of key distribution and key management in general. For more information, we recommend looking at

textbooks on network security. Our favorite is the one by Kaufman et al. [87], which provides an excellent treatment of different protocols for secure key distribution, what they aim to achieve, and how they work.

We highly recommend reading the original paper by Diffie and Hellman [47]. The history of the development of public-key cryptography is a fascinating one; the book by Levy [95] focuses on the political and historical aspects of the public-key revolution.

Exercises

9.1 Consider the following *interactive* protocol Π' for encrypting a message: first, the sender and receiver run a key-exchange protocol Π to generate a shared key k. Next, the sender computes $c \leftarrow \mathsf{Enc}_k(m)$ and sends c to the other party, who can decrypt and recover m using k.

(a) Formulate a definition of indistinguishable encryptions in the presence of an eavesdropper (cf. Definition 3.8) appropriate for this interactive setting.

(b) Prove that if Π is secure in the presence of an eavesdropper and $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a private-key encryption scheme that has indistinguishable encryptions in the presence of an eavesdropper, then Π' satisfies your definition given previously.

9.2 Describe in detail a man-in-the-middle attack on the Diffie-Hellman key-exchange protocol whereby the adversary ends up sharing a key k_A with Alice and a (different) key k_B with Bob, and Alice and Bob cannot detect that anything has gone wrong.

What happens if Alice and Bob try to detect the presence of a man-in-the-middle adversary by sending each other (encrypted) questions that only the other party would know how to answer?

9.3 Consider the following key-exchange protocol:

(a) Alice chooses $k, r \leftarrow \{0,1\}^n$ at random, and sends $s := k \oplus r$ to Bob.

(b) Bob chooses $t \leftarrow \{0,1\}^n$ at random and sends $u := s \oplus t$ to Alice.

(c) Alice computes $w := u \oplus r$ and sends w to Bob.

(d) Alice outputs k and Bob computes $w \oplus t$.

Show that Alice and Bob output the same key. Analyze the security of the scheme (i.e., either prove its security or show a concrete attack).

It remains only to show that families of trapdoor permutations have hard-core predicates. For some natural families, such as the one based on the RSA assumption that was discussed in the previous section, specific hard-core predicates are known. (As one example, it is known that if the RSA assumption holds then the least-significant bit is hard-core for the RSA family of trapdoor permutations.) For the general case, we can rely on the following result that can be proved by a suitable modification of Theorem 6.6:

THEOREM 10.29 *If a family of trapdoor permutations* Π *exists, then there exists a family of trapdoor permutations* $\widehat{\Pi}$ *along with a predicate* hc *that is hard-core for* $\widehat{\Pi}$.

Encrypting longer messages. Using Proposition 10.11, we know that we can extend Construction 10.27 to encrypt ℓ-bit messages for any polynomial ℓ. Doing so, the ciphertext corresponding to an ℓ-bit message $m = m_1 \cdots m_\ell$ encrypted with respect to the public key I would take the form

$$\langle f_I(x_1), \mathsf{hc}_I(x_1) \oplus m_1 \rangle, \ldots, \langle f_I(x_\ell), \mathsf{hc}_I(x_\ell) \oplus m_\ell \rangle$$

with $x_1, \ldots, x_\ell$ chosen independently and uniformly at random from $\mathcal{D}_I$.

We can reduce the size of the ciphertext by having the sender instead proceed as follows: Choose random $x_1 \leftarrow \mathcal{D}_I$ and compute $x_{i+1} := f_I(x_i)$ for $i = 1$ to ℓ. Then output the ciphertext

$$\langle x_{\ell+1}, \mathsf{hc}_I(x_1) \oplus m_1, \cdots, \mathsf{hc}_I(x_\ell) \oplus m_\ell \rangle .$$

A proof that this is secure uses ideas from Section 6.4, and is left as an advanced exercise.

References and Additional Reading

The idea of public-key encryption was first proposed (in the open literature, at least) by Diffie and Hellman [47]. Somewhat amazingly, the El Gamal encryption scheme [59] was not proposed until 1984 even though it can be viewed as a direct transformation of the Diffie-Hellman key exchange protocol introduced by Diffie and Hellman in 1976 (see Exercise 10.4). Rivest, Shamir, and Adleman [122] introduced the RSA assumption and proposed a public-key encryption scheme based on this assumption.

Definition 10.3 is rooted in the seminal work of Goldwasser and Micali [69], who were also the first to recognize the necessity of *probabilistic* encryption for satisfying this definition. A proof of security for a special case of hybrid encryption was first given by Blum and Goldwasser [22].

The public-key encryption scheme suggested by Rivest, Shamir, and Adleman [122] corresponds to the textbook RSA scheme shown here. The attacks described in Section 10.4.2 are due to [73, 45, 133, 27]; see [99, Chapter 8] and [25] for additional attacks and further explanation. The *PKCS #1 RSA Cryptography Standard* (both the latest version and previous versions) is available for download from `http://www.rsa.com/rsalabs`. A proof of Theorem 10.19 can be derived from results in [5, 75].

As noted in Chapter 4, chosen-ciphertext attacks were first formally defined by Naor and Yung [107] and Rackoff and Simon [121]. The chosen-ciphertext attacks on the "textbook RSA" and El Gamal schemes are immediate; the attack on PKCS #1 v1.5 is due to Bleichenbacher [20]. For more recent definitional treatments of public-key encryption under stronger attacks, including chosen-ciphertext attacks, the reader is referred to the works of Dolev et al. [50] and Bellare et al. [10]. The first efficient public-key encryption scheme secure against chosen-ciphertext attack was shown by Cramer and Shoup [39]. The expository article by Shoup [129] contains a discussion of the importance of security against chosen-ciphertext attacks.

The existence of public-key encryption based on arbitrary trapdoor permutations was shown by Yao [149], and the efficiency improvement discussed at the end of Section 10.7.2 is due to Blum and Goldwasser [22]. The reader interested in finding out more about hard-core predicates for the RSA family of trapdoor permutations is invited to peruse [5, 75, 4] and references therein.

When using any encryption scheme in practice the question of what key-length to use arises. This issue should not be taken lightly, and we refer the reader to [94] for a treatment of this issue.

Exercises

10.1 Assume a public-key encryption scheme for single-bit messages. Show that, given pk and a ciphertext c computed via $c \leftarrow \mathsf{Enc}_{pk}(m)$, it is possible for an unbounded adversary to determine m with probability 1. This shows that perfectly-secret public-key encryption is impossible.

10.2 Say a deterministic public-key encryption scheme is used to encrypt a message m that is known to lie in a small set of $\mathcal{L}$ possible values. Show how it is possible to determine m in time linear in $\mathcal{L}$ (assume that encryption of an element takes a single unit of time).

10.3 Show that for any CPA-secure public-key encryption scheme, the size of the ciphertext after encrypting a single bit is superlogarithmic in the security parameter. (That is, for $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$ it must hold that $|\mathsf{Enc}_{pk}(b)| = \omega(\log n)$ for any $b \in \{0, 1\}$.)

Hint: If not, the range of possible ciphertexts is only polynomial in size.

10.4 Show that any 2-round key-exchange protocol (that is, where each party sends a single message) satisfying Definition 9.1 can be converted into a public-key encryption scheme that is CPA-secure.

10.5 Show that in Definition 10.7, we can assume without loss of generality that $\mathcal{A}$ always outputs two vectors containing *exactly* $t(n)$ messages each. That is, show how to construct, for any scheme Π and any adversary $\mathcal{A}$, an adversary $\mathcal{A}'$ that always outputs vectors of the same length $t(n)$ for each fixed value of n and such that

$$\Pr[\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n)=1]=\Pr[\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A}',\Pi}(n)=1].$$

10.6 Prove Claim 10.9.

10.7 Fix N, and assume there exists an adversary $\mathcal{A}$ running in time t for which

$$\Pr\left[\mathcal{A}\left([x^e \bmod N]\right)=x\right]=0.01,$$

where the probability is taken over random choice of $x \leftarrow \mathbb{Z}^*_N$. Show that it is possible to construct an adversary $\mathcal{A}'$ for which

$$\Pr\left[\mathcal{A}\left([x^e \bmod N]\right)=x\right]=0.99.$$

The running time t' of $\mathcal{A}'$ should satisfy $t'=\mathsf{poly}(\|N\|,t)$.

Hint: Use the fact that $y^{1/e}\cdot r=(y\cdot r^e)^{1/e} \bmod N$.

10.8 The public exponent e in RSA can be chosen arbitrarily, subject to $\gcd(e,\phi(N))=1$. Popular choices of e include $e=3$ and $e=2^{16}+1$. Explain why such e are preferable to a random value of the same length.

Hint: See the algorithm for modular exponentiation in Appendix B.2.3.

10.9 Let GenRSA have the usual meaning. Consider the following experiment for an algorithm $\mathcal{A}$ and a function ℓ with $\ell(n)\le 2n-2$ for all n:

The padded RSA experiment $\mathsf{PAD}_{\mathcal{A},\mathsf{GenRSA},\ell}(n)$:

(a) *Run* $\mathsf{GenRSA}(1^n)$ *to obtain output* (N,e,d).

(b) *Give* N,e *to* $\mathcal{A}$, *who outputs a string* $m\in\{0,1\}^{\ell(n)}$.

(c) *Choose random* $y_0 \leftarrow \mathbb{Z}^*_N$. *Choose random* $r\leftarrow\{0,1\}^{\|N\|-\ell(n)-1}$ *and set*

$$y_1 := [(r\|m)^e \bmod N].$$

(d) *Choose random* $b\leftarrow\{0,1\}$. *Adversary* $\mathcal{A}$ *is given* y_b, *and outputs a bit* b'.

(e) *The output of the experiment is defined to be 1 if* $b'=b$, *and 0 otherwise.*

Say the ℓ-padded RSA problem is hard relative to GenRSA if for all probabilistic polynomial-time algorithms $\mathcal{A}$ there exists a negligible function negl such that $\Pr[\mathsf{PAD}_{\mathcal{A},\mathsf{GenRSA},\ell}(n)=1]\le\frac{1}{2}+\mathsf{negl}(n)$.

Prove that if the ℓ-padded RSA problem is hard relative to GenRSA, then the padded RSA encryption scheme (Construction 10.18) using ℓ is CPA-secure.

10.10 Say a function f is *hard-core for* GenRSA if for all PPT algorithms $\mathcal{A}$ there exists a negligible function negl such that

$$\left|\Pr[\mathcal{A}(N,e,y,f(x))=1]-\Pr[\mathcal{A}(N,e,y,f(r))=1]\right|\le\mathsf{negl}(n),$$

where in each case the probabilities are taken over the experiment in which $\mathsf{GenRSA}(1^n)$ outputs (N,e,d), random $x,r\leftarrow\mathbb{Z}^*_N$ are chosen, and y is set equal to $[x^e \bmod N]$.

For $x\in\mathbb{Z}^*_N$, let $\mathsf{lsb}(x)$ (resp., $\mathsf{msb}(x)$) denote the least- (resp., most-) significant bit of x when written as an integer using exactly $\|N\|$ bits. Define $f(x)\overset{\text{def}}{=}\mathsf{msb}(x)\|\mathsf{lsb}(x)$. It can be shown that if the RSA problem is hard relative to GenRSA, then f is hard-core for GenRSA [4]. Prove Theorem 10.19 by relying on this result.

Hint: Note that in Construction 10.18, $\mathsf{msb}(r\|m)$ is always equal to 0.

10.11 Consider the following public-key encryption scheme. The public key is $(\mathbb{G},q,g,h)$ and the private key is x, generated exactly as in the El Gamal encryption scheme. In order to encrypt a bit b, the sender does the following:

(a) If $b=0$ then choose a random $y\leftarrow\mathbb{Z}_q$ and compute $c_1=g^y$ and $c_2=h^y$. The ciphertext is $\langle c_1,c_2\rangle$.

(b) If $b=1$ then choose independent random $y,z\leftarrow\mathbb{Z}_q$, compute $c_1=g^y$ and $c_2=g^z$, and set the ciphertext equal to $\langle c_1,c_2\rangle$.

Show that it is possible to decrypt efficiently given knowledge of x. Prove that this encryption scheme is CPA-secure if the decisional Diffie-Hellman problem is hard relative to $\mathcal{G}$.

10.12 The natural way of applying hybrid encryption to the El Gamal encryption scheme is as follows. The public key is $pk=\langle\mathbb{G},q,g,h\rangle$ as in the El Gamal scheme, and to encrypt a message m the sender chooses random $k\leftarrow\{0,1\}^n$ and sends

$$\langle g^r,\ h^r\cdot k,\ \mathsf{Enc}_k(m)\rangle,$$

where $r\leftarrow\mathbb{Z}_q$ is chosen at random and Enc represents a private-key encryption scheme. Suggest an improvement that results in a shorter ciphertext containing only a *single* group element followed by a private-key encryption of m.

10.13 Show that Proposition 10.11 does not hold in the setting of CCA-security. In contrast, Theorem 10.10 *does* hold in the setting of CCA-security. Explain why in the setting of CPA security there is no distinction between multiple messages and a single long message, whereas in the setting of CCA security there is.

10.14 Consider the following version of padded RSA encryption. Assume that the message m to be encrypted has length $\|N\|/2$ (i.e., roughly half the length of the modulus). To encrypt, first pad m to the left with one byte of zeroes, then 10 random bytes, and then all zeroes; the result is denoted $\overline{m}$ (that is, $\overline{m} = (0^k \| r \| 00000000 \| m)$, where k is the number of zeroes needed to make $\overline{m}$ the appropriate size). Finally, compute $c = [\overline{m}^e \bmod N]$. Describe a chosen-ciphertext attack on this scheme. Why is it easier to construct a chosen-ciphertext attack on this scheme than on PKCS #1 v1.5?

10.15 Let Π be a CCA-secure public-key encryption scheme and let Π' be a CCA-secure private-key encryption scheme. Is Construction 10.12 instantiated using Π and Π' CCA-secure? Prove your answer.

10.16 Consider the following variant of Construction 10.27 which gives an alternative way of encrypting using any family of trapdoor permutations:

CONSTRUCTION 10.30

Let $\widehat{\Pi} = (\widehat{\mathsf{Gen}}, f)$ and hc be as in Construction 10.27.

- Gen: as in Construction 10.27.
- Enc: on input a public key I and a message $m \in \{0,1\}$, choose a random $x \leftarrow \mathcal{D}_I$ such that $\mathsf{hc}_I(x) = m$, and output the ciphertext $f_I(x)$.
- Dec: on input a private key td and a ciphertext y with $y \in \mathcal{D}_{\mathsf{td}}$, compute $x := f_I^{-1}(y)$ and output the message $\mathsf{hc}_I(x)$.

(a) Argue that encryption can be performed in polynomial time.

(b) Prove that if $\widehat{\Pi}$ is a family of trapdoor permutations and hc is a hard-core predicate of $\widehat{\Pi}$, then this construction is CPA-secure.

10.17 Consider the following protocol for two parties A and B to flip a fair coin (more complicated versions of this might be used for Internet gambling): **(1)** a trusted party T publishes her public key pk; **(2)** A chooses a random bit b_A, encrypts it using pk, and announces the ciphertext c_A to B and T; **(3)** next, B acts symmetrically and announces a ciphertext $c_B \neq c_A$; **(4)** T decrypts both c_A and c_B, and the parties XOR the results to obtain the value of the coin.

(a) Argue that even if A is dishonest (but B is honest), the final value of the coin is uniformly distributed.

(b) Assume the parties use El Gamal encryption (where the bit b is encoded as the group element g^b). Show how a dishonest B can bias the coin to any value he likes.

(c) Suggest what type of encryption scheme would be appropriate to use here. Can you define an appropriate notion of security and prove that your suggestion achieves this definition?

Exercises

12.1 Prove that the existence of a one-time signature scheme for 1-bit messages implies the existence of one-way functions.

12.2 For each of the following variants of the definition of security for signatures, state whether textbook RSA is secure and prove your answer:

(a) In this first variant, the experiment is as follows: the adversary is given the public key pk and a random message m. The adversary is then allowed to query the signing oracle once on a single message that does not equal m. Following this, the adversary outputs a signature σ and succeeds if $\mathsf{Vrfy}_{pk}(m, \sigma) = 1$. As usual, security is said to hold if the adversary can succeed in this experiment with at most negligible probability.

(b) The second variant is as above, except that the adversary is not allowed to query the signing oracle at all.

12.3 The textbook Rabin signature scheme is the same as the textbook RSA scheme, except using the Rabin trapdoor permutation (see Section 11.2). Show that textbook Rabin signatures have the property that an adversary can actually obtain the private key using an adaptive chosen-message attack.

12.4 Another approach (besides hashed RSA) to trying to construct secure RSA signatures is to use *encoded RSA*. Here, public and private keys are as in textbook RSA; a public encoding function enc is fixed; and the signature on a message m is computed as $\sigma := [\mathsf{enc}(m)^d \bmod N]$.

(a) How is verification performed in encoded RSA?

(b) Discuss why appropriate choice of an encoding function prevents the "no-message attack" described in Section 12.3.1.

(c) Show that encoded RSA is insecure for $\mathsf{enc}(m) = 0\|m\|0^{\ell/10}$ (where $\ell = \|N\|$, $|m| = 9\ell/10 - 1$, and m is not the all-0 message).

(d) Show that encoded RSA is insecure for $\mathsf{enc}(m) = 0\|m\|0\|m$ (where $|m| = (\|N\| - 1)/2$ and m is not the all-0 message)

12.5 Show that Construction 4.5 for constructing a variable-length MAC from any fixed-length MAC can also be used (with appropriate modifications) to construct a signature scheme for arbitrary-length messages from any signature scheme for messages of fixed length $\ell(n)$. What is the minimal length that $\ell(n)$ can be? What are the advantages and disadvantages

12.6 Let f be a one-way permutation (as in Definition 6.2). Consider the following signature scheme for messages in the set $\{1, \ldots, n\}$:

- To generate keys, choose random $x \leftarrow \{0,1\}^n$ and set $y := f^n(x)$. The public key is y and the private key is x.
- To sign message $i \in \{1, \ldots, n\}$, output $f^{n-i}(x)$ (where $f^0(x) \stackrel{\text{def}}{=} x$).
- To verify signature σ on message i with respect to public key y, check whether $y \stackrel{?}{=} f^i(\sigma)$.

(a) Show that the above is not a one-time signature scheme. Given a signature on a message i, for what messages j can an adversary output a forgery?

(b) Prove that no PPT adversary given a signature on i can output a forgery on any message $j > i$ except with negligible probability.

(c) Suggest how to modify the scheme so as to obtain a one-time signature scheme.

Hint: Include two values y, y' in the public key.

12.7 Consider the Lamport one-time signature scheme. Describe an adversary who obtains signatures on *two* messages of its choice and can then forge signatures on any message it likes.

12.8 The Lamport scheme uses 2ℓ values in the public key to sign messages of length ℓ. Consider the following variant: the private key consists of 2ℓ values $x_1, \ldots, x_{2\ell}$ and the public key contains the values $y_1, \ldots, y_{2\ell}$ where $y_i = f(x_i)$. A message $m \in \{0,1\}^{\ell'}$ is mapped in a one-to-one fashion to a subset $S_m \subset \{1, \ldots, 2\ell\}$ of size ℓ. To sign m, the signer reveals $\{x_i\}_{i \in S_m}$. Prove that this gives a one-time signature scheme. What is the maximum message-length ℓ' that this scheme supports?

12.9 A *strong* one-time signature scheme satisfies the following (informally): given a signature σ on a message m, it is infeasible to output $(m', \sigma') \neq (m, \sigma)$ for which σ' is a valid signature on m' (note that $m = m'$ is allowed).

(a) Give a formal definition of strong one-time signatures.

(b) Assuming the existence of one-way functions, show a one-way function for which Lamport's scheme is *not* a strong one-time signature scheme.

(c) Construct a strong one-time signature scheme based on any assumption used in this book.

12.10 Let (Gen, H) be a collision-resistant hash function, where H maps strings of length $2n$ to strings of length n. Prove that the function family (Gen, Samp, H) is one-way (cf. Definition 7.70), where Samp is the trivial algorithm that samples a random string of length $2n$.

> **Hint:** Choosing random $x \leftarrow \{0,1\}^{2n}$ and finding an inverse of $y = H^s(x)$ does not guarantee a collision. But it does yield a collision most of the time...

12.11 At the end of Section 12.6.2, we show how a pseudorandom function can be used to make Construction 12.12 stateless. Does a similar approach work for the path-based scheme described in Section 12.6.1? If so, sketch a construction and proof. If not, explain why and modify the scheme to obtain a stateless variant.

12.12 Prove Theorem 12.14.

12.13 Assume revocation of certificates is handled in the following way: when a user Bob claims that the private key corresponding to his public key pk_B has been stolen, the user sends to the CA a statement of this fact *signed with respect to* pk_B. Upon receiving such a signed message, the CA revokes the appropriate certificate.

Explain why it is not necessary for the CA to check Bob's identity in this case. In particular, explain why it is of no concern that an adversary who has stolen Bob's private key can forge signatures with respect to pk_B.

Chapter 13

Public-Key Cryptosystems in the Random Oracle Model

In the previous three chapters, we have seen constructions of public-ke cryption schemes and digital signatures based on a variety of assump For the most part, however, the provably-secure schemes we have disc and analyzed are not particularly efficient. Specifically:

- In Section 10.4.3 we saw an encryption scheme that can be proven s based on the RSA assumption (cf. Theorem 10.19), but the efficien this scheme does not come close to the efficiency of the textbook encryption scheme described in Section 10.4.1. In fact, no secure enc tion scheme based on RSA with efficiency comparable to the text RSA encryption scheme is currently known. (The padded RSA enc tion scheme with $\ell = \Theta(n)$ is efficient, but its security is not know follow from the assumption that the RSA problem is hard.)
- We have not shown any public-key encryption scheme that is se against chosen-ciphertext attacks. Though efficient schemes base the decisional Diffie-Hellman assumption and others are known, the no known scheme based on RSA that is even remotely practical.
- We have seen only a single example of a digital signature scheme is existentially unforgeable under an adaptive chosen-message att This construction, shown in Section 12.6.2, is not very practical. signature scheme is currently known that can be proven secure base any of the assumptions introduced in this book, and whose efficienc comparable to the textbook RSA signature scheme.

The above hold even if we are willing to assume efficient pseudorandom fu tions (that could be instantiated using a block cipher such as DES or A and/or efficient collision-resistant hash functions (that could be instantia using a cryptographic hash function such as SHA-1). We conclude that th are few public-key cryptosystems that are both: (1) efficient enough to used in practice, yet (2) can be proven secure based on "standard" cryp graphic assumptions (such as the RSA