Jonathan Katz and Yehuda Lindell

Introduction to Modern Cryptography – 2nd Edition

Solutions Manual

©2016 Jonathan Katz and Yehuda Lindell. All Rights Reserved

CRC PRESS
Boca Raton London New York Washington, D.C.

Contents

Preface

In this solutions manual, we provide solutions to all the exercises in the book *Introduction to Modern Cryptography, second edition.*

A significant number of the exercises ask for proofs. While we give full proofs in many cases, for some exercises we provide only the high-level ideas behind a solution and omit the details. This is especially so when an exercise can be solved using a proof that is very similar to one that already appears in the book, in which case we comment only on the necessary modifications. In all cases, however, we would expect students to provide full and detailed proofs; we find that students learn best by going through this process.

This solutions manual is intended for instructors teaching a course using our book. For obvious reasons, we do not want these solutions to be widely disseminated; please keep this in mind when using them. (For example, do not post them online, even if password-protected. In addition, preferably provide hardcopy print-outs of the relevant sections to TAs.)

If you find errors or typos in the solutions, or if you find an alternative solution that you find superior (e.g., simpler or more instructive), please let us know by emailing us at `jkatz@cs.umd.edu` and `lindell@biu.ac.il` with "Introduction to Modern Cryptography" in the subject line.

Chapter 1

Introduction – Solutions

1.1 Decrypt the ciphertext provided at the end of the section on mono-alphabetic substitution.

Solution: The ciphertext decrypts to the following plaintext (we have added punctuation and capitalization):

> Cryptographic systems are extremely difficult to build. Nevertheless, for some reason many non-experts insist on designing new encryption schemes that seem to them to be more secure than any other scheme on earth. The unfortunate truth, however, is that such schemes are usually trivial to break.

We would expect students to describe the methodology they used to derive the solution.

1.2 Provide a formal definition of the Gen, Enc, and Dec algorithms for the mono-alphabetic substitution cipher.

Solution: For this exercise, we identify numbers and letters in the natural way. That is, $a = 0$, $b = 1$ and so on. We start with the mono-alphabetic substitution cipher:

- Gen: Choose a random permutation π of $\{0, \ldots, 25\}$ and let the key be this permutation. (A random permutation on $\{0, \ldots, 25\}$ can be chosen as follows: for $i = 0$ to 25, set $\pi(i)$ equal to a random number from $\{0, \ldots, 25\}$ that has not been chosen so far.)
- Enc: Given a plaintext $m = m_1, \ldots, m_\ell$ (where $m_i \in \{0, \ldots, 25\}$) and a key π, set $c_i := \pi(m_i)$ and output $c_1, \ldots, c_n$.
- Dec: Given a ciphertext $c = c_1, \ldots, c_n$ and key π, set $m_i := \pi^{-1}(c_i)$ where π^{-1} is the inverse of π and output $m_1, \ldots, m_n$.

1.3 Provide a formal definition of the Gen, Enc, and Dec algorithms for the Vigenère cipher. (Note: there are several plausible choices for Gen; choose one.)

Solution: As in the previous exercise, we identify numbers and letters in the natural way. That is, $a = 0$, $b = 1$ and so on.

- Gen: Choose a random period: this can be chosen uniformly in a fixed set of some size, or it can be chosen according to some valid

probability distribution over the integers (e.g., assign the length $5 + i$ with probability 2^{-i}). Denote the chosen period by t. For $i = 0, \ldots, t-1$ choose uniform k_i in $\{0, \ldots, 25\}$. Output the key $k = k_0, \ldots, k_{t-1}$.

- Enc: Given a plaintext $p = p_0, \ldots, p_n$ and a key $k = k_0, \ldots, k_{t-1}$, set $c_i := [p_i + k_{[i \bmod t]} \bmod 26]$. Output $c_0, \ldots, c_n$.
- Dec: Given a ciphertext $c = c_0, \ldots, c_n$ and a key $k = k_0, \ldots, k_{t-1}$, set $p_i := [c_i - k_{[i \bmod t]} \bmod 26]$. Output $p_0, \ldots, p_n$.

1.4 Implement the attacks described in this chapter for the shift cipher and the Vigenére cipher.

No solution given.

1.5 Show that the shift, substitution, and Vigenère ciphers are all trivial to break using a chosen-plaintext attack. How much known plaintext is needed to completely recover the key for each of the ciphers?

Solution: For the shift cipher: ask for the encryption of any plaintext character p and let c be the ciphertext character returned; the key is simply $k := [c - p \mod 26]$. The encryption of only a single plaintext character thus suffices to recover the key. For the substitution cipher, given a plaintext character p_i and corresponding ciphertext character c_i, we can conclude that $\pi(p_i) = c_i$ (where π is the permutation determining the key as in the solution of Exercise 1.2). In order to fully determine the key, it therefore suffices to ask for an encryption of a plaintext containing 25 distinct letters of the alphabet. (Since π is a permutation, knowing the value of π on 25 inputs fully determines the value of π on the last remaining input.) For the Vigenère cipher, if the period t is known then the encryption of a plaintext of length t (consecutive) suffices to recover the entire key. If only an upper bound $t_{\max}$ on t is known, then t must be learned as well; this can be done using a single plaintext of length $O(t)$. (Note: it is actually a bit challenging to determine the minimal length of a plaintext that suffices to determine t. We only expect students to understand that a plaintext of length $O(t_{\max})$ suffices.)

1.6 Assume an attacker knows that a user's password is either `abcd` or `bedg`. Say the user encrypts his password using the shift cipher, and the attacker sees the resulting ciphertext. Show how the attacker can determine the user's password, or explain why this is not possible.

Solution: In the shift cipher, the relative shift between characters is preserved. Thus an encryption of `abcd` will always be a ciphertext containing 4 consecutive characters (e.g., `lmno`), whereas `bedg` will not.

1.7 Repeat the previous exercise for the Vigenère cipher using period 2, using period 3, and using period 4.

Solution: For the sake of clarity, we will mostly omit the fact that operations are modulo 26 in this solution.

When the period is 2, it is impossible to determine which password was encrypted. This is due to the fact that the shifts used in the first and third (resp., second and fourth) positions are the same, and the difference between the first and third (resp., second and fourth) characters in the first password is the same as the difference between the first and third (resp., second and fourth) characters in the second password.

When the period is 3, it is possible to tell which password was encrypted because the shifts used in the first and fourth positions are the same, but the difference between the first and fourth characters of the first plaintext is not the same as the difference between the first and fourth characters of the second plaintext.

When the period is 4, it is impossible to tell which password was encrypted, because using a 4-character key to encrypt a 4-character plaintext is perfectly secret (by analogy to the one-time pad).

1.8 The shift, substitution, and Vigenère ciphers can also be defined over the 128-character ASCII alphabet (rather than the 26-character English alphabet).

(a) Provide a formal definition of each of these schemes in this case.

(b) Discuss how the attacks we have shown in this chapter can be modified to break each of these modified schemes.

Solution: We describe the solution for the Vigenère cipher, which is the most complex case.

(a) Key generation chooses a period t (say, uniform in $\{1, \ldots, t_{\max}\}$ for some specified $t_{\max}$) and then, for $i = 0, \ldots, t-1$, chooses uniform $k_i \in \{1, \ldots, 127\}$. The encryption of a plaintext $m = m_1, \ldots, m_\ell$ is $c = c_1, \ldots, c_\ell$, where $c_i = [m_i + k_{[i \bmod t}] \bmod 128]$. Decryption is done in the natural way.

(b) The exact same attacks described in the text work, though one must be careful now to let p_i be the frequency of the ith ASCII character (so, e.g., the frequency of 'A' is different from the frequency of 'a,' and frequencies of the space character and punctuation is also taken into account).

Chapter 2

Perfectly Secret Encryption – Solutions

2.1 Prove that, by redefining the key space, we may assume that the key-generation algorithm Gen chooses a key uniformly at random from the key space, without changing $\Pr[C = c \mid M = m]$ for any m, c.

Solution: If Gen is a randomized algorithm, we may view it as a deterministic algorithm that takes as input a random tape ω of some length; the distribution on the output of Gen is, by definition, the distribution obtained by choosing uniform ω and then running $\mathsf{Gen}(\omega)$. So, rather than letting the key be the output of Gen, we can simply let the key be ω itself (and redefine the key space accordingly).

Formally, given a scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ in which Gen is randomized, construct a new scheme $(\mathsf{Enc}', \mathsf{Dec}')$ where the key is a uniform ω. Then define $\mathsf{Enc}'_\omega(m)$ to compute $k := \mathsf{Gen}(\omega)$ followed by $\mathsf{Enc}_k(m)$, and define decryption analogously.

2.2 Prove that, by redefining the key space, we may assume that Enc is deterministic without changing $\Pr[C = c \mid M = m]$ for any m, c.

Solution: As in the previous exercise, if Enc is a randomized algorithm then we may view it as being a deterministic algorithm that also takes a random tape ω as additional input. The distribution on the output of $\mathsf{Enc}_k(m)$ is then, by definition, the distribution obtained by choosing uniform ω and then computing $\mathsf{Enc}_k(m;\omega)$. We then define key generation to include ω as well as k (and redefine the key space accordingly).

Formally, given a scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ in which Enc is randomized, construct a new scheme $(\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}' = \mathsf{Dec})$ as follows. Gen' computes $k \leftarrow \mathsf{Gen}$ and also chooses uniform ω; the key is (k, ω). Then define $\mathsf{Enc}'_{(k,\omega)}(m)$ to be $\mathsf{Enc}_k(m;\omega)$.

2.3 Prove or refute: An encryption scheme with message space $\mathcal{M}$ is perfectly secret if and only if for every probability distribution over $\mathcal{M}$ and every $c_0, c_1 \in \mathcal{C}$ we have $\Pr[C = c_0] = \Pr[C = c_1]$.

Solution: This is not true. Consider modifying the one-time pad so encryption appends a bit that is 0 with probability 1/4 and 1 with probability 3/4. This scheme will still be perfectly secret, but ciphertexts ending in 1 are more likely than ciphertexts ending in 0.

2.4 Prove the second direction of Lemma 2.4.

Solution: Say $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is perfectly secret. Fix two messages m, m' and a ciphertext c that occurs with nozero probability, and consider the uniform distribution over $\{m, m'\}$. Perfect secrecy implies that $\Pr[M = m \mid C = c] = 1/2 = \Pr[M = m' \mid C = c]$. But

$$\begin{aligned}\frac{1}{2} = \Pr[M = m \mid C = c] &= \frac{\Pr[C = c \mid M = m] \cdot \Pr[M = m]}{\Pr[C = c]} \\ &= \frac{\frac{1}{2}\Pr[C = c \mid M = m]}{\Pr[C = c]},\end{aligned}$$

and so $\Pr[C = c \mid M = m] = \Pr[\mathsf{Enc}_K(m) = c] = \Pr[C = c]$. Since an analogous calculation holds for m' as well, we conclude that $\Pr[\mathsf{Enc}_K(m) = c] = \Pr[\mathsf{Enc}_K(m') = c]$.

2.5 Prove Lemma 2.6.

Solution: We begin by proving that any encryption scheme that is perfectly secret is perfectly indistinguishable. Every adversary $\mathcal{A}$ participating in $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$ defines a fixed pair of plaintext messages m_0, m_1 that it outputs in the first step of the experiment. (Note that since $\mathcal{A}$ is a deterministic algorithm, it always outputs the same pair of messages.) Fix $\mathcal{A}$ and fix m_0, m_1 output by $\mathcal{A}$. By Lemma 2.4, for every $m_0, m_1 \in \mathcal{M}'$ and every $c \in \mathcal{C}$,

$$\Pr[\mathsf{Enc}_K(m_0) = c] = \Pr[\mathsf{Enc}_K(m_1) = c]. \tag{2.1}$$

In particular, the above holds for m_0, m_1 output by $\mathcal{A}$ and for any c. Let $\mathcal{C}_0$ (resp., $\mathcal{C}_1$) denote the set of ciphertexts for which $\mathcal{A}$ outputs 0 (resp., 1) at the conclusion of experiment $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$; since $\mathcal{A}$ is deterministic these sets are well-defined. Note that since $\mathcal{A}$ must output either 0 or 1, it follows that $\mathcal{C} = \mathcal{C}_0 \cup \mathcal{C}_1$. We have:

$$\begin{aligned}\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1] &= \frac{1}{2} \cdot \Pr[\mathcal{A} \text{ outputs } 0 \mid b = 0] + \frac{1}{2} \cdot \Pr[\mathcal{A} \text{ outputs } 1 \mid b = 1] \\ &= \frac{1}{2} \cdot \sum_{c \in \mathcal{C}_0} \Pr[c = \mathsf{Enc}_K(m_0)] + \frac{1}{2} \cdot \sum_{c \in \mathcal{C}_1} \Pr[c = \mathsf{Enc}_K(m_1)] \\ &= \frac{1}{2} \cdot \sum_{c \in \mathcal{C}_0} \Pr[c = \mathsf{Enc}_K(m_0)] + \frac{1}{2} \cdot \sum_{c \in \mathcal{C}_1} \Pr[c = \mathsf{Enc}_K(m_0)] \\ &= \frac{1}{2} \cdot \sum_{c \in \mathcal{C}} \Pr[c = \mathsf{Enc}_K(m_0)] \;=\; \frac{1}{2} \cdot (1) = \frac{1}{2}.\end{aligned}$$

where the second last equality is due to Equation (2.1).

We now proceed to the other direction, that any perfectly indistinguishable encryption scheme is also perfectly secret. We prove the contrapositive. Assume encryption scheme Π is not perfectly secret with respect

to Definition 2.1. Then by Lemma 2.4, there must exist two messages $m_0, m_1 \in \mathcal{M}$ and a ciphertext $\tilde{c} \in \mathcal{C}$ for which

$$\Pr[\mathsf{Enc}_K(m_0) = \tilde{c}] \neq \Pr[\mathsf{Enc}_K(m_1) = \tilde{c}]. \tag{2.2}$$

Let $\mathcal{A}$ be an adversary who outputs m_0, m_1 in the first step of $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$. Then if $\mathcal{A}$ receives any ciphertext $c' \neq \tilde{c}$ it outputs a random bit b'; if it receives the ciphertext $\tilde{c}$, it outputs $b' = 0$.

Since each message is chosen with probability $1/2$ in the experiment, we have

$$\begin{aligned}&\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1]\\ &= \frac{1}{2} \cdot \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1 \mid M = m_0] + \frac{1}{2} \cdot \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1 \mid M = m_1].\end{aligned}$$

Next,

$$\begin{aligned}&\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1 \mid M = m_0]\\ &= \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1 \ \wedge \ \tilde{c} = \mathsf{Enc}_K(m_0)]\\ &\quad + \ \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1 \ \wedge \ \tilde{c} \neq \mathsf{Enc}_K(m_0)]\\ &= \Pr[\tilde{c} = \mathsf{Enc}_K(m_0)] + \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1 \ \wedge \ \tilde{c} \neq \mathsf{Enc}_K(m_0)]\\ &= \Pr[\tilde{c} = \mathsf{Enc}_K(m_0)] + \frac{1}{2} \cdot \Pr[\tilde{c} \neq \mathsf{Enc}_K(m_0)]\end{aligned}$$

where the second-to-last equality is because $\mathcal{A}$ always outputs 0 when receiving $\tilde{c}$ (and when $M = m_0$ this means that $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1$), and the last equality is because $\mathcal{A}$ outputs a random bit when given a ciphertext $c' \neq \tilde{c}$. A similar analysis for the case that $M = m_1$ (the only difference is that now $\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1 \ \wedge \ \tilde{c} \neq \mathsf{Enc}_K(m_1)] = 0$) gives

$$\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1 \mid M = m_1] = \frac{1}{2} \cdot \Pr[\tilde{c} \neq \mathsf{Enc}_K(m_1)].$$

Putting this all together we have:

$$\begin{aligned}\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1] &= \frac{1}{2} \cdot \Big(\Pr[\tilde{c} = \mathsf{Enc}_K(m_0)] + \frac{1}{2} \cdot \Pr[\tilde{c} \neq \mathsf{Enc}_K(m_0)]\Big)\\ &\quad + \ \frac{1}{2} \cdot \frac{1}{2} \cdot \Pr[\tilde{c} \neq \mathsf{Enc}_K(m_1)]\\ &= \frac{1}{2} \cdot \Big(\Pr[\tilde{c} = \mathsf{Enc}_K(m_0)] + \frac{1}{2} \cdot (1 - \Pr[\tilde{c} = \mathsf{Enc}_K(m_0)])\Big)\\ &\quad + \ \frac{1}{2} \cdot \frac{1}{2} \cdot \Pr[\tilde{c} \neq \mathsf{Enc}_K(m_1)]\\ &= \frac{1}{4} + \frac{1}{4} \cdot \Pr[\tilde{c} = \mathsf{Enc}_K(m_0)] + \frac{1}{4} \cdot \Pr[\tilde{c} \neq \mathsf{Enc}_K(m_1)]\\ &\neq \frac{1}{4} + \frac{1}{4} \cdot \Pr[\tilde{c} = \mathsf{Enc}_K(m_1)] + \frac{1}{4} \cdot \Pr[\tilde{c} \neq \mathsf{Enc}_K(m_1)]\\ &= \frac{1}{4} + \frac{1}{4} = \frac{1}{2}\end{aligned}$$

where the inequality is by Equation (2.2). Since $\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1] \neq 1/2$ we have that Π is not perfectly indistinguishable.

2.6 For each of the following encryption schemes, state whether the scheme is perfectly secret. Justify your answer in each case.

(a) The message space is $\mathcal{M} = \{0, \ldots, 4\}$. Algorithm Gen chooses a uniform key from the key space $\{0, \ldots, 5\}$. $\mathsf{Enc}_k(m)$ returns $[k + m \bmod 5]$, and $\mathsf{Dec}_k(c)$ returns $[c - k \bmod 5]$.

(b) The message space is $\mathcal{M} = \{m \in \{0,1\}^\ell \mid \text{the last bit of } m \text{ is } 0\}$. Gen chooses a uniform key from $\{0,1\}^{\ell-1}$. $\mathsf{Enc}_k(m)$ returns ciphertext $m \oplus (k \| 0)$, and $\mathsf{Dec}_k(c)$ returns $c \oplus (k \| 0)$.

Solution:

(a) The scheme is not perfectly secret. To see this, we can use the equivalent definition of perfect secrecy given by Equation (2.1). If the message is 0, then the ciphertext is 0 if and only if $k \in \{0, 5\}$. So $\Pr[\mathsf{Enc}_K(0) = 0] = 1/3$. On the other hand, if the message is 1, then the ciphertext is 0 if and only if $k = 4$. So

$$\Pr[\mathsf{Enc}_K(1) = 0] = 1/6 \neq \Pr[\mathsf{Enc}_K(0) = 0].$$

(b) One can prove that this is perfectly secret by analogy with the one-time pad. (Essentially the final bit of the message is being ignored here, since it is always 0.)

2.7 When using the one-time pad with the key $k = 0^\ell$, we have $\mathsf{Enc}_k(m) = k \oplus m = m$ and the message is sent in the clear! It has therefore been suggested to modify the one-time pad by only encrypting with $k \neq 0^\ell$ (i.e., to have Gen choose k uniformly from the set of *nonzero* keys of length ℓ). Is this modified scheme still perfectly secret? Explain.

Solution: The modified scheme is *not* perfectly secret. To see this formally, consider the uniform distribution over $\mathcal{M} = \{0,1\}^\ell$. For any fixed message $\alpha \in \{0,1\}^\ell$, we have

$$\Pr[M = \alpha \mid C = \alpha] = 0 \neq \Pr[M = \alpha].$$

This contradicts perfect secrecy.

We conclude that in order to obtain perfect secrecy, it must be possible to encrypt using the key 0^ℓ. This may seem counter-intuitive, since this key does not change the plaintext. However, note that an eavesdropper has no way of knowing if the key is 0^ℓ, so the fact that the ciphertext is the same as the plaintext in this case is really of no help to the adversary.

2.8 Let Π denote the Vigenère cipher where the message space consists of all 3-character strings (over the English alphabet), and the key is generated by first choosing the period t uniformly from $\{1, 2, 3\}$ and then letting the key be a uniform string of length t.

(a) Define $\mathcal{A}$ as follows: $\mathcal{A}$ outputs $m_0 = \texttt{aab}$ and $m_1 = \texttt{abb}$. When given a ciphertext c, it outputs 0 if the first character of c is the same as the second character of c, and outputs 1 otherwise. Compute $\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1]$.

(b) Construct and analyze an adversary $\mathcal{A}'$ for which $\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A}',\Pi} = 1]$ is greater than your answer from part (a).

Solution:

(a) Say aab is encrypted to give ciphertext c. What is the probability that the first and second characters of c are equal? When $t = 1$ (which occurs 1/3 of the time) this always happens. But when $t \in \{2, 3\}$ this happens only if the first and second characters of the key are equal, which occurs with probability 1/26. So

$$\Pr[\mathcal{A} \text{ outputs } 0 \mid m_0 \text{ is encrypted}] = \frac{1}{3} + \frac{2}{3} \cdot \frac{1}{26} \approx 0.359.$$

If instead abb is encrypted, then the first and second characters of c can never be equal when $t = 1$, but are equal with probability 1/26 when $t \in \{2, 3\}$. Thus,

$$\Pr[\mathcal{A} \text{ outputs } 0 \mid m_1 \text{ is encrypted}] = \frac{2}{3} \cdot \frac{1}{26} \approx 0.026.$$

We therefore have

$$\begin{aligned}
&\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1] \\
&= \frac{1}{2} \cdot \Pr[\mathcal{A} \text{ outputs } 0 \mid m_0 \text{ is encrypted}] \\
&\quad + \frac{1}{2} \cdot \Pr[\mathcal{A} \text{ outputs } 1 \mid m_1 \text{ is encrypted}] \\
&\approx \frac{1}{2} \cdot 0.359 + \frac{1}{2} \cdot 0.974 \approx 0.667.
\end{aligned}$$

(b) There are many possible solutions; we present one. Consider the adversary $\mathcal{A}'$ who outputs $m_0 = \texttt{aaa}$ and $m_1 = \texttt{abc}$ and outputs '0' iff the first and second characters in the ciphertext c are the same, or if the first and last characters in the ciphertext are the same. Call this event E.

Say aaa is encrypted. If $t \in \{1, 2\}$ then E always happens. When $t = 3$ all characters in the ciphertext are uniform and independent;

rather than calculate the probability of E in this case, we just let p denote that probability.

Say `abc` is encrypted. If $t = 1$ then E never happens. When $t = 2$ the first and last characters of c are never equal, but the first and second characters are equal with probability 1/26. When $t = 3$ then all characters in the ciphertext are random and so E occurs with probability p.

Putting everything together gives:

$$\begin{aligned}
&\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A}',\Pi} = 1] \\
&= \frac{1}{2} \cdot \Pr[\mathcal{A}' \text{ outputs } 0 \mid m_0 \text{ is encrypted}] \\
&\quad + \frac{1}{2} \cdot \Pr[\mathcal{A}' \text{ outputs } 1 \mid m_1 \text{ is encrypted}] \\
&= \frac{1}{2} \cdot \left(\frac{2}{3} + \frac{1}{3} \cdot p\right) + \frac{1}{2} \cdot \left(1 - \left(\frac{1}{3} \cdot \frac{1}{26} + \frac{1}{3} \cdot p\right)\right) \approx 0.827.
\end{aligned}$$

2.9 In this exercise, we look at different conditions under which the shift, mono-alphabetic substitution, and Vigenère ciphers are perfectly secret:

(a) Prove that if only a single character is encrypted, then the shift cipher is perfectly secret.

(b) What is the largest message space $\mathcal{M}$ for which the mono-alphabetic substitution cipher provides perfect secrecy?

(c) Prove that the Vigenère cipher using (fixed) period t is perfectly secret when used to encrypt messages of length t.

Reconcile this with the attacks shown in the previous chapter.

Solution:

(a) This can be proved directly (as in the case of the one-time pad) or using Shannon's theorem.

(b) Let $\mathcal{M}$ be the set of all permutations of the alphabet (i.e., all strings of length 26 with no repeated letter). One can prove directly that the mono-alphabetic substitution cipher is perfectly secret for this message space; we prove it using Shannon's theorem. Briefly, we have $|\mathcal{M}| = |\mathcal{K}| = |\mathcal{C}| = 26!$, where the size of $\mathcal{C}$ follows from the fact that the composition of two permutations is a permutation (and furthermore all permutations can be obtained in this way). This latter fact also proves the second requirement of Shannon's theorem: for every two permutations (namely, plaintext and ciphertext) there exists only one permutation π that maps the plaintext to the ciphertext. This is the largest possible plaintext space, because by Theorem 2.7 we must have $|\mathcal{M}| \leq |\mathcal{K}|$ for *any* perfectly secret encryption scheme.

(c) This can be proved directly (as in the case of the one-time pad) or using Shannon's theorem.

The attacks in Chapter 1 rely on longer plaintexts being encrypted.

2.10 Prove that a scheme satisfying Definition 2.5 must have $|\mathcal{K}| \geq |\mathcal{M}|$ without using Lemma 2.6. Specifically, let Π be an arbitrary encryption scheme with $|\mathcal{K}| < |\mathcal{M}|$. Show an $\mathcal{A}$ for which $\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1\right] > \frac{1}{2}$.

Solution: Consider the following $\mathcal{A}$: output uniform $m_0, m_1 \in \mathcal{M}$. Upon receiving ciphertext c, check (by exhaustive search) whether there exists a key k such that $\mathsf{Dec}_k(c) = m_0$. If so, output 0; else output 1.

When m_0 is encrypted then $\mathcal{A}$ always outputs 0. On the other hand, when m_1 is encrypted then there are at most $|\mathcal{K}|$ possible messages that c can decrypt to; since m_0 is uniform and independent of m_1 the probability that m_0 is equal to one of those messages, and so the probability that $\mathcal{A}$ outputs 0 in this case, is at most $|\mathcal{K}|/|\mathcal{M}| < 1$. We conclude that

$$\begin{aligned}\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1\right] &= \frac{1}{2} \cdot \Pr[\mathcal{A}(\mathsf{Enc}_K(m_0)) = 0] + \frac{1}{2} \cdot \Pr[\mathcal{A}(\mathsf{Enc}_K(m_1)) = 1] \\ &= \frac{1}{2} + \frac{1}{2} \cdot (1 - \Pr[\mathcal{A}(\mathsf{Enc}_K(m_1)) = 0]) \;>\; \frac{1}{2}.\end{aligned}$$

2.11 Assume we require only that an encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M}$ satisfy the following: For all $m \in \mathcal{M}$, we have $\Pr[\mathsf{Dec}_K(\mathsf{Enc}_K(m)) = m] \geq 2^{-t}$. (This probability is taken over choice of the key as well as any randomness used during encryption.) Show that perfect secrecy can be achieved with $|\mathcal{K}| < |\mathcal{M}|$ when $t \geq 1$. Prove a lower bound on the size of $\mathcal{K}$ in terms of t.

Solution: Let $\mathcal{K} = \{0,1\}^\ell$ and $\mathcal{M} = \{0,1\}^{\ell+t}$. The key-generation algorithm chooses a uniform string from $\mathcal{K}$. To encrypt a message $m \in \mathcal{M}$ using key k, let m' denote the first ℓ bits of m and output $c := m' \oplus k$ (both m' and k have length ℓ). To decrypt a ciphertext c using key k, choose a random string $r \leftarrow \{0,1\}^t$ and output $m := (c \oplus k) \| r$. Note that $\Pr[\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m] = 2^{-t}$ because decryption is correct if and only if the random string r chosen during decryption happens to equal the last t bits of m (and this occurs with probability 2^{-t}). Perfect secrecy of this scheme follows from the proof of the one-time pad (indeed, this is exactly a one-time pad on the first ℓ bits of the message).

We now prove the following lower bound:

THEOREM *Let* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a perfectly secret encryption scheme over message space* $\mathcal{M}$ *with* $\Pr[\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m] \geq 2^{-t}$. *Then* $|\mathcal{K}| \geq |\mathcal{M}| \cdot 2^{-t}$.

PROOF Let $\Pr^*$ denote the maximum probability with which an unbounded algorithm can cause the stated event to occur. Consider an experiment in which a message m is chosen uniformly from $\mathcal{M}$ and we are interested in the maximum probability of correctly guessing m. Clearly, $\Pr^*[\text{guess } m] = |\mathcal{M}|^{-1}$. Now consider an extension of this experiment where a key k is generated using Gen and a ciphertext $c \leftarrow \mathsf{Enc}_k(m)$ is computed. Perfect secrecy implies that

$$|\mathcal{M}|^{-1} = \Pr^*[\text{guess } m] = \Pr^*[\text{guess } m \text{ given } c].$$

Continuing, we have

$$\begin{aligned}\Pr^*[\text{guess } m \text{ given } c] &\geq \Pr^*[\text{guess } m \text{ and } k \text{ given } c] \\ &= \Pr^*[\text{guess } k \text{ given } c] \cdot \Pr^*[\text{guess } m \text{ given } k \text{ and } c] \\ &\geq |\mathcal{K}|^{-1} \cdot \Pr^*[\text{guess } m \text{ given } k \text{ and } c] \\ &\geq |\mathcal{K}|^{-1} \cdot 2^{-t},\end{aligned}$$

by the correctness guarantee, and using the fact that m and k are independent. We conclude that $|\mathcal{K}| \geq |\mathcal{M}| \cdot 2^{-t}$. ■

2.12 Let $\varepsilon \geq 0$ be a constant. Say an encryption scheme is *ε-perfectly secret* if for every adversary $\mathcal{A}$ it holds that

$$\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1\right] \leq \frac{1}{2} + \varepsilon\,.$$

(Compare to Definition 2.5.) Show that ε-perfect secrecy can be achieved with $|\mathcal{K}| < |\mathcal{M}|$ when $\varepsilon > 0$. Prove a lower bound on the size of $\mathcal{K}$ in terms of ε.

Solution: (See `http://eprint.iacr.org/2012/053` for further details.) We first show that ε-perfect secrecy can be achieved in this case. Let the message space be $\mathcal{M} = \{0,1\}^\ell$ for some ℓ, and let $\mathcal{K} \subset \{0,1\}^\ell$ be an arbitrary set of size $(1-\varepsilon)\cdot 2^\ell$. One can check that for any $\mathcal{A}$ we have $\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1\right] \leq \frac{1}{2} + \varepsilon$. One can show that this scheme is optimal, in the sense that any scheme that is ε-perfectly secret must have $|\mathcal{K}| \geq (1-\varepsilon)\cdot|\mathcal{M}|$.

2.13 In this problem we consider definitions of perfect secrecy for the encryption of *two* messages (using the same key). Here we consider distributions over *pairs* of messages from the message space $\mathcal{M}$; we let M_1, M_2 be random variables denoting the first and second message, respectively. (We stress that these random variables are not assumed to be independent.) We generate a (single) key k, sample a pair of messages (m_1, m_2) according to the given distribution, and then compute ciphertexts $c_1 \leftarrow \mathsf{Enc}_k(m_1)$ and $c_2 \leftarrow \mathsf{Enc}_k(m_2)$; this induces a distribution over pairs of ciphertexts and we let C_1, C_2 be the corresponding random variables.

(a) Say encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is *perfectly secret for two messages* if for all distributions over $\mathcal{M} \times \mathcal{M}$, all $m_1, m_2 \in \mathcal{M}$, and all ciphertexts $c_1, c_2 \in \mathcal{C}$ with $\Pr[C_1 = c_1 \wedge C_2 = c_2] > 0$:

$$\Pr[M_1 = m_1 \wedge M_2 = m_2 \mid C_1 = c_1 \wedge C_2 = c_2] = \Pr[M_1 = m_1 \wedge M_2 = m_2].$$

Prove that *no* encryption scheme can satisfy this definition.

(b) Say encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is *perfectly secret for two distinct messages* if for all distributions over $\mathcal{M} \times \mathcal{M}$ where the first and second messages are guaranteed to be different (i.e., distributions over pairs of *distinct* messages), all $m_1, m_2 \in \mathcal{M}$, and all $c_1, c_2 \in \mathcal{C}$ with $\Pr[C_1 = c_1 \wedge C_2 = c_2] > 0$:

$$\Pr[M_1 = m_1 \wedge M_2 = m_2 \mid C_1 = c_1 \wedge C_2 = c_2] = \Pr[M_1 = m_1 \wedge M_2 = m_2].$$

Show an encryption scheme that provably satisfies this definition.

Solution:

(a) The definition requires the equation to hold for all pairs of plaintexts m, m' and ciphertexts c, c', *even when* $c = c'$ *but* $m \neq m'$. We show that this is impossible. Take the uniform distribution over $\mathcal{M}$ and any c such that $\Pr[C = c \wedge C' = c] > 0$. Let $m, m' \in \mathcal{M}$ be distinct. For any scheme with no decryption error, and any key k, we must have

$$\Pr[C = c \wedge C' = c \mid M = m \wedge M' = m' \wedge K = k] = 0.$$

(If not, then decryption of the ciphertext c using the key k gives an error some of the time.) The above implies

$$\Pr[C = c \wedge C' = c \mid M = m \wedge M' = m'] = 0.$$

So $\Pr[M = m \wedge M' = m' \mid C = c \wedge C' = c] = 0$ but on the other hand $\Pr[M = m \wedge M' = m'] > 0$. This holds for any encryption scheme, and so no scheme can satisfy the given definition.

(b) Let $\mathcal{K}$ be the set of all permutations on $\mathcal{M}$, and let Gen choose a random permutation from $\mathcal{K}$. (Note that Gen is not necessarily efficient; nevertheless, this is allowed in this exercise.) Encryption is carried out by applying the permutation specified by the key to the plaintext, and decryption is carried out by applying the inverse of the permutation to the ciphertext. This scheme satisfies the definition; details omitted.

Note: An efficient scheme meeting this definition can be constructed using *pairwise-independent permutations* (a topic not covered in the book).

Chapter 3

Private-Key Encryption – Solutions

3.1 Prove Proposition 3.6.

Solution: Let negl_1 and negl_2 be negligible functions. We prove that:

(a) The function negl_3 defined by $\mathsf{negl}_3(n) = \mathsf{negl}_1(n) + \mathsf{negl}_2(n)$ is negligible.

(b) For any (positive) polynomial p, the function negl_4 defined by $\mathsf{negl}_4(n) = p(n) \cdot \mathsf{negl}_1(n)$ is negligible.

We begin with item (1). We show that for every polynomial $q(\cdot)$ there exists an integer N such that for every $n > N$ it holds that $\mathsf{negl}_3(n) < \frac{1}{q(n)}$. Fix $q(\cdot)$. Since negl_1 and negl_2 are negligible, we know that there exist integers N_1 and N_2 respectively such that for every $n > N_1$, $\mathsf{negl}_1(n) < \frac{1}{2q(n)}$ and for every $n > N_2$, $\mathsf{negl}_2(n) < \frac{1}{2q(n)}$ (note that $2q(n)$ is also a polynomial so the above holds). Take $N = \mathsf{max}\{N_1, N_2\}$. It follows that for every $n > N$, $\mathsf{negl}_3(n) = \mathsf{negl}_1(n) + \mathsf{negl}_2(n) < \frac{1}{2q(n)} + \frac{1}{2q(n)} = \frac{1}{q(n)}$. We conclude that for every polynomial $q(\cdot)$ there exists a integer N such that for every $n > N$, $\mathsf{negl}_3(n) < \frac{1}{q(n)}$.

We now prove item (2); the proof is very similar so we are brief. Fix $q(\cdot)$; let N be an integer such that for all $n > N$ it holds that $\mathsf{negl}_1(n) < \frac{1}{p(n) \cdot q(n)}$. This implies that for every $n > N$ it holds that $\mathsf{negl}_4(n) = p(n) \cdot \mathsf{negl}_1(n) < \frac{p(n)}{p(n) \cdot q(n)} = \frac{1}{q(n)}$, as required.

3.2 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 $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$.

Solution: The encryption algorithm Enc runs in polynomial time. Let $p(\cdot)$ be the polynomial (in the security parameter n) that bounds the running time of Enc when encrypting a single bit. (Thus, Enc runs for at most $p(n)$ steps to encrypt a single bit when the security parameter is 1^n.) In particular, the length of the ciphertext generated by Enc when encrypting a single bit is at most $p(n)$ because an algorithm cannot output a string that is longer than its running time. Next, note that the ciphertext generated by encrypting a random string of length $p(n) + n$ has length greater than $p(n)$ with all but negligible probability. (Here we assume, as usual, that decryption is always correct.)

Given the above, we can take $\mathcal{A}$ to be an adversary that, on input 1^n, outputs a pair of plaintexts m_0, m_1 in experiment $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$ where m_0 is a single bit and m_1 is a random string of length $p(n)+n$. If $b=0$ then $|c| \leq p(n)$, while if $b=1$ then $|c| > p(n)$ with all but negligible probability, where c is the challenge ciphertext. Thus, $\mathcal{A}$ succeeds with all but negligible probability by outputting $b'=0$ if and only if $|c| \leq p(n)$. The above holds for any encryption scheme and thus such a definition can never be satisfied.

3.3 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 $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$.

Solution: To be clear, we are considering a modification of experiment $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$ where the adversary $\mathcal{A}$ is not required to output m_0 and m_1 of the same length, but instead it is only required that m_0 and m_1 each have length at most $\ell(n)$.

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a scheme that is secure with respect to the original Definition 3.8 (for messages of equal length). Construct a scheme $\Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ as follows:

(a) Gen' is identical to Gen.

(b) Upon input a plaintext message m of length at most $\ell = \ell(n)$ (where n is the length of the key), Enc' first sets $m' := 0^{\ell - |m|}1 \| m$ and then encrypts m' using Enc. Note that m' is always *exactly* $\ell(n)+1$ bits long.

(c) Dec' applies Dec to the ciphertext, and parses the result as $0^t 1 \| m$ for $t \geq 0$. It outputs m.

It is clear that if Π satisfies Definition 3.8 then Π' satisfies the modified definition. A complete answer to this exercise requires a proof showing that the existence of an adversary breaking Π' with respect to the modified definition implies the existence of an adversary breaking Π with respect to Definition 3.8. We describe the reduction informally. Given an adversary $\mathcal{A}'$ who breaks Π', we construct an adversary $\mathcal{A}$ who takes the pair of plaintexts m_0, m_1 output by $\mathcal{A}'$ and pads them in the same way as Enc' would. Then, it outputs the padded messages to be encrypted. Observe that $\mathcal{A}$ outputs equal-length messages, as required. Furthermore, if $\mathcal{A}'$ can correctly guess b with probability non-negligibly greater than $1/2$, then $\mathcal{A}$ guesses correctly with the same probability.

3.4 Prove the equivalence of Definition 3.8 and Definition 3.9.

Solution: Let Π be an encryption scheme. We have:

$$\begin{aligned}
\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1] &= \Pr_{b\leftarrow\{0,1\}}[\mathsf{output}(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,b))=b] \\
&= \frac{1}{2}\cdot\Pr[\mathsf{output}(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,0))=0] \\
&\quad + \frac{1}{2}\cdot\Pr[\mathsf{output}(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,1))=1] \\
&= \frac{1}{2}\cdot\big(1-\Pr[\mathsf{output}(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,0))=1]\big) \\
&\quad + \frac{1}{2}\cdot\Pr[\mathsf{output}(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,1))=1] \\
&= \frac{1}{2}+\frac{1}{2}\cdot\big(\Pr[\mathsf{output}(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,1))=1] \\
&\qquad - \Pr[\mathsf{output}(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,0))=1]\big). \qquad (3.1)
\end{aligned}$$

If Π satisfies Definition 3.9, then there exists a negligible function negl for which

$$\begin{aligned}
\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1] &= \frac{1}{2}+\frac{1}{2}\cdot\big(\Pr[\mathsf{output}(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,1))=1] \\
&\qquad - \Pr[\mathsf{output}(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,0))=1]\big) \\
&\le \frac{1}{2}+\frac{1}{2}\cdot\big|\Pr[\mathsf{output}(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,1))=1] \\
&\qquad - \Pr[\mathsf{output}(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,0))=1]\big| \\
&\le \frac{1}{2}+\mathsf{negl}(n);
\end{aligned}$$

thus, Π satisfies Definition 3.8 also.

For the other direction, assume Π satisfies Definition 3.8. By Equation (3.1) we see that there exists a negligible function negl such that

$$\begin{aligned}
\mathsf{negl}(n) &\ge \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1]-\frac{1}{2} \\
&= \frac{1}{2}\cdot\big(\Pr[\mathsf{output}(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,1))=1] \\
&\qquad - \Pr[\mathsf{output}(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,0))=1]\big). \qquad (3.2)
\end{aligned}$$

Let $\hat{\mathcal{A}}$ denote an adversary that runs identically to $\mathcal{A}$ except that it outputs the complement of whatever $\mathcal{A}$ outputs. Then

$$\begin{aligned}
\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\hat{\mathcal{A}},\Pi}(n)=1] &= 1-\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1] \\
&= \frac{1}{2}+\frac{1}{2}\cdot\big(\Pr[\mathsf{output}(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,0))=1] \\
&\qquad - \Pr[\mathsf{output}(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,1))=1]\big)
\end{aligned}$$

and, arguing as above, there exists a negligible function negl' such that

$$\begin{aligned}\mathsf{negl}'(n) &\geq \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\hat{\mathcal{A}},\Pi}(n) = 1] - \frac{1}{2} \\ &= \frac{1}{2} \cdot \big(\Pr[\mathsf{output}(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,0)) = 1] \\ &\qquad - \Pr[\mathsf{output}(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,1)) = 1]\big)\,. \qquad (3.3)\end{aligned}$$

Combining Equations (3.2) and (3.3), we see that Π satisfies Definition 3.9 as well.

3.5 Let $|G(s)| = \ell(|s|)$ for some ℓ. Consider the following experiment:

The PRG indistinguishability experiment $\mathsf{PRG}_{\mathcal{A},G}(n)$:

(a) A uniform bit $b \in \{0,1\}$ is chosen. If $b = 0$ then choose a uniform $r \in \{0,1\}^{\ell(n)}$; if $b = 1$ then choose a uniform $s \in \{0,1\}^n$ and set $r := G(s)$.

(b) The adversary $\mathcal{A}$ is given r, and outputs a bit b'.

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

Provide a definition of a pseudorandom generator based on this experiment, and prove that your definition is equivalent to Definition 3.14. (That is, show that G satisfies your definition if and only if it satisfies Definition 3.14.)

Solution: The required definition is that G is a pseudorandom generator if for any PPT algorithm $\mathcal{A}$ we have

$$\Pr[\mathsf{PRG}_{\mathcal{A},G}(n) = 1] \leq 1/2 + \mathsf{negl}(n).$$

To see that these definitions are equivalent, note that

$$\begin{aligned}&\Pr[\mathsf{PRG}_{\mathcal{A},G}(n) = 1] \\ &= \frac{1}{2} \cdot \Pr_{r \leftarrow \{0,1\}^{\ell(n)}}[\mathcal{A}(r) = 0] + \frac{1}{2} \cdot \Pr_{s \leftarrow \{0,1\}^n}[\mathcal{A}(G(s)) = 1] \\ &= \frac{1}{2} + \frac{1}{2} \cdot \big(\Pr_{s \leftarrow \{0,1\}^n}[\mathcal{A}(G(s)) = 1] - \Pr_{r \leftarrow \{0,1\}^{\ell(n)}}[\mathcal{A}(r) = 1]\big)\,.\end{aligned}$$

Thus, if there is an efficient $\mathcal{A}$ for which the new definition fails to hold, then Definition 3.14 fails to hold for this $\mathcal{A}$ as well. And if there is an efficient attacker $\mathcal{A}$ for which Definition 3.14 does not hold, we obtain an efficient attacker for which the new definition does not hold either (possibly by flipping the output of $\mathcal{A}$).

3.6 Let G be a pseudorandom generator with expansion factor $\ell(n) > 2n$. In each of the following cases, say whether G' is necessarily a pseudorandom generator. If yes, give a proof; if not, show a counterexample.

(a) Define $G'(s) \stackrel{\text{def}}{=} G(s_1 \cdots s_{\lceil n/2 \rceil})$, where $s = s_1 \cdots s_n$.

(b) Define $G'(s) \stackrel{\text{def}}{=} G\left(0^{|s|} \| s\right)$.

(c) Define $G'(s) \stackrel{\text{def}}{=} G(s) \,\|\, G(s+1)$.

Solution:

(a) Yes! First, since $\ell(n) > 2n$ we have that $|G'(s)| > \frac{1}{2} \cdot 2n = n$ as required for any pseudorandom generator. Let ℓ' be the expansion factor of G'; i.e., ℓ' is such that $|G'(s)| = \ell'(|s|)$. Fix a probabilistic polynomial-time algorithm D and set

$$\varepsilon(n) \stackrel{\text{def}}{=} \left| \Pr_{r \leftarrow \{0,1\}^{\ell'(n)}}[D(r) = 1] - \Pr_{s \leftarrow \{0,1\}^n}[D(G'(s)) = 1] \right|.$$

By definition of G', we have that

$$\Pr_{s \leftarrow \{0,1\}^n}[D(G'(s)) = 1] = \Pr_{s \leftarrow \{0,1\}^{n/2}}[D(G(s)) = 1],$$

and thus

$$\begin{aligned} \left| \Pr_{r \leftarrow \{0,1\}^{\ell'(n)}}[D(r) = 1] - \Pr_{s \leftarrow \{0,1\}^{n/2}}[D(G(s)) = 1] \right| &= \varepsilon(n) \\ &= \varepsilon'(n/2), \end{aligned}$$

where $\varepsilon'(n) \stackrel{\text{def}}{=} \varepsilon(2n)$ (note the change in the length of s). Since ε' must be negligible, we conclude that ε is negligible as well.

(b) G' is not necessarily a pseudorandom generator. To see this, let $H : \{0,1\}^n \to \{0,1\}^{4n}$ be a pseudorandom generator, and define $G(s) = H(s_1 \cdots s_{n/2})$. As in the previous part, G is a pseudorandom generator. But then

$$G'(s) = G(0^{|s|} \| s) = H(0^{|s|}),$$

and clearly G' is not a pseudorandom generator.

Fundamentally, the problem here is that G' runs G on an input that is not uniformly distributed.

(c) G' is not necessarily a pseudorandom generator. To see this, let $H : \{0,1\}^n \to \{0,1\}^{2n}$ be a pseudorandom generator and define $G(s) = H(s_1 \cdots s_{n-1})$. As in the previous parts, G is a pseudorandom generator. But then if the last bit of s is 0 we have

$$G'(s) = G(s) \| G(s+1) = H(s_1 \cdots s_{n-1}) \| H(s_1 \cdots s_{n-1})$$

(because then s and $s+1$ differ only in their final bit), and so with probability $1/2$ the two halves of the output of G' are the same. This is clearly not a pseudorandom generator.

Fundamentally, the problem here is that G' runs G on two correlated (rather than independent) inputs.

3.7 Prove the converse of Theorem 3.18. Namely, show that if G is not a pseudorandom generator then Construction 3.17 does not have indistinguishable encryptions in the presence of an eavesdropper.

Solution: Let D be an efficient algorithm such that

$$\varepsilon(n) \stackrel{\text{def}}{=} \Pr_{r\leftarrow\{0,1\}^{\ell(n)}}[D(r)=1] - \Pr_{s\leftarrow\{0,1\}^n}[D(G(s))=1]$$

is not negligible. Construct the attacker $\mathcal{A}$ that, on input 1^n, outputs (m_0, m_1) where $m_0 = 0^{\ell(n)}$ and m_1 is uniform. Upon receiving ciphertext c, attacker $\mathcal{A}$ runs $D(c)$ and outputs the result. Note that when m_0 is encrypted, the ciphertext is equal to $G(s)$ for a uniform s, while when m_1 is encrypted, the ciphertext is a uniform ℓ-bit string (since m_1) is uniform. Thus,

$$\begin{aligned}
&\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1] \\
&= \frac{1}{2}\cdot\left(\Pr_{s\leftarrow\{0,1\}^n}[D(G(s))=0] + \Pr_{r\leftarrow\{0,1\}^{\ell(n)}}[D(r)=1]\right) \\
&= \frac{1}{2} + \frac{1}{2}\cdot\left(\Pr_{r\leftarrow\{0,1\}^{\ell(n)}}[D(r)=1] - \Pr_{s\leftarrow\{0,1\}^n}[D(G(s))=1]\right),
\end{aligned}$$

which (by assumption) is not negligibly greater than 1/2.

3.8 (a) Define a notion of indistinguishability for the encryption of multiple *distinct* messages, in which a scheme need not hide whether the same message is encrypted twice.

(b) Show that Construction 3.17 does not satisfy your definition.

(c) Give a construction of a *deterministic* (stateless) encryption scheme that satisfies your definition.

Solution:

(a) The definition is a straightforward modification of Definition 3.19, where experiment $\mathsf{PrivK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n)$ is modified so that all the messages in $\vec{M}_0$ are distinct, and similarly for $\vec{M}_1$ (but a message may appear in both $\vec{M}_0$ and $\vec{M}_1$).

(b) It is easy to see that Construction 3.17 does not satisfy the definition since, for example, when two messages that are identical except for their last bit are encrypted, the result is two ciphertexts that are identical except for their last bit.

(c) Let F be a block cipher. Then setting $\mathsf{Enc}_k(m) = F_k(m)$ satisfies the definition even though it is deterministic.

3.9 Prove *unconditionally* the existence of a pseudorandom function $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ with $\ell_{key}(n) = n$ and $\ell_{in}(n) = O(\log n)$.

Solution: Define keyed function $F : \{0,1\}^n \times \{0,1\}^{\log n} \to \{0,1\}$ as follows: $F_k(i)$ outputs the ith bit of k, where the input i is interpreted

as an integer in the range $\{0, \ldots, n-1\}$ and the bits of k are numbered starting at 0. Note that F is exactly implementing a lookup table based on the key k, and so F_k for uniform k is exactly a random function mapping $\log n$-bit inputs to 1-bit outputs. I.e., F is a random function, which is stronger than being pseudorandom.

3.10 Let F be a length-preserving pseudorandom function. For the following constructions of a keyed function $F' : \{0,1\}^n \times \{0,1\}^{n-1} \to \{0,1\}^{2n}$, state whether F' is a pseudorandom function. If yes, prove it; if not, show an attack.

(a) $F'_k(x) \stackrel{\text{def}}{=} F_k(0\|x) \,\|\, F_k(1\|x)$.

(b) $F'_k(x) \stackrel{\text{def}}{=} F_k(0\|x) \,\|\, F_k(x\|1)$.

Solution:

(a) F' is a pseudorandom function. A formal proof is omitted, but relies on the observation that distinct queries to F'_k result in distinct queries to F_k.

(b) F' is not a pseudorandom function. To see this, consider querying on the two inputs 0^{n-1} and $0^{n-2}1$. We have

$$F'_k(0^{n-1}) = F_k(0^n)\|F_k(0^{n-1}1)$$

and

$$F'_k(0^{n-2}1) = F_k(0^{n-1}1)\|F_k(0^{n-2}1^2);$$

note that the second half of $F'_k(0^{n-1})$ is equal to the first half of $F'_k(0^{n-2}1)$.

Formally, define the following attacker A given 1^n and access to some function g:

$\underline{A^g(1^n)}$:

- Query $y_0 = g(0^{n-1})$ and $y_1 = g(0^{n-2}1)$.
- Output 1 if and only if the second half of y_0 is equal to the first half of y_1.

As shown above, we have $\Pr_{k\leftarrow\{0,1\}^n}[A^{F'_k(\cdot)}(1^n) = 1] = 1$. But when g is a random function then y_0 and y_1 are independent, uniform strings of length $2n$, and so the probability that the second half of y_0 is equal to the first half of y_1 is exactly 2^{-n}. Thus, $\Pr_{f\leftarrow\mathsf{Func}}[A^{f(\cdot)}(1^n) = 1] = 2^{-n}$, and the difference

$$\left|\Pr_{k\leftarrow\{0,1\}^n}[A^{F'_k(\cdot)}(1^n) = 1] - \Pr_{f\leftarrow\mathsf{Func}}[A^{f(\cdot)}(1^n) = 1]\right|$$

is not negligible.

3.11 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 3.19), but is not CPA-secure (i.e., is not secure with respect to Definition 3.22).

Solution: Let F be a pseudorandom function, and define scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ as follows:

(a) Gen: On input 1^n, choose $k, s \leftarrow \{0,1\}^n$.

(b) Enc: On input a key $\langle k, s\rangle$ and a message $m \in \{0,1\}^n$, do:

i. If $m = s$ output the ciphertext $\langle 0, k, s, s\rangle$.

ii. If $m \neq s$ then choose random $r \leftarrow \{0,1\}^n$ and output the ciphertext $\langle 1, s, r, F_k(r) \oplus m\rangle$.

(c) Dec: On input a key $\langle k, s\rangle$ and ciphertext $c = \langle b, c_1, c_2, c_3\rangle$ do:

i. If $b = 0$ output s.

ii. If $b = 1$ output $m := F_k(c_2) \oplus c_3$.

One can check that decryption always succeeds.

It is clear that Π is not secure under a chosen-plaintext attack. This is because an adversary $\mathcal{A}$ can query its encryption oracle with input 0^n and receive (except with negligible probability) the ciphertext $c = \langle 1, s, r, F_k(r)\rangle$, where s is the second component of the secret key. Then, $\mathcal{A}$ can query its encryption oracle again with s and receive back a ciphertext containing k. Given k, of course, $\mathcal{A}$ can distinguish encryptions easily. Despite the above, we argue that Π is still secure with respect to Definition 3.18. Intuitively, this is due to the fact that unless one of the plaintexts output by the adversary is equal to s—which occurs with only negligible probability since s is uniform—the scheme is essentially equivalent to Construction 3.30. We omit the formal proof, which is tedious but straightforward.

3.12 Let F be a keyed function and consider the following experiment:

The PRF indistinguishability experiment $\mathsf{PRF}_{\mathcal{A},F}(n)$:

(a) A uniform bit $b \in \{0,1\}$ is chosen. If $b = 1$ then choose uniform $k \in \{0,1\}^n$.

(b) $\mathcal{A}$ is given 1^n for input. If $b = 0$ then $\mathcal{A}$ is given access to a uniform function $f \in \mathsf{Func}_n$. If $b = 1$ then $\mathcal{A}$ is instead given access to $F_k(\cdot)$.

(c) $\mathcal{A}$ outputs a bit b'.

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

Define pseudorandom functions using this experiment, and prove that your definition is equivalent to Definition 3.25.

Solution: The solution is similar to that of Exercise 3.5, and is omitted.

3.13 Consider the following keyed function F: For security parameter n, the key is an $n \times n$ boolean matrix A and an n-bit boolean vector b. Define $F_{A,b} : \{0,1\}^n \to \{0,1\}^n$ by $F_{A,b}(x) \stackrel{\text{def}}{=} Ax + b$, where all operations are done modulo 2. Show that F is not a pseudorandom function.

Solution: Querying $F_{A,b}(0^n)$ yields b. Let e_i denote the n-bit string with a 1 in position i (and 0s elsewhere). Querying $F_{A,b}(e_i)$ yields $a_i + b$, where a_i is the ith column of A. This can be easily extended to a full-fledged attack.

3.14 Prove that if F is a length-preserving pseudorandom function, then $G(s) \stackrel{\text{def}}{=} F_s(1)\|F_s(2)\|\cdots\|F_s(\ell)$ is a pseudorandom generator with expansion factor $\ell \cdot n$.

Solution: Fix an efficient distinguisher D, and let

$$\varepsilon(n) \stackrel{\text{def}}{=} \left|\Pr_{r\leftarrow\{0,1\}^{\ell n}}[D(r) = 1] - \Pr_{s\leftarrow\{0,1\}^n}[D(G(s)) = 1]\right|.$$

Construct $\mathcal{A}(1^n)$ as follows: given access to a function g, let $y_i = g(i)$ for $i = 1, \ldots, \ell$. Output $D(y_1\| \cdots \|y_\ell)$. Clearly

$$\left|\Pr[\mathcal{A}^{f(\cdot)}(1^n) = 1] - \Pr[\mathcal{A}^{F_k(\cdot)}(1^n) = 1]\right| = \varepsilon(n),$$

and so $\varepsilon(n)$ must be negligible.

3.15 Define a notion of perfect secrecy under a chosen-plaintext attack by adapting Definition 3.22. Show that the definition cannot be achieved.

Solution: Here we would simply require that for all $\mathcal{A}$ (i.e., even unbounded), it holds that $\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1] = 1/2$. This definition cannot be achieved. To see this, fix an arbitrary scheme Π and let $\mathcal{A}$ be the following attacker: request an encryption of 0^n, obtaining in return ciphertext c_0. Then output the messages $(m_0 = 0^n, m_1 = 1^n)$, and receive in return a ciphertext c. If $c = c_0$ output 0; otherwise, output 1. Note that when m_0 is encrypted, there is a nonzero probability that $c = c_0$, whereas when m_1 is encrypted we must have $c \neq c_0$. Thus,

$$\begin{aligned}\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1] &= \frac{1}{2} \cdot (\Pr[\mathsf{Enc}_k(m_0) = c_0] + \Pr[\mathsf{Enc}_k(m_1) \neq c_0]) \\ &> 1/2.\end{aligned}$$

3.16 Prove Proposition 3.27.

Solution: Intuitively, as long as a distinguisher does not find a collision (namely, a pair x and y for which $f(x) = f(y)$) the distribution over the

values of a random function and a random permutation are identical. Furthermore, by the analysis of the "birthday problem" (Appendix A.4) we know that a collision is found with only negligible probability as long as only a polynomial number of inputs are queried to f. Thus, a random function and random permutation are computationally indistinguishable. By reduction, the same is true also for a pseudorandom function and pseudorandom permutation.

Formally, let f be a uniform length-preserving function and let g denote a uniform length-preserving permutation. Let D be any algorithm that queries its oracle only polynomially many times (any polynomial-time algorithm can only query its oracle only polynomially many times). We claim that there is a negligible function negl such that

$$\left|\Pr[D^{f(\cdot)}(1^n)=1]-\Pr[D^{g(\cdot)}(1^n)=1]\right|<\mathsf{negl}(n) \tag{3.4}$$

In order to see this, let $q(n)$ be the polynomial bounding the number of queries that D makes to its oracle and assume without loss of generality that D always makes exactly q (distinct) queries. Let coll denote the event that D queries its oracle with two distinct values x and y for which its oracle returns the same reply. Clearly, if D is given access to oracle g then coll occurs with probability zero (because g is a permutation). Furthermore, by Lemma A.15, when D is given access to oracle f, the probability that coll occurs is at most $q^2/2^n$. (Each query to f on a "fresh" input yields an independent uniform output.) Finally, we observe that *conditioned* on coll not occurring, the view of D when given oracle f is identical to its view when given oracle g. This is because all oracle responses are independently and uniformly distributed subject only to the constraint that all responses are distinct. Therefore:

$$\begin{aligned}
&\left|\Pr[D^{f(\cdot)}(1^n)=1]-\Pr[D^{g(\cdot)}(1^n)=1]\right| \\
&\quad=\Big|\Pr[D^{f(\cdot)}(1^n)=1\mid\mathsf{coll}]\cdot\Pr[\mathsf{coll}] \\
&\qquad+\Pr[D^{f(\cdot)}(1^n)=1\mid\overline{\mathsf{coll}}]\cdot\Pr[\overline{\mathsf{coll}}]-\Pr[D^{g(\cdot)}(1^n)=1]\Big| \\
&\quad=\Big|\Pr[D^{f(\cdot)}(1^n)=1\mid\mathsf{coll}]\cdot\Pr[\mathsf{coll}] \\
&\qquad+\Pr[D^{g(\cdot)}(1^n)=1]\cdot\Pr[\overline{\mathsf{coll}}]-\Pr[D^{g(\cdot)}(1^n)=1]\Big| \\
&\quad=\Big|\Pr[D^{f(\cdot)}(1^n)=1\mid\mathsf{coll}]\cdot\Pr[\mathsf{coll}] \\
&\qquad+\Pr[D^{g(\cdot)}(1^n)=1]\cdot(1-\Pr[\mathsf{coll}])-\Pr[D^{g(\cdot)}(1^n)=1]\Big| \\
&\quad=\Big|\Pr[D^{f(\cdot)}(1^n)=1\mid\mathsf{coll}]\cdot\Pr[\mathsf{coll}]-\Pr[D^{g(\cdot)}(1^n)=1]\cdot\Pr[\mathsf{coll}]\Big| \\
&\quad\leq\Pr[\mathsf{coll}]\ \leq\ \frac{q^2}{2^n},
\end{aligned}$$

implying Equation (3.4). Now, let F be a pseudorandom permutation. By the definition, we have that for every probabilistic polynomial-time D there exists a negligible function such that

$$\left|\Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{g(\cdot)}(1^n) = 1]\right| < \mathsf{negl}(n).$$

Combining this with the above, we have that

$$\left|\Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1]\right| < \mathsf{negl}(n) + \frac{q^2}{2^n}.$$

Since $q^2/2^n$ is negligible, we conclude that any pseudorandom permutation F is also a pseudorandom function.

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

Solution: Let F be a length-preserving pseudorandom permutation. Define F' as follows:

$$F'_k(x) = \begin{cases} 0^n & x = k \\ F_k(k) & x = F_k^{-1}(0^n) \\ F_k(x) & \text{otherwise} \end{cases}.$$

F'_k is a permutation for all k, and can be efficiently inverted. We do not prove that F' is still a pseudorandom permutation, but intuitively this holds since the only difference between F'_k and F_k (when an adversary can query in the "forward" direction only) occurs if the adversary queries F'_k either on the key k or on the point $F_k^{-1}(0^n)$. But the fact that F is pseudorandom means that it is infeasible for an adversary to find either of those two inputs when interacting with $F'_k(\cdot)$.

On the other hand, F' is not strongly pseudorandom: since ${F'}_k^{-1}(0^n) = k$, an adversary can learn the key by querying 0^n in the reverse direction. Formally, consider the attacker $\mathcal{A}$ given access to an oracle g and its inverse g^{-1} that acts as follows: query $g^{-1}(0^n)$ and obtain a value k. Then choose any $x \notin \{k, F_k(k)\}$ and query $g(x)$ to obtain y. Output 1 iff $y = F_k(x)$. It is immediate that

$$\Pr[\mathcal{A}^{{F'}_k(\cdot),{F'}_k^{-1}(\cdot)}(1^n) = 1] = 1.$$

On the other hand, it is easy to see that

$$\Pr[\mathcal{A}^{f(\cdot),f^{-1}(\cdot)}(1^n) = 1] = 2^{-n}.$$

Thus, F' is not strongly pseudorandom.

3.18 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.)

Solution: Decryption is computed by first computing $F_k^{-1}(c)$ and then outputting the $n/2$ least significant bits of the result. We now show that this scheme is CPA-secure. In order to show this, we first consider an encryption scheme $\tilde{\Pi}$ that is identical to the above encryption scheme, except that a truly random permutation is used instead of a pseudorandom one. Let $\mathcal{A}$ be an adversary and let $q(\cdot)$ be a polynomial upper-bounding the running-time of $\mathcal{A}$. We claim that:

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Pi}}(n) = 1] \leq \frac{1}{2} + \frac{q(n)}{2^{n/2}}$$

Let r_c denote the random string used to generate the challenge ciphertext $c = F_k(r\|m)$. There are two cases:

(a) *The value r_c is used by the encryption oracle to answer at least one of $\mathcal{A}$'s queries:* In this case, $\mathcal{A}$ can know which message was encrypted, but the probability of this event occurring is upper-bound by $q(n)/2^{n/2}$ (this is obtained by applying the union bound).

(b) *The value r_c is not used by the encryption oracle to answer any of $\mathcal{A}$'s queries:* In this case, $\mathcal{A}$ learns nothing about the plaintext because the challenge ciphertext is a uniform string (subject to being distinct from all other ciphertexts).

Using a similar argument as in the proof of Theorem 3.31 the above equation follows. The rest of the proof follows as in the proof of Theorem 3.31 by showing that the difference when using a pseudorandom permutation instead is at most negligible.

We omit a proof of CCA-security.

3.19 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 $m \in \{0,1\}^{n+1}$, choose uniform $r \in \{0,1\}^n$ and output the ciphertext $\langle r, G(r) \oplus m\rangle$.

(b) To encrypt $m \in \{0,1\}^n$, output the ciphertext $m \oplus F_k(0^n)$.

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

Solution:

(a) This scheme does not even have indistinguishable encryptions in the presence of an eavesdropper because the ciphertext doesn't depend on the key. An eavesdropper can easily compute m from $c = \langle r, s \rangle$ by computing $m := G(r) \oplus s$.

(b) This scheme has indistinguishable encryptions in the presence of an eavesdropper. To see this, note that $F_k(0^n)$ is pseudorandom and so a proof of this fact follows from the proof of Theorem 3.18. The scheme is *not* CPA-secure because encryption is deterministic.

(c) This scheme is CPA-secure. A proof of this is very similar to the proof of Theorem 3.31 except that Repeat denotes the event that $r-1$, r or $r+1$ is chosen in another ciphertext.

A full would include proofs for the above claims.

3.20 Consider a stateful 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.

Solution: Define $\mathcal{A}$ as follows:

(a) Query the encryption oracle with the message $m = 0^{n-1}\|1$. Receive in return a ciphertext $\langle IV, c \rangle$.

(b) If IV is odd (i.e., has low-order bit 1), then output a random bit.

(c) If IV is even, then output a pair of messages m_0, m_1 where $m_0 = 0^n$ and m_1 is any other message. Receive in return a challenge ciphertext $\langle IV+1, c' \rangle$.

(d) If $c' = c$ output 0; else output 1.

We analyze the success probability of $\mathcal{A}$. If IV is odd, then $\mathcal{A}$ succeeds exactly half the time. For any even IV, though, it holds that $IV + 1 = IV \oplus (0^{n-1}\|1)$. Thus, for any even IV we have

$$c = F_k(IV \oplus m) = F_k(IV \oplus 0^{n-1}1 \oplus m \oplus 0^{n-1}1) = F_k((IV+1) \oplus m_0).$$

So if m_0 is encrypted then $c' = c$ and $\mathcal{A}$ outputs 0, while if m_1 is encrypted then $c' \neq c$ and $\mathcal{A}$ outputs 1. The overall success probability of $\mathcal{A}$ is therefore 3/4, and this modified CBC mode is not CPA-secure.

3.21 What is the effect of a single-bit error in the ciphertext when using the CBC, OFB, and CTR modes of operation?

Solution: Say a message $m_1, m_2, \ldots$ is encrypted to give a ciphertext $c_0, c_1, c_2, \ldots$, and then a single bit is flipped somewhere in the ciphertext. We look at the effect of decrypting the resulting (modified) ciphertext using each of the stated modes to obtain a message $m'_1, m'_2, \ldots$.

CBC mode. Say a bit is flipped in c_i to give modified block c'_i. When decrypting, m_i is computed as $m'_i = F_k^{-1}(c'_i) \oplus c_0$ and $F_k^{-1}(c'_i)$ will, in general, be completely unrelated to $F_k^{-1}(c_i)$. Thus, m'_i will have no relation to m_i. The next message block, m'_{i+1}, is computed as $m'_{i+1} = F_k^{-1}(c_{i+1}) \oplus c'_i$ and so m'_{i+1} will be equal to m_{i+1} but with a single bit flipped. The rest of the message blocks will be unchanged.

OFB mode and CTR mode. In these modes, a bit flip in c_i for $i > 0$ only causes a bit flip in message block m_i. However, a bit flip in c_0 will (in general) result in all the plaintext blocks being recovered incorrectly.

3.22 What is the effect of a dropped ciphertext block (e.g., if the transmitted ciphertext $c_1, c_2, c_3, \ldots$ is received as $c_1, c_3, \ldots$) when using the CBC, OFB, and CTR modes of operation?

Solution: Say a message $m_1, m_2, \ldots$ is encrypted to give a ciphertext $c_0, c_1, c_2, \ldots$, and then a single block is dropped. We look at the effect of decrypting the resulting (modified) ciphertext using each of the stated modes to obtain a message $m'_1, m'_2, \ldots$.

CBC mode. Say block c_i is dropped. The result of decrypting the modified ciphertext is $m_1, \ldots, m_{i-1}, m'_{i+1}, m_{i+2}, \ldots$. I.e., message blocks before position i are recovered correctly, message block i is lost entirely, message block $i+1$ is garbled, and message blocks after position $i+1$ are recovered correctly (though shifted over by one block).

OFB and CTR mode. If c_i is dropped then all plaintext blocks before position i are recovered correctly, but from position i and on the remaining plaintext blocks will be incorrect.

3.23 Say CBC-mode encryption is used with a block cipher having a 256-bit key and 128-bit block length to encrypt a 1024-bit message. What is the length of the resulting ciphertext?

Solution: The message is $8 = 1024/128$ blocks long, so the ciphertext (which includes an IV) is 9 blocks long. Thus, the ciphertext is 1152 bits long.

3.24 Give the details of the proof by reduction for Equation (3.12).

Solution: This is exactly analogous to the relevant step in the proof of Theorem 3.31, so is omitted.

3.25 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.

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

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

Solution:

(a) In order for counter mode to be secure, it is essential that the counter not repeat even though an arbitrary polynomial number of blocks are encrypted. In order to ensure this, the counter must have superlogarithmic length and so we require $\ell_{in}(n) = \omega(\log n)$. In contrast, there is no lower or upper bound on the length of ℓ_{out}.

(b) Consider encryption of a *single* message of length ℓ. Here, for security to hold we only need to make sure that the counter value does not "wrap" when encrypting. When encrypting a message of length ℓ, we will apply F to ℓ/ℓ_{out} counter values. Wrapping occurs if $2^{\ell_{in}} < \ell/\ell_{out}$. So the scheme has indistinguishable encryptions in the presence of an eavesdropper as long as $2^{\ell_{in}} \geq \ell/\ell_{out}$.

3.26 For any function $g : \{0,1\}^n \to \{0,1\}^n$, define $g^{\$}(\cdot)$ to be a *probabilistic* oracle that, on input 1^n, chooses uniform $r \in \{0,1\}^n$ and returns $\langle r, g(r)\rangle$. 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 $k \in \{0,1\}^n$ and $f \in \mathsf{Func}_n$ are chosen uniformly.

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

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

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

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

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

(d) Prove that Construction 3.30 is CPA-secure if F is a weak pseudorandom function.

Solution:

(a) Intuitively, if F cannot be distinguished from a random function even when a distinguisher is allowed to *choose* the evaluation points, then F cannot be distinguished from a random function when F is evaluated on random points. A formal reduction is straightforward.

(b) It is clear that F' is not pseudorandom. Specifically, a distinguisher D can query its oracle with an odd x and then with $x+1$. If its oracle is F' then D will receive back the same value twice. Otherwise, if its oracle is random, it will receive back different values (except with probability 2^{-n}).

On the other hand, F' is weakly pseudorandom. Intuitively, this is because the *only* way to distinguish F' from a random function is to query it on two consecutive points (as above), but if F' is evaluated on random inputs then the probability that any two such points happen to be consecutive is negligible. (We omit the formal proof.)

(c) Counter mode instantiated with a weak pseudorandom function does not necessarily have indistinguishable encryptions in the presence of an eavesdropper. To see this, take F' from part (b). When encrypting a message that is at least 3 blocks long, there will be two consecutive plaintext blocks that are XORed with the same value $F'_k(\mathsf{ctr}) = F'_k(\mathsf{ctr}+1)$.

(d) Since a random r is always used in Construction 3.30 as input to F, the proof that this construction is CPA-secure (even when F is only weakly pseudorandom) is identical.

3.27 Let F be a pseudorandom permutation. Consider the mode of operation in which a uniform value $\mathsf{ctr} \in \{0,1\}^n$ is chosen, and the ith ciphertext block c_i is computed as $c_i := F_k(\mathsf{ctr}+i+m_i)$. Show that this scheme does not have indistinguishable encryptions in the presence of an eavesdropper.

Solution: An encryption of the 2-block message $\langle 1\rangle, \langle 0\rangle$ (where $\langle i\rangle$ is an encoding of the integer i using exactly n bits) has the form ctr, c, c (i.e., the final two ciphertext blocks are equal) whereas an encryption of $\langle 0\rangle, \langle 1\rangle$ does not. This can easily be turned into a formal attack.

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

Solution: CBC mode. Let $\mathcal{A}$ be an adversary who outputs a pair of messages $m_0 = 0^n$ and $m_1 = 1^n$, and receives a challenge ciphertext $\langle IV, c\rangle$. For CBC encryption, we have that $c = F_k(IV \oplus m_b)$. Adversary $\mathcal{A}$ then asks for a decryption of ciphertext $\langle 0^n, c\rangle$. (Note that $IV \neq 0^n$ except with negligible probability.) The result that $\mathcal{A}$ receives back is

$m' = F_k^{-1}(c) \oplus 0^n$. Given this, $\mathcal{A}$ computes $m' \oplus IV$ which must equal either m_0 or m_1. Thus, $\mathcal{A}$ knows which plaintext was encrypted.

OFB mode. Let $\mathcal{A}$ be an adversary who outputs a pair of messages $m_0 = 0^n$ and $m_1 = 1^n$, and receives a challenge ciphertext $\langle IV, c\rangle$. For OFB encryption we have that $c = F_k(IV) \oplus m_b$. Adversary $\mathcal{A}$ then asks for a decryption of ciphertext $\langle IV, 0^n\rangle$. (Note that $c \neq 0^n$ except with negligible probability.) The result that $\mathcal{A}$ receives back is $m' = F_k(IV) \oplus 0^n$. Given this, $\mathcal{A}$ computes $c \oplus m'$ which must equal either m_0 or m_1. Thus, $\mathcal{A}$ knows which plaintext was encrypted.

CTR mode. The solution is similar to the solution for OFB mode.

3.29 Let $\Pi_1 = (\mathsf{Enc}_1, \mathsf{Dec}_1)$ and $\Pi_2 = (\mathsf{Enc}_2, \mathsf{Dec}_2)$ be two encryption schemes for which it is known that at least one is CPA-secure (but you don't know which one). 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. Provide a full proof of your solution.

Solution: Define $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ as follows:

(a) $\mathsf{Gen}(1^n)$ computes $k_1 \leftarrow \mathsf{Gen}_1(1^n)$ and $k_2 \leftarrow \mathsf{Gen}_2(1^n)$ and outputs (k_1, k_2).

(b) Upon input $k = (k_1, k_2)$ and m, algorithm Enc chooses a random $r \leftarrow \{0,1\}^{|m|}$ and outputs $c = \langle \mathsf{Enc}_{1k_1}(m \oplus r), \mathsf{Enc}_{2k_2}(r)\rangle$.

(c) Upon input $k = (k_1, k_2)$ and $c = \langle c_1, c_2\rangle$ output $\mathsf{Dec}_{1k_1}(c_1) \oplus \mathsf{Dec}_{2k_2}(c_2)$.

We prove that the above scheme is CPA-secure as long as at least one of Π_1 and Π_2 is CPA-secure. Assume that Π_1 is CPA-secure (the proof for the case that Π_2 is CPA-secure is almost the same). Let $\mathcal{A}$ be a CPA-adversary attacking Π and let ε be such that

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1] = \frac{1}{2} + \varepsilon(n).$$

We construct an adversary $\mathcal{A}_1$ attacking Π_1 as follows. $\mathcal{A}_1$ chooses $k_2 \leftarrow \mathsf{Gen}_2(1^n)$ and invokes $\mathcal{A}$. For every query m that $\mathcal{A}$ makes to its encryption oracle, $\mathcal{A}_1$ chooses a random r, asks for an encryption of $m \oplus r$ from its oracle (let c_1 be the response), computes $c_2 = \mathsf{Enc}_{2k_2}(r)$, and returns $c = \langle c_1, c_2\rangle$ to $\mathcal{A}$. When $\mathcal{A}$ outputs a pair m_0, m_1, $\mathcal{A}_1$ chooses a random $r \leftarrow \{0,1\}^{|m_0|}$ and outputs the pair $m_0 \oplus r, m_1 \oplus r$. Then, upon receiving back a challenge ciphertext c_1, $\mathcal{A}_1$ computes $c_2 = \mathsf{Enc}_{2k_2}(r)$ and hands $\mathcal{A}$ the challenge ciphertext $c = \langle c_1, c_2\rangle$. When $\mathcal{A}$ outputs some b', adversary $\mathcal{A}_1$ outputs b' as well and halts.

First, we observe that the view of $\mathcal{A}$ in this game by $\mathcal{A}_1$ is identical to its view in an execution of experiment $\mathsf{PrivK}^{\mathsf{cpa}}$. The experiment in its entirety is not identical because there is a dependence between the two

messages encrypted. However, since $\mathcal{A}$ only sees one of the messages encrypted, its view is the same. In particular, we have that

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A}_1,\Pi_1}(n) = 1 \mid b = 0] = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1 \mid b = 0]$$

and likewise for $b = 1$. Therefore,

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A}_1,\Pi_1}(n) = 1] = \frac{1}{2} + \varepsilon(n)$$

and so by the CPA-security of Π_1 we have that ε is negligible, as required.

3.30 Write pseudocode for obtaining the entire plaintext via a padding-oracle attack on CBC-mode encryption using PKCS #5 padding, as described in the text.

No solution given.

3.31 Describe a padding-oracle attack on CTR-mode encryption (assuming PKCS #5 padding is used to pad messages to a multiple of the block length before encrypting).

Solution: The high-level idea is the same; the only difference is that now the attacker must modify the bytes of c_i (rather than c_{i-1}) in order to causes a predictable difference in the ith message block.

Chapter 4

Message Authentication Codes – Solutions

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.

Solution: Assume that $t(n) = c \log n$ for some constant c. Then, consider an adversary $\mathcal{A}$ who upon input 1^n just outputs an arbitrary m and a uniform $t \in \{0,1\}^{t(n)}$. Adversary $\mathcal{A}$ succeeds with probability at least $2^{-t(n)}$ since there must be *some* valid tag for m (note also that $m \notin \mathcal{Q}$ always for this $\mathcal{A}$). Since $t(n) = c \log n$ we have that $2^{-t(n)} = n^{-c}$ which is not negligible.

4.2 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 for this case.

(b) Assume Π is a deterministic MAC using canonical verification that satisfies Definition 4.2. Prove that Π also satisfies your definition from part (a).

Solution:

(a) Consider the following modified experiment:

The message authentication experiment $\mathsf{Mac\text{-}forge}'_{\mathcal{A},\Pi}(n)$**:**

i. *Compute* $k \leftarrow \mathsf{Gen}(1^n)$.

ii. *The adversary* $\mathcal{A}$ *is given input* 1^n *and oracle access to* $\mathsf{Mac}_k(\cdot)$ *and* $\mathsf{Vrfy}_k(\cdot,\cdot)$. *The adversary eventually outputs a pair* (m,t). *Let* $\mathcal{Q}$ *denote the set of all queries that* $\mathcal{A}$ *asked to its* $\mathsf{Mac}_k(\cdot)$ *oracle.*

iii. *The output of the experiment is defined to be 1 if and only if (1)* $\mathsf{Vrfy}_k(m,t) = 1$ *and (2)* $m \notin \mathcal{Q}$.

The definition of security is the same as Definition 4.2, except that it relates to $\mathsf{Mac\text{-}forge}'$ instead of $\mathsf{Mac\text{-}forge}$.

(b) When Π is deterministic and has canonical verification, each message has only a single valid tag. Thus, if the scheme is secure according to Definition 4.2, then access to a Vrfy oracle does not help (and so Π is secure in the sense of the definition given in part (a)). To see this, note that for any query (m, t) to the Vrfy oracle there are 3 possibilities:

i. m was previously queried to the Mac oracle, and response t was received. Here the adversary already knows that $\mathsf{Vrfy}_k(m, t) = 1$.

ii. m was previously queried to the Mac oracle, and response $t' \neq t$ was received. Since Π is deterministic, here the adversary already knows that $\mathsf{Vrfy}_k(m, t) = 0$.

iii. m was not previously queried to the Mac oracle. By security of Π, we can argue that $\mathsf{Vrfy}_k(m, t) = 0$ with all but negligible probability.

This can be turned into a formal proof.

4.3 Assume secure MACs exist. Give a construction of a MAC that is secure with respect to Definition 4.2 but that is not secure when the adversary is additionally given access to a Vrfy oracle (cf. the previous exercise).

Solution: Let $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ be any secure MAC. Define the scheme $\Pi' = (\mathsf{Gen}, \mathsf{Mac}', \mathsf{Vrfy}')$ as follows:

(a) $\mathsf{Mac}'_k(m)$ computes $t = \mathsf{Mac}_k(m)$ and outputs $\langle 0, t, 0, 0\rangle$.

(b) $\mathsf{Vrfy}'_k(m, \langle c, t, i, b\rangle)$ works as follows: if $c = 0$ then it outputs 1 if and only if $\mathsf{Vrfy}_k(m, t) = 1$; if $c = 1$ then it outputs 1 if and only if $\mathsf{Vrfy}_k(m, t) = 1$ and the ith bit of the key k is equal to b.

First observe that Π' is secure in the sense of Definition 4.2 because the tags returned in scheme Π' can be generated from the tags returned from scheme Π. Furthermore, a forgery in Π' requires, in particular, a forgery in Π. This can be turned into a formal proof.

On the other hand, Π' is insecure when an adversary $\mathcal{A}$ is given access to both a Mac and a Vrfy oracle. The attack is to query an arbitrary message m to the Mac oracle to obtain the tag $\langle 0, t, 0, 0\rangle$. Then query the Vrfy oracle with $(m, \langle 1, t, i, 0\rangle)$ for $i = 1, \ldots, n$. By doing so, $\mathcal{A}$ learns the entire key k and can then forge a valid tag for any message of its choice.

4.4 Prove Proposition 4.4.

Solution: In experiment $\mathsf{Mac\text{-}sforge}$ the attacker succeeds if it outputs (m, t) with $\mathsf{Vrfy}_k(m, t) = 1$ and either (1) m was never queried to the Mac oracle or (2) m was queries to the Mac oracle, but the response was not equal to t. If Π uses canonical verification, then condition (2) cannot occur. Condition (1) is ruled out (except with negligible probability) because Π is a secure MAC.

4.5 Assume secure MACs exist. Prove that there exists a MAC that is secure (by Definition 4.2) but is *not* strongly secure (by Definition 4.3).

Solution: Let $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ be a secure MAC. Construct $\Pi' = (\mathsf{Gen}, \mathsf{Mac}', \mathsf{Vrfy}')$. such that $\mathsf{Mac}'_k(m)$ outputs $\mathsf{Mac}_k(m)\|0$ and such that $\mathsf{Vrfy}'_k(m, t\|b) = \mathsf{Vrfy}_k(m, t)$. (I.e., Mac' appends a 0-bit to the original tag, and Vrfy' ignores this extra bit.) It is immediate that the scheme is not strongly secure (since the attacker can simply flip the final bit of a tag on some message), but one can show that the scheme is still secure in the sense of Definition 4.2.

4.6 Consider the following 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 $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ secure? Prove your answer.

Solution: This scheme is not secure. Let $\mathcal{A}$ be an adversary that queries its oracle with two messages $m = m_0\|m_1$ and $m' = m'_0\|m'_1$, where $m_0 \neq m'_0$ and $m_1 \neq m'_1$. Let $t = t_0\|t_1$ and $t' = t'_0\|t'_1$ be the respective responses from its oracle. $\mathcal{A}$ then outputs the message $\tilde{m} = m_0\|m'_1$ and tag $\tilde{t} = t_0\|t'_1$. By the definition of Mac, it follows that $\tilde{t}$ is a correct tag for $\tilde{m}$ and thus $\mathsf{Vrfy}_k(\tilde{m}, \tilde{t}) = 1$ always. Furthermore, since $m_0 \neq m'_0$ and $m_1 \neq m'_1$ we have that $\tilde{m} \notin \mathcal{Q}$. Thus $\mathcal{A}$ succeeds with probability 1 and the scheme is not secure.

4.7 Let F be a pseudorandom function. Show that each of the following MACs is insecure, even if used to authenticate fixed-length messages. (In each case Gen outputs a uniform $k \in \{0,1\}^n$. Let $\langle i \rangle$ denote an $n/2$-bit encoding of the integer i.)

(a) To authenticate a message $m = m_1, \ldots, m_\ell$, 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 $m = m_1, \ldots, m_\ell$, where $m_i \in \{0,1\}^{n/2}$, compute $t := F_k(\langle 1 \rangle \| m_1) \oplus \cdots \oplus F_k(\langle \ell \rangle \| m_\ell)$.

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

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

and let the tag be $\langle r, t \rangle$.

Solution:

(a) Let $m_1, m_2 \in \{0,1\}^n$ be distinct. Then, the tag on the message m_1, m_2 is identical to the tag on m_2, m_1. Thus, an adversary $\mathcal{A}$ can ask for a tag on m_1, m_2 and output the message m_2, m_1 together with the tag it received.

(b) Let $m_1, m_1', m_2, m_2' \in \{0,1\}^{n/2}$ with $m_1 \neq m_1'$ and $m_2 \neq m_2'$. The attacker obtains tag t_1 on the message m_1, m_2; tag t_2 on the message m_1, m_2'; and tag t_3 on the message m_1', m_2. One can then verify that $t_1 \oplus t_2 \oplus t_3$ is a valid tag on m_1', m_2'.

(c) Let $m_1 \in \{0,1\}^{n/2}$ be arbitrary. The attacker can set $r := \langle 1 \rangle \| m_1$ and output the forgery $\langle r, 0^n \rangle$ on the message m_1.

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

Solution: One possible solution can be obtained by modifying the solution for Exercise 4.6. Other attacks are also possible.

4.9 Given any *deterministic* MAC $(\mathsf{Mac}, \mathsf{Vrfy})$, we may view Mac as a keyed function. In both Constructions 4.5 and 4.11, Mac is a pseudorandom function. Give a construction of a secure, deterministic MAC in which Mac is *not* a pseudorandom function.

Solution: There are many possible answers; we give one. Let F be a pseudorandom function and define $\mathsf{Mac}_k(m) = \langle F_k(m), F_k(m) \rangle$ (verification is done in the obvious way). This is not pseudorandom, but is still a secure MAC.

4.10 Is Construction 4.5 necessarily secure when instantiated using a weak pseudorandom function (cf. Exercise 3.26)? Explain.

Solution: No, it is not necessarily secure in this case. For concreteness, consider the weak pseudorandom function F' from Exercise 3.26, part (b). In this case for an odd x it holds that $F'_k(x) = F'_k(x+1)$. Thus, an adversary can query its oracle to obtain a tag on an odd message m and output a valid forgery for the even message $m+1$.

4.11 Prove that Construction 4.7 is secure even when the adversary is additionally given access to a Vrfy oracle (cf. Exercise 4.2).

Solution: (Note that the stated result assumes Π' is a secure MAC that uses canonical verification.) As in the proof of Theorem 4.8, a forgery in Π yields a forgery in Π'. Moreover, a verification oracle for Π' can be simulated using a verification oracle for Π. Since Π is secure even when given access to a verification oracle, it follows that Π' is secure in that case as well.

4.12 Prove that Construction 4.7 is secure if it is changed as follows: 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. (Assume for simplicity that the length of all messages being authenticated is always an integer multiple of $n/2 - 1$.) What is the advantage of this modification?

Solution: Intuitively, the scheme is still secure because the length of the message is only included to prevent a "truncation" attack where some number of blocks at the end of the message are dropped.

Formally, the proof is largely the same as the proof of Theorem 4.8, and we use the same notation as there. Recall that the adversary outputs (m, t) with $m = m_1, \ldots, m_d$ and $t = \langle r, t_1, \ldots, t_d \rangle$; now it is required that $m_i \in \{0,1\}^{n/2-1}$ and no padding is applied. $\Pr[\mathsf{Repeat}]$ is still negligible. When Repeat does not occur, if $r \notin \{r_1, \ldots, r_q\}$ then clearly $\mathsf{NewBlock}$ occurs. Otherwise $r = r_j$ for some unique j, and then:

(a) If $\ell \neq \ell_j$ then the block $r \| 1 \| d \| m_d$ was never authenticated before by the MAC oracle, and so $\mathsf{NewBlock}$ occurs in this case.

(b) If $\ell = \ell_j$ then $m' = m'_1, \ldots, m'_d$ but there must be an i such that $m'_i \neq m_i$. Then the block $r \| b \| i \| m_i$ was never previously authenticated by the MAC oracle (where b is 1 or 0 depending on whether $i = d$ or not).

The advantage is that computation is reduced and the tag is shorter, since more bits of the message are processed with each invocation of F.

4.13 We explore what happens when the basic CBC-MAC construction is used with messages of different lengths.

(a) Say the sender and receiver do not agree on the message length in advance (and so $\mathsf{Vrfy}_k(m, t) = 1$ iff $t \stackrel{?}{=} \mathsf{Mac}_k(m)$, regardless of the length of m), but the sender is careful to only authenticate messages of length $2n$. Show that an adversary can forge a valid tag on a message of length $4n$.

(b) Say the receiver only accepts 3-block messages (so $\mathsf{Vrfy}_k(m, t) = 1$ only if m has length $3n$ and $t \stackrel{?}{=} \mathsf{Mac}_k(m)$), but the sender authenticates messages of any length a multiple of n. Show that an adversary can forge a valid tag on a new message.

Solution: There are multiple solutions; we provide one in each case.

(a) Let $m_1, m_2 \in \{0,1\}^n$ be arbitrary. The attacker requests a tag on the message m_1, m_2, and obtains in return a tag t. One can check that t is also a valid tag on the message $m_1, m_2, m_1 \oplus t, m_2$.

(b) Let $m_1, m_2, m_3 \in \{0,1\}^n$ be arbitrary. Obtain tag t_1 on the message m_1, tag t_2 on the message $t_1 \oplus m_2$, and tag t_3 on the message $t_2 \oplus m_3$. Then t_3 is a valid tag for the 3-block message m_1, m_2, m_3.

4.14 Prove that the following modifications of basic CBC-MAC do not yield a secure MAC (even for fixed-length messages):

(a) Mac outputs all blocks $t_1, \ldots, t_\ell$, rather than just t_ℓ. (Verification only checks whether t_ℓ is correct.)

(b) A random initial block is used each time a message is authenticated. That is, choose uniform $t_0 \in \{0,1\}^n$, run basic CBC-MAC over the "message" $t_0, m_1, \ldots, m_\ell$, and output the tag $\langle t_0, t_\ell \rangle$. Verification is done in the natural way.

Solution:

(a) Consider the following attack: Adversary $\mathcal{A}$ queries its oracle with arbitrary m_1, m_2. It receives back the tag t_1, t_2, where $t_1 = F_k(m_1)$ and $t_2 = F_k(t_1 \oplus m_2)$. Next, $\mathcal{A}$ outputs the message $t_1 \oplus m_2 \| t_2 \oplus m_1$ and the tag t_2, t_1. This is a valid forgery (unless $t_1 \oplus m_2 = m_1$ and $t_2 \oplus m_1 = m_2$, which occurs with only negligible probability), since $t_2 = F_k(t_1 \oplus m_2)$ and

$$t_1 = F_k(t_2 \oplus t_2 \oplus m_1) = F_k(m_1).$$

(b) Consider the following attack: obtain tag (t_0, t) on the one-block message m. Then output (m, t) as a valid tag for the message t_0.

4.15 Show that appending the message length to the *end* of the message before applying basic CBC-MAC does not result in a secure MAC for arbitrary-length messages.

Solution: (We provide the solution, but it will help to draw a diagram to confirm that the solution is correct.) One attack is as follows: obtain tag t on the 1-block message m, the tag s on the 3-block message $m, \langle n \rangle, t$ (where $\langle n \rangle$ denotes the integer n encoded as an n-bit string), and the tag t' on the 1-block message m' (with $m' \neq m$). Then one can verify that s is a valid tag on the 3-block message $m', \langle n \rangle, t'$.

4.16 Show that the encoding for arbitrary-length messages described in Section 4.4.2 is prefix-free.

Solution: Let X and X' be two different encoded strings. Note that X can be parsed as $X = \langle i \rangle \| 0^t \| m$ where i is the length of m, the notation $\langle i \rangle$ denote the n-bit encoding of i, and $t \in \{0, \ldots, n-1\}$ is such that $n+t+i$ is a multiple of n. We can similarly parse X' as $X' = \langle j \rangle \| 0^{t'} \| m'$. If $i \neq j$ then neither X nor X' is a prefix of the other. If $i = j$ then $t = t'$ and m and m' have the same length. But since $m \neq m'$ neither X nor X' is a prefix of the other.

4.17 Consider the following encoding that handles messages whose length is less than $n \cdot 2^n$: We encode a string $m \in \{0,1\}^*$ by first appending as many 0s as needed to make the length of the resulting string $\hat{m}$ a nonzero multiple of n. Then we prepend the number of *blocks* in $\hat{m}$

(equivalently, prepend the integer $|\hat{m}|/n$), encoded as an n-bit string. Show that this encoding is *not* prefix-free.

Solution: The encoding of $m = 1$ is the string $\langle 1\rangle \| 1 \| 0^{n-1}$. The encoding of $m = 10$ is $\langle 1\rangle \| 10 \| 0^{n-2}$. These are identical!

4.18 Prove that the following modification of basic CBC-MAC gives a secure MAC for arbitrary-length messages (for simplicity, assume all messages have length a multiple of the block length). $\mathsf{Mac}_k(m)$ first computes $k_\ell = F_k(\ell)$, where ℓ is the length of m. The tag is then computed using basic CBC-MAC with key k_ℓ. Verification is done in the natural way.

Solution: Assume F is a pseudorandom function. Roughly speaking, this construction reduces to fixed-length CBC-MAC where an independent key k_ℓ is used for messages of length ℓ. Since fixed-length CBC-MAC is secure, the overall construction is secure. A full solution would require a formal proof.

4.19 Let F be a keyed function that is a secure (deterministic) MAC for messages of length n. (Note that F need not be a pseudorandom permutation.) Show that basic CBC-MAC is not necessarily a secure MAC (even for fixed-length messages) when instantiated with F.

Solution: Several solutions are possible; we present one. Let F' be a pseudorandom function with n-bit input length and $n/2$-bit output length. Define $F_k(m) = F'_k(m) \| \langle i \rangle$, where $\langle i \rangle$ is an $n/2$-bit encoding of the number of initial 0s in m. This F is still a secure MAC. But now consider extending it as in CBC-MAC, for simplicity for messages of length $2n$. By requesting tags on messages of the form m_1, m_2 with m_1 fixed and m_2 varying bit-by-bit starting from the left, the value t_1 of the intermediate result $F'_k(m_1)$ can be learned. It is then easy to forge a tag on the message $m_1, m_1 \oplus t$.

4.20 Show that Construction 4.7 is strongly secure.

Solution: (Note that the stated result assumes Π' is strongly secure.) Note that we only need to consider the case where the attacker outputs (m, t) for which m was previously authenticated (possibly multiple times), but never using tag t (since the case where m is a message that was never previously authenticated is already handled by Theorem 4.8). We show how to modify the proof of Theorem 4.8 for this case, and borrow notation from there. Repeat is defined identically, and Pr[Repeat] is still negligible. We now let NewBlock the event that either one of the blocks $r \| \ell \| i \| m_i$ was never previously authenticated, or it was previously authenticated but not using tag t_i. As before, Pr[NewBlock] is negligible, now based on *strong* security of Π'. Using a case analysis, one can show that if neither Repeat nor NewBlock occur, then the attacker cannot succeed.

4.21 Show that Construction 4.18 might not be CCA-secure if it is instantiated with a secure MAC that is *not* strongly secure.

Solution: Consider the MAC from the solution to Exercise 4.5, which is secure but not strongly secure. It is easy to give a chosen-ciphertext attack on Construction 4.18 when constructed using that MAC, by having the attacker just flip the final bit of the challenge ciphertext and request decryption of the resulting, modified ciphertext.

4.22 Prove that Construction 4.18 is unforgeable when instantiated with any encryption scheme (even if not CPA-secure) and any secure MAC (even if the MAC is not strongly secure).

Solution We describe the intuition; a full solution would require a formal proof. Say the attacker makes encryption-oracle queries for messages $m_1, \ldots, m_\ell$, obtaining in return ciphertexts $\langle c_1, t_1\rangle, \ldots, \langle c_\ell, t_\ell\rangle$. Let $\langle c, t\rangle$ be the ciphertext output by the attacker. If $c \notin \{c_1, \ldots, c_\ell\}$, then security of the MAC implies that decryption fails. But if $c \in \{c_1, \ldots, c_\ell\}$ then either decryption fails or else the resulting message m is equal to one of $m_1, \ldots, m_\ell$ and so this does not violate unforgeability.

4.23 Consider a strengthened version of unforgeability (Definition 4.16) where $\mathcal{A}$ is additionally given access to a decryption oracle.

(a) Write a formal definition for this version of unforgeability.

(b) Prove that Construction 4.18 satisfies this stronger definition if Π_M is a strongly secure MAC.

(c) Show by counterexample that Construction 4.18 need not satisfy this stronger definition if Π_M is a secure MAC that is not strongly secure. (Compare to the previous exercise.)

Solution:

(a) The definition is similar to Definition 4.16, except that the experiment $\mathsf{Enc\text{-}Forge}_{\mathcal{A},\Pi}(n)$ is modified to give $\mathcal{A}$ access to a decryption oracle.

(b) A proof follows from the proof of Theorem 4.19, which shows that (except with negligible probability) any "new" ciphertexts the attacker submits to the decryption oracle will be invalid.

(c) Consider the MAC from the solution to Exercise 4.3. Since the decryption oracle effectively acts (also) as a verification oracle, the same attack described in the solution to Exercise 4.3 allows an attacker to learn k_M. At that point, the attacker can forge new, valid ciphertexts if, say, Π_E is CBC-MAC.

4.24 Prove that the authenticate-then-encrypt approach, instantiated with any CPA-secure encryption scheme and any secure MAC, yields a CPA-secure encryption scheme that is unforgeable.

Solution: We provide the intuition; a full solution would require formal proofs. For unforgeability, note that if the attacker were given the encryption key k_E, the encryption oracle effectively provides the attacker with a MAC oracle; any ciphertext c produced by the attacker can be decrypted to some message/tag pair. Thus, security of the MAC implies unforgeability. CPA-security of the construction is immediately implied by CPA-security of Π_E.

4.25 Let F be a strong pseudorandom permutation, and define the following fixed-length encryption scheme: On input a message $m \in \{0,1\}^{n/2}$ and key $k \in \{0,1\}^n$, algorithm Enc chooses a uniform $r \in \{0,1\}^{n/2}$ and computes $c := F_k(m\|r)$. (See Exercise 3.18.) Prove that this scheme is CCA-secure, but is not an authenticated encryption scheme.

Solution: The scheme trivially does not satisfy unforgeability, since any n-bit string is a valid ciphertext. But the scheme is CCA-secure—once again, we provide the intuition but would expect formal proofs for a full solution. To see this, imagine replacing F with a random permutation π. Let E denote the set of random strings used to answer the attacker's encryption-oracle queries, and let D denote the set of $n/2$-bit suffixes in the answers to the attacker's decryption-oracle queries. The elements of E are uniform and independent; the elements in D are "close" to uniform and independent. As long as the random string used when generating the challenge ciphertext is not equal to any of the elements in $E \cup D$, the attacker learns nothing about which of its two messages was encrypted.

4.26 Show a CPA-secure private-key encryption scheme that is unforgeable but is not CCA-secure.

Solution: One solution is given by Construction 4.18 instantiated with any CPA-secure encryption scheme and the MAC from the solution to Exercise 4.5. (See also the solution to Exercise 4.21.)

4.27 Fix $\ell > 0$ and a prime p. Let $\mathcal{K} = \mathbb{Z}_p^{\ell+1}$, $\mathcal{M} = \mathbb{Z}_p^{\ell}$, and $\mathcal{T} = \mathbb{Z}_p$. Define $h : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ as

$$h_{k_0,k_1,\ldots,k_\ell}(m_1,\ldots,m_\ell) = \left[k_0 + \textstyle\sum_i k_i m_i \bmod p\right].$$

Prove that h is strongly universal.

Solution: Fix distinct $m = (m_1,\ldots,m_\ell)$ and $m' = (m'_1,\ldots,m'_\ell)$ and arbitrary $t, t' \in \mathbb{Z}_p$. Then $h_{k_0,k_1,\ldots,k_\ell}(m) = t$ and $h_{k_0,k_1,\ldots,k_\ell}(m') = t'$ if and only if

$$t = k_0 + \textstyle\sum_i k_i m_i \quad \text{and} \quad t' = k_0 + \textstyle\sum_i k_i m'_i, \tag{4.1}$$

where everything is modulo p. We count the number of keys satisfying the above. Let i^* be maximal such that $m_{i^*} \neq m'_{i^*}$. Let $k_1,\ldots,k_{i^*-1}$

and $k_{i^*+1}, \ldots, k_\ell$ be arbitrary. We then obtain the equations

$$t - \textstyle\sum_{i \neq i^*} k_i m_i - k_0 = k_{i^*} m_{i^*} \quad \text{and} \quad t' - \sum_{i \neq i^*} k_i m'_i - k_0 = k_{i^*} m'_{i^*}.$$

Subtracting, we obtain

$$t - t' - \textstyle\sum_{i \neq i^*} k_i m_i + \sum_{i \neq i^*} k_i m'_i = k_{i^*}(m_{i^*} - m'_{i^*}),$$

which uniquely determines k_{i^*} since $m_{i^*} - m'_{i^*} \neq 0$ and p is prime. The original equations then uniquely determine k_0. We conclude that exactly $p^{\ell-1}$ keys satisfy (4.1), and so the probability that a uniform key satisfies those equations is $p^{\ell-1}/p^{\ell+1} = 1/|\mathcal{T}|^2$, as required.

4.28 Fix $\ell, n > 0$. Let $\mathcal{K} = \{0,1\}^{\ell \times n} \times \{0,1\}^\ell$ (interpreted as a boolean $\ell \times n$ matrix and an ℓ-dimensional vector), let $\mathcal{M} = \{0,1\}^n$, and let $\mathcal{T} = \{0,1\}^\ell$. Define $h : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ as $h_{K,v}(m) = K \cdot m \oplus v$, where all operations are performed modulo 2. Prove that h is strongly universal.

Solution: Fix distinct m, m' and arbitrary t, t'. Then $h_{K,v}(m) = t$ and $h_{K,v}(m') = t'$ if and only if

$$Km \oplus v = t \quad \text{and} \quad Km' \oplus v = t'. \tag{4.2}$$

Subtracting, we see the above hold only if $K \cdot (m - m') = t - t'$. Let i be the largest index for which the ith coordinate of $m - m'$ is nonzero. Then we may set all columns of K other than the ith column arbitrarily to any of $s^{\ell \cdot (n-1)}$ possibilities; there is then a unique value of the ith column for which $K \cdot (m - m') = t - t'$. Having fixed K, there is a unique v such that (4.2) hold. We conclude that the number of keys satisfying (4.2) is $2^{\ell \cdot (n-1)}$, and so the probability that a uniform key satisfies those equations is $2^{\ell \cdot (n-1)}/2^{\ell \cdot (n+1)} = 1/2^{2\ell} = 1/|\mathcal{T}|^2$, as required.

4.29 A *Toeplitz matrix* K is a matrix in which $K_{i,j} = K_{i-1,j-1}$ when $i, j > 1$; i.e., the values along any diagonal are equal. So an $\ell \times n$ Toeplitz matrix (for $\ell > n$) has the form

$$\begin{bmatrix} K_{1,1} & K_{1,2} & K_{1,3} & \cdots & K_{1,n} \\ K_{2,1} & K_{1,1} & K_{1,2} & \cdots & K_{1,n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ K_{\ell,1} & K_{\ell-1,1} & K_{\ell-2,1} & \cdots & K_{\ell-n+1,1} \end{bmatrix}.$$

Let $\mathcal{K} = T^{\ell \times n} \times \{0,1\}^\ell$ (where $T^{\ell \times n}$ denotes the set of $\ell \times n$ Toeplitz matrices), let $\mathcal{M} = \{0,1\}^n$, and let $\mathcal{T} = \{0,1\}^\ell$. Define $h : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ as $h_{K,v}(m) = K \cdot m \oplus v$, where all operations are performed modulo 2. Prove that h is strongly universal. What is the advantage here as compared to the construction in the previous exercise?

Solution: We first count the number of Toeplitz matrices. In the first row there are n values that may be chosen. In each of the successive

$(\ell - 1)$ rows, the final $n - 1$ elements are determined by the previous row so there is only one new value to choose. Thus, the total number of Toeplitz matrices is $2^{n+\ell-1}$, and the size of the key space is $2^{n+2\ell-1}$.

As in the previous exercise, we now count the number of keys for which $h_{K,v}(m) = t$ and $h_{K,v}(m') = t'$. These equations hold only if $K(m - m') = t - t'$. Let i again be the largest index for which the ith coordinate of $m - m'$ is nonzero. We fill in the entries of K starting from the last row. The first $i-1$ entries of that row can be set arbitrarily; there is then a unique value for the ith entry such that the equation $K(m-m') = t-t'$ can possibly be satisfied. Proceeding to the second-to-last row (we do not yet set the final $n-i$ entries of the last row), we see that the first $i-1$ entries are already fixed and there is a unique value for the ith entry such that the equation $K(m - m') = t - t'$ can possibly be satisfied; note that this determines also the $(i + 1)$st entry of the bottom row. Continuing in this manner to the top row, we see that all entries of the matrix are uniquely determined except for the final $n - i$ elements of the top row (which in turn determine the final $n - i - 1$ elements of the second-to-top row, etc.). These can be set arbitrarily. Having fixed K, the original pair of equations then uniquely determine v. We conclude that the number of keys for which $h_{K,v}(m) = t$ and $h_{K,v}(m') = t'$ is $2^{(i-1)+(n-i)} = 2^{n-1}$, and so the probability that a uniform key satisfies those equations is $2^{n-1}/2^{n+2\ell-1} = 1/2^{2\ell} = 1/|\mathcal{T}|^2$, as required.

The advantage of this construction is that it has shorter keys.

4.30 Define an appropriate notion of a *two-time* ε-secure MAC, and give a construction that meets your definition.

Solution: A definition is obtained by a straightforward modification of Definition 4.22, where the experiment now allows the attacker to (adaptively) request tags on any *two* messages of its choice.

The natural primitive for constructing a two-time ε-secure MAC is a three-wise independent function, i.e., a keyed function $h : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ such that for all distinct $m, m', m'' \in \mathcal{M}$ and all $t, t', t'' \in \mathcal{T}$ we have

$$h_k(m) = t \wedge h_k(m') = t' \wedge h_k(m'') = t''] = 1/|\mathcal{T}|^3$$

(where the probability is over choice of k). An example of such a function is given by quadratic polynomials over a field. I.e., let $\mathcal{M}, \mathcal{T} = \mathbb{Z}_p$ and $\mathcal{K} = \mathbb{Z}_p^3$, and define $h_{k_1,k_2,k_3}(m) = k_1 m^2 + k_2 m + k_3$. One can check that this satisfies the requirement.

4.31 Let $\{h_n : \mathcal{K}_n \times \{0,1\}^{10\cdot n} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ be such that h_n is strongly universal for all n, and let F be a pseudorandom function. (When $K \in \mathcal{K}_n$ we write $h_K(\cdot)$ instead of $h_{n,K}(\cdot)$.) Consider the following MAC: $\mathsf{Gen}(1^n)$ chooses uniform $K \in \mathcal{K}_n$ and $k \in \{0,1\}^n$, and outputs (K, k). To authenticate a message $m \in \{0,1\}^{10\cdot n}$, choose uniform

$r \in \{0,1\}^n$ and output $\langle r,\ h_K(m) \oplus F_k(r)\rangle$. Verification is done in the natural way. Prove that this gives a (computationally) secure MAC for messages of length $10n$.

Solution: We provide a proof sketch; a full solution would require formal proof. First note that we may, in the standard way, replace F_k by a truly random function f. Let Repeat be the event that, in answering the attacker's Mac queries, an r-value is repeated. This event occurs with negligible probability; we show that as long as it does not occur, the attacker's success probability is 2^{-n}.

Assume Repeat does not occur. In this case, note that in each tag $\langle r_i, t_i\rangle$ obtained by the attacker on a message m_i, the value t_i is uniform and independent of other values, and in particular independent of $h_K(m_i)$. That is, intuitively, the attacker gets no information about K. Now let $m, \langle r, t\rangle$ be the candidate forgery output by the attacker. If r was never used to answer any Mac query by the attacker, then $f(r)$ is uniform and independent of anything else and it is immediate that the attacker succeeds only with probability 2^{-n}. Otherwise, since Repeat did not occur, there is a unique Mac query—say, for message m_i—for which the tag $\langle r, t_i\rangle$ was returned. The message/tag pair output by the attacker is valid if and only if

$$h_K(m) \oplus h_K(m_i) = (t \oplus f(r)) \oplus (t_i \oplus f(r)) = t \oplus t_i.$$

But the fact that h is strongly universal implies that the above occurs only with probability 2^{-n}.

Chapter 5

Hash Functions and Applications – Solutions

5.1 Provide formal definitions for second preimage resistance and preimage resistance. Prove that any hash function that is collision resistant is second preimage resistant, and any hash function that is second preimage resistant is preimage resistant.

Solution: We define two experiments for a hash function $\Pi = (\mathsf{Gen}, H)$, adversary $\mathcal{A}$, and security parameter n. We assume for simplicity that H maps inputs of length $2n$ to outputs of length n, but this is inessential.

DEFINITION Π *is* preimage resistant *if for all* PPT *adversaries* $\mathcal{A}$ *there is a negligible function* negl *such that the success probability of* $\mathcal{A}$ *in the following experiment is negligible:*

(a) A key s is generated by running $\mathsf{Gen}(1^n)$, and a random $x \in \{0,1\}^{2n}$ is chosen.
(b) The adversary $\mathcal{A}$ is given s and $H^s(x)$, and outputs x'.
(c) We say that $\mathcal{A}$ succeeds if $H^s(x') = H^s(x)$.

DEFINITION Π *is* second preimage resistant *if for all* PPT *adversaries* $\mathcal{A}$ *there is a negligible function* negl *such that the success probability of* $\mathcal{A}$ *in the following experiment is negligible:*

(a) A key s is generated by running $\mathsf{Gen}(1^n)$, and a random $x \in \{0,1\}^{2n}$ is chosen.
(b) The adversary $\mathcal{A}$ is given s and x, and outputs $x' \neq x$.
(c) We say that $\mathcal{A}$ succeeds if $H^s(x') = H^s(x)$.

We first prove that any collision-resistant hash function is also second preimage resistant. Let Π be a collision-resistant hash function. Let $\mathcal{A}$ be an adversary and ε a function such that

$$\Pr\left[\mathcal{A}(s,x) = x' \ \wedge \ x \neq x' \ \wedge \ H^s(x) = H^s(x')\right] = \varepsilon(n).$$

We construct an adversary $\mathcal{A}'$ that finds a collision in Π as follows. Upon input s, adversary $\mathcal{A}'$ chooses uniform $x \in \{0,1\}^{2n}$ and runs $\mathcal{A}(s,x)$.

If $\mathcal{A}$ outputs $x' \neq x$ such that $H^s(x) = H^s(x')$ then $\mathcal{A}'$ outputs the collision (x, x'). It is immediate that $\mathcal{A}'$ outputs a collision—or, stated differently, succeeds in experiment Hash-coll—with probability exactly ε. Thus, ε must be negligible, and so Π is second preimage resistant.

We now prove that any second preimage resistant hash function is also preimage resistant. Let Π be a second preimage resistant hash function. Let $\mathcal{A}$ be an adversary and set

$$\varepsilon(n) \stackrel{\text{def}}{=} \Pr\left[\mathcal{A}(s, H^s(x)) = x' \ \wedge \ H^s(x') = H^s(x)\right].$$

We construct an adversary $\mathcal{A}'$ that finds a second preimage in Π as follows. Upon input s and $x \leftarrow \{0,1\}^{2n}$, $\mathcal{A}'$ invokes $\mathcal{A}$ upon $(s, H^s(x))$. If $\mathcal{A}$ outputs a value $x' \neq x$ such that $H^s(x') = H^s(x)$ then $\mathcal{A}'$ outputs x' and halts. The interesting part of the proof is to show that with high probability $\mathcal{A}$ outputs a *different* preimage $x' \neq x$. Intuitively, this is the case because inputs of length $2n$ are mapped to outputs of length n. Thus, on average there are many inputs mapped to the same $H^s(x)$, and so the probability that $x' = x$ is small.

Formally, we first show that with probability almost one over choice of a random x, there exists at least one other x' such that $H^s(x') = H^s(x)$. We prove this using a simple counting argument. Fix s, and define

$$T \stackrel{\text{def}}{=} \{x \in \{0,1\}^{2n} : H^s(x) \text{ has only one preimage under } H^s\}.$$

Clearly, $|T| \leq 2^n$ (since the range of H^s has size 2^n). It follows that $x \notin T$ with probability at least $1 - 2^n/2^{2n} = 1 - 2^{-n}$.

If $x \notin T$ and $\mathcal{A}$ outputs a value x' such that $H^s(x') = H^s(x)$, the probability that $x' = x$ is at most $1/2$. (This conclusion follows from the above because the view of $\mathcal{A}$ is identical whether x or x' is chosen.) We conclude that:

$$\begin{aligned}
&\Pr_x\left[\mathcal{A}'(s,x) = x' \ \wedge \ x \neq x' \ \wedge \ H^s(x) = H^s(x')\right] \qquad (5.1)\\
&\geq \Pr_x\left[\mathcal{A}'(s,x) = x' \ \wedge \ x \neq x' \ \wedge \ H^s(x) = H^s(x') \mid x \notin T\right] \cdot \Pr_x[x \notin T]\\
&= \Pr_x\left[x' \neq x \mid \mathcal{A}(s, H^s(x)) = x' \ \wedge \ H^s(x') = H^s(x) \ \wedge \ x \notin T\right]\\
&\quad \cdot \Pr_x\left[\mathcal{A}(s, H^s(x)) = x' \ \wedge \ H^s(x') = H^s(x) \mid x \notin T\right] \cdot \Pr_x[x \notin T]\\
&\geq \frac{1}{2} \cdot \Pr_x\left[\mathcal{A}(s, H^s(x)) = x' \ \wedge \ H^s(x') = H^s(x) \mid x \notin T\right] \cdot \Pr_x[x \notin T].
\end{aligned}$$

To complete the proof, note that:

$$\begin{aligned}
&\Pr\left[\mathcal{A}(s, H^s(x)) = x' \ \wedge\ H^s(x') = H^s(x)\right] \\
&= \Pr\left[\mathcal{A}(s, H^s(x)) = x' \ \wedge\ H^s(x') = H^s(x) \mid x \notin T\right] \cdot \Pr[x \notin T] \\
&\quad + \Pr\left[\mathcal{A}(s, H^s(x)) = x' \ \wedge\ H^s(x') = H^s(x) \mid x \in T\right] \cdot \Pr[x \in T] \\
&\leq \Pr\left[\mathcal{A}(s, H^s(x)) = x' \ \wedge H^s(x') = H^s(x) \mid x \notin T\right] \cdot \Pr[x \notin T] + \Pr[x \in T] \\
&\leq \Pr\left[\mathcal{A}(s, H^s(x)) = x' \ \wedge\ H^s(x') = H^s(x) \mid x \notin T\right] \cdot \Pr[x \notin T] + 2^{-n}.
\end{aligned}$$

Thus,

$$\Pr\left[\mathcal{A}(s, H^s(x)) = x' \ \wedge\ H^s(x') = H^s(x) \mid x \notin T\right] \cdot \Pr[x \notin T] \geq \varepsilon(n) -$$

Combining this with Equation (5.1), we have

$$\Pr_x\left[\mathcal{A}'(s, x) = x' \ \wedge\ x \neq x' \ \wedge\ H^s(x) = H^s(x')\right] \geq \frac{\varepsilon(n)}{2} - 2^{-n-1}.$$

Since H is second preimage resistant, ε must be negligible. Because $\mathcal{A}$ was arbitrary, this proves that H is preimage resistant.

5.2 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 $H^{s_1,s_2}(x) = H_1^{s_1}(x) \| H_2^{s_2}(x)$.

(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 preimage resistance and preimage resistance, respectively. Prove your answer in each case.

Solution:

(a) Let $\mathcal{A}$ be an adversary that with probability $\varepsilon(n)$ outputs a pair x, x' such that $H^{s_1,s_2}(x) = H^{s_1,s_2}(x')$. In such a case, by the definition of H, we have that $H_1^{s_1}(x) = H_1^{s_1}(x')$ *and* $H_2^{s_2}(x) = H_2^{s_2}(x')$. Thus, $\mathcal{A}$ finds a collision in both H_1 and H_2 with probability ε. A full reduction here is straightforward (to attack H_1 with key s_1, run Gen_2 to get s_2, invoke $\mathcal{A}$ with key (s_1, s_2), and then output the pair that $\mathcal{A}$ outputs).

(b) An analogous claim does hold for second preimage resistance. Informally, this is again because a collision in H yields a collision in both H_1 and H_2. A formal proof is straightforward.

The same does not hold for preimage resistance, i.e., it is possible that (say) H_1 is preimage resistant yet H is not. We sketch a counter-example demonstrating this. Let $\hat{H}$ be a preimage resistant

hash function mapping n-bit inputs to $n/2$-bit outputs, and let trunc output the first $n/2$ bits of its input. Define

$$H_1^s(x_1\|x_2) = \mathsf{trunc}(x_1) \parallel \hat{H}^s(x_2),$$

where $x_1, x_2 \in \{0,1\}^n$. Note that H_1 is preimage resistant. Let H_2 be a keyed function that (for every key s) outputs the last n bits of its input. Then

$$\begin{aligned} H^{s_1,s_2}(x_1\|x_2) &= H_1^{s_1}(x_1\|x_2) \parallel H_2^{s_2}(x_1\|x_2) \\ &= \mathsf{trunc}(x_1) \parallel \hat{H}^{s_1}(x_2) \parallel x_2, \end{aligned}$$

and it is trivial to find a preimage in H.

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

Solution: Yes. Let x, x' be a collision for $\hat{H}$; that is, $H^s(H^s(x)) = H^s(H^s(x'))$. There are two cases:

(a) $H^s(x) = H^s(x')$: in this case, x, x' is a collision for the original H. Therefore, such a pair can only be found with negligible probability.

(b) $H^s(x) \neq H^s(x')$: set $y = H^s(x)$ and $y' = H^s(x')$. By the assumption in this case, $y \neq y'$ and yet $H^s(y) = H^s(y')$. Thus, y, y' is a collision in H. Once again, this implies that such a pair x, x' can be found with only negligible probability (otherwise, by computing H^s once on each one obtains a collision y, y' in H^s).

5.4 Provide a formal proof of Theorem 5.4 (i.e., describe the reduction).

Solution: Let $\mathcal{A}$ be a probabilistic polynomial-time adversary and ε a function such that $\mathcal{A}$ succeeds in the $\mathsf{Hash\text{-}coll}$ experiment with (Gen, H) with probability ε. We construct an adversary $\mathcal{A}'$ that succeeds in the $\mathsf{Hash\text{-}coll}$ experiment with (Gen, h) with probability ε. Upon receiving s, $\mathcal{A}'$ invokes $\mathcal{A}$ upon s and receives back a pair x, x'. If $H^s(x) \neq H^s(x')$ then $\mathcal{A}'$ just halts. Otherwise, $\mathcal{A}'$ uses the strategy described in the portion of the proof of Theorem 5.4 to find a collision in h. Specifically, if $L \neq L'$ then $\mathcal{A}'$ outputs $z_B\|L$ and $z'_{B'}\|L'$ and halts. Otherwise, if $L = L'$, then $\mathcal{A}'$ works backwards from the last block to the first to find the maximal i^*. Then, $\mathcal{A}'$ outputs $z_{i^*-1}\|x_{i^*}$ and $z'_{i^*-1}\|x'_{i^*}$ and halts. Note that $\mathcal{A}'$ can compute all the z-values by computing H and storing all the intermediate values. By what is shown in the proof in the book, whenever $\mathcal{A}$ finds a collision, $\mathcal{A}'$ also finds a collision. Thus, $\mathcal{A}'$ succeeds in $\mathsf{Hash\text{-}coll}$ for (Gen, h) with probability ε, implying that $\varepsilon(n)$ is negligible. Thus, (Gen, H) is collision-resistant as required.

5.5 Generalize the Merkle-Damgård transform (Construction 5.3) for the case when the fixed-length hash function h has input length $n + \kappa$ (with

$\kappa > 0$) and output length n, and the length of the input to H should be encoded as an ℓ-bit value (as discussed in Section 5.3.2). Prove collision resistance of (Gen, H), assuming collision resistance of (Gen, h).

Solution: See Construction 5.1-S below. This construction is interesting as long as κ is large enough so that 2^κ captures the maximum length of message needed. A proof that this is collision-resistant is similar, though not identical, to the proof of Theorem 5.4.

CONSTRUCTION 5.1-S

Let (Gen, h) be a fixed-length collision-resistant hash function for inputs of length $n+\kappa$ and with output length n, where $\kappa > 0$. Construct a variable-length hash function (Gen, H) as follows:

- Gen: remains unchanged.
- H: on input a key s and a string $x \in \{0,1\}^*$ of length $L < 2^\kappa$, do the following:
  (a) Pad x with zeroes so its length is a multiple of κ. Append L, encoded in binary using exactly κ bits. Parse the resulting string as a sequence of κ-bit blocks $X_1, \ldots, X_B$.
  (b) Set $z_0 := 0^n$.
  (c) For $i = 1, \ldots, B$, compute $z_i := h^s(z_{i-1} \| X_i)$.
  (d) Output z_B.

Generalized Merkle-Damgård transform.

When κ is small (e.g., $\kappa = 1$), a solution is obtained by allocating n additional blocks, always encoding the length using exactly n bits, and appending it to the input before applying the Merkle-Damgård transform. This ensures that if the lengths of two messages are different, then they differ in their last n bits. The proof that this is collision resistant is similar again to the proof of Theorem 5.4.

5.6 For each of the following modifications to the Merkle-Damgård transform (Construction 5.3), determine whether the result is collision resistant. 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 $z_{B+1} = h^s(z_B \| L)$). (Assume the resulting hash function is only defined for inputs whose length is an integer multiple of the block length.)

(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 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.

(d) 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.

Solution:

(a) This is not collision resistant. In particular, take any x of length that is not an exact multiple of ℓ and then set $x' := x\|0$. (Counterexamples are also possible in which there are collisions between two inputs whose lengths are multiples of ℓ, though they rely on a contrived compression function h.)

(b) This is collision resistant. The only difference in the proof of Theorem 5.4 is with respect to Case 1. However, here it is even easier because when $L \neq L'$ the result cannot be a collision (note that $z_B\|L$ cannot equal $z'_{B'}\|L'$ when $L \neq L'$).

(c) Likewise, this is collision resistant and the proof is the same.

(d) This is not necessarily secure. Assume there is a collision-resistant compression function such that it is possible to efficiently find some $L, x \in \{0,1\}^n$ such that $h^s(L, x) = L - n$ for all s). (A full answer would show that it is possible to construct a hash function with this property, assuming collision-resistant hash functions exist.) In this case, the hash of any message y of length $L-1$ is equal to the hash of the message (x, y) of length L.

5.7 Assume collision-resistant hash functions exist. Show a construction of a fixed-length hash function (Gen, h) that is *not* collision resistant, but such that the hash function (Gen, H) obtained from the Merkle-Damgård transform to (Gen, h) as in Construction 5.3 *is* collision resistant.

Solution: Let $\hat{h} : \{0,1\}^{2n-1} \to \{0,1\}^{n-1}$ be a collision-resistant compression function, and define $h : \{0,1\}^{2n} \to \{0,1\}^n$ such that

$$h^s(0\|x) = 0\|\hat{h}^s(x) \quad \text{and} \quad h^s(1\|x) = 1^n$$

(where $x \in \{0,1\}^{2n-1}$). Clearly (Gen, h) is not collision resistant. But one can show that the hash function (Gen, H) obtained from applying the Merkle-Damgård transform to (Gen, h) is collision-resistant.

5.8 Prove or disprove: if (Gen, h) is preimage resistant, then so is the hash function (Gen, H) obtained by applying the Merkle-Damgård transform to (Gen, h) as in Construction 5.3.

Solution: (Note: for preimage-resistance to be defined we must specify some input length. We consider inputs of length L, where L is arbitrary.) This is, in general, false. To see this, let (Gen, h) be preimage

resistant but such that $h^s(x\|L) = 0^n$ for all s and all $x \in \{0,1\}^n$. (A full answer would prove that such a hash function exists assuming preimage-resistant hash functions exist at all.) Then the Merkle-Damgård transform applied to (Gen, h) yields a hash function (Gen, H) for which every L-bit input hashes to 0^n (and so is clearly not preimage-resistant).

5.9 Prove or disprove: if (Gen, h) is second preimage resistant, then so is the hash function (Gen, H) obtained by applying the Merkle-Damgård transform to (Gen, h) as in Construction 5.3.

Solution: This is, in general, false. To see this, let (Gen, h) be second preimage resistant but with the property that, for any $x \in \{0,1\}^n$,

$$h^s([s]_n\|x) = h^s(x\|[s]_n) = 0^n \text{ and } h^s(0^{2n}) = [s]_n,$$

where $[s]_n$ denotes the first n bits of s. (A full answer would prove that such a hash function exists assuming second preimage-resistant hash functions exist at all.) Then the Merkle-Damgård transform applied to (Gen, h) yields a hash function (Gen, H) for which there is the following attacker finding a second preimage:

(a) Output 0^n.
(b) Obtain key s.
(c) Output $[s]_n$.

This attacker succeeds with probability 1 since

$$H^s(0^n) = h^s(h^s(0^n, 0^n), \langle n\rangle) = h^s([s]_n, \langle n\rangle) = 0^n$$

and

$$H^s([s]_n) = h^s(h^s(0^n, [s]_n), \langle n\rangle) = h^s([s]_n, \langle n\rangle) = 0^n,$$

where $\langle n\rangle$ denotes an n-bit encoding of the integer n.

5.10 Before HMAC, it was common to define a MAC for arbitrary-length messages by $\mathsf{Mac}_{s,k}(m) = H^s(k\|m)$ where H is a collision-resistant hash function.

(a) Show that this is never a secure MAC when H is constructed via the Merkle-Damgård transform. (Assume the hash key s is known to the attacker, and only k is kept secret.)
(b) Prove that this is a secure MAC if H is modeled as a random oracle.

Solution:

(a) (For simplicity we omit s.) Let $\langle i\rangle$ denote an n-bit encoding of the integer i. We demonstrate an attack: first request a tag for an arbitrary message $m \in \{0,1\}^n$ and receive in return a tag t. Then compute $t' = h(t, \langle 3n\rangle)$, and output the forged tag t' on the 3-block message $m, t, \langle 2n\rangle$. This succeeds with probability 1 since $t = h(h(h(0^n, k), m), \langle 2n\rangle)$ so then $t' = h(h(h(h(0^n, k), m), \langle 2n\rangle), \langle 3n\rangle)$.

(b) While security (when H is modeled as a random oracle) can be proven directly, a proof also follows along the lines of the proof of Theorem 4.6 using the result of the following exercise. We therefore omit further details.

5.11 Prove that the construction of a pseudorandom function given in Section 5.5.1 is secure in the random-oracle model.

Solution: Let us be clear about the probability spaces of the experiments we will be considering. Let the security parameter be n. In the first experiment, a random function $H : \{0,1\}^{2n} \to \{0,1\}^n$ and a random key $k \in \{0,1\}^n$ are chosen, and $\mathcal{A}$ is given access to H as well as the function $F_k(\cdot) \stackrel{\text{def}}{=} H(k\|\cdot)$. In the second experiment, a random function $H : \{0,1\}^{2n} \to \{0,1\}^n$ and an independent random function $f : \{0,1\}^n \to \{0,1\}^n$ are chosen, and $\mathcal{A}$ is given access to H and f. We want to show that the following is negligible:

$$\left|\Pr[\mathcal{A}^{H(\cdot),H(k\|\cdot)}(1^n) = 1] - \Pr[\mathcal{A}^{H(\cdot),f(\cdot)}(1^n) = 1]\right|.$$

Observe that the views of $\mathcal{A}$ in the two experiments are *identical* (i.e., $\mathcal{A}$ gets uniform and independent responses from each of its oracles) until $\mathcal{A}$ make a query of the form $k\|\star$ to its H-oracle in the first experiment. Since $\mathcal{A}$ has no information about k, the probability that it ever makes such a query is $q/2^n$, where q is a bound on the total number of queries made by $\mathcal{A}$. Since q must be polynomial, this occurs with negligible probability.

5.12 Prove Theorem 5.11.

Solution: The proof is similar to the proof of Theorem 5.4; namely, we show that a collision in $\mathcal{MT}_t$ implies a collision in H. (For simplicity, we leave the key s implicit.) Consider two inputs $(x_1, \ldots, x_t) \neq (x'_1, \ldots, x'_t)$ for which $\mathcal{MT}_t(x_1, \ldots, x_t) = h = \mathcal{MT}_t(x'_1, \ldots, x'_t)$. Following Figure 5.5, we can imagine a pair of binary trees with $1+\log t$ levels where h is at the 0th level of both trees and the inputs $(x_1, \ldots, x_t$ in one case and $x'_1, \ldots, x'_t$ in the other case) are at the tth level of the trees. Let i denote the least integer for which the values at level i are equal in the two trees but the values at level $i+1$ are not. Then there is a collision in H at that level.

5.13 Show how to find a collision in the Merkle tree construction if t is not fixed. Specifically, show how to find two sets of inputs $x_1, \ldots, x_t$ and $x'_1, \ldots, x'_{2t}$ such that $\mathcal{MT}_t(x_1, \ldots, x_t) = \mathcal{MT}_{2t}(x'_1, \ldots, x'_{2t})$.

Solution: We give a solution for $t = 2$ for concreteness. Let x'_1, x'_2, x'_3, x'_4 be arbitrary. Set $x_1 = H(x'_1, x'_2)$ and $x_2 = H(x'_3, x'_4)$. Then notice that $\mathcal{MT}_2(x_1, x_2) = \mathcal{MT}_4(x'_1, \ldots, x'_4)$.

5.14 Consider the scenario introduced in Section 5.6.2 in which a client stores files on a server and wants to verify that files are returned unmodified.

(a) Provide a formal definition of security for this setting.

(b) Formalize the construction based on Merkle trees as discussed in Section 5.6.2.

(c) Prove that your construction is secure relative to your definition under the assumption that (Gen_H, H) is collision resistant.

Solution:

(a) We first define the semantics of a scheme (we note that our formulation is not the only one possible). Let $\Pi = (\mathsf{Upload}, \mathsf{Return}, \mathsf{Vrfy})$ be a triple of algorithms that work as follows:

- Upload takes as input files $x_1, \ldots, x_t$ and returns state s.
- Return takes as input an index i and files $x_1, \ldots, x_t$, and returns a proof π_i.
- Vrfy takes as input state s, an index i, a file x, and a proof π and outputs a bit indicating acceptance or rejection.

In intended operation of the scheme, the client begins by computing $s \leftarrow \mathsf{Upload}(x_1, \ldots, x_t)$, sending $x_1, \ldots, x_t$ to the server, and storing s locally. When the client wants file i, it sends i to the server who then computes $\pi \leftarrow \mathsf{Return}(i, x_1, \ldots, x_t)$ and sends back x_i and π. The client then computes $\mathsf{Vrfy}(s, i, x_i, \pi)$. Correctness requires that, when run as described, Vrfy returns 1.

We now define security. (Again, other definitions are possible.)

DEFINITION Π *is* secure *if for all* PPT *adversaries* $\mathcal{A}$ *there is a negligible function* negl *such that the success probability of* $\mathcal{A}$ *in the following experiment is negligible:*

i. $\mathcal{A}$ outputs $x_1, \ldots, x_t, i, x$, and π.
ii. Compute $s \leftarrow \mathsf{Upload}(x_1, \ldots, x_t)$.
iii. $\mathcal{A}$ $\mathsf{succeeds}$ if $\mathsf{Vrfy}(s, i, x, \pi) = 1$ and $x \neq x_i$.

(b)
- $\mathsf{Upload}(x_1, \ldots, x_t)$ computes $h := \mathcal{MT}_t(x_1, \ldots, x_t)$ and lets $s := (h, t)$.
- $\mathsf{Return}(i, x_1, \ldots, x_t)$ computes $\mathcal{MT}_t(x_1, \ldots, x_t)$ and lets π be the values of the nodes in the Merkle tree adjacent to the path from x_i to the root (as described in Section 5.6.2).
- $\mathsf{Vrfy}(s, i, x, \pi)$, where $s = (h, t)$, uses t, π, x, and i to re-compute a candidate value h' for the root of the Merkle tree (as described in Section 5.6.2). It outputs 1 iff $h' = h$.

(c) A proof of security follows by showing that if $\mathcal{A}$ succeeds in the experiment above, a collision in H is found. We omit the details.

5.15 Prove that the commitment scheme discussed in Section 5.6.5 is secure in the random-oracle model.

Solution: Binding follows immediately from the fact that H is collision-resistant (as discussed in Section 5.5.1). For hiding, we provide only a sketch of the proof. Consider an attacker $\mathcal{A}$ in experiment $\mathsf{Hiding}_{\mathcal{A},\mathsf{Com}}(n)$ that makes q queries to H. Let r denote the random value used by the sender, and note that (1) the view of $\mathcal{A}$ is independent of b (i.e., independent of which message is chosen) unless $\mathcal{A}$ makes a query of the form $H(\star\|r)$ (since then both the commitment as well as all answers from the random oracle are uniform and independent); (2) the probability that $\mathcal{A}$ makes such a query is at most $q/2^n$.

Chapter 6

Practical Constructions of Symmetric-Key Primitives – Solutions

6.1 Assume a degree-6 LFSR with $c_0 = c_5 = 1$ and $c_1 = c_2 = c_3 = c_4 = 0$.

(a) What are the first 10 bits output by this LFSR if it starts in initial state $(1, 1, 1, 1, 1, 1)$?

(b) Is this LFSR maximal length?

Solution:

(a) $1, 1, 1, 1, 1, 1, 0, 1, 0, 1$

(b) Yes, one can check that this LFSR has maximal length.

6.2 In this question we consider a nonlinear combination generator, where we have a degree-n LFSR but the output at each time step is not s_0 but instead $g(s_0, \ldots, s_{n-1})$ for some nonlinear function g. Assume the feedback coefficients of the LFSR are known, but its initial state is not. Show that each of the following choices of g does not yield a good pseudorandom generator:

(a) $g(s_0, \ldots, s_{n-1}) = s_0 \wedge s_1$.

(b) $g(s_0, \ldots, s_{n-1}) = (s_0 \wedge s_1) \oplus s_2$.

Solution:

(a) The first bit of the output is 0 with probability 3/4. This easily leads to a distinguisher.

(b) Let $s_0^{(0)}, \ldots, s_{n-1}^{(0)}$ be the initial state, and denote by $y_1, \ldots, y_n$ the first n bits of output. If $s_1^{(0)} = 0$ (which occurs with probability 1/2) then $y_1 = (s_0^{(0)} \wedge s_1^{(0)}) \oplus s_2^{(0)} = s_2^{(0)}$. Furthermore, in the next step, we have $s_0^{(1)} = s_1^{(0)} = 0$ and so $y_2 = s_2^{(1)} = s_3^{(0)}$. At this point, the attacker knows $s_2^{(0)}$ and $s_3^{(0)}$ which equal $s_0^{(2)}$ and $s_1^{(2)}$, respectively. But $y_3 = (s_0^{(2)} \wedge s_1^{(2)}) \oplus s_2^{(2)}$, and so from y_3 the attacker can compute $s_2^{(2)} = s_4^{(0)}$. Continuing in the same way, the attacker can compute $s_2^{(0)}, \ldots, s_{n-1}^{(0)}$ from $y_1, \ldots, y_n$.

Using the above observation, the attacker works as follows. It assumes that $s_1^{(0)} = 0$ and computes values $s_2^{(0)}, \ldots, s_{n-1}^{(0)}$ based on this assumption. Then, for each of the two possible values of $s_0^{(0)} \in \{0,1\}$ and under the assumption that $s_1^{(0)} = 0$, it computes two possibilities for the next n output bits of the generator. If either of these is correct, the attacker outputs 1; otherwise, it outputs 0. An easy calculation shows that the attacker outputs 1 with probability at least $1/4$ when receiving the output of the generator, but outputs 1 with probability at most $2/2^n$ when receiving a uniform string. This is a distinguishing attack. In reality the attack is much more severe since with probability $1/2$, the attacker learns the entire initial seed.

6.3 Let F be a block cipher with n-bit key length and block length. Say there is a key-recovery attack on F that succeeds with probability 1 using n chosen plaintexts and minimal computational effort. Prove formally that F cannot be a pseudorandom permutation.

Solution: A distinguisher D runs the strategy for learning k based on its oracle queries. Then, D asks a new query x (that it did not query previously) and receives back the oracle response y. If $y \stackrel{?}{=} y'$, where $y' \stackrel{\text{def}}{=} F_k(x)$, then D outputs 1; otherwise it outputs 0.

By the assumption of the question

$$\Pr[D^{F_k(\cdot)}(1^n) = 1] = 1.$$

In contrast, we claim that when D has access to a random permutation:

$$\Pr[D^{f(\cdot)}(1^n) = 1] \leq \frac{1}{2^n - n}.$$

This holds because given k and x, the value $y' = F_k(x)$ is fixed. There are then two cases: either $y' \in \{f(x_1), \ldots, f(x_n)\}$, where $x_1, \ldots, x_n$ are the n queries made by D to its oracle in the first phase of its execution, or not. In the first case, $f(x)$ cannot possibly be equal to y' (since f is a permutation). In the second case, the probability that $f(x) = y'$ is exactly $1/(2^n - n)$. We conclude that

$$\left|\Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1]\right| \geq 1 - \frac{1}{2^n - n},$$

and F is therefore not a pseudorandom permutation.

6.4 In our attack on a one-round SPN, we considered a block length of 64 bits and 8 S-boxes that each take a 8-bit input. Repeat the analysis for the case of 16 S-boxes, each taking a 4-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.

Solution: For the case of 16 S-boxes each taking a 4-bit input, we have that for each guess of 4 bits of k_2 there is exactly one possibility for k_1. The attacker therefore prepares 16 lists, each of length 2^4. The overall cost is thus $16 \cdot 2^4 = 2^8$ (less than for the case studied in the text, as is to be expected).

Next for the case of a 128-bit block length and 16 S-boxes taking 8-bit inputs, the analysis is exactly the same as in the text except that now there are 16 lists instead of 8 lists. Thus, the complexity is $16 \cdot 2^8 = 2^{12}$ (instead of 2^{11} for the case in the text).

Observe that the size of the S-box is much more important than the size of the block.

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

Solution: This is equivalent to a *single*-round of a substitution-permutation network. To see this, note that r rounds of key-mixing are equivalent to one round of key-mixing (since XORing the r keys $k_1, \ldots, k_r$ is equivalent to XORing the single key $k_1 \oplus \cdots \oplus k_r$). Similarly, r rounds of substitution are equivalent to one round of substitution, and applying r mixing permutations sequentially is equivalent to applying a single mixing permutation.

6.6 In this question we assume a two-round SPN with 64-bit block length.

(a) Assume independent 64-bit sub-keys are used in each round, so the master key is 192 bits long. Show a key-recovery attack using much less than 2^{192} time.

(b) Assume the first and third sub-keys are equal, and the second sub-key is independent, so the master key is 128 bits long. Show a key-recovery attack using much less than 2^{128} time.

Solution: Since this question does not mention the size of the S-boxes, we describe attacks that work independently of the S-box size.

(a) If an attacker guesses the first two sub-keys k_1, k_2, then it can compute the first two rounds of the SPN and is left only with the sub-key mixing of k_3. Now, given a plaintext/ciphertext pair (x, y), it can carry out this computation forward from x using its guess of k_1, k_2. Then, k_3 will equal the XOR of the result of the first two rounds with y. Thus, the attacker can prepare a list of 2^{128} candidates for k_1, k_2, k_3. Since the size of the block is 64 bits, an incorrect key will map a plaintext x to its correct ciphertext y

with probability approximately 2^{-64}. Thus, an incorrect key will correctly map 3 given plaintexts to their given ciphertexts with probability only $(2^{-64})^3 = 2^{-192}$. Since there are 2^{128} candidates, only 1 will remain after 3 more plaintext/ciphertext pairs, except with probability 2^{-64}. The complexity of the attack is therefore $4 \cdot 2^{128} = 2^{130}$ time and 2^{128} memory.

(b) Denote the first and third sub-keys by k_1, and the second sub-key by k_2. Let (x, y) be a plaintext/ciphertext pair. For every possible guess of k_1, the attacker can compute the SPN forward from x to before the mixing of sub-key k_2 (this involves mixing k_1 which is "known" by the guess, S-box mixing and permutation). Likewise, the attacker can invert the SPN from y up to just after the mixing of sub-key k_2. Since the third sub-key is also k_1, this is also "known". Then, given the values before and after the mixing of k_2, we have that k_2 is just the XOR of these values. Thus, the attacker can prepare a list of 2^{64} candidates of (k_1, k_2) in time 2^{64}. These can then be checked on another two ciphertexts and with high probability only one will remain (as in the previous solution).

6.7 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.

Solution:

(a) Using the formula in Equation (6.3) we have that $L_1 = R_0$ and $R_1 = L_0 \oplus f_1(R_0) = L_0 \oplus 0^n = L_0$. Similarly, $L_2 = R_1 = L_0$ and $R_2 = L_1 = R_0$. Thus, if r is even then the output is (L_0, R_0), and if r is odd then the output is (R_0, L_0).

(b) Again, using the formula in Equation (6.3), we have that $L_1 = R_0$ and $R_1 = L_0 \oplus f_1(R_0) = L_0 \oplus R_0$. Then, $L_2 = R_1 = L_0 \oplus R_0$ and $R_2 = L_1 \oplus R_1 = R_0 \oplus L_0 \oplus R_0 = L_0$. Continuing for one more round, we have $L_3 = R_2 = L_0$ and $R_3 = L_2 \oplus R_2 = L_0 \oplus R_0 \oplus L_0 = R_0$, bringing us back to the beginning.

We conclude that if r is a multiple of 3, then the output equals (L_0, R_0). If $r = 1 \bmod 3$ then the output equals $(R_0, L_0 \oplus R_0)$, and if $r = 2 \bmod 3$ then the output equals $(L_0 \oplus R_0, L_0)$.

6.8 Let $\mathsf{Feistel}_{f_1,f_2}(\cdot)$ denote a two-round Feistel network using functions f_1 and f_2 (in that order). Show that if $\mathsf{Feistel}_{f_1,f_2}(L_0, R_0) = (L_2, R_2)$, then $\mathsf{Feistel}_{f_2,f_1}(R_2, L_2) = (R_0, L_0)$.

Solution: The formula for computing $\mathsf{Feistel}_{f_1,f_2}(L_0, R_0)$ is:

$$\begin{aligned} L_1 &= R_0 \\ R_1 &= L_0 \oplus f_1(R_0) \\ L_2 &= R_1 = L_0 \oplus f_1(R_0) \qquad (6.1) \\ R_2 &= L_1 \oplus f_2(R_1) = R_0 \oplus f_2(L_2). \qquad (6.2) \end{aligned}$$

Now, taking this pair (L_2, R_2), we compute $\mathsf{Feistel}_{f_2,f_1}(R_2, L_2)$. We denote $L'_0 = R_2$ and $R'_0 = L_2$ and the intermediate values computed by L'_1, R'_1 and L'_2, R'_2. Using the same formula as above, we have:

$$\begin{aligned} L'_1 &= R'_0 \\ R'_1 &= L'_0 \oplus f_2(R'_0) \\ L'_2 &= L'_0 \oplus f_2(R'_0) \\ R'_2 &= L'_1 \oplus f_1(R'_1) = R'_0 \oplus f_1(L'_2) \end{aligned}$$

(note that f_2 and f_1 have been reversed). Plugging in $L'_0 = R_2$ and $R'_0 = L_2$ we have that

$$L'_2 = R_2 \oplus f_2(L_2) \qquad \text{and} \qquad R'_2 = L_2 \oplus f_1(L'_2).$$

Using Equation (6.2), we have that $R_2 \oplus f_2(L_2) = R_0$ and thus $L'_2 = R_0$. Thus, $R'_2 = L_2 \oplus f_1(L'_2) = L_2 \oplus f_1(R_0)$. However, by Equation (6.1), we have that $L_2 \oplus f_1(R_0) = L_0$ and so $R'_2 = L_0$. We conclude that $\mathsf{Feistel}_{f_2,f_1}(R_2, L_2) = (R_0, L_0)$, as required.

6.9 For this exercise, rely on the description of DES given in this chapter, but use the fact that 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 of the Feistel network is (L_{16}, R_{16}), then the output of DES is (R_{16}, L_{16}).

(a) Show that the only difference between computation of DES_k and DES_k^{-1} is the order in which sub-keys are used. (Rely on the previous exercise.)

(b) Show that for $k = 0^{56}$ it holds that $DES_k(DES_k(x)) = x$.

(c) Find three other DES keys with the same property. These keys are known as *weak keys* for DES. (Note: the keys you find will differ from the actual weak keys of DES because of differences in our presentation.)

(d) Do these 4 weak keys represent a serious vulnerability in the use of triple-DES as a pseudorandom permutation? Explain.

Solution:

(a) Denote by $\mathsf{Feistel}'_{f_1,f_2}$ a *modified* two-round Feistel network using round functions f_1 and f_2 in that order, where the modification is that the two halves of the output of the final round are swapped. In the previous exercise we showed that if $y = \mathsf{Feistel}'_{f_1,f_2}(x)$, then $x = \mathsf{Feistel}'_{f_2,f_1}(y)$. (This is not true for the "regular" Feistel network, where the two halves of the output of the final round are *not* swapped.)

Now consider the computation $\mathsf{Feistel}'_{f_2,f_1}$ on input $(L'_0, R'_0) = (R_2, L_2)$. We call the results of the first round L'_1 and R'_1, and the results of the second round L'_2 and R'_2. Then $L'_1 = R'_0 = R_2 = L_0 \oplus f_1(R_0)$ and $R'_1 = L'_0 \oplus f_2(R'_0) = R_0$. Next, $L'_2 = R'_1 = L_1 = R_0$, and

$$R'_2 = L'_1 \oplus f_1(R'_1) = L_0 \oplus f_1(R_0) \oplus f_1(L_1) = L_0 \oplus f_1(R_0) \oplus f_1(R_0) = L_0.$$

Swapping the result, we obtain output (L_0, R_0) which is the original input we started with. The extension to the 16 rounds of DES follows by applying the above argument 8 times.

(b) When $k = 0^{56}$, all sub-keys in all rounds are exactly the same (every sub-key is an all-0 string). By the previous exercise, this means that decryption and encryption are identical (because all sub-keys are the same). Thus, $DES_{0^{56}}(DES_{0^{56}}(x)) = DES^{-1}_{0^{56}}(DES_{0^{56}}(x)) = x$. The use of such a key is a vulnerability because it means that an adversary can decrypt a challenge ciphertext using a chosen-plaintext attack.

(c) Three other keys with this property are 1^{56}, $0^{28}\|1^{28}$ and $1^{28}\|0^{28}$. The two latter keys are also weak keys because, as described in the book, the sub-keys are derived by taking subsets of the left and right halves of the key in each round. Thus, in all rounds we have the sub-key $0^{24}\|1^{24}$ (or $1^{24}\|0^{24}$) and so once again encryption and decryption are the same. (We remark that the actual weak keys in DES are different because the description in the book is a simplification of DES. Specifically, the two halves of the master key that are used for deriving the sub-keys are not the left and right halves of the master key.)

(d) The existence of these keys does not constitute any significant problem for DES. The reason for this is that the probability that one of these keys is chosen is $4/2^{56} = 2^{-54}$.

6.10 Show that DES has the property that $DES_k(x) = \overline{DES_{\bar{k}}(\bar{x})}$ for every key k and input x (where $\bar{z}$ denotes the bitwise complement of z). (This is called the *complementarity property* of DES.) Does this represent a serious vulnerability in the use of triple-DES as a pseudorandom permutation? Explain.

Solution: Let $\hat{f}$ be the DES mangler function. We first claim that for every key k and every input x, it holds that $\hat{f}(k,x) = \hat{f}(\bar{k},\bar{x})$. To see this, notice that for input x and key k, the input to the S-boxes equals $E(x) \oplus k$ where E is the expansion function. Since E simply duplicates half of the bits of its input, we have that $E(\bar{x})$ equals $\overline{E(x)}$. Therefore $E(\bar{x}) \oplus \bar{k} = \overline{E(x)} \oplus \bar{k} = E(x) \oplus k$. Since the input to the S-boxes is the same, the output from the S-boxes is also the same. Applying the mixing permutation does not change the fact that the outputs are equal. We conclude that $\hat{f}(k,x) = \hat{f}(\bar{k},\bar{x})$.

Next look at the entire Feistel structure. For input L_0, R_0 and key k, the values after the first round are $L_1 = R_0$ and $R_1 = L_0 \oplus \hat{f}(k_1, R_0)$. By the above, for input $\overline{L}_0, \overline{R}_0$ and key $\bar{k}$ the values after the first round are $L'_1 = \overline{R}_0 = \overline{L}_1$ and

$$R'_1 = \overline{L}_0 \oplus \hat{f}(\bar{k}_1, \overline{R}_0) = \overline{L}_0 \oplus \hat{f}(k_1, R_0) = \overline{L_0 \oplus \hat{f}(k_1, R_0)} = \overline{R}_1.$$

Since the above continues to hold after each round of the Feistel network is applied, we see that $DES_k(x) = \overline{DES_{\bar{k}}(\bar{x})}$.

6.11 Describe attacks on the following modifications to DES:

(a) Each sub-key is 32 bits long, and the round function simply XORs the sub-key with the input to the round (i.e., $\hat{f}(k,R) = k_i \oplus R$). For this question, the key schedule is unimportant and you can treat the sub-keys 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 in $\ll 2^{48}$ time.

Solution:

(a) Let us follow a few rounds of the computation, where k_i denotes the round-i sub-key. Starting with (L_0, R_0) we get $L_1 = R_0$ and

$$R_1 = L_0 \oplus \hat{f}(k_1, R_0) = L_0 \oplus R_0 \oplus k_1.$$

Then $L_2 = L_0 \oplus R_0 \oplus k_1$ and

$$R_2 = R_0 \oplus L_0 \oplus R_0 \oplus k_1 \oplus k_2 = L_0 \oplus k_1 \oplus k_2.$$

Continuing, we see that the final output is a simple linear function of L_0, R_0, and the sub-keys. It can thus be distinguished easily from a random permutation.

(b) By the result of Exercise 6.9, we know that when all sub-keys are the same encryption and decryption are identical. It is therefore trivial to distinguish this cipher from a random permutation.

6.12 (This exercise relies on Exercise 6.9.) Our goal is to show that for any weak key k of DES, it is easy to find an input x such that $DES_k(x) = x$.

(a) Assume we evaluate DES_k on input (L_0, R_0), and the output after 8 rounds of the Feistel network is (L_8, R_8) with $L_8 = R_8$. Show that the output of $DES_k(L_0, R_0)$ is (L_0, R_0). (Recall from Exercise 6.9 that DES swaps the two halves of the 16th round of the Feistel network before outputting the result.)

(b) Show how to find an input (L_0, R_0) with the property in part (a).

Solution:

(a) By Exercise 6.9(a), the only difference between DES_k and DES_k^{-1} is the order in which sub-keys are used. When k is a DES weak key, it follows that all round functions are the same. For this reason, $DES_k(DES_k(x)) = x$ as shown in Exercise 6.9(b). However, the argument used to show these facts in Exercise 6.9 is the same for just 8 rounds. Thus, denoting $\widetilde{DES}$ to be DES reduced to 8 rounds (including swapping the two halves before outputting the result), it follows that for every x: $\widetilde{DES}_k(\widetilde{DES}_k(x)) = x$. Equivalently, it follows that for every x: $\widetilde{DES}_k(x) = \widetilde{DES}_k^{-1}(x)$. Now, the only difference between $\widetilde{DES}_k(\widetilde{DES}_k(x))$ and $DES_k(x)$ is that in $\widetilde{DES}_k(\widetilde{DES}_k(x))$ the halves are swapped after 8 rounds. If an input x is such that (L_8, R_8) fulfills that $L_8 = R_8$, then this implies that for such an x we have that $\widetilde{DES}_k(\widetilde{DES}_k(x)) = DES_k(x)$. However, we have already seen that $\widetilde{DES}_k(\widetilde{DES}_k(x)) = x$. Thus, we concludes that $DES_k(x) = x$ (for k a weak key and for x with the property that $L_8 = R_8$).

(b) In order to find such an x, take any arbitrary 32-bit value z and set $L_8 = R_8 = z$. Then, for a weak key k, compute $x = (L_0, R_0) = \widetilde{DES}_k(L_8, R_8)$. As we have already shown, it also holds that $x = \widetilde{DES}_k^{-1}(L_8, R_8)$. Thus, $\widetilde{DES}_k(\widetilde{DES}_k(x)) = DES_k(x) = DES_k(x) = x$.

6.13 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 $F'_{k_1,k_2}(x) \stackrel{\text{def}}{=} F_{k_1}(F_{k_2}^{-1}(F_{k_1}(x)))$.

(a) Assume that given a pair (m_1, m_2) it is possible to find in *constant* time all keys k_2 such that $m_2 = F_{k_2}^{-1}(m_1)$. 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 preprocessing step taking 2^n

time it is possible, given m_2, to find in (essentially) constant time all keys k_2 such that $m_2 = F_{k_2}^{-1}(0^n)$.

(c) Assume k_1 is known and that the pre-processing step above has already been run. Show how to use the value $y = F'_{k_1,k_2}(x)$ for a single *chosen* plaintext x to determine k_2 in constant time.

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

Solution:

(a) Let $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ be three plaintext/ciphertext pairs that were computed using F'_{k_1,k_2} (for unknown keys k_1, k_2). Initialize a set $K = \emptyset$. Then for every key $\hat{k}_1 \in \{0,1\}^n$ do:

i. Compute $m_1 = F_{\hat{k}_1}(x_1)$ and $m_2 = F_{\hat{k}_1}^{-1}(y_1)$.

ii. Find all keys $\hat{k}_2$ such that $m_2 = F_{\hat{k}_2}^{-1}(m_1)$. Call this set S. For each $\hat{k}_2 \in S$ do:

If $y_2 \stackrel{?}{=} F'_{\hat{k}_1,\hat{k}_2}(x_2)$ and $y_3 \stackrel{?}{=} F'_{\hat{k}_1,\hat{k}_2}(x_3)$, then add $(\hat{k}_1, \hat{k}_2)$ to K.

Let us first analyze the running time of the attack. For a given $\hat{k}_1$, the running time of the loop above is linear in $|S|$ (under the assumption that we can find all keys $\hat{k}_2$ such that $m_2 = F_{\hat{k}_2}^{-1}(m_1)$ in constant time). Furthermore, for a given (m_1, m_2) we expect S to contain only a single key $\hat{k}_2$. Since the algorithm tries all possible values for $\hat{k}_1$, the total running time is roughly 2^n.

It is easy to see that the correct key (k_1, k_2) will be in K. We argue that with high probability this is the *only* key in K. Let S_i be the set S in the ith iteration of the algorithm. As we have said above, we expect that each set S_i contains only a single key $\hat{k}_2$. Summing over all iterations of the algorithm, we expect a total of 2^n keys in $S* \stackrel{\text{def}}{=} \cup_i S_i$. But the probability that an incorrect key in S^* gets put into K is roughly 2^{-2n} (because this is roughly the probability that an incorrect key $(\hat{k}_1, \hat{k}_2)$ satisfies $y_2 = F'_{\hat{k}_1,\hat{k}_2}(x_2)$ and $y_3 = F'_{\hat{k}_1,\hat{k}_2}(x_3)$). Taking a union bound over all incorrect keys in S^*, we expect an incorrect key to get put into K only with probability 2^{-n}.

(b) Using 2^n preprocessing time, we can compute all pairs (m_2, k_2) such that $m_2 = F_{k_2}^{-1}(0^n)$ and put these in a list, sorted by m_2. Then given any m_2, we can find in essentially constant time all keys k_2 such that (m_2, k_2) is in the list. (Note that, in expectation [and treating F as a random function], for any given m_2 there is only one k_2 such that $m_2 = F_{k_2}^{-1}(0^n)$.)

(c) If k_1 is already known, then we can compute $x = F_{k_1}^{-1}(0^n)$ and ask for an encryption of x; let the result be y. Then, given y we can compute $m_2 = F_{k_1}^{-1}(y)$. By the definition of F' we have that $m_2 = F_{k_2}^{-1}(0^n)$. This is because $y = F'_{k_1,k_2}(x) = F_{k_1}(F_{k_2}^{-1}(F_{k_1}(x))) = F_{k_1}(F_{k_2}^{-1}(0^n))$ and so $m_2 = F_{k_1}^{-1}(y) = F_{k_2}^{-1}(0^n)$. We can therefore use the method of the previous step to find k_2 in constant time.

(d) The full attack works as follows. First run the preprocessing step as described in part (b). Then for every $k_1 \in \{0,1\}^n$, run the attack described in part (c). This will result in a set of 2^n possible values for the key (one for each value of k_1). Using the plaintext/ciphertext pairs we already have, we can narrow this down to the correct key.

For each $k_1 \in \{0,1\}^n$ we request the encryption of only one chosen plaintext, for a total of 2^n chosen plaintexts. Each step of the attack takes time 2^n. We remark that 2^n memory is also required.

6.14 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}.

Solution: The solution to this exercise is to run a meet-in-the-middle attack on DES as follows. Use an encryption oracle to obtain plaintext/ciphertext pairs $(x_1, y_1), (x_2, y_2), (x_3, y_3)$. Then, prepare a list of values (z_1, k_1) where $k_1 \in \{0,1\}^{28}$ and z_1 is the result of 8 rounds of DES encryption of x_1 using k_1. Likewise, prepare another list of values (z_2, k_2) where $k_2 \in \{0,1\}^{28}$ and z_2 is the result of 8 rounds of DES *decryption* of y_1 using k_2. As in the meet-in-the-middle attack on 2DES, it is possible to find pairs k_1, k_2 that constitute candidate keys and then verify them using (x_2, y_2) and (x_3, y_3).

6.15 Let $f : \{0,1\}^m \times \{0,1\}^\ell \to \{0,1\}^\ell$ and $g : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^\ell$ be block ciphers with $m > n$, and define $F_{k_1,k_2}(x) = f_{k_1}(g_{k_2}(x))$. Show a key-recovery attack on F using time $\mathcal{O}(n \cdot 2^m)$ and space $\mathcal{O}(\ell \cdot 2^n)$.

Solution: This is a generalization of the meet-in-the-middle attack. Assume $\ell = \mathcal{O}(n)$. Briefly, given a plaintext/ciphertext pair (x, y), construct a list of size $(\ell + n) \cdot 2^n = \mathcal{O}(\ell \cdot 2^n)$ of all the pairs (k_2, z) where $k_2 \in \{0,1\}^n$ and $z = g_{k_2}(x)$, and sort this list by the second element. Next, for every $k_1 \in \{0,1\}^m$, search for $f_{k_1}^{-1}(y)$ in this list. If it exists, and k_2 is the first element in the pair, then (k_1, k_2) is a potential candidate for the key. By using a few plaintext/ciphertext pairs (with the exact number depending on ℓ), a single candidate key remains. The time to carry out the attack is dominated by the $\mathcal{O}(2^m)$

time to enumerate over all k_1 (technically, if binary search is used, this requires time $\mathcal{O}(n \cdot 2^m)$).

6.16 Define $DESY_{k,k'}(x) = DES_k(x \oplus k')$. The key length of $DESY$ is 120 bits. Show a key-recovery attack on $DESY$ taking time and space $\approx 2^{64}$.

Solution: The observation is that an optimized meet-in-the-middle attack can be used here. Let $\{(x_i, y_i)\}$ be a set of plaintext/ciphertext pairs. For every $k \in \{0,1\}^{56}$ compute $z_1 = DES_k^{-1}(y_1)$. Since $y_1 = DES_k(x_1 \oplus k')$ we have that if k is the correct first half of the key, then $k' = x_1 \oplus z_1$. That is, for every guess of k, we obtain a single possibility for k'. The other plaintext/ciphertext pairs can then be used to check whether this candidate key (k, k') is correct.

This attack requires time 2^{56} and constant space.

6.17 Choose random S-boxes and mixing permutations for SPNs of different sizes, and develop differential attacks against them. We recommend trying five-round SPNs with 16-bit and 24-bit block lengths, using S-boxes with 4-bit input/output. Write code to compute the differential tables, and to carry out the attack.

No solution given.

6.18 Implement the time/space tradeoff for 40-bit DES (i.e., fix the first 16 bits of the key of DES to 0). Calculate the time and memory needed, and empirically estimate the probability of success. Experimentally verify the increase in success probability as the number of tables is increased. (Warning: this is a big project!)

No solution given.

6.19 For each of the following constructions of a compression function h from a block cipher F, either show an attack or prove collision resistance in the ideal-cipher model:

(a) $h(k, x) = F_k(x)$.

(b) $h(k, x) = F_k(x) \oplus k \oplus x$.

(c) $h(k, x) = F_k(x) \oplus k$.

Solution: In each case we try the proof approach used to prove Theorem 6.5.

(a) There are two ways the attacker can learn the value of h on some input. If it queries $F_k(x)$ and gets response y, then it learns that $h(k, x) = y$. In this case y is (essentially) uniform. The attacker can also query $F_k^{-1}(y)$ to get response x; then it learns that $h(k, x) = y$. Observe that here the attacker *has complete control over* y, and so the proof approach does not work. In fact, this suggests an attack:

pick arbitrary y and distinct k, k'. Compute $x := F_k^{-1}(y)$ and $x' := F_{k'}^{-1}(y)$. Note that $h(k,x) = h(k',x') = y$ and so (k,x) and (k',x') are a collision.

(b) There are two ways the attacker can learn the value of h on some input. If it queries $F_k(x)$ and gets response y, then it learns that $h(k,x) = y \oplus k \oplus x$. Note that y is (essentially) uniform and independent of k, x, and so $h(k,x)$ is (essentially) uniform as well. The attacker can also query $F_k^{-1}(y)$ to get response x, in which case it learns that $h(k,x) = y \oplus k \oplus x$. Here, again, x is (essentially) uniform, and so $h(K,x)$ is (essentially) uniform as well. Thus, every h-output learned by the attacker is uniform, and the proof approach used to prove Theorem 6.5 works here as well to show that h is collision-resistant.

(c) There are two ways the attacker can learn the value of h on some input. If it queries $F_k(x)$ and gets response y, then it learns that $h(k,x) = y \oplus k$. In this case that y is (essentially) uniform, and so $h(k,x)$ is (essentially) uniform as well. The attacker can also query $F_k^{-1}(y)$ to get response x, in which case it learns that $h(k,x) = y \oplus k$. Here the attacker *has complete control over k and y*, and so the proof approach does not work. In fact, this suggests an attack: pick (distinct) k, k', y, y' such that $k \oplus y = k' \oplus y'$, then compute $x := F_k^{-1}(y)$ and $x' := F_{k'}^{-1}(y')$. Then $h(k,x) = h(k',x')$ and so (k,x) and (k',x') are a collision.

6.20 Consider using DES to construct a compression function in the following way: Define $h : \{0,1\}^{112} \to \{0,1\}^{64}$ as $h(x_1,x_2) \stackrel{\text{def}}{=} DES_{x_1}(DES_{x_2}(0^{64}))$ where $|x_1| = |x_2| = 56$.

(a) Write down an explicit collision in h.

(b) Show how to find a preimage of an arbitrary 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 preimage attack that runs in roughly 2^{32} time and succeeds with high probability.

Solution:

(a) The pair $x = 1^{112}$ and $x' = 0^{112}$ constitutes a collision. This is because in both cases encryption equals decryption and so

$$h(x) = DES_{1^{56}}(DES_{1^{56}}(0^{64})) = DES^{-1}_{1^{56}}(DES_{1^{56}}(0^{64})) = 0^{64},$$

and similarly $h(0^{112}) = 0^{64}$.

(b) Use the same meet-in-the-middle attack as for double encryption, viewing $(0^{64}, y)$ as a plaintext/ciphertext pair encrypted using some unknown "key" $x = x_1, x_2$. (Any key that is found will be a preimage for the given value y.)

(c) The key observation here is that we can run a modified version of the meet-in-the-middle attack from part (b), using the fact that there is no "right" answer. (That is, we are not searching for some fixed key x_1, x_2 such that $DES_{x_1}(DES_{x_2}(0^{64})) = y$; rather, we are searching for *any* key satisfying this constraint.) We modify the attack as follows: choose 2^{28} *random* values of x_2, and for each such value compute $z := DES_{x_2}(0^{64})$ and store (z, x_2) in a list L. Then choose 2^{28} random values of x_1, and for each such value compute $z := DES^{-1}_{x_1}(y)$ and store (z, x_1) in a list L'. Roughly speaking, we can view all the computed z-values as being "random"; since each z-value is 64-bits long, we therefore expect a "match" (i.e., entries $(z_2, x_2) \in L$ and $(z_1, x_1) \in L'$ such that $z_2 = z_1$) to occur with constant probability. Any match immediately yields a pre-image for the given value y.

6.21 Let F be a block cipher for which it is easy to find fixed points for some key: namely, there is a key k for which it is easy to find inputs x for which $F_k(x) = x$. Find a collision in the Davies–Meyer construction when applied to F. (Consider this in light of Exercise 6.12.)

Solution: Davies-Meyer is defined by $h(k, x) = F_k(x) \oplus x$. Now, for any k, x such that $F_k(x) = x$, it holds that $h(k, x) = x \oplus x = 0$. Thus, if for some key k it is easy to find two distinct inputs x, x' such that $F_k(x) = x$ and $F_k(x') = x'$ then it follows that $h(k, x) = 0 = h(k, x')$, which is a collision. (Likewise, if it is easy to find $k' \neq k$.)

As we have seen in Exercise 6.12, in DES it is easy to find many such values x for any of the four DES weak keys. Thus, this implies that Davies-Meyer is *not* collision resistant when used with DES. Observe that this does not contradict Theorem 6.5 since that assumes the F is an ideal cipher. This should serve as a strong warning regarding the ideal-cipher model.

Chapter 7

Theoretical Constructions of Symmetric-Key Primitives – Solutions

7.1 Prove that if there exists a one-way function, then there exists a one-way function f such that $f(0^n) = 0^n$ for every n. Note that for infinitely many values y, it is easy to compute $f^{-1}(y)$. Why does this not contradict one-wayness?

Solution: We provide a painfully detailed proof. Let f be one-way function (that exists by the assumption) and define $g(x) = f(x)$ for every $x \neq 0^{|x|}$ and $g(0^n) = 0^n$ for every n. Clearly, g fulfills the requirements. It remains to prove that it is one-way. First, g is efficiently computable. Second, assume by contradiction that there exists a probabilistic polynomial-time algorithm $\mathcal{A}$ and set

$$\varepsilon(n) \stackrel{\text{def}}{=} \Pr[\mathsf{Invert}_{\mathcal{A},g}(n) = 1].$$

We begin by analyzing the probability that $\mathcal{A}$ succeeds in inverting g on non-zero inputs (the value x referred to below is the x chosen in the Invert experiment):

$$\begin{aligned}\Pr[\mathsf{Invert}_{\mathcal{A},g}(n) = 1] &= \Pr[\mathsf{Invert}_{\mathcal{A},g}(n) = 1 \mid x \neq 0^n] \cdot \Pr[x \neq 0^n] \\ &\quad + \Pr[\mathsf{Invert}_{\mathcal{A},g}(n) = 1 \mid x = 0^n] \cdot \Pr[x = 0^n] \\ &\leq \Pr[\mathsf{Invert}_{\mathcal{A},g}(n) = 1 \mid x \neq 0^n] + \Pr[x = 0^n] \\ &= \Pr[\mathsf{Invert}_{\mathcal{A},g}(n) = 1 \mid x \neq 0^n] + \frac{1}{2^n}.\end{aligned}$$

Therefore, $\Pr[\mathsf{Invert}_{\mathcal{A},g}(n) = 1 \mid x \neq 0^n] \geq \varepsilon(n) - \frac{1}{2^n}$. We now construct $\mathcal{B}$ that inverts f as follows. Upon receiving an input y, algorithm $\mathcal{B}$

invokes $\mathcal{A}$ and returns whatever $\mathcal{A}$ outputs. We analyze $\mathcal{B}$'s success:

$$\begin{aligned}
\Pr[\mathsf{Invert}_{\mathcal{B},f}(n)=1] &= \Pr[\mathsf{Invert}_{\mathcal{B},f}(n)=1 \mid x\neq 0^n]\cdot \Pr[x\neq 0^n] \\
&\quad + \Pr[\mathsf{Invert}_{\mathcal{B},f}(n)=1 \mid x= 0^n]\cdot \Pr[x= 0^n] \\
&\geq \Pr[\mathsf{Invert}_{\mathcal{B},f}(n)=1 \mid x\neq 0^n]\cdot \Pr[x\neq 0^n] \\
&= \Pr[\mathsf{Invert}_{\mathcal{B},f}(n)=1 \mid x\neq 0^n]\cdot \left(1-\frac{1}{2^n}\right) \\
&= \Pr[\mathsf{Invert}_{\mathcal{B},g}(n)=1 \mid x\neq 0^n]\cdot \left(1-\frac{1}{2^n}\right) \\
&\geq \left(\varepsilon(n)-\frac{1}{2^n}\right)\cdot\left(1-\frac{1}{2^n}\right) = \varepsilon(n)-\mathsf{negl}(n)
\end{aligned}$$

for some negligible function negl. Since f is one-way, we must have that ε is negligible. Since this holds for arbitrary PPT $\mathcal{A}$, we conclude that g is one-way as well.

This does not contradict one-wayness since although there are infinitely many values that can be easily inverted, the probability of receiving such a value is negligible (specifically, 2^{-n}).

7.2 Prove that if f is a one-way function, then the function g defined by $g(x_1,x_2)\stackrel{\text{def}}{=}(f(x_1),x_2)$, where $|x_1|=|x_2|$, is also a one-way function. Observe that g reveals half of its input, but is nevertheless one-way.

Solution: Let $\mathcal{A}$ be a probabilistic polynomial-time adversary and let

$$\varepsilon(n)\stackrel{\text{def}}{=}\Pr[\mathsf{Invert}_{\mathcal{A},g}(n)=1].$$

We construct an adversary $\mathcal{A}'$ who inverts f as follows: $\mathcal{A}'$ receives for input 1^n and some y, chooses $x_2\leftarrow\{0,1\}^n$ and invokes $\mathcal{A}$ upon 1^{2n} and $y\|x_2$. Then, $\mathcal{A}'$ outputs the first half of whatever $\mathcal{A}$ outputs. Now, if $\mathcal{A}$ successfully inverts $y\|x_2$ under g then it returns $x\|x_2$ such that $f(x)=y$. Thus, $\mathcal{A}'$ successfully inverts y under f. This implies that

$$\Pr[\mathsf{Invert}_{\mathcal{A}',f}(n)=1]=\varepsilon(n),$$

and so ε must be negligible. We conclude that g is a one-way function.

7.3 Prove that if there exists a one-way function, then there exists a length-preserving one-way function.

Solution: We provide only a proof sketch here. We first show how to construct a length-regular one-way function. (A function is length regular if $|x|=|y|\Rightarrow|f(x)|=|f(y)|$.) Let f be a one-way function. Since f is one-way, it is efficiently computable. Thus, there exists a polynomial $p(\cdot)$ such that for every x, $|f(x)|\leq p(|x|)$. Assume $p(n)\geq n$ (if not, just set $p(n)=n$). Define $f'(x)=f(x)10^{p(|x|)-|f(x)|}$. Clearly,

f' is length-regular. This is due to the fact that the output-length of $f'(x)$ for every x of length n is exactly $p(n)+1$. We now prove that f' is one-way.

Assume by contradiction that there exists a probabilistic polynomial-time adversary $\mathcal{A}'$ and a non-negligible function $\varepsilon(\cdot)$ such that

$$\Pr[\mathsf{Invert}_{\mathcal{A}',f'}(n)=1]=\varepsilon(n).$$

We construct an adversary $\mathcal{A}$ who inverts f as follows: $\mathcal{A}$ receives 1^n and y for input, invokes $\mathcal{A}'$ on 1^n and $y10^{p(|x|)-|f(x)|}$ and returns whatever $\mathcal{A}'$ returns (we can assume that $\mathcal{A}$ knows the polynomial $p(\cdot)$). The important point to note here is that due to the fact that "padding" includes a single one followed by zeroes, the output of $f'(x)$ uniquely defines the length of the portion that is $f(x)$. Therefore, the set of pre-images of $f'(x)$ equals the set of pre-images of $f(x)$. This implies that if $\mathcal{A}'$ inverts f', then $\mathcal{A}$ will have inverted f. By our assumption, $\mathcal{A}'$ inverts with success probability of $\varepsilon(n)$; therefore the same is true of $\mathcal{A}$. This contradicts the assumed one-wayness of f.

We next construct a length-preserving one-way function g. Define $q(n)=|f'(0^n)|$ where f' is as above; since f' is length-regular, this means that the output length of f' on any n-bit input is exactly $q(n)$. Note also that by construction of f', we have $q(n)\geq n$ for all n. Now compute $g(x)$ as follows:

(a) Set $n:=|x|$.

(b) Find the largest n' such that $q(n')<n$. (Note that $n'\neq n$.) Let x' denote the n'-bit prefix of x.

(c) Output $f'(x')10^{n-q(n')-1}$.

We omit the (tedious, but straightforward) proof that g is one-way.

7.4 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 $(\mathsf{Gen},\mathsf{Samp},H)$ is one-way (cf. Definition 7.3), where Samp is the trivial algorithm that samples a uniform string of length $2n$.

Solution: Let $\Pi=(\mathsf{Gen},\mathsf{Samp},H)$ and fix an adversary $\mathcal{A}$. Consider the following algorithm C attempting to find a collision in Π.

> **Algorithm** C:
> The algorithm is given s and tries to find a collision in H^s.
>
> - Choose $x\leftarrow\{0,1\}^{2n}$ (recall that 1^n is implicit is s).
> - Compute $y:=H^s(x)$ and run $\mathcal{A}(s,y)$ to obtain x'.
> - Output (x,x').

Before we analyze C, we argue about the behavior of $\mathcal{A}$. For any fixed key s output by $\mathsf{Gen}(1^n)$, let

$$\mathsf{light}_s \stackrel{\text{def}}{=} \{x \mid H^s(x) \text{ has at most } 2^{n/2} \text{ pre-images under } H^s\}.$$

Since the output of H^s is an n-bit string, the maximum possible size of light_s is $2^n \cdot 2^{n/2} = 2^{3n/4}$. So the probability that a random $x \in \{0,1\}^{2n}$ is in light_s is at most $2^{3n/4}/2^{2n} = 2^{-n/4}$.

We have

$$\begin{aligned}\Pr[\mathsf{Invert}_{\mathcal{A},\Pi}(n) = 1] &= \Pr[H^s(x') = y] \\ &\leq \Pr[H^s(x') = y \wedge x \notin \mathsf{light}_s] + \Pr[x \in \mathsf{light}_s] \\ &\leq \Pr[H^s(x') = y \wedge x \notin \mathsf{light}_s] + 2^{-n/4}, \qquad (7.1)\end{aligned}$$

where in all cases the probability is over the experiment in which s is output by $\mathsf{Gen}(1^n)$; x is then chosen at random from $\{0,1\}^{2n}$ and y is computed as $y := H^s(x)$; and then $\mathcal{A}(s,y)$ is run to obtain x'. (This is exactly the inverting experiment considered in Definition 7.70.)

Returning to an analysis of C, we have

$$\Pr\left[\mathsf{Hash\text{-}coll}_{C,\Pi}(n) = 1\right] \geq \Pr[H^s(x') = y \wedge x' \neq x \wedge x \notin \mathsf{light}_s],$$

where the probability is over the exact same experiment as above (this follows from the construction of C). So

$$\begin{aligned}&\Pr\left[\mathsf{Hash\text{-}coll}_{C,\Pi}(n) = 1\right] \\ &\geq \Pr[H^s(x') = y \wedge x' \neq x \wedge x \notin \mathsf{light}_s] \\ &= \Pr[H^s(x') = y \wedge x \notin \mathsf{light}_s] \cdot (1 - \Pr[x' = x \mid H^s(x') = y \wedge x \notin \mathsf{light}_s]) \\ &\geq \Pr[H^s(x') = y \wedge x \notin \mathsf{light}_s] \cdot \left(1 - 2^{-n/2}\right),\end{aligned}$$

using the fact that for $y = H^s(x)$ with $x \notin \mathsf{light}_s$, there are at least $2^{n/2}$ pre-images of y under H^s, all of which are equally likely to have been chosen (from the point of view of $\mathcal{A}$). Combined with Equation (7.1) and the assumption that (Gen, H) is collision resistant (along with the fact that $2^{-n/2}$ and $2^{-n/4}$ are negligible), this shows that there exists a negligible function negl such that $\Pr[\mathsf{Invert}_{\mathcal{A},\Pi}(n) = 1] \leq \mathsf{negl}(n)$.

7.5 Let F be a (length-preserving) pseudorandom permutation.

(a) Show that the function $f(x,y) = F_x(y)$ is not one-way.

(b) Show that the function $f(y) = F_{0^n}(y)$ (where $n = |y|$) is not one-way.

(c) Prove that the function $f(x) = F_x(0^n)$ (where $n = |x|$) is one-way.

7.6 Let f be a length-preserving one-way *function*, and let hc be a hard-core predicate of f. Define G as $G(x) = f(x)\|\mathsf{hc}(x)$. Is G necessarily a pseudorandom generator? Prove your answer.

Solution: No, G is not necessarily a pseudorandom generator. For example, let g be a one-way function and define $f(x_1, x_2) = (g(x_1), 0^{|x_2|})$ where $|x_1| = |x_2|$. The fact that f is a one-way function can be proved in an almost identical way as in the solution to the previous exercise. Using this f in the construction of the exercise, we obtain a G which is clearly not pseudorandom (since on input a seed of length n, the output of G contains a sequence of $n/2$ consecutive 0s).

7.7 Prove that there exist one-way functions if and only if there exist families of one-way functions. Discuss why your proof does not carry over to the case of one-way permutations.

Solution: It is trivial to see that the existence of any one-way function f implies the existence of a family of one-way functions: just take Gen to be the trivial algorithm that outputs $I = 1^n$ on input 1^n, and let Samp be the trivial algorithm that returns a random n-bit string on input $I = 1^n$; the family $(\mathsf{Gen}, \mathsf{Samp}, f)$ is then one-way. We therefore focus on the more interesting direction.

Let $\Pi = (\mathsf{Gen}, \mathsf{Samp}, f)$ be a family of one-way functions. Let $p_G(\cdot)$ denote the (polynomial) running time of Gen and let $p_S(\cdot)$ denote the (polynomial) running time of Samp. Note that $p_G(n)$ and $p_S(n)$ constitute an upper bound on the number of random coins used by Gen and Samp, respectively, on security parameter 1^n. Furthermore, $p_S(n)$ also constitutes an upper bound on the length of values in the range of D_I for I output by $\mathsf{Gen}(1^n)$.

We define a single one-way function g as follows. Upon input x of length n, let k be maximal such that $p_G(k) + p_S(k) \leq n$; then divide x into three parts x_1, x_2, x_3 such that $|x_1| = p_G(k)$ and $|x_2| = p_S(k)$. The function g is defined as (we ignore x_3):

$$g(x) = g(x_1, x_2) = \left(\mathsf{Gen}(1^k; x_1), f\left(\mathsf{Gen}(1^k; x_1), \mathsf{Samp}(\mathsf{Gen}(1^k; x_1); x_2)\right)\right),$$

where $\mathsf{Gen}(1^k; x_1)$ denotes the output of algorithm Gen upon input 1^k and *random tape* x_1, and $\mathsf{Samp}(\mathsf{Gen}(1^k; x_1); x_2)$ denotes the output of Samp upon input $\mathsf{Gen}(1^k; x_1)$ and *random tape* x_2. Note that $\mathsf{Gen}(1^k; x_1)$ is included in the output to force any inverting algorithm to find a pre-image of $f\left(\mathsf{Gen}(1^k; x_1), \mathsf{Samp}(\mathsf{Gen}(1^k; x_1); x_2)\right)$ under $\mathsf{Gen}(1^k; x_1)$ and not under some other function in the family.

It remains to prove that g is a one-way function. First, it is clearly an efficiently computable function. Second, assume by contradiction that it can be inverted with probability $\varepsilon(n)$ for some non-negligible function $\varepsilon(\cdot)$. By the construction, this implies that the family $(\mathsf{Gen}, \mathsf{Samp}, f)$

can be inverted with probability $\varepsilon(n)$. This holds because for every n there exists a k that results from the above derivation via p_G and p_S, and g can be inverted for these k's with probability $\varepsilon(n)$. Now, let c be a constant such that $n = k^c$ (such a constant exists because the lengths of k and n are polynomially related). Thus, we have that the family $(\mathsf{Gen}, \mathsf{Samp}, f)$ can be inverted with probability $\varepsilon(k^c)$. If ε is non-negligible in k^c then it is also non-negligible in k. This contradicts the one-wayness of $(\mathsf{Gen}, \mathsf{Samp}, f)$.

For the case of permutations, any one-way permutation implies a family of one-way permutations using the same trivial construction given earlier. On the other hand, the transformation of a one-way function family to a one-way function given above does not work for permutations since the g that results is not necessarily a permutation even if f is.

7.8 Let f be a one-way function. Is $g(x) \stackrel{\text{def}}{=} f(f(x))$ necessarily a one-way function? What about $g'(x) \stackrel{\text{def}}{=} (f(x), f(f(x)))$? Assume that g' is encoded so that its output can be uniquely parsed into $f(x)$ and $f(f(x))$. Prove your answers.

Solution: We give a counterexample showing that taking $g(x) = f(f(x))$ for an arbitrary one-way function f does not necessarily make g a one-way function. Let h be any length-preserving one-way function and define f as follows: If $x_{n/2+1} \cdots x_n = 0^{n/2}$, then $f(x) = 0^{|x|}$; else, $f(x) = h(x_1 \cdots x_{n/2})0^{n/2}$. We show that f is one-way, but that g (defined as above) is not.

To show that f is one-way, let $\mathcal{A}$ be a probabilistic polynomial-time adversary with

$$\varepsilon(n) \stackrel{\text{def}}{=} \Pr[\mathsf{Invert}_{\mathcal{A},f}(1^n) = 1].$$

We construct an adversary $\mathcal{A}'$ that inverts h as follows. $\mathcal{A}'$ receives $1^{n/2}$ and a value $y \in \{0,1\}^{n/2}$ and attempts to find a value $x \in h^{-1}(y)$. The adversary $\mathcal{A}'$ just invokes $\mathcal{A}$ upon 1^n and $y0^{n/2}$ and outputs the first $n/2$ bits of $\mathcal{A}$'s output. We now analyze the success probability of $\mathcal{A}'$. First, denote $S_n = \{x \mid x_{n/2+1} \cdots x_n = 0^{n/2}\}$ and note that $\Pr[x \in S_n] = 2^{-n/2}$. Next, note that

$$\Pr[\mathsf{Invert}_{\mathcal{A}',h}(n) = 1] \geq \Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 1 \mid x \notin S_n].$$

Now,

$$\begin{aligned}
\Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 1] &= \Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 1 \mid x \in S_n] \cdot \Pr[x \in S_n] \\
&\quad + \Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 1 \mid x \notin S_n] \cdot \Pr[x \notin S_n] \\
&\leq \Pr[x \in S_n] + \Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 1 \mid x \notin S_n] \\
&= \Pr[x \in S_n] + \Pr[\mathsf{Invert}_{\mathcal{A}',h}(n) = 1] \\
&= \frac{1}{2^{n/2}} + \Pr[\mathsf{Invert}_{\mathcal{A}',h}(n) = 1].
\end{aligned}$$

We therefore have that

$$\Pr[\mathsf{Invert}_{\mathcal{A}',h}(n)=1] \geq \Pr[\mathsf{Invert}_{\mathcal{A},f}(n)=1] - \frac{1}{2^{n/2}} = \varepsilon(n) - \frac{1}{2^{n/2}}.$$

Since $\mathcal{A}'$ is a probabilistic polynomial-time algorithm and h is one-way, ε must be negligible. We therefore conclude that f is one-way.

Having established that f is one-way, it remains to show that g is not one-way. In order to see this, observe that $f(f(x)) = 0^{|x|}$ for every input x; thus, it is trivial to find a pre-image of $0^{|x|}$ under g (just take $0^{|x|}$) and $g(x) = f(f(x))$ is *not* one-way.

Next, we claim that $g(x) = (f(x) \| f(f(x)))$ is one-way. Assume by contradiction that there exists a probabilistic polynomial-time adversary $\mathcal{A}$ and a non-negligible function $\varepsilon(\cdot)$ such that

$$\Pr[\mathsf{Invert}_{\mathcal{A},g}(n)=1] = \varepsilon(n).$$

We construct a probabilistic polynomial-time $\mathcal{A}'$ who inverts f with probability $\varepsilon(n)$. Namely, $\mathcal{A}'$ receives input 1^n and y, computes $z = f(y)$ and invokes $\mathcal{A}$ on input 1^n and (y,z). Adversary $\mathcal{A}'$ then outputs whatever $\mathcal{A}$ outputs. Clearly,

$$\Pr[\mathsf{Invert}_{\mathcal{A}',f}(n)=1] = \Pr[\mathsf{Invert}_{\mathcal{A},g}(n)=1] = \varepsilon(n),$$

in contradiction to the one-wayness of f.

7.9 Let $\Pi = (\mathsf{Gen}, \mathsf{Samp}, f)$ be a function family. A function $\mathsf{hc} : \{0,1\}^* \to \{0,1\}$ is a *hard-core predicate of* Π if it is efficiently computable and if for every PPT algorithm $\mathcal{A}$ there is a negligible function negl such that

$$\Pr_{I \leftarrow \mathsf{Gen}(1^n),\, x \leftarrow \mathsf{Samp}(I)}[\mathcal{A}(I, f_I(x)) = \mathsf{hc}(I,x)] \leq \frac{1}{2} + \mathsf{negl}(n).$$

Prove a version of the Goldreich–Levin theorem for this setting, namely, if a one-way function (resp., permutation) family Π exists, then there exists a one-way function (resp., permutation) family Π' and a hard-core predicate hc of Π'.

Solution: The only difference between this and the original Goldreich-Levin proof is that here the set S_n of Claim 7.18 depends on the function in the family. The proof is otherwise the same as in the book.

7.10 Show a construction of a pseudorandom generator from any one-way permutation family. You may use the result of the previous exercise.

Solution: This works in the same way as the proof of Theorem 7.19. However, the seed must also be used to sample the permutation using the sampling algorithm. Let $p(n)$ be an upper bound on the amount of randomness used to sample a function in the family. Then, given

a seed of length $p(n)+n$, first use $p(n)$ bits to sample a function f_I. Then, it is possible to use the constructions of Sections 7.4.1 and 7.4.2 to obtain $p(n)+n+1$ bits. Observe that increasing the expansion factor here is essential since applying the basic generator of Section 7.4.1 only once will yield an output of just $n+1$ bits which is less than the length of the seed and thus not a generator. By applying $p(n)$ times as in Section 7.4.1, expansion is achieved.

7.11 This exercise is for students who have taken a course in complexity theory or are otherwise familiar with $\mathcal{NP}$ completeness.

(a) Show that the existence of one-way functions implies $\mathcal{P} \neq \mathcal{NP}$.

(b) Assume that $\mathcal{P} \neq \mathcal{NP}$. Show that there exists a function f that is: (1) computable in polynomial time, (2) hard to invert *in the worst case* (i.e., for all probabilistic polynomial-time $\mathcal{A}$, $\Pr_{x \leftarrow \{0,1\}^n}[f(\mathcal{A}(f(x))) = f(x)] \neq 1$), but (3) is *not* one-way.

Solution:

(a) Consider the language $L = \{(y, x') \mid \exists x'' : f(x'\|x'') = y\}$; that is, the language of all pairs (y, x') where x' is is the prefix of *some* pre-image of y. It is clear that $L \in \mathcal{NP}$ (one can non-deterministically guess x'' and then compute f). Now, if $\mathcal{P} = \mathcal{NP}$ then there exists a polynomial-time algorithm A that decides L. We use A to invert f. Specifically, given 1^n and y, we use A to determine the bits of a pre-image of y one at a time as follows. Given the prefix x' of a pre-image of y (initially x' is of length zero), we extend it by one bit by using A to determine if $(y, x'\|0) \in L$. If yes, then we set $x' := x'\|0$ and proceed; otherwise, we set $x' := x'\|1$ and proceed. After n iterations, we obtain x such that $f(x) = y$ and have thus successfully inverted f. The above holds for any function f. Therefore, if there exists a one-way function, $\mathcal{P}$ cannot equal $\mathcal{NP}$.

(b) Let G be a graph and let ϕ be a 3-coloring of G. Then, consider the function $f(G, \phi) = (G, 1)$ if ϕ is a valid coloring of G and $f(G, \phi) = (G, 0)$ if ϕ is not a valid coloring of G. (Note, that G can be represented as a string of length $\binom{n}{2}$ where each bit in the string denotes the existence or nonexistence of the appropriate edge.)

f is easy to compute, because the validity of a coloring can be checked in polynomial-time. Further, f is hard to invert in the worst case. This is due to the assumption that $\mathcal{P} \neq \mathcal{NP}$ and so given $(G, 1)$ it is hard – in the worst case – to find a valid 3-coloring for G. Finally, note that f is *not* a one-way function. This is due to the fact that a random 3-coloring of a random graph will be valid with very low probability. Thus, with high probability over a *random input* the output of f will be some pair $(G, 0)$ which is easily invertible (just compute any invalid coloring).

7.12 Let $x \in \{0,1\}^n$ and denote $x = x_1 \cdots x_n$. Prove that if there exists a one-way function, then there exists a one-way function f such that for every i there exists an algorithm A_i such that

$$\Pr_{x \leftarrow \{0,1\}^n}[A_i(f(x)) = x_i] \geq \frac{1}{2} + \frac{1}{2n}\,.$$

(This exercise demonstrates that it is not possible to claim that every one-way function hides at least one *specific* bit of the input.)

Solution: Let f be a one-way function. We define a function g that receives an input of length $n + \log n$. For convenience, we denote the input of g by a pair (x, j) where $|x| = n$ and $|j| = \log n$; thus j is a number between 0 and n. Now, define g as follows:

$$g(x\|j) = f(x), j, x_j$$

where x_j denotes the j^{th} bit of x.

Consider inputs $x\|j$ of length $n + \log n$ and fix $i \leq n$. We show an algorithm A_i such that $A_i(g(x\|j))$ outputs x_i with probability $1/2 + 1/2(n + \log n)$. (Note that outputting the ith bit of the input for $i > n$ is trivial since j is revealed in its entirety in the output of g.) Algorithm A_i receives $\langle y, j, b\rangle$. If $j = i$, then A_i outputs b; otherwise, A_i outputs a uniform bit. Now,

$$\begin{aligned}\Pr\left[A_i(g(x\|j)) = x_i\right] &= \frac{1}{2} \cdot \Pr[j \neq i] + 1 \cdot \Pr[j = i] \\ &= \frac{1}{2}\left(1 - \frac{1}{n}\right) + \frac{1}{n} = \frac{1}{2} + \frac{1}{2n} \\ &> \frac{1}{2} + \frac{1}{2(n + \log n)}.\end{aligned}$$

It remains to prove that g is one-way. However, this is easy. Assume by contradiction that it can be inverted with non-negligible probability by some probabilistic polynomial-time algorithm $\mathcal{A}$. Then we can invert f with at least the same probability as follows. Upon receiving input $y = f(x)$, invoke $\mathcal{A}$ upon input $(y, j, 0)$ and $(y, j, 1)$ for all $j = 1, \ldots, n$. Note that at least one of these tuples $(y, j, 0)$ or $(y, j, 1)$ is in the range of g. Therefore, $\mathcal{A}$ inverts this tuple, returning x, with the same probability that it inverts g. We can check all of the results and return the one that is correct, if one exists. (Note that we invoke $\mathcal{A}$ exactly $2n$ times so the inversion procedure remains polynomial.) This inversion procedure succeeds with probability at least as high as the probability that $\mathcal{A}$ inverts g, in contradiction to the assumption that f is one-way.

7.13 Show that if a one-to-one function has a hard-core predicate, then it is one-way.

Solution: Let f be an efficiently-computable one-to-one function and let hc be a hard-core predicate of f. We show that if f is not one-way then hc cannot be a hard-core predicate. Assume by contradiction that there exists a probabilistic polynomial-time adversary $\mathcal{A}$ and a non-negligible function $\varepsilon(\cdot)$ such that

$$\Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 1] = \varepsilon(n).$$

We now construct an adversary $\mathcal{B}$ that contradicts the assumption that hc is a hard-core predicate of f. The adversary $\mathcal{B}$ receives for input some $y = f(x)$, with $x \leftarrow \{0,1\}^n$, and attempts to guess $\mathsf{hc}(x)$. In order to do this, $\mathcal{B}$ first invokes $\mathcal{A}$ upon input 1^n and y; let x' be the output of $\mathcal{A}$. Next, $\mathcal{B}$ checks if $f(x') = y$ (notice that f is efficiently computable, so $\mathcal{B}$ can do this). If yes, then $\mathcal{B}$ outputs $\sigma = \mathsf{hc}(x')$ and halts (in this case, $\mathcal{B}$ is correct with probability 1). Otherwise, $\mathcal{B}$ outputs a uniformly chosen bit $\sigma \leftarrow \{0,1\}$. It remains to analyze $\mathcal{B}$'s success. We denote the event of $\mathcal{A}$ successfully inverting f by $\mathsf{Succ}_{\mathcal{A}}$. Then,

$$\begin{aligned}
&\Pr[\mathcal{B}(f(x)) = \mathsf{hc}(x)] \\
&= \Pr[\mathcal{B}(f(x)) = \mathsf{hc}(x) \mid \mathsf{Succ}_{\mathcal{A}}] \cdot \Pr[\mathsf{Succ}_{\mathcal{A}}] \\
&\quad + \Pr[\mathcal{B}(f(x)) = \mathsf{hc}(x) \mid \neg\mathsf{Succ}_{\mathcal{A}}] \cdot \Pr[\neg\mathsf{Succ}_{\mathcal{A}}] \\
&= \Pr[\mathcal{B}(f(x)) = \mathsf{hc}(x) \mid \mathsf{Succ}_{\mathcal{A}}] \cdot \Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 1] \\
&\quad + \Pr[\mathcal{B}(f(x)) = \mathsf{hc}(x) \mid \neg\mathsf{Succ}_{\mathcal{A}}] \cdot \Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 0] \\
&= 1 \cdot \Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 1] + \frac{1}{2} \cdot \Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 0] \\
&= 1 \cdot \Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 1] + \frac{1}{2} \cdot (1 - \Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 1]) \\
&= \frac{1}{2} + \frac{1}{2} \cdot \Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 1] \;\geq\; \frac{1}{2} + \frac{\varepsilon(n)}{2}\,.
\end{aligned}$$

Since hc is a hard-core predicate for f, we see that ε must be negligible. We conclude that f must be one-way.

Note that the condition that f is one-to-one is necessary to ensure that whenever $\mathcal{A}$ succeeds in inverting $f(x)$, it follows that $\mathcal{B}$ succeeds in guessing $\mathsf{hc}(x)$. For example, consider the function $f(x) = 0^{|x|}$ and $\mathsf{hc}(x) = x_1$ where $x = x_1 \cdots x_n$. Then, clearly hc is a hard-core predicate of f. However, f is *not* one-way.

7.14 Show that if Construction 7.21 is modified in the natural way so that $F_k(x)$ is defined for every nonempty string x of length at most n, then the construction is no longer a pseudorandom function.

Solution: Let D be a distinguisher who first queries its oracle on any $x \in \{0,1\}^{n-1}$; let y be the response. Next, D queries its oracle on $x' =$

$x\|0$ of length n; let y' be the response. Finally, D checks if $y' = G_0(y)$. If yes, then D outputs 1; otherwise, it outputs 0. We have

$$\Pr[D^{F_k(\cdot)}(1^n) = 1] = 1$$

whereas for a random function f

$$\Pr[D^{f(\cdot)}(1^n) = 1] = \frac{1}{2^n}.$$

D therefore distinguishes F_k from random.

7.15 Prove that if there exists a pseudorandom function that, using a key of length n, maps n-bit inputs to single-bit outputs, then there exists a pseudorandom function that maps n-bit inputs to n-bit outputs.

Solution: Let F' take a key of length n^2, and define its output as

$$F'_{\langle k_1,\ldots,k_n\rangle}(x) = F_{k_1}(x), \ldots, F_{k_n}(x).$$

This has output length n, as requested.

To show that F' is pseudorandom, fix a PPT algorithm $\mathcal{A}$ distinguishing F' from a random function. We assume without loss of generality that $\mathcal{A}$ does not query its oracle on the same input more than once. Define an algorithm D as follows:

> **Algorithm** D:
> The algorithm is given access to an oracle that is either equal to $F_k(\cdot)$ for uniform $k \in \{0,1\}^n$, or is equal to $f(\cdot)$ for a random function f.
>
> - Choose uniform $i \in \{1,\ldots,n\}$.
> - For $j = i+1$ to n, choose uniform $k_j \in \{0,1\}^n$.
> - Run $\mathcal{A}(1^n)$. When $\mathcal{A}$ queries its oracle on input x, do:
>   - For $j = 1$ to $i-1$, choose uniform $y_j \in \{0,1\}$.
>   - For $j = i$, query x to the oracle and set y_j equal to the answer received.
>   - For $j = i+1$ to n, set $y_j := F_{k_j}(x)$.
> - Output whatever $\mathcal{A}$ outputs.

For a fixed value of n and $i \in \{0,\ldots,n\}$, let $\mathcal{O}_n^i$ be a distribution over *functions* mapping n-bit inputs to n-bit outputs and defined as follows: choose (independent) random functions $f_1,\ldots,f_i$ mapping n-bit inputs to 1-bit outputs, along with uniform $k_{i+1},\ldots,k_n \in \{0,1\}^n$. The function then maps input x to the output

$$f_1(x),\ldots,f_i(x),F_{k_{i+1}}(x),\ldots,F_{k_n}(x).$$

By inspection of D we have

$$\Pr[D^{f(\cdot)}(1^n) = 1] = \sum_{i=1}^{n} \frac{1}{n} \cdot \Pr[\mathcal{A}^{\mathcal{O}_n^i(\cdot)}(1^n) = 1]$$

(where f denotes a random function mapping n-bit inputs to 1-bit outputs) and

$$\Pr[D^{F_k(\cdot)}(1^n) = 1] = \sum_{i=1}^{n} \frac{1}{n} \cdot \Pr[\mathcal{A}^{\mathcal{O}_n^{i-1}(\cdot)}(1^n) = 1].$$

Therefore

$$\begin{aligned}
&\left|\Pr[D^{f(\cdot)}(1^n) = 1] - \Pr[D^{F_k(\cdot)}(1^n) = 1]\right| \\
&= \left|\sum_{i=1}^{n} \frac{1}{n} \cdot \Pr[\mathcal{A}^{\mathcal{O}_n^i(\cdot)}(1^n) = 1] - \sum_{i=1}^{n} \frac{1}{n} \cdot \Pr[\mathcal{A}^{\mathcal{O}_n^{i-1}(\cdot)}(1^n) = 1]\right| \\
&= \left|\frac{1}{n} \cdot \Big(\Pr[\mathcal{A}^{\mathcal{O}_n^n(\cdot)}(1^n) = 1] - \Pr[\mathcal{A}^{\mathcal{O}_n^0(\cdot)}(1^n) = 1]\Big)\right| \\
&= \frac{1}{n} \cdot \left|\Pr[\mathcal{A}^{\hat{f}(\cdot)}(1^n) = 1] - \Pr[\mathcal{A}^{F'_{\langle k_1,\ldots,k_n\rangle}(\cdot)}(1^n) = 1]\right|,
\end{aligned}$$

where $\hat{f}$ denotes a random function mapping n-bit inputs to n-bit outputs. Since F is pseudorandom, we know that there exists a negligible function negl such that

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

therefore,

$$\left|\Pr[\mathcal{A}^{\hat{f}(\cdot)}(1^n) = 1] - \Pr[\mathcal{A}^{F'_{\langle k_1,\ldots,k_n\rangle}(\cdot)}(1^n) = 1]\right| \leq n \cdot \mathsf{negl}(n),$$

which is still negligible. Since $\mathcal{A}$ was arbitrary, this completes the proof.

7.16 Prove that a two-round Feistel network using pseudorandom round functions (as in Equation (7.15)) is not pseudorandom.

Solution: We show an attack that works (with high probability) regardless of what round functions are used. The attacker $\mathcal{A}$, given access to an oracle $\mathcal{O} : \{0,1\}^{2n} \to \{0,1\}^{2n}$ works as follows: choose arbitrary $L_0, L'_0, R_0 \in \{0,1\}^n$ with $L_0 \neq L'_0$. Query $\mathcal{O}(L_0, R_0)$ and obtain output (L_2, R_2). Then query $\mathcal{O}(L'_0, R_0)$ and obtain output (L'_2, R'_2). Output 1 iff $L_0 \oplus L'_0 \stackrel{?}{=} L_2 \oplus L'_2$.

Observe that if $\mathcal{O}$ is a permutation chosen at random, then (L_2, R_2) and (L'_2, R'_2) are uniform subject to the constraint that they are unequal;

thus, the probability that $\mathcal{A}$ outputs 1 is roughly 2^{-n}. On the other hand, if $\mathcal{O}$ is a 2-round Feistel network with round functions f_1, f_2 then

$$L_2 = R_1 = L_0 \oplus f_1(R_0)$$

and, similarly,

$$L_2' = L_0' \oplus f_1(R_0).$$

So,

$$L_2 \oplus L_2' \;=\; L_0 \oplus f_1(R_0) \oplus L_0' \oplus f_1(R_0) \;=\; L_0 \oplus L_0',$$

and so $\mathcal{A}$ outputs 1 with probability 1 in this case. $\mathcal{A}$ thus has a significant advantage in distinguishing between a two-round Feistel network and a truly random permutation.

7.17 Prove that a three-round Feistel network using pseudorandom round functions (as in Equation (7.16)) is not *strongly* pseudorandom.

Solution: We show an attack that works (with high probability) regardless of what round functions are used. The attack exploits the fact that we can introduce a known "shift" δ in the input to the second round-function of the Feistel network in two ways: by XORing δ into the left half of the input when querying the function in the forward direction (and keeping the right half fixed), or by XORing δ into the right half of the input when querying the function in the *reverse* direction (and keeping the left half fixed). It is easiest to see this by tracing through the effects of these modifications in Figure 7.3. We now give the details.

The attacker $\mathcal{A}$, given access to an oracle $\mathcal{O} : \{0,1\}^{2n} \to \{0,1\}^{2n}$ and its inverse $\mathcal{O}^{-1} : \{0,1\}^{2n} \to \{0,1\}^{2n}$ works as follows: choose arbitrary $L_0, R_0, \delta \in \{0,1\}^n$ with $\delta \neq 0^n$. Then:

(a) Query $\mathcal{O}(L_0, R_0)$ to obtain (L_3, R_3).

(b) Query $\mathcal{O}^{-1}(L_3, R_3 \oplus \delta)$ to obtain (L_0', R_0').

(c) Query $\mathcal{O}(L_0 \oplus \delta, R_0)$ to obtain (L_3'', R_3'').

Output 1 iff $L_3 \oplus R_0' \stackrel{?}{=} L_3'' \oplus R_0$.

When $\mathcal{O}$ is a random permutation, $\mathcal{A}$ outputs 1 with only negligible probability. To see this, note that $(L_3, R_3 \oplus \delta) \neq (L_3, R_3)$ and so (L_0', R_0') is uniform subject to being unequal to (L_0, R_0) (because $\mathcal{O}$ is a permutation). The probability that $(L_0', R_0') = (L_0 \oplus \delta, R_0)$ is, therefore, roughly 2^{-2n}. Assuming $(L_0', R_0') \neq (L_0 \oplus \delta, R_0)$, the value (L_3'', R_3'') is uniform subject to being different from both (L_3, R_3) and $(L_3, R_3 \oplus \delta)$, and so $L_3 \oplus R_0' = L_3'' \oplus R_0$ only with probability roughly 2^{-n}.

On the other hand, if $\mathcal{O}$ is a 3-round Feistel network with round functions f_1, f_2, f_3 then

$$\begin{aligned} L_3 \oplus R'_0 &= L_3 \oplus \Big(L_3 \oplus f_2(R_3 \oplus \delta \oplus f_3(L_3))\Big) \\ &= f_2(R_3 \oplus \delta \oplus f_3(L_3)) \end{aligned} \tag{7.2}$$

and

$$\begin{aligned} L''_3 \oplus R_0 &= \Big(R_0 \oplus f_2(L_0 \oplus \delta \oplus f_1(R_0))\Big) \oplus R_0 \\ &= f_2(L_0 \oplus \delta \oplus f_1(R_0)). \end{aligned} \tag{7.3}$$

Since $R_3 \oplus f_3(L_3) = L_0 \oplus f_1(R_0)$, we see that Equations (7.2) and (7.3) are equal, and so $\mathcal{A}$ outputs 1 with probability 1 in this case. $\mathcal{A}$ thus has a significant advantage in distinguishing between a three-round Feistel network and a truly random permutation.

7.18 Consider the keyed permutation F^* defined by

$$F^*_k(x) \stackrel{\text{def}}{=} \mathsf{Feistel}_{F_k,F_k,F_k}(x).$$

(Note that the same key is used in each round.) Show that F^* is not pseudorandom.

Solution: When the same key k is used in all rounds, computing F^* in the forward direction is essentially the same as inverting it. In particular, for every $L_0, R_0 \in \{0,1\}^{n/2}$ it holds that $F^*_k(L_0, R_0) = {F^*}^{-1}_k(R_0, L_0)$. In order to see this, observe that the only difference between computing F^* in the forward or reverse direction is that in the forward direction the first input to F_k in the first round is R_0, whereas in the reverse direction the first input to F_k is $R_2 = L_3$ (see Figure 7.3 in the book).

This observation yields the following distinguisher D: set $x = 0^n$ and query the oracle, receiving back y. Denote $y = (L_3, R_3)$. Then, query $x' = (R_3, L_3)$ to the oracle. If the response equals 0^n then output 0; otherwise, output 1. Based on what we have explained above

$$\Pr\Big[D^{F^*_k(\cdot)}(1^n) = 1\Big] = 1$$

whereas

$$\Pr\Big[D^{f(\cdot)}(1^n) = 1\Big] = 2^{-n}$$

since for a pseudorandom permutation, the probability that $f(f(0^n)) = 0^n$ is exactly 2^{-n}. (In order to see this, choose f by first choosing the output of 0^n, and then continuing to choose the outputs one at a time.)

7.19 Let G be a pseudorandom generator with expansion factor $\ell(n) = n+1$. Prove that G is a one-way function.

Solution: Assume by contradiction that there is a probabilistic polynomial-time algorithm $\mathcal{A}$ and a non-negligible function $\varepsilon(\cdot)$ such that

$$\Pr[\mathsf{Invert}_{\mathcal{A},G}(n) = 1] = \varepsilon(n).$$

We construct a distinguisher D for the pseudorandom generator as follows. Upon input r of length $n+1$, distinguisher D invokes $\mathcal{A}$ upon 1^n and r; let s be $\mathcal{A}$'s output. If $r = G(s)$ then D outputs 1; otherwise, D outputs a random bit $b \leftarrow \{0,1\}$. We have:

$$\begin{aligned}
&\Pr_{s\in\{0,1\}^n}[D(G(s)) = 1] \\
&= 1 \cdot \Pr[\mathsf{Invert}_{\mathcal{A},G}(n) = 1] + \frac{1}{2} \cdot \Pr[\mathsf{Invert}_{\mathcal{A},G}(n) = 0] \\
&= \varepsilon(n) + \frac{1}{2} \cdot (1 - \varepsilon(n)) = \frac{1}{2} + \frac{\varepsilon(n)}{2}\,.
\end{aligned}$$

In contrast,

$$\Pr_{r\in\{0,1\}^{n+1}}[D(r) = 1] \leq \frac{1}{2}.$$

This holds because the range of G (on inputs of length n) is of size at most 2^n, and so at least half the strings in $\{0,1\}^{n+1}$ are not even in the range of G (and so there doesn't exist an $s \in \{0,1\}^n$ such that $G(s) = r$). We conclude that

$$\Pr_{s\in\{0,1\}^n}[D(G(s)) = 1] - \Pr_{r\in\{0,1\}^{n+1}}[D(r) = 1] \geq \frac{\varepsilon(n)}{2}$$

in contradiction to the assumption that G is a pseudorandom generator.

7.20 Let $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ be probability ensembles. Prove that if $\mathcal{X} \stackrel{\text{c}}{\equiv} \mathcal{Y}$ and $\mathcal{Y} \stackrel{\text{c}}{\equiv} \mathcal{Z}$, then $\mathcal{X} \stackrel{\text{c}}{\equiv} \mathcal{Z}$.

Solution: Let D be a probabilistic polynomial-time algorithm. We wish to show that there exists a negligible function $\mathsf{negl}(n)$ such that

$$\left| \Pr_{x\leftarrow X_n}[D(1^n, x)) = 1] - \Pr_{z\leftarrow Z_n}[D(1^n, z) = 1] \right| \leq \mathsf{negl}(n). \tag{7.4}$$

By the assumption, there exist negligible functions $\mathsf{negl}_1(n)$ and $\mathsf{negl}_2(n)$ such that:

$$\left| \Pr_{x\leftarrow X_n}[D(1^n, x)) = 1] - \Pr_{y\leftarrow Y_n}[D(1^n, y) = 1] \right| \leq \mathsf{negl}_1(n)$$

and

$$\left| \Pr_{y\leftarrow Y_n}[D(1^n, y)) = 1] - \Pr_{z\leftarrow Z_n}[D(1^n, z) = 1] \right| \leq \mathsf{negl}_2(n).$$

Thus,

$$\begin{aligned}
&\left|\Pr_{x\leftarrow X_n}[D(1^n,x))=1]-\Pr_{z\leftarrow Z_n}[D(1^n,z)=1]\right| \\
&\quad=\left|\Pr_{x\leftarrow X_n}[D(1^n,x))=1]-\Pr_{y\leftarrow Y_n}[D(1^n,y)=1]\right. \\
&\qquad\left.+\Pr_{y\leftarrow Y_n}[D(1^n,y))=1]-\Pr_{z\leftarrow Z_n}[D(1^n,z)=1]\right| \\
&\quad\leq\left|\Pr_{x\leftarrow X_n}[D(1^n,x))=1]-\Pr_{y\leftarrow Y_n}[D(1^n,y)=1]\right| \\
&\qquad+\left|\Pr_{y\leftarrow Y_n}[D(1^n,y))=1]-\Pr_{z\leftarrow Z_n}[D(1^n,z)=1]\right| \\
&\quad\leq \mathsf{negl}_1(n)+\mathsf{negl}_2(n),
\end{aligned}$$

where the first inequality is by the triangle inequality. Since the sum of negligible functions is negligible, we conclude that Equation (7.4) holds.

7.21 Prove Theorem 7.32.

Solution: The proof is by reduction. We show that if there exists a probabilistic polynomial-time distinguisher D that distinguishes $\overline{\mathcal{X}}$ from $\overline{\mathcal{Y}}$ with non-negligible success, then there exists a probabilistic polynomial-time distinguisher D' that distinguishes a single sample of X_n from a single sample of Y_n with non-negligible success. Our proof uses a *hybrid argument*.

Let D be a probabilistic polynomial-time distinguisher and let $\varepsilon(n)$ be a function such that

$$\left|\Pr\left[D(X_n^{(1)},\ldots,X_n^{(p(n))})=1\right]-\Pr\left[D(Y_n^{(1)},\ldots,Y_n^{(p(n))})=1\right]\right|=\varepsilon(n). \tag{7.5}$$

For every i, we define a *hybrid* random variable H_n^i as a sequence containing i independent copies of X_n followed by $p(n)-i$ independent copies of Y_n. That is:

$$H_n^i=\left(X_n^{(1)},\ldots,X_n^{(i)},Y_n^{(i+1)},\ldots,Y_n^{(p(n))}\right)$$

Denote $\overline{X}_n=(X_n^{(1)},\ldots,X_n^{(p(n))})$ and likewise for $\overline{Y}_n$. Then, we have that $H_n^0=\overline{Y}_n$ and $H_n^{p(n)}=\overline{X}_n$. Thus,

$$\begin{aligned}
&\left|\Pr[D(\overline{X}_n)=1]-\Pr[D(\overline{Y}_n)=1]\right| \\
&\quad=\left|\Pr[D(H_n^{p(n)})=1]-\Pr[D(H_n^0)=1]\right| \\
&\quad=\left|\sum_{i=0}^{p(n)-1}\Pr[D(H_n^i)=1]-\sum_{i=0}^{p(n)-1}\Pr[D(H_n^{i+1})=1]\right|
\end{aligned}$$

where the second equality follows from the fact that the only remaining terms in this telescopic sum are $\Pr[D(H_n^0) = 1]$ and $\Pr[D(H_n^{p(n)}) = 1]$.

We now construct a probabilistic polynomial-time distinguisher D' for a single sample of X_n and Y_n. Upon input some sample α of X_n or Y_n, D' chooses a random $i \leftarrow \{0, \ldots, p(n) - 1\}$, generates the vector $\overline{H}_n = (X_n^{(1)}, \ldots, X_n^{(i)}, \alpha, Y_n^{(i+2)}, \ldots, Y_n^{(p(n))})$, invokes D on the vector $\overline{H}_n$, and outputs whatever D does. Now, if α is distributed according to X_n, then $\overline{H}_n$ is distributed exactly like H_n^{i+1}. In contrast, if α is distributed according to Y_n, then $\overline{H}_n$ is distributed exactly like H_n^i. (Note that we use the independence of the samples in making this argument.) Furthermore, each i is chosen with probability exactly $1/p(n)$. Therefore,

$$\Pr[D'(X_n) = 1] = \frac{1}{p(n)} \cdot \sum_{i=0}^{p(n)-1} \Pr[D(H_n^{i+1}) = 1]$$

and

$$\Pr[D'(Y_n) = 1] = \frac{1}{p(n)} \cdot \sum_{i=0}^{p(n)-1} \Pr[D(H_n^{i}) = 1]$$

It therefore follows that:

$$\begin{aligned}
&|\Pr[D'(X_n) = 1] - \Pr[D'(Y_n) = 1]| \\
&\quad = \frac{1}{p(n)} \cdot \left| \sum_{i=0}^{p(n)-1} \Pr[D(H_n^{i+1}) = 1] - \sum_{i=0}^{p(n)-1} \Pr[D(H_n^{i}) = 1] \right| \\
&\quad = \frac{1}{p(n)} \cdot \left| \Pr[D(H_n^{p(n)}) = 1] - \Pr[D(H_n^{0}) = 1] \right| \\
&\quad = \frac{1}{p(n)} \cdot \left| \Pr[D(\overline{X}_n) = 1] - \Pr[D(\overline{Y}_n) = 1] \right| \;=\; \frac{\varepsilon(n)}{p(n)}.
\end{aligned}$$

By the assumption that $\mathcal{X}$ and $\mathcal{Y}$ are indistinguishable, we conclude that $\frac{\varepsilon(n)}{p(n)}$ is negligible, implying that $\varepsilon(n)$ itself is negligible. By Equation (7.5), this implies that X is indistinguishable from Y, completing the proof of the theorem.

7.22 Let $\mathcal{X} = \{X_n\}_{n\in\mathbb{N}}$ and $\mathcal{Y} = \{Y_n\}_{n\in\mathbb{N}}$ be computationally indistinguishable probability ensembles. Prove that for any probabilistic polynomial-time algorithm $\mathcal{A}$, the ensembles $\{\mathcal{A}(X_n)\}_{n\in\mathbb{N}}$ and $\{\mathcal{A}(Y_n)\}_{n\in\mathbb{N}}$ are computationally indistinguishable.

Solution: Assume there is a probabilistic polynomial-time $\mathcal{A}$, a distinguisher D and a non-negligible function $\varepsilon(\cdot)$ such that

$$|\Pr[D(1^n, \mathcal{A}(X_n)) = 1] - \Pr[D(1^n, \mathcal{A}(Y_n)) = 1]| = \varepsilon(n).$$

We construct a distinguisher D' that receives a sample z (from X_n or Y_n), runs $\mathcal{A}$ on the input (i.e. z, but not the 1^n part) and then runs D on 1^n together with the result from $\mathcal{A}$. Finally, D' outputs whatever D does. Clearly, for every z, $D'(1^n, z) = 1$ if and only if $D(1^n, \mathcal{A}(z)) = 1$. Therefore,

$$|\Pr[D'(1^n, X_n) = 1] - \Pr[D'(1^n, Y_n) = 1]| = \varepsilon(n)$$

a contradiction.

Chapter 8

Number Theory and Cryptographic Hardness Assumptions – Solutions

8.1 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.

Solution: Let $\mathbb{G}$ be a group, and let $e, e' \in \mathbb{G}$ with $eg = ge = g$ and $e'g = ge' = g$ for all $g \in \mathbb{G}$. Then $e' = e \cdot e' = e$ and so e and e' are the same element.

Let 1 denote the identity in $\mathbb{G}$. Fix $g \in \mathbb{G}$ and let $h, h' \in \mathbb{G}$ be such that $gh = hg = 1$ and $gh' = h'g = 1$. Then

$$hgh' = (hg)h' = 1 \cdot h' = h'$$

and

$$hgh' = h(gh') = h \cdot 1 = h,$$

and so $h' = h$.

8.2 Show that Proposition 8.36 does not necessarily hold when $\mathbb{G}$ is infinite.

Solution: Take $\mathbb{G} = \mathbb{R}^+$, i.e., the group of all positive real numbers with respect to multiplication. Let $\mathbb{H} = \{1\} \cup \{2, 4, 6, 8, \ldots\}$ be a subset of $\mathbb{G}$. Note that $\mathbb{H}$ satisfies the conditions of Proposition 8.36, but is not a subgroup since no elements in $\mathbb{H}$ (other than 1) have inverses in $\mathbb{H}$.

8.3 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?

Solution: We verify that $\langle g \rangle$ satisfies the conditions of Proposition 8.36. Since $g^0 = 1$, clearly $\langle g \rangle$ is nonempty. Furthermore, for any $a, b \in \langle g \rangle$ we know that there exist positive x, y such that $a = g^x$ and $b = g^y$. But then $ab = g^{x+y} \in \langle g \rangle$. So $\langle g \rangle$ is a subgroup when $\mathbb{G}$ is finite.

The set $\{g^0, g^1, \ldots\}$ need not be a subgroup of $\mathbb{G}$ when $\mathbb{G}$ is infinite. For example, consider the group $\mathbb{R}^+$ of all positive real numbers with respect to multiplication. The set $S \stackrel{\text{def}}{=} \{2^0, 2^1, 2^2, \ldots\}$ is *not* a subgroup of $\mathbb{R}^+$ since, for example, the element $2 \in S$ has no inverse in S.

8.4 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 use the Chinese remainder theorem.)

(c) Prove Theorem 8.19.

Solution:

(a) The integers in $\{1, \ldots, p^e\}$ that are not relatively prime to p are exactly the multiples of p, and there are exactly $p^e/p = p^{e-1}$ such integers. So $\phi(p^e) = p^e - p^{e-1} = p^{e-1}(p-1)$.

(b) We provide a solution that does not use the Chinese remainder theorem. Let p and q be relatively prime. Consider the integers $\{1, \ldots, pq\}$ arranged in an array as follows:

$$\begin{array}{cccc} 1 & p+1 & \cdots & (q-1)p+1 \\ 2 & p+2 & \cdots & (q-1)p+2 \\ \vdots & \vdots & \ddots & \vdots \\ p & 2p & \cdots & qp \end{array}$$

For any r having a factor in common with p, every element in the row

$$r \;\; p+r \;\; 2p+r \;\; \cdots \;\; (q-1)p+r$$

has a factor in common with p and hence also has a factor in common with pq. Eliminate those rows from consideration, and consider the remaining $\phi(p)$ rows. Let

$$s \;\; p+s \;\; 2p+s \;\; \cdots \;\; (q-1)p+s$$

be such a row (i.e., $\gcd(s, p) = 1$). We claim that

$$[s \bmod q] \;\; [p+s \bmod q] \;\; [2p+s \bmod q] \;\; \cdots \;\; [(q-1)p+s \bmod q]$$

contains the elements of $\mathbb{Z}_q$ in permuted order. This follows since there are q such elements, and $ip + s = jp + s \bmod q$ implies $j = i$ (using $\gcd(p, q) = 1$). From this it follows that each remaining row contains exactly $\phi(q)$ elements relatively prime to q. The leaves a total of $\phi(pq) \stackrel{\text{def}}{=} \phi(p) \cdot \phi(q)$ elements relatively prime to pq.

(c) Let $N = \prod_i p_i^{e_i}$. Parts (a) and (b) immediately give

$$\phi(N) = \prod_i \phi(p_i^{e_1}) = \prod_i p_i^{e_i - 1}(p_i - 1).$$

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

Solution: Note that $|\mathbb{Z}_{100}^*| = \phi(100) = 40$. So

$$3^{1000} = 3^{1000 \bmod 40} = 3^0 = 1 \bmod 100.$$

8.6 Compute $[101^{4,800,000,002} \bmod 35]$ (by hand).

Solution: We use the Chinese remainder theorem. First,

$$101^{4,800,000,002} = 1^{4,800,000,002} = 1 \bmod 5.$$

Next,

$$101^{4,800,000,002} = 3^{4,800,000,002 \bmod 6} = 3^2 = 2 \bmod 7.$$

Finally, we can explicitly compute (by trial-and-error) that $(1, 2) \leftrightarrow 16$. So, the answer is 16.

8.7 Compute $[46^{51} \bmod 55]$ (by hand) using the Chinese remainder theorem.

Solution: $55 = 11 \cdot 5$ and so we will work mod 11 and mod 5. First, $[46 \bmod 11] = 2$. Furthermore $\phi(11) = 10$ and thus $[46^{51} \bmod 11] = [2^{51 \bmod 10} \bmod 11] = [2^1 \bmod 11]$. Similarly, $[46 \bmod 5] = 1$. We conclude that $[46^{51} \bmod 55] \leftrightarrow (2, 1)$. We thus need to find the value $x \in \mathbb{Z}_{55}$ such that $x \bmod 11 = 2$ and $x \bmod 5 = 1$. But we have already seen that 46 is such a value. So $46 = 46^{51} \bmod 55$.

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

Solution: The conditions of Definition 7.9 are easy to verify. Closure and associativity are obvious. If e, e' are the identity elements of $\mathbb{G}$ and $\mathbb{H}$, respectively, then (e, e') is the identity of $\mathbb{G} \times \mathbb{H}$. Finally, any $(g, h) \in \mathbb{G} \times \mathbb{H}$ has inverse (g^{-1}, h^{-1}), where g^{-1} (resp., h^{-1}) is the inverse of g (resp., h) in $\mathbb{G}$ (resp., $\mathbb{H}$).

8.9 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 $[X \bmod N]$.

Solution: Let $b = [X \bmod N]$ so that X can be written as $X = aN + b$ with $a \in \mathbb{Z}$. Let $\beta = [b \bmod p]$ so that b can be written as $b = \alpha p + \beta$ with $\alpha \in \mathbb{Z}$. Then $X = aN + \alpha p + \beta$ and, since $p \mid N$, we therefore have $[X \bmod p] = \beta = [[X \bmod N] \bmod p\,]$, as desired.

On the other hand, let $X = 15$, $N = 10$, and $p = 5$. Then we have $[X \bmod N] = 5$ but $[[X \bmod p\,] \bmod N] = 0$.

8.10 Corollary 8.21 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 $x \in \{0, \ldots, N-1\}$.

Solution: If $ed = 1 \bmod \phi(N)$ then, because $\phi(N) = (p-1)(q-1) = \phi(p) \cdot \phi(q)$, it also holds that $ed = 1 \bmod \phi(p)$. So for any $x \in \mathbb{Z}_p^*$ we know that $(x^e)^d = x \bmod p$. But for $x = 0$ it also holds, trivially, that $(x^e)^d = x \bmod p$. We conclude that for any $x \in \mathbb{Z}_p$, it holds that $(x^e)^d = x \bmod p$. The same holds with p replaced by q.

Now let $x \in \mathbb{Z}_N$. Then $x \leftrightarrow (x_p, x_q)$ with $x_p \in \mathbb{Z}_p$ and $x_q \in \mathbb{Z}_q$. Now

$$\begin{aligned}(x^e)^d &\leftrightarrow ((x_p, x_q)^e)^d \\ &= \left(\left[\left(x_p^e\right)^d \bmod p\right], \left[\left(x_q^e\right)^d \bmod q\right]\right) \\ &= (x_p, x_q) \;\leftrightarrow\; x.\end{aligned}$$

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

Solution: Let f be as defined in Theorem 8.24, restricted to domain $\mathbb{Z}_N^*$. It follows from the proof in the text that f is a bijection. It therefore only remains to show that for any $a, b \in \mathbb{Z}_N^*$ we have $f(a \cdot_N b) = f(a) \boxdot f(b)$, where $\cdot_N$ denotes multiplication modulo N and $\boxdot$ denotes the group operation in $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$ (namely, multiplication modulo p in the first component and modulo q in the second component).

To see that this is true, note that

$$\begin{aligned}f(a \cdot_N b) &= ([(a \cdot_N b) \bmod p\,], [(a \cdot_N b) \bmod q]) \\ &= ([(a \cdot b) \bmod p\,], [(a \cdot b) \bmod q]) \\ &= ([a \bmod p\,], [a \bmod q]) \boxdot ([b \bmod p\,], [b \bmod q]) \\ &= f(a) \boxdot f(b).\end{aligned}$$

8.12 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 $\hat{N}^e = N$.

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

Solution:

(a) If $N = \hat{N}^e$ then $e = \log_{\hat{N}} N \leq \log_2 N \leq \|N\| + 1$.

ALGORITHM 8.1-S
Computing $N^{1/e}$

Input: $N, e > 1$
Output: $N^{1/e}$ if this is an integer; **false** otherwise

low $:= 1$, **high** $:= N$
while low $\leq$ **high**: {
  $\hat{N} := \lfloor (\textbf{low} + \textbf{high})/2 \rfloor$
  if $\hat{N}^e = N$ then **return** $\hat{N}$ and halt
  if $\hat{N}^e > N$ then **high** $:= \hat{N} - 1$
  if $\hat{N}^e < N$ then **low** $:= \hat{N} + 1$ }
return false

(b) Consider Algorithm 8.1-S. Clearly, the algorithm always outputs false when $N^{1/e}$ is not an integer. Otherwise, the algorithm outputs $N^{1/e}$ in at most $\mathcal{O}(\log N) = \mathcal{O}(\|N\|)$ iterations.

(c) Run Algorithm 8.1-S for each of at most $\mathcal{O}(\|N\|)$ values of e.

8.13 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.

Solution: See Algorithm 8.2-S.

ALGORITHM 8.2-S
Testing strong witnesses

Input: odd integer N, $a \in \mathbb{Z}_N^*$
Output: Decide whether a is a strong witness

Compute r, u such that $N - 1 = 2^r u$ with u odd
$x := [a^u \bmod N]$
if $x = \pm 1$ then **return false**
for $i = 1$ **to** $r - 1$: {
  $x := [x^2 \bmod N]$
  if $x = N - 1$ then **return false** }
return true

8.14 Fix N, e with $\gcd(e, \phi(N)) = 1$, and assume there is 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 uniform choice of $x \in \mathbb{Z}_N^*$. Show that it is possible to construct an adversary $\mathcal{A}'$ for which

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

for *all* x. The running time t' of $\mathcal{A}'$ should be polynomial in t and $\|N\|$.

Solution: Let s be a parameter, fixed later. Construct $\mathcal{A}'$ as follows:

Algorithm $\mathcal{A}'$
On input N, y, e do:

(a) For $i = 1$ to s:
  i. Choose $r_i \leftarrow \mathbb{Z}_N^*$.
  ii. Run $\mathcal{A}([(r_i)^e \cdot y \bmod N])$ to obtain x_i.
  iii. If $(x_i)^e = (r_i)^e \cdot y \bmod N$ then output $[x_i/r_i \bmod N]$ and terminate.

(b) If the algorithm has not yet terminated, output fail.

Let y be arbitrary. The key point to note is that, in every iteration, $\mathcal{A}$ is run on a *uniform* element of $\mathbb{Z}_N^*$, irrespective of how y is distributed. This is so since r_i is uniform, hence $[(r_i)^e \bmod N]$ is uniform (since raising to eth powers is a permutation), and thus $[(r_i)^e \cdot y \bmod N]$ is uniform (because $\mathbb{Z}_N^*$ is a group). Furthermore, if $\mathcal{A}$ ever correctly computes an eth root in any iteration, then $\mathcal{A}'$ outputs the eth root of y: this follows since $(x_i)^e = (r_i)^e \cdot y \bmod N$ implies $(x_i/r_i)^e = y \bmod N$.

Combining the observations above and setting $s = 100 \ln 100$, we see that the probability that $\mathcal{A}'$ *fails* to output an inverse is

$$\left(1 - \frac{1}{100}\right)^{100 \ln 100} = \left(\left(1 - \frac{1}{100}\right)^{100}\right)^{\ln 100} \leq e^{-\ln 100} = \frac{1}{100}$$

(using Proposition A.2). The running time of $\mathcal{A}'$ is $\mathcal{O}(t \cdot \mathsf{poly}(\|N\|))$.

8.15 Formally define the CDH assumption. Prove that hardness of the CDH problem relative to $\mathcal{G}$ implies hardness of the discrete-logarithm problem relative to $\mathcal{G}$, and that hardness of the DDH problem relative to $\mathcal{G}$ implies hardness of the CDH problem relative to $\mathcal{G}$.

Solution: We first formally define what it means for the CDH problem to be hard.

The CDH experiment $\mathsf{CDH}_{\mathcal{A},\mathcal{G}}(n)$:

(a) Run $\mathcal{G}(1^n)$ to obtain $(\mathbb{G}, q, g)$, where $\mathbb{G}$ is a cyclic group of order q (with $\|q\| = n$), and g is a generator of $\mathbb{G}$.
(b) Choose $h_1, h_2 \leftarrow \mathbb{G}$.
(c) $\mathcal{A}$ is given $\mathbb{G}, q, g, h_1, h_2$, and outputs $h_3 \in \mathbb{G}$.
(d) The output of the experiment is defined to be 1 if $h_3 = \mathsf{DH}_g(h_1, h_2)$, and 0 otherwise.

As described, it is unclear how to test whether $h_3 = \mathsf{DH}_g(h_1, h_2)$ efficiently. However, by choosing uniform h_1 with $\log_g h_1$ known (which

can be done by choosing uniform $x_1 \in \mathbb{Z}_q$ and setting $h_1 := g^{x_1}$) it becomes easy to perform.

DEFINITION *We say that* the CDH problem is hard relative to $\mathcal{G}$ *if for all probabilistic polynomial-time algorithms $\mathcal{A}$ there exists a negligible function* negl *such that*

$$\Pr[\mathsf{CDH}_{\mathcal{A},\mathcal{G}}(n) = 1] \leq \mathsf{negl}(n).$$

Assume the CDH problem is hard relative to $\mathcal{G}$. Let $\mathcal{A}$ be a probabilistic polynomial-time algorithm and set $\varepsilon(n) \stackrel{\text{def}}{=} \Pr[\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(n) = 1]$. We show how $\mathcal{A}$ can be used by a probabilistic polynomial-time algorithm $\mathcal{A}'$ to solve the CDH problem with success probability $\varepsilon(n)$:

Algorithm $\mathcal{A}'$:
The algorithm is given $\mathbb{G}, q, g, h_1, h_2$ as input.

(a) Run $\mathcal{A}(\mathbb{G}, q, g, h_1)$ and obtain output x_1.
(b) Return $h_2^{x_1}$.

Observe that the input given to $\mathcal{A}$ when run as a subroutine by $\mathcal{A}'$ is distributed exactly as in experiment $\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(n)$. So with probability $\varepsilon(n)$ it holds that $x_1 = \log_g h_1$. Whenever this occurs, $\mathcal{A}'$ outputs the correct answer $\mathsf{DH}_g(h_1, h_2)$. Since the CDH problem was assumed to be hard, this implies that $\varepsilon(n)$ is negligible and so the discrete logarithm problem must be hard as well.

We now proceed to prove that the hardness of the DDH problem relative to $\mathcal{G}$ implies the hardness of the CDH problem relative to $\mathcal{G}$. Let $\mathcal{A}$ be a probabilistic polynomial-time algorithm and set $\varepsilon(n) \stackrel{\text{def}}{=} \Pr[\mathsf{CDH}_{\mathcal{A},\mathcal{G}}(n) = 1]$. We show how $\mathcal{A}$ can be used by a probabilistic polynomial-time algorithm $\mathcal{A}'$ to solve the DDH problem with roughly the same probability:

Algorithm $\mathcal{A}'$:
The algorithm is given $\mathbb{G}, q, g, h_1, h_2, h_3$ as input.

(a) Run $\mathcal{A}(\mathbb{G}, q, g, h_1, h_2)$ and obtain output h_3'.
(b) If $h_3' = h_3$ output 1; else output 0.

We need to analyze the probability that $\mathcal{A}'$ outputs 1 when the final component of its input h_3 is equal to $\mathsf{DH}_g(h_1, h_2)$, as compared to the probability that it outputs 1 when h_3 is chosen uniformly at random from $\mathbb{G}$. The input given to $\mathcal{A}$ when run as a subroutine by $\mathcal{A}'$ is distributed exactly as in experiment $\mathsf{CDH}_{\mathcal{A},\mathcal{G}}(n)$. So with probability $\varepsilon(n)$, it holds that $h_3' = \mathsf{DH}_g(h_1, h_2)$. It follows that when $h_3 = \mathsf{DH}_g(h_1, h_2)$, algorithm $\mathcal{A}'$ outputs 1 with probability at least $\varepsilon(n)$.

On the other hand, when h_3 is chosen uniformly at random from $\mathbb{G}$, then regardless of the behavior of $\mathcal{A}'$ the probability that $h'_3 = h_3$ is exactly $1/|\mathbb{G}| = 1/q$. Thus,

$$\left|\Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] - \Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]\right| \geq \varepsilon(n) - 1/q.$$

Since the above must be negligible by the assumption that the DDH problem is hard, and $1/q$ is negligible, it follows that $\varepsilon(n)$ must be negligible. We conclude that the CDH problem is hard as well.

8.16 Determine the points on the elliptic curve $E : y^2 = x^3 + 2x + 1$ over $\mathbb{Z}_{11}$. How many points are on this curve?

Solution: This simply involves finding all values of x for which x^3+2x+1 is either 0 or a quadratic residue modulo 11. The quadratic residues modulo 11 are 1, 3, 4, 5, and 9. So we have the points $(0, 1)$, $(0, 10)$, $(1, 2)$, $(1, 9)$, $(3, 1)$, $(3, 10)$, $(5, 2)$, $(5, 9)$, $(6, 3)$, $(6, 8)$, $(8, 1)$, $(8, 10)$, $(9, 0)$, $(10, 3)$, $(10, 8)$, plus the point at infinity.

8.17 Consider the elliptic-curve group from Example 8.67. (See also Example 8.69.) Compute $(1, 0)+(4, 3)+(4, 3)$ in this group by first converting to projective coordinates and then using Equations (8.3) and (8.4).

Solution: In projective coordinates these points can be represented as $(1, 0, 1)$, $(4, 3, 1)$, and $(4, 3, 1)$. To compute $(1, 0, 1) + (4, 3, 1)$ we first compute $u = 3$, $v = 3$, and $w = 6$. Then

$$(1, 0, 1) + (4, 3, 1) = (4, 2, 6) = (-4, -2, -6) = (3, 5, 1).$$

To compute $(3, 5, 1)+(4, 3, 1)$ we proceed similarly. The answer, in affine coordinates, is $(4, 4)$ in agreement with Exercise 8.69.

8.18 Prove the fourth statement in Proposition 8.68.

Solution: To compute the slope of the line tangent to E at $P_1 = P_2 = (x_1, y_1)$, we first compute the partial derivatives of E as

$$2y \cdot dy = 3x^2 \cdot dx + A \cdot dx,$$

so that the slope at (x_1, y_1) is given by

$$m \stackrel{\text{def}}{=} \frac{dy}{dx}(x_1, y_1) = \left[\frac{3x_1^2 + A}{2y_1} \bmod p\right].$$

The rest of the proof follows the derivation of Proposition 8.68 verbatim, using the above as the value of m.

8.19 Can the following problem be solved in polynomial time? Given a prime p, a value $x \in \mathbb{Z}^*_{p-1}$, and $y := [g^x \bmod p]$ (where g is a uniform value in $\mathbb{Z}^*_p$), find g, i.e., compute $y^{1/x} \bmod p$. If your answer is "yes," give a polynomial-time algorithm. If your answer is "no," show a reduction to one of the assumptions introduced in this chapter.

Solution: The problem can be solved efficiently. Since $x \in \mathbb{Z}^*_{p-1}$ (and $p-1$ is known), we can compute $d := [x^{-1} \bmod p-1]$. Then

$$y^d = (g^x)^d = g^{xd} = g^{[xd \bmod p-1]} = g \bmod p.$$

8.20 Let GenRSA be as in Section 8.2.4. Prove that if the RSA problem is hard relative to GenRSA then the construction shown below is a fixed-length collision-resistant hash function.

CONSTRUCTION 8.3-S

Define (Gen, H) as follows:

- Gen: on input 1^n, run $\mathsf{GenRSA}(1^n)$ 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 $f^s_0(x) \stackrel{\text{def}}{=} [x^e \bmod N]$ and $f^s_1(x) \stackrel{\text{def}}{=} [y \cdot x^e \bmod N]$. For a $3n$-bit long string $x = x_1 \cdots x_{3n}$, define

$$H^s(x) \stackrel{\text{def}}{=} f^s_{x_{3n}}\left(f^s_{x_{3n-1}}\left(\cdots\left(1\right)\cdots\right)\right).$$

Solution: We show that an eth root of y can be computed from any collision; given this, a formal proof follows along the lines of the proof of Theorem 8.79.

Fix $s = \langle N, e, y \rangle$, and assume $y \neq 1$ (computing an eth root of y is trivial if $y = 1$). Let x, z be distinct inputs with $H^s(x) = H^s(z)$. Define $x^0 = 1$ and, for for $i = 1$ to $3n$, inductively define $x^i = f^s_{x_i}(x^{i-1})$ (note that $x^1, \ldots, x^{3n}$ is the sequence of values produced when computing $H^s(x)$). Define z^i analogously. Since $x \neq z$, there is some index i with $x^i \neq z^i$ (in particular, take i to be the first position where the strings x and z differ, i.e., where $x_i \neq z_i$). But since $x^{3n} = z^{3n}$, there must be an index $j < 3n$ with

$$x^j \neq z^j \quad \text{but} \quad x^{j+1} = z^{j+1}.$$

Since f^s_0, f^s_1 are permutations (this can be easily verified), it must be the case that $x_{j+1} \neq z_{j+1}$. Assume without loss of generality that $x_{j+1} = 0$. The above shows that given a collision we can efficiently

compute distinct elements $X, Z \in \mathbb{Z}_N^*$ with $f_0^s(X) = f_1^s(Z)$. Then

$$X^e = y \cdot Z^e \bmod N,$$

and we see that $X/Z \bmod N$ is an eth root of y.

8.21 Consider the following generalization of Construction 8.78:

CONSTRUCTION 8.4-S

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 $H^s(x_1, \ldots, x_t) := \prod_i h_i^{x_i}$.

(a) Prove that if the discrete logarithm problem is hard relative to $\mathcal{G}$, and q is prime, then for any $t = \mathsf{poly}(n)$ this construction is a fixed-length collision-resistant hash function.

(b) 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).

Solution:

(a) The easiest way to prove this is by reducing to the security of Construction 8.78. That is, we show that any algorithm $\mathcal{A}$ finding a collision in Construction 8.4-S can be used as a subroutine to find collisions in Construction 8.78.

Let $\Pi = (\mathsf{Gen}, H)$ denote Construction 8.4-S and let $\widehat{\Pi} = (\widehat{\mathsf{Gen}}, \widehat{H})$ denote Construction 8.78. Given a probabilistic polynomial-time algorithm $\mathcal{A}$ attacking Π, let $\varepsilon(n) \stackrel{\text{def}}{=} \Pr[\mathsf{Hash\text{-}coll}_{\mathcal{A},\Pi}(n) = 1]$. Construct algorithm $\hat{\mathcal{A}}$ attacking $\widehat{\Pi}$ as follows:

Algorithm $\hat{\mathcal{A}}$:

The algorithm is given $\hat{s} = \langle \mathbb{G}, q, g, h \rangle$ as input.

i. Set $h_1 := g$ and let $\beta_1 := 1$. Choose $i \leftarrow \{2, \ldots, t\}$ and set $h_i := h$.

ii. For $j \in \{2, \ldots, t\} \setminus \{i\}$, choose $\beta_j \leftarrow \mathbb{Z}_q$ and set $h_j := g^{\beta_j}$. Set $s := \langle \mathbb{G}, q, (h_1, \ldots, h_t) \rangle$.

iii. Run $\mathcal{A}(s)$ and obtain outputs $(x_1, \ldots, x_t)$ and $(x'_1, \ldots, x'_t)$.

iv. If $x_i \neq x'_i$ then

A. Set $X_1 := \sum_{j \neq i} \beta_j \cdot x_j$ and $X_2 := x_i$.

B. Set $X'_1 := \sum_{j \neq i} \beta_j \cdot x'_j$ and $X'_2 := x'_i$.

v. Output (X_1, X_2) and (X'_1, X'_2).

We first claim that whenever $(x_1, \ldots, x_t)$ and $(x'_1, \ldots, x'_t)$ form a collision in H^s with $x_i \neq x'_i$, then $\hat{\mathcal{A}}$ outputs a collision in $\widehat{H}^{\hat{s}}$. To see this, note that

$$\begin{aligned} H^s(x_1, \ldots, x_t) &= \textstyle\prod_j h_j^{x_j} \\ &= h^{x_i} \cdot \textstyle\prod_{j \neq i} \left(g^{\beta_j}\right)^{x_j} \\ &= h^{x_i} \cdot g^{\sum_{j \neq i} \beta_j \cdot x_j} \\ &= g^{X_1} \cdot h^{X_2} \;\; = \;\; \widehat{H}^{\hat{s}}(X_1, X_2), \end{aligned}$$

and similarly $H^s(x'_1, \ldots, x'_t) = \widehat{H}^{\hat{s}}(X'_1, X'_2)$. Since the values output by $\mathcal{A}$ collide, $\widehat{H}^{\hat{s}}(X_1, X_2) = \widehat{H}^{\hat{s}}(X'_1, X'_2)$; since $x_i \neq x'_i$ we have $X'_1 \neq X'_2$ and so (X_1, X_2) and (X'_1, X'_2) are a collision in $\widehat{H}^{\hat{s}}$.

We now analyze the probability that $\hat{\mathcal{A}}$ outputs a collision. First note that the distribution on the inputs given to $\mathcal{A}$ when run as a subroutine by $\hat{\mathcal{A}}$ is identical to the distribution in experiment $\mathsf{Hash\text{-}coll}_{\mathcal{A},\Pi}(n)$, and $\mathcal{A}$ therefore outputs a collision $(x_1, \ldots, x_t)$, $(x'_1, \ldots, x'_t)$ with probability exactly $\varepsilon(n)$. We claim that for at least one value $k \in \{2, \ldots, t\}$ we have $x_k \neq x'_k$. Since $(x_1, \ldots, x_t)$ and $(x'_1, \ldots, x'_t)$ are distinct, the only other possibility is that $x_1 \neq x'_1$; in this case, however, it follows from the fact that $\prod_j h_j^{x_j} = \prod_j h_j^{x'_j}$ that $(x_2, \ldots, x_t)$ and $(x'_2, \ldots, x'_t)$ cannot all be the same. (Note that $h_1 \neq 1$ since it is a generator of $\mathbb{G}$.)

$\mathcal{A}$ has no information about $\hat{\mathcal{A}}$'s choice of i, since each element $h_2, \ldots, h_t$ is a uniform element in $\mathbb{G}$. It follows that $x_i \neq x'_i$ with probability at least $1/(t-1)$. Combining this with the argument given earlier, we see that $\hat{\mathcal{A}}$ outputs a collision with probability at least $\varepsilon(n)/(t-1)$. Since this must be negligible (by security of $\widehat{\Pi}$) and $t = \mathsf{poly}(n)$, it follows that $\varepsilon(n)$ must be negligible as well.

(b) The input length of Construction 8.4-S can be $t \cdot (n-1)$ (since $\|q\| = n$), and the output length is given by the length of a bit-string needed to represent an element of $\mathbb{G}$. The latter is $p(n)$ for some polynomial p. Setting $t > p(n)/(n-1)$, we obtain compression.

Chapter 9

Algorithms for Factoring and Computing Discrete Logarithms – Solutions

9.1 In order to speed up the key generation algorithm for RSA, it has been suggested to generate a prime by generating many small random primes, multiplying them together and adding one (of course, then checking that the result is prime). Ignoring the question of the probability that such a value really is prime, what do you think of this method?

Solution: This is not a good idea, since if a prime p is generated using this method then $p-1$ will have small prime factors and so Pollard's $p-1$ algorithm can be applied.

9.2 In an execution of Algorithm 9.2, define $x^{(i)} \stackrel{\text{def}}{=} F^{(i)}(x^{(0)})$. Show that if, in a given execution of the algorithm, there exist $i, j \leq 2^{n/2}$ such that $x^{(i)} \neq x^{(j)}$ but $x^{(i)} = x^{(j)} \bmod p$, then that execution of the algorithm outputs p with overwhelming probability.

Solution: Let $x_p^{(i)} \stackrel{\text{def}}{=} [x^{(i)} \bmod p]$. As in the proof of Claim 5.10, if $x_p^{(I)} = x_p^{(J)}$ for $1 \leq I < J \leq 2^{n/2}$ then there is an $i < J$ such that $x_p^{(i)} = x_p^{(2i)}$. In addition (as noted already in Section 9.1.2), $x^{(i)} \neq x^{(2i)}$ with overwhelming probability if we model F as a random function

In iteration i of Algorithm 9.2, x is set equal to $x^{(i)}$ and x' is set equal to $x^{(2i)}$. Thus, iteration i of the algorithm will output p with overwhelming probability.

9.3 (a) Show that if $ab = c \bmod N$ and $\gcd(b, N) = d$, then:

i. $d \mid c$;

ii. $a \cdot (b/d) = (c/d) \bmod (N/d)$; and

iii. $\gcd(b/d, N/d) = 1$.

(b) Describe how to use the above to compute discrete logarithms in $\mathbb{Z}_N$ efficiently even when the base g is not a generator of $\mathbb{Z}_N$.

Solution:

(a) Since $ab = c \bmod N$ we have $ab - \alpha N = c$ for some integer α. Since $d \mid b$ and $d \mid N$, it follows immediately that $d \mid c$, proving (i).

Furthermore, $a(b/d) - \alpha(N/d) = c/d$ proving (ii). To prove (iii), let d' be a positive integer dividing both b/d and N/d. Then dd' divides both b and N. Since $d = \gcd(b, N)$, this implies $d' = 1$.

(b) Say we are given N along with $g, y \in \mathbb{Z}_N$, and want to compute $x \in \{0, \ldots, N-1\}$ such that $x \cdot g = y \bmod N$. If g is a generator then, as discussed in Section 8.2.3, we could compute $x := [y \cdot g^{-1} \bmod N]$. If g is not a generator then $\gcd(g, N) = d > 1$. If $d \not| y$ then, by (i), we know that no solution exists. If $d \,|\, y$, then by (ii) we need only solve the equation

$$x \cdot (g/d) = (y/d) \bmod (N/d).$$

By (iii), we have $\gcd(g/d, N/d) = 1$ and so we can compute the solution $x = [(y/d) \cdot (g/d)^{-1} \bmod (N/d)]$.

9.4 Here we show how to solve the discrete logarithm problem in a cyclic group of order $q = p^e$ in time $\mathcal{O}(\mathsf{polylog}(q) \cdot \sqrt{p}\,)$. Given as input a generator g of known order p^e and a value y, we want to compute $x = \log_g y$.

(a) Show how to find $[x \bmod p]$ in time $\mathcal{O}(\mathsf{polylog}(q) \cdot \sqrt{p}\,)$.

(b) Say $x = x_0 + x_1 \cdot p + \cdots + x_{e-1} \cdot p^{e-1}$ with $0 \leq x_i < p$. In the previous step we determined x_0. Show how to compute in $\mathsf{polylog}(q)$ time a value y_1 such that $(g^p)^{x_1 + x_2 \cdot p + \cdots + x_{e-1} \cdot p^{e-2}} = y_1$.

(c) Use recursion to obtain the claimed running time for the original problem. (Note that $e = \log(q)$.)

Solution:

(a) As suggested by the hint, set $g_0 := g^{p^{e-1}}$ and $y_0 := y^{p^{e-1}}$. Note that $g_0^x = y_0$, and so there must exist an $x_0 \in \{0, \ldots, \mathsf{ord}(g_0) - 1\}$ such that $g_0^{x_0} = y_0$. Since $\mathsf{ord}(g_0) = p^e / p^{e-1} = p$ (cf. Lemma 8.7), it follows from Proposition 7.50 that $x_0 = [x \bmod p]$. Working in the group of order p generated by g_0, we can compute x_0 in time $\sqrt{p}$ using, e.g, the baby-step/giant-step algorithm.

(b) Setting $y_1 := y \cdot g^{-x_0}$ works. Thus is so since

$$\begin{aligned} y \cdot g^{-x_0} &= g^{x_0 + x_1 \cdot p + \cdots + x_{e-1} \cdot p^{e-1}} \cdot g^{-x_0} \\ &= (g^p)^{x_1 + x_2 \cdot p + \cdots + x_{e-1} \cdot p^{e-2}}. \end{aligned}$$

(c) Note that $\mathsf{ord}(g^p) = p^{e-1}$, and so the group $\mathbb{G}_1$ generated by g^p has order p^{e-1}. Letting $x' \stackrel{\text{def}}{=} x_1 + x_2 \cdot p + \cdots + x_{e-1} \cdot p^{e-2}$, we can apply the result from part (a) in group $\mathbb{G}_1$ to compute $[x' \bmod p] = x_1$. Repeatedly applying parts (b) and (a) thus allows us to compute $x_2, \ldots, x_{e-1}$ and hence all of x. The computation requires a total of $\mathcal{O}(e) = \mathcal{O}(\log q)$ computations, each taking time $\mathcal{O}(\mathsf{polylog}(q) \cdot \sqrt{p}\,)$, giving the claimed running time.

9.5 Let q have prime factorization $q = \prod_{i=1}^{k} p_i^{e_i}$. Using the result from the previous problem, show a modification of the Pohlig-Hellman algorithm that solves the discrete logarithm problem in a group of order q in time $\mathcal{O}\left(\mathsf{polylog}(q) \cdot \sum_{i=1}^{k} e_i\sqrt{p_i}\right) = \mathcal{O}\left(\mathsf{polylog}(q) \cdot \max\{\sqrt{p_i}\}\right)$.

Solution: Using the Pohlig-Hellman algorithm, a discrete-logarithm computation in a group of order q reduces to k discrete-logarithm computations in groups of order $p_1^{e_1}, \ldots, p_k^{e_k}$. The previous exercise shows that discrete logarithms in a group of order p^e can be computed in time $\mathcal{O}(\mathsf{polylog}(p^e) \cdot \sqrt{p})$. Thus, the total computation is

$$\mathcal{O}\left(\sum_{i=1}^{k} \mathsf{polylog}(p_i^{e_i}) \cdot \sqrt{p_i}\right).$$

Since $p_i^{e_i} = \mathcal{O}(q)$ and $k = \mathcal{O}(\log q)$, this simplifies to a complexity of $\mathcal{O}\left(\mathsf{polylog}(q) \cdot \max_i\{\sqrt{p_i}\}\right)$.

9.6 Give pseudocode for the small-space algorithm for computing discrete logarithms described in Section 9.2.3, and give a heuristic analysis of the probability with which it succeeds.

Solution: Let $g, h \in \mathbb{G}$ be given; we want to compute $\log_g h$. To do this, define $F : \mathbb{G} \to \mathbb{Z}_q \times \mathbb{Z}_q$ (modeled as a random function), $H_{g,h} : \mathbb{Z}_q \times \mathbb{Z}_q \to \mathbb{G}$, and $H = H_{g,h} \circ F : \mathbb{G} \to \mathbb{G}$ as in Section 9.2.3. Use Algorithm 5.9 to find a collision k, k' in H using $\sqrt{q}$ invocations of H, and hence $\sqrt{q}$ invocations of F. With overwhelming probability, $F(k) \neq F(k')$; this follows from Lemma A.15 because the range of F has size q^2. Thus, $F(k), F(k')$ are a collision in $H_{g,h}$, and as discussed in Theorem 8.79 allow computation of $\log_g h$.

Chapter 10

Key Management and the Public-Key Revolution – Solutions

10.1 Let Π be a key-exchange protocol, and $(\mathsf{Enc}, \mathsf{Dec})$ be a private-key encryption scheme. Consider the following *interactive* protocol Π' for encrypting a message: first, the sender and receiver run 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 decrypts and recovers 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 $(\mathsf{Enc}, \mathsf{Dec})$ has indistinguishable encryptions in the presence of an eavesdropper, then Π' satisfies your definition.

Solution:

(a) Consider the following experiment, defined for any interactive protocol Π' (that is in turn composed of a key-exchange protocol Π and a private-key encryption scheme $\Pi_{\mathsf{enc}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$), any adversary $\mathcal{A}$, and any value n for the security parameter:

> **The eavesdropping indistinguishability experiment $\mathsf{KE}\text{-}\mathsf{Enc}_{\mathcal{A},\Pi,\Pi_{\mathsf{enc}}}(n)$:**
>
> *(a) The adversary $\mathcal{A}$ is given input 1^n, and outputs a pair of messages m_0, m_1 of the same length.*
>
> *(b) Two parties holding 1^n execute protocol Π, which results in a transcript trans and a key k that is output by each of the parties.*
>
> *(c) A random bit $b \leftarrow \{0,1\}$ is chosen, and then a ciphertext $c \leftarrow \mathsf{Enc}_k(m_b)$ is computed.*
>
> *(d) $\mathcal{A}$ is given trans and c, and outputs a bit b'.*
>
> *(e) The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise.*

DEFINITION *Interactive protocol Π', composed of Π and Π_{enc}, has* indistinguishable encryptions in the presence of an eavesdropper *if for all*

probabilistic polynomial-time adversaries $\mathcal{A}$ there exists a negligible function negl *such that*

$$\Pr\left[\mathsf{KE\text{-}Enc}_{\mathcal{A},\Pi,\Pi_{\mathsf{enc}}}(n)=1\right] \leq \frac{1}{2}+\mathsf{negl}(n),$$

where the probability is taken over the random coins used by $\mathcal{A}$, as well as the random coins used in the experiment.

(b) Fix an arbitrary probabilistic polynomial-time adversary $\mathcal{A}$. Our goal is to prove that there exists a negligible function negl such that

$$\Pr\left[\mathsf{KE\text{-}Enc}_{\mathcal{A},\Pi,\Pi_{\mathsf{enc}}}(n)=1\right] \leq \frac{1}{2}+\mathsf{negl}(n).$$

Let us write

$$\Pr\left[\mathsf{KE\text{-}Enc}_{\mathcal{A},\Pi,\Pi_{\mathsf{enc}}}(n)=1\right] = \Pr_{(\mathsf{trans},k)\leftarrow\Pi}[\mathcal{A}(\mathsf{trans},\mathsf{Enc}_k(m_b))=b],$$

where b is chosen uniformly at random, trans is generated by running an execution of protocol Π, and k is the key resulting from that execution. (To emphasize the latter, we include $(\mathsf{trans},k)\leftarrow\Pi$ as a subscript.)

We first prove that there is a negligible function negl_1 such that

$$\begin{aligned}&\Pr_{(\mathsf{trans},k)\leftarrow\Pi}[\mathcal{A}(\mathsf{trans},\mathsf{Enc}_k(m_b))=b] \\ &\quad\leq \Pr_{(\mathsf{trans},k)\leftarrow\Pi}[\mathcal{A}(\mathsf{trans},\mathsf{Enc}_{k'}(m_b))=b]+\mathsf{negl}_1(n),\end{aligned} \tag{10.1}$$

where k' is now chosen uniformly at random, *independently* of trans. The above follows from the assumed security of Π. We then show that there exists a negligible function negl_2 with

$$\Pr_{(\mathsf{trans},k)\leftarrow\Pi}[\mathcal{A}(\mathsf{trans},\mathsf{Enc}_{k'}(m_b))=b] \leq \frac{1}{2}+\mathsf{negl}_2(n). \tag{10.2}$$

This follows from security of Π_{enc}. Taken together, Equations (10.1) and (10.2) prove the desired result.

We now prove Equation (10.1). Consider the following adversary $\mathcal{A}_1$ that eavesdrops on an execution of Π:

Adversary $\mathcal{A}_1$:

(a) $\mathcal{A}_1$ is given a transcript trans and a key $\hat{k}$. Let $n=|\hat{k}|$.
(b) $\mathcal{A}_1$ runs $\mathcal{A}(1^n)$ to obtain two messages (m_0,m_1).
(c) $\mathcal{A}_1$ chooses a bit $b\leftarrow\{0,1\}$ and computes $c\leftarrow\mathsf{Enc}_{\hat{k}}(m_b)$. It then runs $\mathcal{A}(\mathsf{trans},c)$ to obtain a bit b'.
(d) If $b'=b$ then $\mathcal{A}_1$ outputs 1; otherwise, $\mathcal{A}_1$ outputs 0.

Consider the experiment $\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A}_1,\Pi}(n)$. When the random bit (call it b_{ke}) used in *that* experiment is equal to 1, then $\hat{k}$ is equal to the actual key generated by the execution of Π resulting in trans. Furthermore, in this case the experiment evaluates to 1 exactly when $\mathcal{A}_1$ outputs 1 (since this implies that $\mathcal{A}_1$ has correctly guessed the value of $b_{\mathsf{ke}} = 1$). Since this occurs exactly when $\mathcal{A}$ correctly guesses the value of b, we have

$$\Pr[\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A}_1,\Pi}(n) = 1 \mid b_{\mathsf{ke}} = 1] = \Pr_{(\mathsf{trans},k)\leftarrow\Pi}[\mathcal{A}(\mathsf{trans}, \mathsf{Enc}_k(m_b)) = b].$$

On the other hand, when $b_{\mathsf{ke}} = 0$ then $\hat{k}$ is a uniformly random key, chosen independently of anything else. In this case, the experiment evaluates to 1 if and only if $\mathcal{A}_1$ outputs 0. Since this occurs only when $\mathcal{A}$ correctly fails to guess the value of b, we have

$$\Pr\left[\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A}_1,\Pi}(n) = 1 \mid b_{\mathsf{ke}} = 0\right] = 1 - \Pr_{(\mathsf{trans},k)\leftarrow\Pi}\left[\mathcal{A}(\mathsf{trans}, \mathsf{Enc}_{k'}(m_b)) = b\right].$$

Altogether, then, we have

$$\begin{aligned}
&\Pr[\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A}_1,\Pi}(n) = 1] \\
&= \frac{1}{2} \cdot \Pr[\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A}_1,\Pi}(n) = 1 \mid b_{\mathsf{ke}} = 1] + \frac{1}{2} \cdot \Pr[\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A}_1,\Pi}(n) = 1 \mid b_{\mathsf{ke}} = 0] \\
&= \frac{1}{2} + \frac{1}{2} \cdot \left(\Pr_{(\mathsf{trans},k)\leftarrow\Pi}[\mathcal{A}(\mathsf{trans}, \mathsf{Enc}_k(m_b)) = b] \right. \\
&\qquad\qquad \left. - \Pr_{(\mathsf{trans},k)\leftarrow\Pi}[\mathcal{A}(\mathsf{trans}, \mathsf{Enc}_{k'}(m_b)) = b] \right).
\end{aligned}$$

But since Π is secure in the presence of an eavesdropper, we know there exists a negligible function negl'_1 such that

$$\Pr[\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A}_1,\Pi}(n) = 1] \leq \frac{1}{2} + \mathsf{negl}'_1(n).$$

These last two equations imply Equation (10.1).

We now prove Equation (10.2). Consider the following adversary $\mathcal{A}_2$, attacking Π_{enc} in the sense of Definition 3.8:

Adversary $\mathcal{A}_2$:

(a) $\mathcal{A}_2$ is given 1^n, and runs $\mathcal{A}(1^n)$ to obtain two messages (m_0, m_1). These same messages are output by $\mathcal{A}_2$.

(b) $\mathcal{A}_2$ is given a ciphertext c.

(c) $\mathcal{A}_2$ runs Π itself, playing the roles of both Alice and Bob, to generate a transcript trans and a key k. (It will ignore k.)

(d) $\mathcal{A}_2$ runs $\mathcal{A}(\mathsf{trans}, c)$ to obtain a bit b. This same bit is output by $\mathcal{A}_2$.

In experiment $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A}_2,\Pi_{\mathsf{enc}}}(n)$, the key k' used (by the experiment) to generate the ciphertext c is indeed chosen uniformly and independently of trans. Furthermore, $\mathcal{A}$ is given an encryption of m_0 (resp., m_1) exactly when $\mathcal{A}_2$ is given an encryption of m_0 (resp., m_1), and $\mathcal{A}_2$ outputs as its guess whatever is output by $\mathcal{A}$. Thus,

$$\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A}_2,\Pi_{\mathsf{enc}}}(n) = 1] = \Pr_{(\mathsf{trans},k)\leftarrow\Pi}[\mathcal{A}(\mathsf{trans}, \mathsf{Enc}_{k'}(m_b)) = b].$$

Since Π_{enc} satisfies Definition 3.8, we know that there exists a negligible function negl_2 such that

$$\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A}_2,\Pi_{\mathsf{enc}}}(n) = 1] \leq \frac{1}{2} + \mathsf{negl}_2(n).$$

Equation (10.2) follows.

10.2 Show that, for either of the groups considered in Sections 8.3.3 or 8.3.4, a uniform group element (expressed using the natural representation) is easily distinguishable from a uniform bit-string of the same length.

Solution: Consider first the case of a subgroup $\mathbb{G}$ of order q of $\mathbb{Z}_p^*$, with $p = rq + 1$. As we have seen in Section 8.3.3, $h \in \mathbb{G}$ if and only if $h^q = 1 \bmod p$. Now, a uniform string of the same length will be in $\mathbb{G}$ with probability *at most* $1/r$, and so can easily be distinguished. However, a much more serious problem arises with the higher order bits of a uniform string. To be concrete, let $p = 23$, with a representation of 5 bits. Then, a uniform 5-bit string will be greater than or equal to p with probability $9/32$, whereas an element of any subgroup of $\mathbb{Z}_{32}^*$ will never be greater than or equal to p. Thus, elements of the group will have the most significant bit equal to 1 with probability significantly lower than a uniform string.

In Section 8.3.4, elements of the group are pairs (x, y) that satisfy the equation $y^2 = x^3 + Ax + B \bmod p$. The number of elements on the curve is $q \stackrel{\text{def}}{=} |E(\mathbb{Z}_p)|$, whereas the number of pairs (x, y) is 2^{2n} (where n denotes the number of bits in the representation of p). By Theorem 8.70, we have $q \leq p + 1 + 2\sqrt{p} \ll 2^{2n}$. Thus, the vast majority of pairs (x, y) will not satisfy the equation, and a uniform string can be easily distinguished from a uniform group element.

10.3 Describe a man-in-the-middle attack on the Diffie–Hellman protocol where the adversary shares a key k_A with Alice and a (different) key k_B with Bob, and Alice and Bob cannot detect that anything is wrong.

Solution: The man-in-the-middle attack consists of simply running independent executions of the protocol with Alice and with Bob. (I.e.,

the adversary plays the role of Bob when interacting with Alice, and plays the role of Alice when interacting with Bob.) This results in a key k_A output by Alice and a key k_B output by Bob, both of which are known to the adversary. Since the adversary ran the protocol honestly with each party, neither party can detect that anything is amiss.

10.4 Consider the following key-exchange protocol:

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

(b) Bob chooses uniform $t \in \{0,1\}^n$, 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 outputs $w \oplus t$.

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).

Solution: Alice outputs k, while Bob outputs

$$w \oplus t = (u \oplus r) \oplus t = ((s \oplus t) \oplus r) \oplus t = (((k \oplus r) \oplus t) \oplus r) \oplus t = k.$$

The scheme, however, is not secure. Given a transcript (s, u, w) of an execution of the protocol, an adversary can compute $s \oplus u \oplus w$ and this is equal to the key since

$$s \oplus u \oplus w = (k \oplus r) \oplus u \oplus (u \oplus r) = k.$$

Chapter 11

Public-Key Encryption – Solutions

11.1 Assume a public-key encryption scheme for single-bit messages with no decryption error. 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.

Solution: This is just a trivial brute-force search. We describe the attack assuming that m is a single bit, but the attack generalizes easily to m of arbitrary length or even m of unknown length. The attack is as follows: let $\ell = \ell(n)$ denote the length of the random coins used by Enc. Given pk and c, compute $c_r := \mathsf{Enc}_{pk}(0; r)$ for all $r \in \{0,1\}^\ell$. If $c = c_r$ for some r, then output "$m = 0$"; else, output "$m = 1$." This adversary is clearly far from efficient, but that is ok since in this exercise we allow the adversary to be unbounded.

If c is an encryption of 0 then $c = \mathsf{Enc}_{pk}(0; r)$ for some r and so the above adversary always (correctly) outputs "$m = 0$." But if c is an encryption of 1 then $c \neq c_r$ for all r. (This follows from correctness.) So in this case the adversary always (correctly) outputs "$m = 1$."

11.2 Show that for any CPA-secure public-key encryption scheme for single-bit messages, the length of the ciphertext must be superlogarithmic in the security parameter.

Solution: We aim for simplicity rather than trying to optimize parameters. Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. For a given public key pk generated using Gen and a bit b, let $\mathcal{C}_{pk}(b)$ denote the set of ciphertexts corresponding to possible encryptions of b with respect to pk. Say that for some fixed bit b the length of the ciphertext after encrypting b is at most logarithmic in the security parameter (for every public key). Then for any public key, the set $\mathcal{C}_{pk}(b)$ has size at most $p(n)$ for some polynomial p. We exploit this in the following attack:

Adversary $\mathcal{A}$:

(a) $\mathcal{A}$ is given pk, and computes $c_0 \leftarrow \mathsf{Enc}_{pk}(0)$ and $c_1 \leftarrow \mathsf{Enc}_{pk}(1)$.

(b) $\mathcal{A}$ outputs the two messages $(0, 1)$, and is given in return a ciphertext c.

(c) If $c = c_0$ then $\mathcal{A}$ outputs 0; if $c = c_1$ then $\mathcal{A}$ outputs 1; otherwise $\mathcal{A}$ outputs a random bit.

As above, let b be the bit for which $|\mathcal{C}_{pk}(b)| \leq p(n)$ (if this holds for both 0 and 1 then set b arbitrarily). Let $c_b^* \in \mathcal{C}_{pk}(b)$ be the ciphertext that occurs most frequently when encrypting b, and note that $\Pr[\mathsf{Enc}_{pk}(b) = c_b^*] \geq 1/|\mathcal{C}_{pk}(b)|$. We now analyze the behavior of $\mathcal{A}$.

$$\begin{aligned}\Pr[\mathcal{A} \text{ outputs } b \mid c \text{ is an encryption of } b] &= \Pr[c = c_b] + \frac{1}{2} \cdot \Pr[c \neq c_b] \\ &= \frac{1}{2} + \frac{1}{2} \cdot \Pr[c = c_b] \\ &= \frac{1}{2} + \frac{1}{2} \cdot \Pr[c = c_b^* \bigwedge c_b = c_b^*] \\ &\geq \frac{1}{2} + \frac{1}{2p^2(n)}.\end{aligned}$$

(In fact, one can show that $\Pr[c = c_b] \geq 1/|\mathcal{C}_{pk}(b)| \geq 1/p(n)$ but this is not needed for the proof.) We also have

$$\begin{aligned}&\Pr[\mathcal{A} \text{ outputs } \bar{b} \mid c \text{ is an encryption of } \bar{b}\,] \\ &= \Pr[c = c_{\bar{b}}] + \frac{1}{2} \cdot \Pr[c \neq c_{\bar{b}}] \;\; \geq \;\; \frac{1}{2}.\end{aligned}$$

From the above, we have

$$\begin{aligned}\Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] &= \frac{1}{2} \cdot \Pr[\mathcal{A} \text{ outputs } b \mid c \text{ is an encryption of } b\,] \\ &\quad + \frac{1}{2} \cdot \Pr[\mathcal{A} \text{ outputs } \bar{b} \mid c \text{ is an encryption of } \bar{b}\,] \\ &\geq \frac{1}{2} \cdot \left(\frac{1}{2} + \frac{1}{2p^2(n)}\right) + \frac{1}{2} \cdot \frac{1}{2} \;\; = \;\; \frac{1}{2} + \frac{1}{4p^2(n)}.\end{aligned}$$

Since $1/4p^2(n)$ is not negligible, Π is not CPA-secure.

11.3 Say a public-key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ for n-bit messages is *one-way* if any PPT adversary $\mathcal{A}$ has negligible probability of success in the following experiment:

- $\mathsf{Gen}(1^n)$ *is run to obtain keys* (pk, sk).
- *A message* $m \in \{0,1\}^n$ *is chosen uniformly at random, and a ciphertext* $c \leftarrow \mathsf{Enc}_{pk}(m)$ *is computed.*
- $\mathcal{A}$ *is given* pk *and* c*, and outputs a message* m'*. We say* $\mathcal{A}$ *succeeds if* $m' = m$.

(a) Give a construction of a CPA-secure KEM in the random-oracle model based on any one-way public-key encryption scheme.

(b) Can a *deterministic* public-key encryption scheme be one-way? If not, prove impossibility; if so, give a construction based on any of the assumptions introduced in this book.

Solution:

(a) See Construction 11.1-S.

CONSTRUCTION 11.1-S

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme, and let $\ell(n)$ be an arbitrary polynomial. Let H be a function whose domain is $\{0,1\}^*$ and whose range can be set to $\{0,1\}^{\ell(n)}$ for any n. Construct a public-key encryption scheme Π' as follows:

- Gen': on input 1^n, run $\mathsf{Gen}(1^n)$ to obtain (pk, sk). The public key is pk and the private key is sk.
- Enc': on input a public key pk and a message $m \in \{0,1\}^{\ell(n)}$, choose a random $r \leftarrow \{0,1\}^n$ and output the ciphertext

$$\langle \mathsf{Enc}_{pk}(r),\ H(r) \oplus m \rangle .$$

- Dec': on input a private key sk and a ciphertext $\langle c_1, c_2 \rangle$, compute $r := \mathsf{Dec}_{sk}(c_1)$ and then output the message $H(r) \oplus c_2$.

A proof of security is similar-in-spirit to the proof of Theorem 11.38, though simpler because only CPA-security is being shown.

(b) A public-key encryption scheme where encryption is deterministic *can* be one-way. Textbook RSA encryption (Construction 11.26) serves as one example: this scheme is one-way as long as the RSA problem is hard relative to GenRSA.

11.4 Show that any two-round key-exchange protocol (that is, where each party sends a single message) satisfying Definition 10.1 can be converted into a CPA-secure public-key encryption scheme.

Solution: Let Π be a 2-round key-exchange protocol where Alice goes first. A public-key encryption scheme can be constructed as in Construction 11.2-S. Correctness follows from the fact that $\mathsf{msg}_1, \mathsf{msg}_2$ jointly form an execution of Π, and so the sender and receiver generate the same key k. Thus, the receiver obtains the sender's intended message m.

Let Π_{enc} denote Construction 11.2-S. We prove that Π_{enc} has indistinguishable encryptions in the presence of an eavesdropper.

Let $\mathcal{A}$ be a probabilistic, polynomial-time adversary, and define

$$\varepsilon(n) \stackrel{\text{def}}{=} \Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi_{\mathsf{enc}}}(n) = 1].$$

CONSTRUCTION 11.2-S

Let Π be as in the text. Define a public-key encryption scheme as follows:

- Gen: on input 1^n run Alice's side of Π to obtain msg_1 along with some state information s. The public key is msg_1 and the private key is s.
- Enc: on input public key $pk = \mathsf{msg}_1$ and message $m \in \{0,1\}^n$, the sender runs Bob's side of Π to obtain msg_2 and a key $k \in \{0,1\}^n$. Output the ciphertext $\langle \mathsf{msg}_2,\ k \oplus m \rangle$.
- Dec: on input a private key $sk = s$ and a ciphertext $\langle \mathsf{msg}_2, c \rangle$, run Alice's side of Π (using state s and incoming message msg_2) to obtain a key k. Then output the message $m := c \oplus k$.

Public-key encryption from key exchange.

Consider the modified "encryption scheme" $\widetilde{\Pi}$ where Gen is the same as in Π_{enc}, but encryption of a message m with respect to the public key msg_1 is done by running Bob's side of Π to obtain msg_2 but then choosing a random $k \in \{0,1\}^n$ and outputting the ciphertext

$$\langle \mathsf{msg}_2,\ k \oplus m \rangle.$$

Although $\widetilde{\Pi}$ is not actually an encryption scheme (as there is no way for the receiver to decrypt), the experiment $\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\widetilde{\Pi}}(n)$ is still well-defined since that experiment depends only on the key generation and encryption algorithms.

As in the case of the one-time pad, the second component of the ciphertext in scheme $\widetilde{\Pi}$ is a uniform string and, in particular, is independent of the message m being encrypted. The first component of the ciphertext is also independent of m. It follows that

$$\Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] = \frac{1}{2}.$$

Now consider the following PPT algorithm D attacking Π:

Algorithm D:
The algorithm is given $\mathsf{trans} = (\mathsf{msg}_1, \mathsf{msg}_2), k$ as input.

- Set $pk = \mathsf{msg}_1$ and run $\mathcal{A}(pk)$ to obtain two messages m_0, m_1.
- Choose a uniform bit b, and give $\langle \mathsf{msg}_2,\ k \oplus m_b \rangle$ to $\mathcal{A}$.
- When $\mathcal{A}$ outputs b', output 1 if $b' = b$ and 0 otherwise.

Let us analyze the behavior of D. There are two cases to consider:

Case 1: Say the input to D is generated by running Π to obtain a transcript trans and a key k. In this case the view of $\mathcal{A}$ when run as a subroutine by D is distributed identically to $\mathcal{A}$'s view in experiment $\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi_{\mathsf{enc}}}(n)$, and so D outputs 1 in this case with probability $\Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi_{\mathsf{enc}}}(n) = 1]$.

Case 2: Say the input to D is generated by running Π to obtain a transcript trans and a key k', but then choosing k uniformly from $\{0,1\}^n$ and giving D the pair (trans, k). In this case the view of $\mathcal{A}$ when run as a subroutine by D is distributed identically to $\mathcal{A}$'s view in experiment $\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\widetilde{\Pi}}(n)$. Since D outputs 1 exactly when the output b' of $\mathcal{A}$ is equal to b, the probability that D outputs 1 in this case is exactly $\Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] = \frac{1}{2}$.

Since Π is a secure key-exchange protocol, there exists a negligible function negl such that

$$\begin{aligned}
&\frac{1}{2} + \mathsf{negl}(n) \geq \Pr[\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] \\
&= \frac{1}{2} \cdot \Pr[D \text{ outputs } 1 \mid k \text{ is output by } \Pi] \\
&\quad + \frac{1}{2} \cdot \Pr[D \text{ outputs } 0 \mid k \text{ is random}] \\
&= \frac{1}{2} \cdot \Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi_{\mathsf{enc}}}(n) = 1] + \frac{1}{2} \cdot \frac{1}{2}\,.
\end{aligned}$$

We thus conclude that

$$\Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi_{\mathsf{enc}}}(n) = 1] \leq \frac{1}{2} + 2 \cdot \mathsf{negl}(n).$$

Since $\mathcal{A}$ was arbitrary, this proves that Π_{enc} has indistinguishable encryptions in the presence of an eavesdropper.

11.5 Show that Claim 11.7 does not hold in the setting of CCA-security.

Solution: Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a CCA-secure encryption scheme for 1-bit messages, and let $\Pi' = (\mathsf{Gen}, \mathsf{Enc}', \mathsf{Dec}')$ denote the encryption scheme for arbitrary-length messages where

$$\mathsf{Enc}_{pk}(m_1 \cdots m_t) = \mathsf{Enc}_{pk}(m_1), \ldots, \mathsf{Enc}_{pk}(m_t).$$

We show that Π' is *not* CCA-secure. Consider the following adversary:

Algorithm $\mathcal{A}$:

(a) Given pk, output messages $m_0 = 00$ and $m_1 = 11$.
(b) Receive in return a ciphertext $c = c_1, c_2$.

(c) Compute $c_2' \leftarrow \mathsf{Enc}_{pk}(0)$. Submit the ciphertext $c' = c_1, c_2'$ to the decryption oracle, and receive in return the message $b\,0$, with $b \in \{0,1\}$.

(d) Output b.

First note that this is a legal attack: m_0 and m_1 have the same length, and the ciphertext c' submitted to the decryption oracle satisfies $c' \neq c$. It is easy to see that this attacker succeeds in guessing the bit b used by the experiment with probability 1; thus, Π' is not CCA-secure.

11.6 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 uniform $y \in \mathbb{Z}_q$ 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 uniform $y, z \in \mathbb{Z}_q$, 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 $\mathcal{G}$.

Solution: Decryption is easy. Given a ciphertext $\langle c_1, c_2 \rangle$ and secret key x, output 0 iff $c_1^x = c_2$. When the ciphertext is an encryption of 0 then decryption is always correct since

$$c_1^x = (g^y)^x = (g^x)^y = h^y = c_2.$$

When the ciphertext is an encryption of 1 then $c_1^x = g^{yx}$. Unless it happens that $z = xy$, decryption will correctly output 1. But $z = xy$ only with probability $1/q$, which is negligible.

Let Π denote the encryption scheme of the exercise. Security of Π follows fairly immediately from the observation that when $\langle c_1, c_2 \rangle$ is an encryption of 0 then $c_2 = \mathsf{DH}_g(h, c_1)$, whereas if $\langle c_1, c_2 \rangle$ is an encryption of 1 then c_1 and c_2 are independent, random elements of $\mathbb{G}$. Formally, given an adversary $\mathcal{A}$ attacking the encryption scheme we can construct an algorithm $\mathcal{A}'$ solving the DDH problem as follows:

Algorithm $\mathcal{A}'$:
The algorithm is given $\mathbb{G}, q, g, h, c_1, c_2$ as input.

(a) Run $\mathcal{A}$ on the public key $pk = \langle \mathbb{G}, q, g, h \rangle$. Assume for simplicity that $\mathcal{A}$ outputs $(m_0 = 0, m_1 = 1)$.

(b) Give to $\mathcal{A}$ the ciphertext $\langle c_1, c_2 \rangle$. Output whatever bit b' is output by $\mathcal{A}$.

Consider the case when the input to $\mathcal{A}'$ is generated by running $\mathcal{G}(1^n)$ to obtain $(\mathbb{G}, q, g)$, then choosing uniform $x, y, z \in \mathbb{Z}_q$, and finally setting $h := g^x$, $c_1 := g^y$, and $c_2 := g^z$. Then, from the point of view of $\mathcal{A}$ (run as a sub-routine by $\mathcal{A}'$), the experiment is equivalent to $\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$ conditioned on $b = 1$ (i.e., conditioned on the ciphertext $\langle c_1, c_2\rangle$ being an encryption of 1). Furthermore, $\mathcal{A}'$ outputs 1 in this case exactly when $\mathcal{A}$ outputs 1, i.e., when $\mathcal{A}$ succeeds. Thus:

$$\Pr[\mathcal{A}'(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] = \Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1 \mid b = 1].$$

When the input to $\mathcal{A}'$ is generated by running $\mathcal{G}(1^n)$ to obtain $(\mathbb{G}, q, g)$, then choosing uniform $x, y \in \mathbb{Z}_q$, and finally setting $h := g^x$, $c_1 := g^y$, and $c_2 := g^{xy}$ then, from the point of view of $\mathcal{A}$, the experiment is equivalent to $\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$ conditioned on $b = 0$. Again, $\mathcal{A}'$ outputs 1 when $\mathcal{A}$ does; now, however, this means that $\mathcal{A}$ did *not* succeed. So,

$$\Pr[\mathcal{A}'(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1] = 1 - \Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1 \mid b = 0].$$

Putting this together and assuming the DDH problem is hard relative to $\mathcal{G}$, there exists a negligible function negl such that

$$\begin{aligned}
&\mathsf{negl}(n)\\
&\geq \left|\Pr[\mathcal{A}'(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] - \Pr[\mathcal{A}'(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]\right|\\
&= \left|\Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1 \mid b = 1] + \Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1 \mid b = 0] - 1\right|\\
&= 2 \cdot \left|\Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] - \frac{1}{2}\right|.
\end{aligned}$$

Since $\mathcal{A}$ was arbitrary, it follows that Π is CPA-secure.

11.7 Consider the following variant of El Gamal encryption. Let $p = 2q + 1$, let $\mathbb{G}$ be the group of squares modulo p (so $\mathbb{G}$ is a subgroup of $\mathbb{Z}_p^*$ of order q), and let g be a generator of $\mathbb{G}$. The private key is $(\mathbb{G}, g, q, x)$ and the public key is $(\mathbb{G}, g, q, h)$, where $h = g^x$ and $x \in \mathbb{Z}_q$ is chosen uniformly. To encrypt a message $m \in \mathbb{Z}_q$, choose a uniform $r \in \mathbb{Z}_q$, compute $c_1 := g^r \bmod p$ and $c_2 := h^r + m \bmod p$, and let the ciphertext be $\langle c_1, c_2\rangle$. Is this scheme CPA-secure? Prove your answer.

Solution: This scheme is *not* secure. In particular, consider an adversary $\mathcal{A}$ that chooses uniform $m_0, m_1 \in \mathbb{Z}_q$ and then receives the challenge ciphertext $\langle c_1, c_2\rangle$. Observe that since $c_2 = h^y + m \bmod p$ it is not necessarily the case that $c_2 \in \mathbb{G}$ (since addition is not the group operation). However, it *is* guaranteed that $[c_2 - m_b \bmod p]$ is in $\mathbb{G}$, where m_b is the value that was encrypted in $\langle c_1, c_2\rangle$. Furthermore, as we have seen in the text, it is possible to efficiently verify if a value is in such a group $\mathbb{G}$. The question remains as to the probability that $[c_2 - m_{1-b} \bmod p]$ is *also* in $\mathbb{G}$. However, as we know, $\mathbb{G}$ includes exactly

half of the elements of $\mathbb{Z}_p^*$. Thus, since m_{1-b} is a random value, it follows that $[c_2 - m_{1-b} \bmod p] \in \mathbb{G}$ with probability only 1/2. Thus, $\mathcal{A}$'s strategy is to check if $[c_2 - m_0 \bmod p] \in \mathbb{G}$ and if $[c_2 - m_1 \bmod p] \in \mathbb{G}$. If both elements are in the group, then $\mathcal{A}$ outputs a uniform bit. Otherwise, it outputs the bit b for which $[c_2 - m_b \bmod p] \in \mathbb{G}$. A simple calculation shows that $\mathcal{A}$ succeeds with probability 3/4.

11.8 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) then A chooses a uniform 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 before being encrypted—note that efficient decrypt is still possible). 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?

Solution:

(a) The ciphertext c_A output by A corresponds to *some* bit b_A. (It could also be an invalid ciphertext that does not decrypt at all, though this cannot occur [without being detected] when using El Gamal encryption. If T detects an invalid ciphertext upon decryption, then it can just use c_B as the coin.) If B is honest, then it chooses a uniform c_B, and so the coin $c_A \oplus c_B$ (or even just c_B in case c_A is invalid) is uniform.

(b) The problem when trying to apply the argument in part (a) to the case when A is honest but B is not is that B might choose c_B in a way that *depends* on c_A (and hence b_A); this was not possible when the dishonest party went first. We show that, when El Gamal encryption is used, B can bias the coin any way it likes.

B can bias the coin toward 0 as follows. Given the ciphertext $c_A = \langle c_1, c_2 \rangle$ broadcast by A, player B chooses a random $r \leftarrow \mathbb{Z}_q$ and outputs $c_B = \langle c_1', c_2' \rangle$ with

$$c_1' := c_1 \cdot g^r \quad \text{and} \quad c_2' := c_2 \cdot h^r.$$

One can check that c_B always decrypts to b_A, and so the coin is always 0 if B follows this strategy. Furthermore, it holds that c_B is distributed exactly like a *random* encryption of b_A (this is not too much more difficult to show), and so this kind of cheating by B is undetectable (since it is indistinguishable whether B is cheating or whether it just happened that B chose $b_B = b_A$).

To bias toward 1, player B can do the following. Given c_A as above, choose random $r \leftarrow \mathbb{Z}_q$ and output $c_B = \langle c_1', c_2' \rangle$ with

$$c_1' := (c_1)^{-1} \cdot g^r \quad \text{and} \quad c_2' := (c_2)^{-1} \cdot g \cdot h^r.$$

One can check that now $b_B = 1 - b_A$, and so the coin always comes out to be 1. Once again, c_B is in fact a random encryption of $1 - b_A$ and so this form of cheating is undetectable.

(c) A CCA-secure encryption scheme should be used to prevent the problems sketched in part (b).

We define security informally, since security of two-party protocols is not covered in the book. Say a coin-tossing protocol Π is secure if, for any PPT adversary corrupting either one of the parties, the probability $p = p(n)$ that the resulting coin is 0 is negligibly close to $\frac{1}{2}$. (I.e., there is a negligible function negl such that $|p(n) - \frac{1}{2}| \leq \mathsf{negl}(n)$.) Note that we use absolute value here since we don't want the coin to be biased toward either 0 or 1. We claim that the protocol in this exercise, when instantiated with a CCA-secure public-key encryption scheme, is a secure coin-tossing protocol. (We add one point of clarification for the protocol: if either party's ciphertext is invalid, including the case when it does not decrypt to 0/1, then T outputs the bit obtained by decrypting the other party's ciphertext.)

We have seen already in part (a) that an adversary corrupting A cannot bias the coin. So consider the case when B is corrupted. Let Π denote the CCA-secure encryption scheme used in the protocol. Define the following adversary $\mathcal{A}$ attacking Π:

Adversary $\mathcal{A}$:

i. $\mathcal{A}$ is given pk, and outputs $m_0 = 0$ and $m_1 = 1$.

ii. $\mathcal{A}$ is given in return a ciphertext c_A. It then runs $B(c_A)$ to obtain a ciphertext $c_B \neq c_A$. $\mathcal{A}$ submits c_B to its decryption oracle and receives in return a bit b'. Then:

- If decryption of c_B returns an error, or if $b' \notin \{0, 1\}$ then $\mathcal{A}$ outputs 0.
- If $b' = 0$ then $\mathcal{A}$ outputs 0.
- If $b' = 1$ then $\mathcal{A}$ outputs 1.

We have

$$\begin{aligned}\Pr[\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n)=1] &= \frac{1}{2}\cdot\Pr[\mathcal{A}\text{ outputs }0 \mid c_A\text{ is an encryption of }0]\\ &\quad+\frac{1}{2}\cdot\Pr[\mathcal{A}\text{ outputs }1 \mid c_A\text{ is an encryption of }1]\\ &= \frac{1}{2}\cdot\Pr[\text{coin=}0 \mid c_A\text{ is an encryption of }0]\\ &\quad+\frac{1}{2}\cdot\Pr[\text{coin=}0 \mid c_A\text{ is an encryption of }1]\\ &= \Pr[\text{coin=}0] \stackrel{\text{def}}{=} p(n).\end{aligned}$$

CCA-security of Π implies that $p(n) \leq \frac{1}{2} + \mathsf{negl}(n)$ for some negligible function negl. A similar argument, flipping the output of $\mathcal{A}$, shows that $1-p(n) \leq \frac{1}{2}+\mathsf{negl}(n)$ as well. Together, these complete the proof of security for the protocol.

11.9 Prove formally that the El Gamal encryption scheme is not CCA-secure.

Solution: Consider the following attacker $\mathcal{A}$: given public key $(\mathbb{G}, q, g, h)$, choose arbitrary (distinct) $h_0, h_1 \in \mathbb{G}$ as the challenge messages. Upon receiving the challenge ciphertext $\langle c_1, c_2\rangle$, compute $c_2' = h \cdot c_2$ and request decryption of $\langle c_1, c_2'\rangle$, receiving some u in return. Finally, output 0 if $\frac{u}{h} = h_0$ and output 1 if $\frac{u}{h} = h_1$.

We prove that $\mathcal{A}$ succeeds in the CCA-experiment with probability 1. Observe that $\langle c_1, c_2\rangle = \langle g^y, h^y \cdot h_b\rangle$ and so $\langle c_1, c_2'\rangle = \langle g^y, h^y \cdot h_b \cdot h\rangle$. This implies that the decryption oracle returns $u = h_b \cdot h$ and thus $\frac{u}{h} = h_b$, as required.

11.10 In Section 11.4.4 we showed that El Gamal encryption is malleable, and specifically that given a ciphertext $\langle c_1, c_2\rangle$ that is the encryption of some unknown message m, it is possible to produce a ciphertext $\langle c_1, c_2'\rangle$ that is the encryption of $\alpha \cdot m$ (for known α). A receiver who receives both these ciphertexts might be suspicious since both ciphertexts share the first component. Show that it is possible to generate $\langle c_1', c_2'\rangle$ that is the encryption of $\alpha \cdot m$, with $c_1' \neq c_1$ and $c_2' \neq c_2$.

Solution: Let $\langle c_1, c_2\rangle = \langle g^y, h^y \cdot m\rangle$. Then, for any r we have $\langle c_1', c_2'\rangle \stackrel{\text{def}}{=} \langle c_1 \cdot g^r, c_2 \cdot h^r \cdot \alpha\rangle = \langle g^{y+r}, h^{y+r} \cdot \alpha \cdot m\rangle$. Thus, $\langle c_1', c_2'\rangle$ is an encryption of $\alpha \cdot m$ with $c_1' \neq c_1$ and $c_2' \neq c_2$. (In fact, c_1' is uniform and independent of c_1 and $\langle c_1', c_2'\rangle$ is an independent random encryption of $\alpha \cdot m$.)

11.11 Prove Theorem 11.22.

Solution: The proof is similar to that of Theorem 11.38, and is omitted.

11.12 One of the attacks on plain RSA discussed in Section 11.5.1 involves a sender who encrypts two related messages using the same public key.

Formulate an appropriate definition of security ruling out such attacks, and show that any CPA-secure public-key encryption scheme satisfies your definition.

Solution: There are multiple ways a definition can be formulated; we provide one. Consider the following experiment involving an attacker $\mathcal{A}$ and an encryption scheme Π:

$\mathsf{PubK}^{\mathsf{related}}_{\mathcal{A},\Pi}(n)$**:**

(a) $\mathsf{Gen}(1^n)$ *is run to obtain keys* (pk, sk)*.*

(b) Adversary $\mathcal{A}$ *is given* pk*, and outputs a pair of messages* m_0, m_1 *and a function* f*. We require that* $|m_0| = |m_1|$ *and* $|f(m_0)| = |f(m_1)|$ *and that* $m_0, m_1, f(m_0), f(m_1)$ *are in the message space.*

(c) A *uniform bit* $b \in \{0,1\}$ *is chosen, and then ciphertexts* $c \leftarrow \mathsf{Enc}_{pk}(m_b)$ *and* $c' \leftarrow \mathsf{Enc}_{pk}(f(m_b))$ *are computed and given to* $\mathcal{A}$*.*

(d) $\mathcal{A}$ *outputs a bit* b'*. The output of the experiment is* 1 *if* $b' = b$*, and* 0 *otherwise.*

$\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is secure for encryption of related messages if for all probabilistic, polynomial-time adversaries $\mathcal{A}$ there is a negligible function negl such that

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

Any attacker in the above experiment can be simulated by an attacker in the $\mathsf{PubK}^{\mathsf{LR\text{-}cpa}}$ experiment (by submitting (m_0, m_1) to the LR-oracle, followed by $(f(m_0), f(m_1))$). Since any CPA-secure scheme is secure in that experiment, this implies that any CPA-secure scheme is also secure for encryption of related messages according to the definition above.

11.13 One of the attacks on plain RSA discussed in Section 11.5.1 involves a sender who encrypts the same message to three different receivers. Formulate an appropriate definition of security ruling out such attacks, and show that any CPA-secure public-key encryption scheme satisfies your definition.

Solution: There are multiple ways a definition can be formulated; we provide one. Consider the following experiment involving an attacker $\mathcal{A}$ and an encryption scheme Π:

$\mathsf{PubK}^{\ell\text{-}\mathsf{keys}}_{\mathcal{A},\Pi}(n)$**:**

(a) $\mathsf{Gen}(1^n)$ *is run* $\ell = \ell(n)$ *times to obtain* $(pk_1, sk_1), \ldots, (pk_\ell, sk_\ell)$*.*

(b) $\mathcal{A}$ *is given* $pk_1, \ldots, pk_\ell$*, and outputs a pair of equal-length messages* m_0, m_1*.*

(c) A uniform $b \in \{0,1\}$ is chosen, and (for $i = 1, \ldots, \ell$) ciphertexts $c_i \leftarrow \mathsf{Enc}_{pk_i}(m_b)$ are computed and given to $\mathcal{A}$.

(d) $\mathcal{A}$ outputs a bit b'. The output of the experiment is 1 if $b' = b$, and 0 otherwise.

$\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is secure for encrypting to different receivers if for all polynomials ℓ and all PPT adversaries $\mathcal{A}$ there is a negligible function negl such that

$$\Pr[\mathsf{PubK}^{\ell\text{-keys}}_{\mathcal{A},\Pi}(n) = 1] \leq \mathsf{negl}(n).$$

A proof that CPA-security of Π implies security for encrypting to different receivers uses a hybrid argument as in the proof of Theorem 11.6, and is omitted.

11.14 Consider the following modified version of padded RSA encryption: Assume messages to be encrypted have length exactly $\|N\|/2$. To encrypt, first compute $\hat{m} := \texttt{0x00}\|r\|\texttt{0x00}\|m$ where r is a uniform string of length $\|N\|/2 - 16$. Then compute the ciphertext $c := [\hat{m}^e \bmod N]$. When decrypting a ciphertext c, the receiver computes $\hat{m} := [c^d \bmod N]$ and returns an error if $\hat{m}$ does not consist of `0x00` followed by $\|N\|/2 - 16$ arbitrary bits followed by `0x00`. Show that this scheme is not CCA-secure. Why is it easier to construct a chosen-ciphertext attack on this scheme than on PKCS #1 v1.5?

Solution: Consider the following adversary:

Algorithm $\mathcal{A}$:

(a) Given the public key N, e, output $m_0 = 00\cdots0$ and $m_1 = 01\cdots1$. Receive in return a ciphertext c.

(b) Set $c' := [2^e \cdot c \bmod N]$ and submit c' to the decryption oracle.

(c) If the decryption oracle returns an error, then output a random bit. Otherwise, let m' denote the message returned by the decryption oracle.

(d) Output the most-significant bit of m'.

Why does this attack succeed (with high probability)? Let b denote the bit such that c is an encryption of m_b, and let $k = \|N\|/2 - 16$. Then

$$c = [(\texttt{0x00} \parallel r_1 \cdots r_k \parallel \texttt{0x00} \parallel 0b\cdots b)^e \bmod N]$$

and

$$\begin{aligned} c' = 2^e \cdot c &= [(2 \cdot (\texttt{0x00} \parallel r_1 \cdots r_k \parallel \texttt{0x00} \parallel 0b\cdots b))^e \bmod N] \\ &= [(0^7 r_1 \parallel r_2 \cdots r_k 0 \parallel \texttt{0x00} \parallel b\cdots b0)^e \bmod N]. \end{aligned}$$

(Multiplication by 2 corresponds to a left-wise bit shift; there is no overflow since the leading bit of the multiplicand is 0.) Decryption of

c' results in an error only if $r_1 = 1$, which occurs with probability $\frac{1}{2}$; if no error occurs, then the high-order bit of the decrypted message is exactly equal to b. Thus the above attack correctly determines b with probability 3/4, which is significantly better than guessing at random.

This simple attack does not work on PKCS #1 v1.5. In that scheme, encryption of a message m is done by computing

$$c := [(00000000 \parallel 00000010 \parallel r_1 r_2 \cdots \parallel 00000000 \parallel m_1 m_2 \cdots)^e \bmod N],$$

where r is uniform. Thus, setting $c' := [2^e \cdot c \bmod N]$ would give

$$c' = [(00000000 \parallel 0000010 r_1 \parallel r_2 \cdots 0 \parallel 0000000 m_1 \parallel m_2 \cdots)^e \bmod N],$$

which will never decrypt correctly (since the second byte is formatted incorrectly, regardless of the value of r_1).

11.15 Consider the RSA-based encryption scheme in which a user encrypts a message $m \in \{0,1\}^\ell$ with respect to the public key $\langle N, e\rangle$ by computing $\hat{m} := H(m) \| m$ and outputting the ciphertext $[\hat{m}^e \bmod N]$. (Here, let $H : \{0,1\}^\ell \to \{0,1\}^n$ and assume $\ell + n < \|N\|$.) The receiver recovers $\hat{m}$ in the usual way and verifies that it has the correct form before outputting the ℓ least-significant bits as m. Prove or disprove that this scheme is CCA-secure if H is modeled as a random oracle.

Solution: This scheme is not even CPA-secure, since encryption is deterministic.

11.16 Show a chosen-ciphertext attack on Construction 11.34.

Solution: There are many possible answers; we provide one. We only sketch an attack. Given a ciphertext c that encapsulates an unknown key $k_1 \cdots k_n$, compute $c' := [c^e \bmod N]$ and request decryption of c'. The result will be $k_2 \cdots k_n k$, where $k = \mathsf{lsb}(c)$, thus revealing all-but-one of the bits of the encapsulated key.

11.17 Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a CPA-secure public-key encryption scheme, and let $\Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ be a CCA-secure private-key encryption scheme. Does Construction 11.3-S have indistinguishable encryptions under a chosen-*ciphertext* attack, if H is modeled as a random oracle? If yes, provide a proof. If not, where does the approach used to prove Theorem 11.38 break down?

Solution: Π^* does *not*, in general, have indistinguishable encryptions under a chosen-ciphertext attack. The issue is that it might be easy to take a ciphertext c_1 encrypted using Π and generate a *different* ciphertext c_1' that decrypts to the same value as c_1. For example, El Gamal encryption has this property (see Section 11.4.4). If Π has this property, then there is an easy chosen-ciphertext attack on Π^*: given ciphertext

CONSTRUCTION 11.3-S

Let $H : \{0,1\}^n \to \{0,1\}^n$ be a function. Construct a public-key encryption scheme as follows:

- Gen^*: on input 1^n, run $\mathsf{Gen}(1^n)$ to obtain (pk, sk). Output these as the public and private keys, respectively.
- Enc^*: on input a public key pk and a message $m \in \{0,1\}^n$, choose a uniform $r \in \{0,1\}^n$ and output the ciphertext

$$\left\langle \mathsf{Enc}_{pk}(r),\ \mathsf{Enc}'_{H(r)}(m) \right\rangle .$$

- Dec^*: on input a private key sk and a ciphertext $\langle c_1, c_2 \rangle$, compute $r := \mathsf{Dec}_{sk}(c_1)$ and set $k := H(r)$. Then output $\mathsf{Dec}'_k(c_2)$.

$C = \langle c_1, c_2 \rangle$ generate a ciphertext $C' = \langle c'_1, c_2 \rangle$ (where c'_1 decrypts to the same value as c_1 with respect to the given public key) and then submit this ciphertext to the decryption oracle. Note that the decryption of C' is the same as the decryption of C.

The feature of Construction 11.37 that prevents this attack is that textbook RSA encryption is a *permutation*; that is, every message corresponds to a *unique* ciphertext. Thus, given a ciphertext c_1 encrypted using textbook RSA encryption, there does not exist a different ciphertext c'_1 decrypting to the same value.

11.18 Consider the following variant of Construction 11.32:

CONSTRUCTION 11.4-S

Let GenRSA be as usual, and define a public-key encryption scheme as follows:

- Gen: on input 1^n, run $\mathsf{GenRSA}(1^n)$ to obtain (N, e, d). Output the public key $pk = \langle N, e \rangle$, and the private key $sk = \langle N, d \rangle$.
- Enc: on input a public key $pk = \langle N, e \rangle$ and a message $m \in \{0,1\}$, choose a uniform $r \in \mathbb{Z}_N^*$. Output the ciphertext $\langle [r^e \bmod N],\ \mathsf{lsb}(r) \oplus m \rangle$.
- Dec: on input a private key $sk = \langle N, d \rangle$ and a ciphertext $\langle c, b \rangle$, compute $r := [c^d \bmod N]$ and output $\mathsf{lsb}(r) \oplus b$.

Prove that this scheme is CPA-secure. Discuss its advantages and disadvantages relative to Construction 11.32.

Solution: The proof is (conceptually) very similar to the proof of Theorem 11.33. Let Π denote the above variant scheme, and let $\mathcal{A}$ be an adversary attacking this scheme. Assume for simplicity (and without loss

of generality) that $\mathcal{A}$ always outputs $m_0 = 0$ and $m_1 = 1$. Construct an algorithm $\mathcal{A}'$ for experiment $\mathsf{RSA\text{-}lsb}$ as follows: $\mathcal{A}'$, given N, e, c, chooses a uniform bit $\hat{b}$ and computes $b' \leftarrow \mathcal{A}(N, e, \langle c, \hat{b}\rangle)$; it then outputs $\hat{b} \oplus b'$. Let r be the (unknown) value such that $c = [r^e \bmod N]$. Note that

$$\begin{aligned}&\Pr[\mathcal{A}'(N,e,c) = 0 \mid \mathsf{lsb}(r) = 0]\\ &= \frac{1}{2} \cdot (\Pr[\mathcal{A}(N,e,\langle c, 0\rangle) = 0 \mid \mathsf{lsb}(r) = 0] + \Pr[\mathcal{A}(N,e,\langle c,1\rangle) = 1 \mid \mathsf{lsb}(r) = 0])\end{aligned}$$

and

$$\begin{aligned}&\Pr[\mathcal{A}'(N,e,c) = 1 \mid \mathsf{lsb}(r) = 1]\\ &= \frac{1}{2} \cdot (\Pr[\mathcal{A}(N,e,\langle c, 0\rangle) = 1 \mid \mathsf{lsb}(r) = 1] + \Pr[\mathcal{A}(N,e,\langle c,1\rangle) = 0 \mid \mathsf{lsb}(r) = 1])\,.\end{aligned}$$

Moreover,

$$\begin{aligned}&\Pr[\mathcal{A} \text{ outputs } 0 \mid 0 \text{ is encrypted}]\\ &= \frac{1}{2} \cdot (\Pr[\mathcal{A}(N,e,\langle c, 0\rangle) = 0 \mid \mathsf{lsb}(r) = 0] + \Pr[\mathcal{A}(N,e,\langle c,1\rangle) = 0 \mid \mathsf{lsb}(r) = 1])\,,\end{aligned}$$

and similarly for $\Pr[\mathcal{A} \text{ outputs } 1 \mid 1 \text{ is encrypted}]$. Putting everything together gives

$$\Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] = \Pr[\mathsf{RSA\text{-}lsb}_{\mathcal{A}',\mathsf{GenRSA}}(n) = 1],$$

which shows that the scheme is CPA-secure if the RSA problem is hard relative to GenRSA.

11.19 Say three users have RSA public keys $\langle N_1, 3\rangle$, $\langle N_2, 3\rangle$, and $\langle N_3, 3\rangle$ (i.e., they all use $e = 3$), with $N_1 < N_2 < N_3$. Consider the following method for sending the same message $m \in \{0,1\}^\ell$ to each of these parties: choose a uniform $r \leftarrow \mathbb{Z}^*_{N_1}$, and send to everyone the same ciphertext

$$\left\langle [r^3 \bmod N_1], [r^3 \bmod N_2], [r^3 \bmod N_3], H(r) \oplus m \right\rangle,$$

where $H : \mathbb{Z}^*_{N_1} \to \{0,1\}^\ell$. Assume $\ell \gg n$.

(a) Show that this is not CPA-secure, and an adversary can recover m from the ciphertext even when H is modeled as a random oracle.

(b) Show a simple way to fix this and get a CPA-secure method that transmits a ciphertext of length $3\ell + \mathcal{O}(n)$.

(c) Show a better approach that is still CPA-secure but with a ciphertext of length $\ell + \mathcal{O}(n)$.

Solution:

(a) The second attack presented in Section 10.4.2 shows how r can be recovered in polynomial time given $[r^3 \bmod N_1]$, $[r^3 \bmod N_2]$, and $[r^3 \bmod N_3]$. Given r, the message m can be easily recovered.

(b) One simple way to fix the scheme is to use an independent value r for each user. I.e., to encrypt the sender would choose random $r_i \in \mathbb{Z}_{N_i}$ for $i \in \{1,2,3\}$ and then send the ciphertext

$$\left\langle \begin{array}{c} [r_1^3 \bmod N_1], [r_2^3 \bmod N_2], [r_3^3 \bmod N_3], \\ H(r_1) \oplus m, \;\; H(r_2) \oplus m, \;\; H(r_3) \oplus m \end{array} \right\rangle.$$

(c) A better idea is to use a variant of *hybrid encryption*: use the approach from part (b) to encrypt a key k, and then use the same key k to encrypt m. In detail: let $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ denote a CPA-secure private-key encryption scheme. Then to encrypt the sender would choose random $r_i \in \mathbb{Z}_{N_i}$ for $i \in \{1,2,3\}$ as well as random $k \in \{0,1\}^n$, and then send the ciphertext

$$\left\langle \begin{array}{c} [r_1^3 \bmod N_1], [r_2^3 \bmod N_2], [r_3^3 \bmod N_3], \\ H(r_1) \oplus k, \;\; H(r_2) \oplus k, \;\; H(r_3) \oplus k, \\ \mathsf{Enc}_k(m) \end{array} \right\rangle.$$

11.20 Fix an RSA public key $\langle N, e \rangle$ and assume we have an algorithm $\mathcal{A}$ that always correctly computes $\mathsf{lsb}(x)$ given $[x^e \bmod N]$. Write full pseudocode for an algorithm $\mathcal{A}'$ that computes x from $[x^e \bmod N]$.

Solution: Let $\gamma \stackrel{\text{def}}{=} [2^{-1} \bmod N]$. The intuition is that $x^e \cdot \gamma^e = (x\gamma)^e \bmod N$; thus, multiplication by γ^e can be used to effect a bit-wise right-shift (i.e., division by 2), which can in turn be used to learn all the bits of x one-by-one. There are, however, some technicalities.

ALGORITHM 11.5-S
GetBits

Input: $\langle N, e \rangle$; $c \in \mathbb{Z}_N^*$; ℓ
Output: the ℓ least significant bits of $[c^{1/e} \bmod N]$

if $\ell = 1$, return $\mathcal{A}(N, e, c)$
$\gamma := [2^{-1} \bmod N]$
$x_0 := \mathcal{A}(N, e, c)$
$x' := \mathsf{GetBits}(N, e, [c \cdot \gamma^e \bmod N], \ell - 1)$
if $x_0 = 0$ **return** $x' \| x_0$
else return $[2x' - N \bmod 2^\ell]$

When $\mathsf{lsb}(x) = 0$ then $[\gamma \cdot x \bmod N]$ is indeed just a right-shift of x (since x, viewed as an integer, is divisible by 2). But when $\mathsf{lsb}(x) = 1$, then

$[\gamma \cdot x \bmod N] = \frac{x+N}{2}$. (Note that N is odd.) We take this into account in Algorithm 11.5-S, which is described recursively. The algorithm relies on the assumed algorithm $\mathcal{A}$ for computing $\mathsf{lsb}(x)$. When called with $\ell = \|N\|$ it returns all the bits of $x = [c^{1/e} \bmod N]$.

11.21 Fix an RSA public key $\langle N, e\rangle$ and define

$$\mathsf{half}(x) = \begin{cases} 0 & \text{if } 0 < x < N/2 \\ 1 & \text{if } N/2 < x < N \end{cases}$$

Prove that half is a hard-core predicate for the RSA problem.

Solution: The key observation is that

$$\mathsf{half}(x) = 0 \Leftrightarrow \mathsf{lsb}([2x \bmod N]) = 0.$$

This is because when $x < N/2$ then $[2x \bmod N] = 2x$ (there is no overflow), which is even. On the other hand, when $x > N/2$ then $[2x \bmod N] = 2x - N$, which is odd.

Turning this around, given an algorithm $\mathcal{A}$ for computing $\mathsf{half}(x)$ from $N, e, [x^e \bmod N]$, we can compute $\mathsf{lsb}(x)$ from $N, e, c = [x^e \bmod N]$ by calling $\mathcal{A}(N, e, [c/2^e \bmod N])$.

Chapter 12

Digital Signature Schemes – Solutions

12.1 Show that Construction 4.7 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 $\ell(n) \geq n$.

Solution: The proof that Construction 4.7 works also in the context of signatures follows along exactly the same lines as the proof of Theorem 4.8.

As long as $\ell(n) = \omega(\log n)$ (see Appendix A.2 for a definition of this notation), the construction works. To see this, recall that the construction needs to compute a signature on "messages" of the form $r\|\mathsf{len}\|i\|m_i$, where r is a uniform string of some length; len denotes the message length (encoded in binary); i denotes the block number (encoded in binary); and m_i is the i block of the message. Using 1-bit blocks (so $|m_i| = 1$), we need $|\mathsf{len}|$ to be $\omega(\log n)$ so that messages of any polynomial length can be handled (at least asymptotically). It then suffices for $|i|$ to have length $\omega(\log n)$ as well. For the security proof, we require $|r| = \omega(\log n)$ so that a "repeat" does not occur too often (cf. the proof of Claim 4.7). Altogether, then, $\ell(n) = \omega(\log n)$ is sufficient.

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

Solution: Let $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a one-time signature scheme for 1-bit messages. Since Gen runs in polynomial time, there exists a polynomial p such that the number of random bits the algorithm uses on input 1^n is at most $p(n)$. For simplicity, assume that Gen always uses exactly $p(n)$ random bits on input 1^n, and further than $p(n)$ is strictly increasing. Define the following function f: on input a string x, compute n such that $p(n) \leq |x| \leq p(n+1)$. Then compute $(pk, sk) := \mathsf{Gen}(1^n; x)$ and output pk.

A proof that f as defined above is one-way follows along the same lines as the proof of Theorem 8.74. The only difference here is to note that inverting f on a given pk allows computation of an associated private

key sk' (which need not be equal to sk), which then enables signature forgery with respect to pk.

12.3 In Section 12.4 we showed an attack on the plain RSA signature scheme in which an attacker forges a signature on an arbitrary message using two signing queries. Show how an attacker can forge a signature on an arbitrary message using a *single* signing query.

Solution: Let (N, e) be the public key and let $x \in \mathbb{Z}_N$ be the message that the attacker $\mathcal{A}$ wishes to obtain a signature for. Then, $\mathcal{A}$ chooses an arbitrary $r \in \mathbb{Z}_N$ and computes $s = r^e \bmod N$. Then, $\mathcal{A}$ asks for a signature on $x \cdot s \bmod N$ and receives back some σ. Finally, $\mathcal{A}$ outputs the pair $(x, \hat{\sigma})$ where $\hat{\sigma} = \frac{\sigma}{r} \bmod N$. In order to see why this is correct, observe that $\hat{\sigma}^e = \frac{\sigma^e}{r^e} = \frac{((x \cdot s)^d)^e}{r^e} = \frac{x \cdot s}{r^e} = \frac{x \cdot r^e}{r^e} = x \bmod N$, as required.

12.4 Assume the RSA problem is hard. Show that the plain RSA signature scheme satisfies the following weak definition of security: an attacker is given the public key $\langle N, e \rangle$ and a uniform message $m \in \mathbb{Z}_N^*$. The adversary succeeds if it can output a valid signature on m without making any signing queries.

Solution: A formal proof follows easily from the fact that computing a signature on a random message m is equivalent to solving the RSA problem on a random instance.

12.5 Another approach (besides hashing) that has been tried to construct secure RSA-based signatures is to *encode* the message before applying the RSA permutation. Here the signer fixes a public encoding function $\mathsf{enc} : \{0,1\}^\ell \to \mathbb{Z}_N^*$ as part of its public key, and the signature on a message m is $\sigma := [\mathsf{enc}(m)^d \bmod N]$.

(a) How is verification performed in encoded RSA?

(b) Discuss why appropriate choice of encoding function for $\ell \ll \|N\|$ prevents the "no-message attack" described in Section 12.4.1.

(c) Show that encoded RSA is insecure if $\mathsf{enc}(m) = 0^{\kappa/10} \| m \| 0^{\kappa/10}$ (where $\kappa \stackrel{\text{def}}{=} \|N\|$, $\ell = |m| \stackrel{\text{def}}{=} 4\kappa/5$, and m is not the all-0 message). Assume $e = 3$.

(d) Show that encoded RSA is insecure for $\mathsf{enc}(m) = 0 \| m \| 0 \| m$ (where $\ell = |m| \stackrel{\text{def}}{=} (\|N\| - 1)/2$ and m is not the all-0 message). Assume $e = 3$.

(e) Solve parts (c) and (d) for arbitrary e.

Solution:

(a) To verify a signature σ on a message m with respect to a public key $pk = \langle N, e \rangle$, simply check whether

$$\sigma^e \stackrel{?}{=} \mathsf{enc}(m) \bmod N.$$

(b) The "no-message attack" from Section 12.3.1 would work by choosing a uniform $\sigma \in \mathbb{Z}_N^*$ and then trying to find an m such that $\mathsf{enc}(m) = \sigma^e \bmod N$. If enc is such that a uniform element of $\mathbb{Z}_N^*$ lies in the range of enc with only negligible probability, then this attack will not work since $[\sigma^e \bmod N]$ is not even in the range of enc except with negligible probability.

We remark that the encoding schemes used in parts (c) and (d) both have the above property.

(c) The key observation is that $\mathsf{enc}(m) = m \cdot 2^{\kappa/10} \bmod N$, as long as m can be expressed as an ℓ-bit binary integer.

An attack on the scheme is as follows: Set $m := 0 \cdots 01000 = 2^e$ and request a signature σ on m. Then output the forged signature $[\sigma \cdot 2^{-1} \bmod N]$ on the message $m' = 0 \cdots 01 = 1$. This works since

$$\left(\sigma \cdot 2^{-1}\right)^e = \mathsf{enc}(m) \cdot 2^{-e} = 2^e 2^{\kappa/10} 2^{-e} = 2^{\kappa/10} = \mathsf{enc}(m') \bmod N.$$

(d) Observe that $\mathsf{enc}(m) = m \cdot (2^{\ell+1} + 1) \bmod N$ as long as m can be expressed as an ℓ-bit binary integer. The same attack as in the previous part works here as well.

(e) Both parts use an encoding of the form $\mathsf{enc}(m) = m \cdot \alpha$, so we show one attack that applies to both. Request a signature σ_4 on the message $0 \cdots 0100 = 4$ and a signature σ_2 on the message $0 \cdots 010 = 2$. Then output the forgery $\sigma := [\sigma_4^2/\sigma_2 \bmod N]$ on the message $0 \cdots 01000 = 8$. This is a valid forgery since

$$\begin{aligned} \sigma^e &= \left(\sigma_4^2/\sigma_2\right)^e \\ &= \mathsf{enc}(4)^2/\mathsf{enc}(2) \\ &= \frac{(4 \cdot \alpha)^2}{2 \cdot \alpha} \\ &= 8 \cdot \alpha \;\; = \;\; \mathsf{enc}(8) \bmod N. \end{aligned}$$

12.6 Consider a variant of the Fiat–Shamir transform in which the signature is (I, s) rather than (r, s) and verification is changed in the natural way. Show that if the underlying identification scheme is secure, then the resulting signature scheme is secure here as well.

Solution: This follows immediately from the fact that, given only the public key, it is possible to convert a signature of the first type to a signature of the second type, and vice versa.

12.7 Consider a variant of DSA in which the message space is $\mathbb{Z}_q$ and H is omitted. (So the second component of the signature is now $s := [k^{-1} \cdot (m + xr) \bmod q]$.) Show that this variant is not secure.

Solution: The key observation is that k can be set implicitly and then m can be chosen to make things cancel appropriately. The easiest example

is to set k implicitly equal to x, by using y in place of g^k in the signing calculation. I.e., compute $r := F(y)$ (so $k = x$), and then observe that

$$s := [k^{-1} \cdot (m + xr) \bmod q] = [x^{-1}m + r \bmod q].$$

Thus, if we set $m := 0$ then s can be computed without knowing x.

More generally, we can compute $r := F(g^\alpha y^\beta)$ so $k = \alpha + \beta x$. Then

$$s := [(\alpha + \beta x)^{-1} \cdot (m + xr) \bmod q],$$

and we see that by setting $m = r\alpha/\beta$ it is possible to compute s without knowing x.

12.8 Let f be a one-way permutation. Consider the following signature scheme for messages in the set $\{1, \ldots, n\}$:

- To generate keys, choose uniform $x \in \{0,1\}^n$ and set $y := f^{(n)}(x)$ (where $f^{(i)}(\cdot)$ refers to i-fold iteration of f, and $f^{(0)}(x) \stackrel{\text{def}}{=} x$). The public key is y and the private key is x.
- To sign message $i \in \{1, \ldots, n\}$, output $f^{(n-i)}(x)$.
- 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-secure 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-secure signature scheme.

Solution:

(a) Here is an attack. Request a signature on the message n. This gives the secret key $x = f^{n-n}(x)$, after which a signature on any other message can be computed.

More generally, given a signature on a message i it is possible to forge a signature for any message j with $j < i$. (In detail, if σ_i is a signature on i, then $\sigma_j \stackrel{\text{def}}{=} f^{i-j}(\sigma_i)$ is a signature on j. Since $i > j$ and f is known, computation of σ_j can be done efficiently.)

(b) We first give a formal security definition. Consider the following experiment for an adversary $\mathcal{A}$ and security parameter n (the experiment is tailored for the scheme proposed in this exercise):

The modified signature experiment $\mathsf{Sig\text{-}forge}'_{\mathcal{A}}(n)$**:**

i. *Choose* $x \leftarrow \{0,1\}^n$ *and set* $y := f^n(x)$.

ii. *Adversary* $\mathcal{A}$ *is given* y. *It then outputs any* $i \in \{1,\ldots,n\}$ *and is given in return the signature* $\sigma_i = f^{n-i}(x)$ *on* i.

iii. $\mathcal{A}$ *outputs* (j, σ_j) *with* $j > i$.

iv. *The output of the experiment is defined to be 1 if and only if* $f^j(\sigma_j) = y$.

DEFINITION *The signature scheme is* weakly secure *if for all probabilistic polynomial-time adversaries* $\mathcal{A}$, *there exists a negligible function* negl *such that* $\Pr[\mathsf{Sig\text{-}forge}'_{\mathcal{A}}(n) = 1] \leq \mathsf{negl}(n)$.

We now show that if f is a one-way permutation then the scheme is weakly secure. Let $\mathcal{A}$ be any PPT adversary. Consider the following algorithm trying to invert f on a given point:

Algorithm D:
The algorithm is given z and its goal is to compute $f^{-1}(z)$.

- Choose $i^* \leftarrow \{1,\ldots,n\}$.
- Set $y = f^{i^*}(z)$.
- Run $\mathcal{A}$ on the public key y. When $\mathcal{A}$ requests a signature on i, abort unless $i = i^*$. If $i = i^*$ then give the adversary the signature $\sigma_i = z$.
- When $\mathcal{A}$ outputs (j, σ_j), output $f^{j-i^*-1}(\sigma_j)$.

Let us analyze D when its input z satisfies $z = f(x)$ for uniform $x \in \{0,1\}^n$. Note first that, since f is a permutation, the public key y is uniform in $\{0,1\}^n$ just as in the real signature scheme. Furthermore, the probability that $i^* = i$ is exactly $1/n$ because i^* is chosen uniformly from $\{1,\ldots,n\}$. Conditioned on this being the case, the signature σ_i given to the adversary is correct. Finally, if $\mathcal{A}$ outputs a valid forgery, then the output of D is indeed the inverse of z. Putting everything together, we have:

$$\Pr[\mathsf{Invert}_{D,f}(n) = 1] = \frac{1}{n} \cdot \Pr[\mathsf{Sig\text{-}forge}'_{\mathcal{A}}(n) = 1].$$

If f is one-way, then this implies the existence of a negligible function negl such that $\Pr[\mathsf{Sig\text{-}forge}'_{\mathcal{A}}(n) = 1] < \mathsf{negl}(n)$.

(c) Consider the following modified scheme, which signs messages in $\{1,\ldots,n-1\}$:

- To generate keys, choose uniform $x, x' \leftarrow \{0,1\}^n$ and set $y := f^n(x)$ and $y' := f^n(x')$. The public key is $\langle y, y' \rangle$ and the private key is $\langle x, x' \rangle$.

- To sign message $i \in \{1, \ldots, n\}$, output $\sigma = f^{n-i}(x)$ and $\sigma' = f^i(x')$.
- To verify signature $\langle \sigma, \sigma' \rangle$ on message i with respect to public key $\langle y, y' \rangle$, check whether $y \stackrel{?}{=} f^i(\sigma)$ and $y' \stackrel{?}{=} f^{n-i}(\sigma')$.

Note that a signature on i in the modified scheme consists of *two* signatures in the original scheme: one on i, and one on $n-i$. Given the result from part (b), one-time security of this modified scheme follows easily.

12.9 A *strong* one-time-secure 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-secure 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-secure signature scheme.

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

Solution:

(a) Let $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme, and consider the following experiment for an adversary $\mathcal{A}$ and parameter n:

> **The strong one-time signature experiment $\mathsf{Sig\text{-}forge}^{\mathsf{1\text{-}strong}}_{\mathcal{A},\Pi}(n)$:**
>
> *i.* $\mathsf{Gen}(1^n)$ *is run to obtain keys* (pk, sk).
>
> *ii. Adversary* $\mathcal{A}$ *is given* pk *and asks a* single *query* m' *to oracle* $\mathsf{Sign}_{sk}(\cdot)$. *Let* σ' *denote the signature that was returned.* $\mathcal{A}$ *then outputs* (m, σ) *where* $(m, \sigma) \neq (m', \sigma')$.
>
> *iii. The output of the experiment is defined to be 1 if and only if* $\mathsf{Vrfy}_{pk}(m, \sigma) = 1$.

DEFINITION *A signature scheme* $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is is a* strong one-time signature scheme, *if for all probabilistic polynomial-time adversaries* $\mathcal{A}$, *there exists a negligible function* negl *such that:*

$$\Pr[\mathsf{Sig\text{-}forge}^{\mathsf{1\text{-}strong}}_{\mathcal{A},\Pi}(n) = 1] \leq \mathsf{negl}(n).$$

(b) Let f be a one-way function and define f' as follows: $f'(x\|b) = f(x)$ (where we let b denote the last bit of the input to f'). It is not

hard to see that f' is a one-way function, but given an inverse of any point y it is easy to come up with a different inverse of y. (Namely, given $x' = x\|b$ such that $f'(x') = y$ we know that $x'' = x\|\bar{b}$ also satisfies $f'(x'') = y$.)

When f' is used in Lamport's scheme, the resulting signature scheme is clearly not a strong one-time signature scheme. E.g., take $\ell = 1$ and note that if $x' = x\|b$ is a signature on the message 0, then so is $x'' = x\|\bar{b}$.

(c) Take f to be a collision-resistant hash function mapping, say, $2n$-bit strings to n-bit outputs. (See the next exercise for a proof that such a collision-resistant hash function is one-way.) Formally, as part of key generation the signer generates a uniform key s for a collision-resistant hash function and includes s as part of the public key; the rest is exactly as in Construction 12.7 except that H^s is used in place of f.

Assuming the result of the following exercise, this scheme is certainly a one-time signature scheme. But any two signatures σ, σ' on the same message m yield a collision in the hash function. A formal proof that this scheme is a strong one-time signature scheme is immediate.

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

Solution: The attack is simple. Consider the scheme defined for messages of length $\ell = \ell(n)$. The adversary can request signatures on the two messages 0^ℓ and 1^ℓ. The values contained in the two signatures together yield the entire private key, at which point a signature on any message can be forged.

12.11 The Lamport scheme uses 2ℓ values in the public key to sign messages of length ℓ. Consider the variant in which the private key contains 2ℓ values $x_1, \ldots, x_{2\ell}$ and the public key contains the values $y_1, \ldots, y_{2\ell}$ with $y_i := f(x_i)$. 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-secure signature scheme. What is the maximum message length ℓ' that this scheme supports?

Solution: A proof that this gives a one-time signature scheme is very similar to the proof of Theorem 12.8. Let Π denote the scheme, and let $\mathcal{A}$ be a probabilistic polynomial-time adversary. Consider the following PPT algorithm $\mathcal{I}$ attempting to invert the one-way function f:

Algorithm $\mathcal{I}$:
The algorithm is given y and 1^n as input.

- Choose $i^* \leftarrow \{1, \ldots, 2\ell\}$, and set $y_{i^*} := y$.
- For $i \in \{1, \ldots, 2\ell\} \setminus \{i^*\}$, do:
  - Choose $x_i \leftarrow \{0,1\}^n$.
  - Set $y_i := f(x_i)$.
- Set $pk = (y_1, \ldots, y_{2\ell})$ and run $\mathcal{A}(pk)$.
- When $\mathcal{A}$ requests a signature of the message m', map m' to a subset $S_{m'}$ of size ℓ. If $i^* \in S_{m'}$, then abort. Otherwise, give $\{x_i\}_{i \in S_{m'}}$ to $\mathcal{A}$.
- When $\mathcal{A}$ outputs $(m, \sigma' = (x_{i_1}, \ldots, x_{i_\ell}))$, map m to a subset S_m of size ℓ. If i^* is the first index for which $i^* \in S_m$ but $i^* \notin S_{m'}$, then output x_{i^*}. Otherwise, abort.

In an execution of $\mathcal{A}$, let m' denote the message whose signature is requested by $\mathcal{A}$, and let $m \neq m'$ denote the message whose signature is (purportedly) forged by $\mathcal{A}$. Let $S_{m'}$ and S_m be defined in the natural way, and let $j \in \{1, \ldots, 2\ell\}$ denote the first index such that $j \in S_m$ but $j \notin S_{m'}$. (Note that such j must always exist.)

In the execution of $\mathcal{I}$, above, observe that the public key given to $\mathcal{A}$ is distributed identically to the public key in a real execution of Π. Furthermore, if $\mathcal{I}$ correctly guesses $i^* = j$, then the signature given by $\mathcal{I}$ to $\mathcal{A}$ is correct. Finally, conditioned on $i^* = j$, whenever $\mathcal{A}$ outputs a valid forgery it is the case that $\mathcal{I}$ correctly inverts the given value y. Since $i^* = j$ with probability exactly $1/2\ell$, we have

$$\Pr[\mathsf{Invert}_{\mathcal{I},f}(n) = 1] = \frac{1}{2\ell} \cdot \Pr[\mathsf{Sig\text{-}forge}^{\mathsf{1\text{-}time}}_{\mathcal{A},\Pi}(n) = 1].$$

Since f is one-way, this implies the existence of a negligible function negl such that $\Pr[\mathsf{Sig\text{-}forge}^{\mathsf{1\text{-}time}}_{\mathcal{A},\Pi}(n) = 1] < \mathsf{negl}(n)$.

The number of subsets of size ℓ is $\binom{2\ell}{\ell}$, and so this is the number of messages that can be signed. So the maximum message length is given by $\ell' = \lfloor \log_2 \binom{2\ell}{\ell} \rfloor$.

12.12 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 chain-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.

Solution: A similar approach cannot be directly applied to the chain-based scheme, since that scheme required the signer's state to include the entire history of messages that have been signed so far.

We describe a way to adapt the chain-based scheme so that the message history does not need to be stored as part of the signer's state; this variant can be made stateless following the approach used in Section 12.6.2.

On a high level, the modification will be to use each public key pk_i to sign the concatenation of *two* public keys $pk'_{i+1} \| pk_{i+1}$, and to use pk'_{i+1} to sign the message m_i. (Looking at Figure 12.2, imagine replacing each message m_i with a public key pk'_{i+1} along with an additional arrow from pk'_{i+1} to m_i.) We now give the details.

In the variant we describe, the public key again consists of just a single public key pk_1 generated using Gen and the private key contains the associated private key sk_1. To sign the first message m_1, the signer first generates *two* new key-pairs (pk_2, sk_2), (pk'_2, sk'_2) using Gen, and then signs both pk'_2 and pk_2 using sk_1 to obtain $\sigma_1 \leftarrow \mathsf{Sign}_{sk_1}(pk'_2 \| pk_2)$. Finally, the signer signs m_1 using sk'_2 to obtain $\sigma'_1 \leftarrow \mathsf{Sign}_{sk'_2}(m_1)$. The signature that is output includes pk'_2, pk_2, σ_1, and σ'_1 and the signer adds $(pk'_2, pk_2, sk_2, \sigma_1)$ to its current state. We stress that neither m_1 nor σ'_1 need to be stored.

In general, when it comes time to sign the ith message the signer will have stored $\{(pk'_{j+1}, pk_{j+1}, sk_{j+1}, \sigma_j)\}_{j=1}^{i-1}$ as part of its state. To sign the ith message m_i, the signer generates two new key-pairs (pk_{i+1}, sk_{i+1}), (pk'_{i+1}, sk'_{i+1}) using Gen; signs pk'_{i+1} and pk_{i+1} using sk_i to obtain a signature $\sigma_i \leftarrow \mathsf{Sign}_{sk_i}(pk'_{i+1} \| pk_{i+1})$; and signs m_i using sk'_{i+1} to obtain $\sigma'_i \leftarrow \mathsf{Sign}_{sk'_{i+1}}(m_i)$. The actual signature that is output includes pk'_{i+1}, pk_{i+1}, σ_i, σ'_i, and also the values $\{pk'_{j+1}, pk_{j+1}, \sigma_j\}_{j=1}^{i-1}$. The signer then adds $(pk'_{i+1}, pk_{i+1}, sk_{i+1}, \sigma_i)$ to its state. Verification is done in the expected way.

As in the case of the original chain-based scheme, security follows on an intuitive level from the fact that every key-pair generated by the signer is used to sign only a single "message".

12.13 Prove Theorem 12.22.

Solution: Let Π^* denote Construction 12.20 and let Π denote the stateless variant, instantiated with a pseudorandom function F as discussed in the book. We assume that the signing algorithm of the underlying one-time signature scheme used in Construction 12.20 is deterministic. Fix a PPT adversary $\mathcal{A}$. We now construct an algorithm D that is given access to an oracle $\mathcal{O}(\cdot)$ and must decide whether $\mathcal{O}(\cdot)$ is a function chosen truly at random, or whether $\mathcal{O}(\cdot) = F_k(\cdot)$ for a uniform key k.

Algorithm D:
The algorithm is given 1^n and access to an oracle $\mathcal{O}(\cdot)$.

- Implement Construction 12.20 for $\mathcal{A}$. In the course of this experiment, every time randomness r_w is needed to generate (pk_w, sk_w) the algorithm obtains r_w by querying $\mathcal{O}(w)$.

- Eventually, $\mathcal{A}$ outputs a message/signature pair (m, σ). If this is a valid forgery then $\mathcal{A}$ outputs 1; otherwise, it outputs 0.

It is easy to see that if $\mathcal{O}(\cdot)$ is a function chosen at random, then D exactly implements scheme Π^* for $\mathcal{A}$; on the other hand, if $\mathcal{O}(\cdot) = F_k(\cdot)$ for a uniform key k then D implements Π for $\mathcal{A}$. Therefore,

$$\begin{aligned}&\left|\Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1]\right| \\ &= \left|\Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi}(n) = 1] - \Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi^*}(n) = 1]\right|\end{aligned}$$

By Theorem 12.21 says that there exists a negligible function negl for which $\Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi^*}(n) = 1] \leq \mathsf{negl}(n)$. The fact that F is a pseudorandom function means that there is a negligible function negl' such that $\left|\Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1]\right| \leq \mathsf{negl}'(n)$. Thus,

$$\Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi}(n) = 1] \leq \mathsf{negl}(n) + \mathsf{negl}'(n).$$

Since $\mathcal{A}$ was arbitrary, this proves the theorem.

12.14 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.

Solution: Say the CA receives a message, signed with respect to pk_B, requesting revocation of B's key. There are two cases: either B signed this message or not. If B signed the message, then the legitimate user is requesting revocation of the key and so it is ok for the CA to revoke it. If B did not sign the message, then B's key must have been compromised (or the scheme has been broken in some other way) in which case B would want its key to be revoked, anyway!

Chapter 13

Advanced Topics in Public-Key Encryption – Solutions

13.1 Construct and prove CPA-security for a KEM based on any trapdoor permutation by suitably generalizing Construction 11.34.

Solution: See Construction 13.1-S. A proof of security is similar to that of Theorem 11.35.

CONSTRUCTION 13.1-S

Let $\widehat{\Pi} = (\widehat{\mathsf{Gen}}, f, \mathsf{Inv})$ be a family of trapdoor permutations with hard-core predicate hc. Define a public-key encryption scheme as follows:

- Gen: on input 1^n, run $\widehat{\mathsf{Gen}}(1^n)$ to obtain (I, td). Output the public key I and the private key td.
- Encaps: on input $pk = I$ and 1^n, choose uniform $c_1 \in \mathcal{D}_I$. Then for $i = 1, \ldots, n$ do:
  (a) Compute $k_i := \mathsf{hc}_I(c_i)$.
  (b) Compute $c_{i+1} := f_I(c_i)$.

  Output the ciphertext c_{n+1} and the key $k = k_1 \cdots k_n$.
- Decaps: on input $sk = \mathsf{td}$, a ciphertext c_{n+1}, and 1^n, for $i = n, \ldots, 1$ do:
  (a) Compute $c_i := \mathsf{Inv}_I(c_{i+1})$.
  (b) Compute $k_i := \mathsf{hc}_I(c_i)$.

  Output the key $k = k_1 \cdots k_n$.

A KEM based on any trapdoor permutation.

13.2 Show that the isomorphism of Proposition 13.6 can be efficiently inverted when the factorization of N is known.

Solution: We are given $y \in \mathbb{Z}_{N^2}^*$ and want to find $a \in \mathbb{Z}_N$ and $b \in \mathbb{Z}_N^*$ such that

$$y = (1+N)^a \cdot b^N \bmod N^2.$$

The decryption algorithm of the Paillier scheme allows recovery of a. To recover b, we compute

$$b := \left[[y \bmod N]^{[N^{-1} \bmod \phi(N)]} \bmod N \right]$$

(recall that $\gcd(\phi(N), N) = 1$ and so $[N^{-1} \bmod \phi(N)]$ is well-defined). Let us see why this works. We claim first that

$$[y \bmod N] = [b^N \bmod N].$$

To see this, observe that $(1+N)^a \cdot b^N = y + r \cdot N^2$ for some integer r. (Note that this is an equation over the integers.) So

$$[y \bmod N] = \left[(1+N)^a \cdot b^N - rN^2 \bmod N \right] = [1 \cdot b^N \bmod N].$$

Given this, we have

$$\begin{aligned} \left[[y \bmod N]^{[N^{-1} \bmod \phi(N)]} \bmod N \right] &= \left[\left(b^N\right)^{[N^{-1} \bmod \phi(N)]} \bmod N \right] \\ &= \left[b^{[N \cdot N^{-1} \bmod \phi(N)]} \right] \\ &= [b \bmod N] \;=\; b. \end{aligned}$$

13.3 Let $\Phi(N^2)$ denote the set $\{(a,1) \mid a \in \mathbb{Z}_N\} \subset \mathbb{Z}_{N^2}^*$. Show that it is *not* hard to decide whether a given element $y \in \mathbb{Z}_{N^2}^*$ is in $\Phi(N^2)$.

Solution: Using Proposition 13.6(2), we have that

$$\Phi(N^2) = \{[1 + aN \bmod N^2] \mid a \in \mathbb{Z}_N\}.$$

It is easy to decide whether $y \in \Phi(N^2)$ by checking whether the integer $y - 1$ is divisible by N.

13.4 Let $\mathbb{G}$ be an abelian group. Show that the set of quadratic residues in $\mathbb{G}$ forms a subgroup.

Solution: Let $Q = \{x^2 \mid x \in \mathbb{G}\}$ denote the set of quadratic residues. We show that Q is a subgroup.

- Let $a, b \in Q$. So there exist $x, y \in \mathbb{G}$ such that $a = x^2$ and $b = y^2$. Then
$$ab = x^2 \cdot y^2 = (xy)^2,$$
and so $ab \in Q$. (Note we use here the fact that $\mathbb{G}$ is abelian.)
- Let 1 denote the identity in $\mathbb{G}$. Then $1^2 = 1$ and so $1 \in Q$.
- Let $a \in Q$. So there exists $x \in \mathbb{G}$ such that $a = x^2$. Then
$$a^{-1} = \left(x^2\right)^{-1} = \left(x^{-1}\right)^2,$$
and so $a^{-1} \in Q$.

- Associativity in Q is inherited from $\mathbb{G}$.

13.5 This question concerns the quadratic residues in the additive group $\mathbb{Z}_N$. (An element $y \in \mathbb{Z}_N$ is a quadratic residue if and only if there exists an $x \in \mathbb{Z}_N$ with $2x = y \bmod N$.)

(a) What are the quadratic residues in $\mathbb{Z}_p$ for p an odd prime?

(b) Let $N = pq$ be a product of two odd primes p and q. What are the quadratic residues in $\mathbb{Z}_N$?

(c) Let N be an even integer. What are the quadratic residues in $\mathbb{Z}_N$?

Solution:

(a) All elements are quadratic residues. To see this, note that for any $a \in \mathbb{Z}_p$, the equation $a = 2x \bmod p$ has the solution $x = [2^{-1} \cdot a \bmod p]$. (This follows since 2 is invertible modulo p.)

(b) As in the previous part, all elements are quadratic residues.

(c) Only even integers are quadratic residues in $\mathbb{Z}_N$. If $a \in \mathbb{Z}_N$ is even, then the equation $a = 2x \bmod N$ clearly has the solution $x = a/2$ (it also has the solution $x = (N + a)/2$). On the other hand, if a is odd then $a = 2x \bmod N$ cannot have a solution since, if it did, then there would exist an integer b with $a = 2x + bN$ in contradiction to the fact that a is odd.

13.6 Let $N = pq$ with p, q distinct, odd primes. Show a PPT algorithm for choosing a uniform element of $\mathcal{QNR}_N^{+1}$ *when the factorization of* N *is known.* (Your algorithm can have failure probability negligible in $\|N\|$.)

Solution: We can choose a uniform quadratic non-residue $x_p \in \mathcal{QNR}_p$ by repeatedly choosing uniform elements from $\mathbb{Z}_p^*$ until we find one that is not a quadratic residue; since half the elements of $\mathbb{Z}_p^*$ are not quadratic residues, if we choose n uniform elements from $\mathbb{Z}_p^*$ this procedure fails with only negligible probability. A uniform quadratic non-residue $x_q \in \mathcal{QNR}_q$ can be chosen similarly. We then use the bijection provided by the Chinese remainder theorem to compute $x \in \mathbb{Z}_N^*$ with $(x_p, x_q) \leftrightarrow x$.

13.7 Let $N = pq$ with p, q distinct, odd primes. Prove that if $x \in \mathcal{QR}_N$ then $[x^{-1} \bmod N] \in \mathcal{QR}_N$, and if $x \in \mathcal{QNR}_N^{+1}$ then $[x^{-1} \bmod N] \in \mathcal{QNR}_N^{+1}$.

Solution: If $x \in \mathcal{QR}_N$ then there exists a $y \in \mathbb{Z}_N^*$ such that $x = y^2 \bmod N$. So

$$x^{-1} = \left(y^2\right)^{-1} = \left(y^{-1}\right)^2 \bmod N\,,$$

and $[x^{-1} \bmod N]$ is a quadratic residue.

Let $x \leftrightarrow (x_p, x_q)$. If $x \in \mathcal{QNR}_N^{+1}$, then this implies that $x_p \in \mathcal{QNR}_p$ and $x_q \in \mathcal{QNR}_q$. This means that $[x_p^{-1} \bmod p] \in \mathcal{QNR}_p$ and $[x_q^{-1} \bmod q] \in \mathcal{QNR}_q$. (If not, then $[\left(x_p^{-1}\right)^{-1} \bmod p] \in \mathcal{QR}_p$, a contradiction.)

Since $[x^{-1} \bmod N] \leftrightarrow \big([x_p^{-1} \bmod p], [x_q^{-1} \bmod q]\big)$, this means that we have $[x^{-1} \bmod N] \in \mathcal{QNR}_N^{+1}$.

13.8 Let $N = pq$ with p, q distinct, odd primes, and fix $z \in \mathcal{QNR}_N^{+1}$. Show that choosing uniform $x \in \mathcal{QR}_N$ and setting $y := [z \cdot x \bmod N]$ gives a y that is uniformly distributed in $\mathcal{QNR}_N^{+1}$. That is, for any $\hat{y} \in \mathcal{QNR}_N^{+1}$

$$\Pr[z \cdot x = \hat{y} \bmod N] = 1/|\mathcal{QNR}_N^{+1}|,$$

where the probability is taken over uniform choice of $x \in \mathcal{QR}_N$.

Solution: $zx = \hat{y} \bmod N$ iff $x = z^{-1}\hat{y} \bmod N$. By the previous exercise, we know that $[z^{-1} \bmod N] \in \mathcal{QNR}_N^{+1}$. Using Corollary 13.25(2), we see that $[z^{-1}\hat{y} \bmod N] \in \mathcal{QR}_N$. So

$$\Pr[zx = \hat{y} \bmod N] = \Pr[x = z^{-1}\hat{y} \bmod N] = 1/|\mathcal{QR}_N|.$$

Since $|\mathcal{QR}_N| = |\mathcal{QNR}_N^{+1}|$, this proves the desired result.

13.9 Let N be the product of 5 distinct odd primes. If $y \in \mathbb{Z}_N^*$ is a quadratic residue, how many solutions are there to the equation $x^2 = y \bmod N$?

Solution: Let $N = \prod_{i=1}^{5} p_i$, with the p_i distinct odd primes. Applying the Chinese remainder theorem several times, we see there is an isomorphism $x \leftrightarrow (x_1, x_2, x_3, x_4, x_5)$ between elements $x \in \mathbb{Z}_N^*$ and elements $(x_1, x_2, x_3, x_4, x_5) \in \times_{i=1}^{5} \mathbb{Z}_{p_i}^*$. Using arguments as in this chapter, $y \leftrightarrow (y_1, y_2, y_3, y_4, y_5)$ is a quadratic residue modulo N iff each y_i is a quadratic residue modulo p_i, and $x \leftrightarrow (x_1, x_2, x_3, x_4, x_5)$ is a square root of y modulo N iff each x_i is a square root of y_i modulo p_i. Since each y_i has two square roots modulo p_i, there are 2^5 square roots of y modulo N.

13.10 Show that the Goldwasser-Micali encryption scheme is homomorphic if the message space $\{0, 1\}$ is viewed as the group $\mathbb{Z}_2$.

Solution: In Goldwasser-Micali encryption, a ciphertext corresponding to the bit m is of the form $[z^m \cdot x^2 \bmod N]$ for some $x \in \mathbb{Z}_N^*$. So if c_1, c_2 are encryptions of m_1, m_2, respectively, we have

$$c \stackrel{\text{def}}{=} c_1 \cdot c_2 = z^{m_1} \cdot x_1^2 \cdot z^{m_2} \cdot x_2^2 = z^{m_1+m_2} \cdot (x_1 x_2)^2 \bmod N.$$

If either m_1 or m_2 is 0, then c is clearly a ciphertext corresponding to the bit $[m_1 + m_2 \bmod 2]$. If $m_1 = m_2 = 1$, then

$$c = z^2 \cdot (x_1 x_2)^2 = z^0 \cdot (z \cdot x_1 x_2)^2 \bmod N,$$

and so here also c is a ciphertext corresponding to $[m_1 + m_2 \bmod 2]$.

13.11 Consider the following variation of the Goldwasser-Micali encryption scheme: $\mathsf{GenModulus}(1^n)$ is run to obtain (N, p, q) where $N = pq$ and $p = q = 3 \bmod 4$. (I.e., N is a Blum integer.) The public key is N and the secret key is $\langle p, q \rangle$. To encrypt $m \in \{0, 1\}$, the sender chooses uniform $x \in \mathbb{Z}_N$ and computes the ciphertext $c := [(-1)^m \cdot x^2 \bmod N]$.

(a) Prove that for N of the stated form, $[-1 \bmod N] \in \mathcal{QNR}_N^{+1}$.

(b) Prove that the scheme described has indistinguishable encryptions under a chosen-plaintext attack if deciding quadratic residuosity is hard relative to $\mathsf{GenModulus}$.

Solution:

(a) Looking at Algorithm 13.18, we see that $[-1 \bmod p]$ is a quadratic non-residue when $p = 3 \bmod 4$. This holds since $p = 4s + 3$ for some integer s, and so

$$(-1)^{\frac{p-1}{2}} = (-1)^{2s+1} = -1 \bmod p.$$

Similarly, $[-1 \bmod q]$ is a quadratic non-residue. It follows that $[-1 \bmod N] \in \mathcal{QNR}_N^{+1}$.

(b) Let Π denote the modified scheme. We prove that Π has indistinguishable encryptions in the presence of an eavesdropper.

Let $\mathcal{A}$ be a probabilistic polynomial-time adversary, and define

$$\varepsilon(n) \stackrel{\text{def}}{=} \Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1].$$

Consider the following PPT adversary D that attempts to solve the quadratic residuosity problem relative to $\mathsf{GenModulus}$:

> **Algorithm** D:
> The algorithm is given N and z as input and its goal is to determine if $z \in \mathcal{QR}_N$ or $z \in \mathcal{QNR}_N^{+1}$.
> - Set $pk = N$ and run $\mathcal{A}(pk)$ to obtain m_0, m_1. Assume without loss of generality that $m_0 = 0$ and $m_1 = 1$.
> - Set $c := z$. Give the ciphertext c to $\mathcal{A}$ and obtain a bit b'. Output b'.

Observe that

$$\Pr[D(N, \mathsf{qr}) = 0] = \Pr[\mathcal{A} \text{ outputs } 0 \mid c \text{ is an encryption of } 0],$$

where qr denotes a uniform quadratic residue. This follows because an encryption of 0 in Π is just a uniform quadratic residue. Similarly,

$$\Pr[D(N, \mathsf{qnr}) = 0] = \Pr[\mathcal{A} \text{ outputs } 0 \mid c \text{ is an encryption of } 1],$$

where qnr denotes a uniform element of $\mathcal{QNR}_N^{+1}$. This holds since an encryption of 1 in Π is just a uniform element in $\mathcal{QNR}_N^{+1}$. So

$$\begin{aligned}
&\Big|\Pr[D(N,\mathsf{qr})=1]-\Pr[D(N,\mathsf{qnr})=1]\Big| \\
&=\Big|\Pr[D(N,\mathsf{qr})=0]-\Pr[D(N,\mathsf{qnr})=0]\Big| \\
&=\Big|\Pr[\mathcal{A}\text{ outputs }0\mid c\text{ is an encryption of }0] \\
&\quad-\Pr[\mathcal{A}\text{ outputs }0\mid c\text{ is an encryption of }1]\Big| \\
&=\Big|\Pr[\mathcal{A}\text{ outputs }0\mid c\text{ is an encryption of }0 \\
&\quad+\Pr[\mathcal{A}\text{ outputs }1\mid c\text{ is an encryption of }1]-1\Big| \\
&=2\cdot\left|\varepsilon(n)-\frac{1}{2}\right|.
\end{aligned}$$

So if deciding quadratic residuosity is hard relative to $\mathsf{GenModulus}$, there must be a negligible function negl with $\varepsilon(n)\leq\frac{1}{2}+\mathsf{negl}(n)$.

13.12 Assume deciding quadratic residuosity is hard for $\mathsf{GenModulus}$. Show that this implies the hardness of distinguishing a uniform element of $\mathcal{QR}_N$ from a uniform element of $\mathcal{J}_N^{+1}$.

Solution: Fix a probabilistic polynomial-time algorithm D, and define

$$\varepsilon(n)\stackrel{\text{def}}{=}\left|\Pr[D(N,\mathsf{qr})=1]-\Pr[D(N,\mathcal{J})=1]\right|,$$

where the probabilities are taken over N generated by $\mathsf{GenModulus}$ and, in the first case, qr chosen uniformly from $\mathcal{QR}_N$ and, in the second case, $\mathcal{J}$ chosen uniformly from $\mathcal{J}_N^{+1}$. We know that with probability $1/2$ we have $\mathcal{J}\in\mathcal{QR}_N$ (in which case $\mathcal{J}$ is uniformly distributed over $\mathcal{QR}_N$) and with probability $1/2$ we have $\mathcal{J}\in\mathcal{QNR}_N^{+1}$ (in which case $\mathcal{J}$ is uniformly distributed over $\mathcal{QNR}_N^{+1}$). So,

$$Pr[D(N,\mathcal{J})=1]=\frac{1}{2}\cdot\Pr[D(N,\mathsf{qr})=1]+\frac{1}{2}\cdot\Pr[D(N,\mathsf{qnr})=1],$$

where qnr is chosen uniformly from $\mathcal{QNR}_N^{+1}$. Thus,

$$\varepsilon(n)=\frac{1}{2}\cdot\left|\Pr[D(N,\mathsf{qr})=1]-\Pr[D(N,\mathsf{qnr})=1]\right|.$$

So if deciding quadratic residuosity is hard relative to $\mathsf{GenModulus}$, there must exist a negligible function negl with $\varepsilon(n)\leq\mathsf{negl}(n)$.

13.13 Show that plain RSA encryption of a message m leaks $\mathcal{J}_N(m)$.

Solution: We prove that $\mathcal{J}_p(m^e) = \mathcal{J}_p(m)$ when p is prime and e is relatively prime to $p-1$. This implies that $\mathcal{J}_N(m^e) = \mathcal{J}_N(m)$. Since $\mathcal{J}_N(x)$ can be computed efficiently for any x, it follows that plain RSA encryption of m leaks $\mathcal{J}_N(m)$.

To see that $\mathcal{J}_p(m^e) = \mathcal{J}_p(m)$, note first that if $m = x^2 \bmod p$ is a quadratic residue modulo p, then so is $m^e = (x^e)^2 \bmod p$. Moreover, if $m^e = y^2 \bmod p$ is a quadratic residue modulo p, then so is $m = (m^e)^d = (y^d)^2 \bmod p$, where d is such that $ed = 1 \bmod p-1$.

13.14 Consider the following variation of the Goldwasser-Micali encryption scheme: $\mathsf{GenModulus}(1^n)$ is run to obtain (N, p, q). The public key is N and the secret key is $\langle p, q\rangle$. To encrypt a 0, the sender chooses n uniform elements $c_1, \ldots, c_n \leftarrow \mathcal{QR}_N$. To encrypt a 1, the sender chooses n uniform elements $c_1, \ldots, c_n \leftarrow \mathcal{J}_N^{+1}$. In each case, the resulting ciphertext is $c^* = \langle c_1, \ldots, c_n\rangle$.

(a) Show how the sender can generate a uniform element of $\mathcal{J}_N^{+1}$ in polynomial time.

(b) Suggest a way for the receiver to decrypt efficiently, though with error probability negligible in n.

(c) Prove that if deciding quadratic residuosity is hard relative to $\mathsf{GenModulus}$, then this scheme is CPA-secure.

Solution:

(a) A uniform element of $\mathcal{J}_N^{+1}$ can be generated in polynomial time by repeatedly sampling elements from $\mathbb{Z}_N^*$ and outputting the first such element with Jacobi symbol +1. This works because (1) it is possible to compute the Jacobi symbol in polynomial time, and (2) half the elements of $\mathbb{Z}_N^*$ are in $\mathcal{J}_N^{+1}$.

(b) Given a ciphertext $\langle c_1, \ldots, c_n\rangle$, the receiver can determine whether each c_i is a quadratic residue or not. If every c_i is a quadratic residue, the receiver outputs 0; otherwise, the receiver outputs 1.

When $\langle c_1, \ldots, c_n\rangle$ is an encryption of 0, then $c_i \in \mathcal{QR}_N$ for all i and so the receiver always outputs the correct result. When $\langle c_1, \ldots, c_n\rangle$ is an encryption of 1, then for every i we have $c_i \in \mathcal{QR}_N$ with probability 1/2 and so the probability that $c_i \in \mathcal{QR}_N$ for *all* i is 2^{-n}. Thus, the receiver errs in this case only with probability 2^{-n}.

(c) Let Π denote the encryption scheme above. We prove that Π has indistinguishable encryptions in the presence of an eavesdropper; as usual, this implies that it is CPA-secure.

Let $\mathcal{A}$ be a probabilistic polynomial-time adversary, and define

$$\varepsilon(n) \stackrel{\text{def}}{=} \Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1].$$

Consider the following PPT adversary D that attempts to distinguish uniform quadratic residues from uniform elements of $\mathcal{QNR}_N^{+1}$:

> **Algorithm** D:
> The algorithm is given N and z as input and its goal is to determine if $z \in \mathcal{QR}_N$ or $z \in \mathcal{QNR}_N^{+1}$.
>
> - Set $pk = N$ and run $\mathcal{A}(pk)$ to obtain m_0, m_1. Assume without loss of generality that $m_0 = 0$ and $m_1 = 1$.
> - For $i = 1$ to n, choose uniform $x_i \in \mathcal{QR}_N$ and uniform $b_i \in \{0,1\}$, and set $c_i := [z^{b_i} \cdot x_i \bmod N]$.
> - Give the ciphertext $c = \langle c_1, \ldots, c_n \rangle$ to $\mathcal{A}$ and obtain an output bit b'. Output $1 - b'$.

The thing to notice about the algorithm above is the following: if $z \in \mathcal{QR}_N$, then each c_i is a uniform element of $\mathcal{QR}_N$ and hence c corresponds to a random encryption of 0. This follows since for any $y \in \mathcal{QR}_N$ we have

$$\begin{aligned}\Pr[z^{b_i} \cdot x_i = y] &= \frac{1}{2} \cdot \Pr[z^0 \cdot x_i = y] + \frac{1}{2} \cdot \Pr[z^1 \cdot x_i = y] \\ &= \frac{1}{2} \cdot \frac{1}{|\mathcal{QR}_N|} + \frac{1}{2} \cdot \Pr[x_i = y/z] \;=\; \frac{1}{|\mathcal{QR}_N|},\end{aligned}$$

where x_i is chosen uniformly from $\mathcal{QR}_N$ in the above. On the other hand, when $z \in \mathcal{QNR}_N^{+1}$ then each c_i is a uniform element of $\mathcal{J}_N^{+1}$ and so c corresponds to a random encryption of 1. This is true because for any $y \in \mathcal{J}_N^{+1}$ we have

$$\Pr[z^{b_i} \cdot x_i = y] = \frac{1}{2} \cdot \Pr[x_i = y] + \frac{1}{2} \cdot \Pr[x_i = y/z],$$

while

$$\Pr[x_i = y] = \begin{cases} \frac{1}{|\mathcal{QR}_N|} & y \in \mathcal{QR}_N \\ 0 & y \in \mathcal{QNR}_N^{+1} \end{cases}$$

and (cf. Exercise 13.7)

$$\Pr[x_i = y/z] = \begin{cases} 0 & y \in \mathcal{QR}_N \\ \frac{1}{|\mathcal{QR}_N|} & y \in \mathcal{QNR}_N^{+1} \end{cases}.$$

So for any $y \in \mathcal{J}_N^{+1}$ we have $\Pr[z^{b_i} \cdot x_i = y] = \frac{1}{2 \cdot |\mathcal{QR}_N|} = \frac{1}{|\mathcal{J}_N^{+1}|}$.

From the above, we have

$$\begin{aligned}
&|\Pr[D(N,\mathsf{qr})=1]-\Pr[D(N,\mathsf{qnr})=1]| \\
&= \Big|\Pr[\mathcal{A}\text{ outputs }0 \mid c\text{ is an encryption of }0] \\
&\quad - \Pr[\mathcal{A}\text{ outputs }0 \mid c\text{ is an encryption of }1]\Big| \\
&= \Big|\Pr[\mathcal{A}\text{ outputs }0 \mid c\text{ is an encryption of }0 \\
&\quad + \Pr[\mathcal{A}\text{ outputs }1 \mid c\text{ is an encryption of }1]-1\Big| \\
&= 2\cdot\left|\varepsilon(n)-\frac{1}{2}\right|.
\end{aligned}$$

So if deciding quadratic residuosity is hard relative to $\mathsf{GenModulus}$, there must be a negligible function negl with $\varepsilon(n) \le \frac{1}{2} + \mathsf{negl}(n)$.

13.15 Let $\mathcal{G}$ be a polynomial-time algorithm that, on input 1^n, outputs a prime p with $\|p\| = n$ and a generator g of $\mathbb{Z}_p^*$. Prove that the DDH problem is *not* hard relative to $\mathcal{G}$.

Solution: Consider Algorithm 13.2-S. Note that the algorithm can be implemented efficiently since deciding quadratic residuosity modulo a prime can be done efficiently.

ALGORITHM 13.2-S
Solving DDH in $\mathbb{Z}_p^*$

Input: prime p and generator g of $\mathbb{Z}_p^*$; elements $h_1, h_2, h' \in \mathbb{Z}_p^*$
Output: Decide whether $h' = \mathsf{DH}_g(h_1, h_2)$

If $h_1 \in \mathcal{QR}_p$, set $b_1 := 1$; else set $b_1 := 0$
If $h_2 \in \mathcal{QR}_p$, set $b_2 := 1$; else set $b_2 := 0$
If $h' \in \mathcal{QR}_p$, set $b' := 1$; else set $b' := 0$
If $b' = b_1 \vee b_2$ then **return** "yes"; else **return** "no"

Let us analyze how well this algorithm performs. We first claim that when $h' = \mathsf{DH}_g(h_1, h_2)$ then it is always the case that $b' = b_1 \vee b_2$. To see this, note that $h \in \mathbb{Z}_p^*$ is a quadratic residue iff $\log_g h$ is even. (Here we rely on the fact that the order of $\mathbb{Z}_p^*$ is even. See the following exercise for a proof.) Since

$$\log_g h' = \left[\left(\log_g h_1 \cdot \log_g h_2\right) \bmod (p-1)\right],$$

$\log_g h'$ is even iff either $\log_g h_1$ or $\log_g h_2$ is even. The claim follows. We conclude that the algorithm always outputs "yes" when $h' = \mathsf{DH}_g(h_1, h_2)$.

On the other hand, when h' is uniform them b' is equally likely to be 0 or 1. So the probability that $b' = b_1 \vee b_2$ is exactly $1/2$.

Letting D denote Algorithm 13.2-S, we thus have (cf. Definition 8.63):

$$\left|\Pr[D(p,g,h_1,h_2,\hat{h})=1]-\Pr[D(p,g,h_1,h_2,\mathsf{DH}_g(h_1,h_2))=1]\right|=\frac{1}{2}$$

(above, $\hat{h}$ represents a uniformly-chosen element of $\mathbb{Z}_p^*$). This, of course, is not negligible.

13.16 The discrete logarithm problem is believed to be hard for $\mathcal{G}$ as in the previous exercise. This means that the function (family) $f_{p,g}$ where $f_{p,g}(x) \stackrel{\text{def}}{=} [g^x \bmod p]$ is one-way. Let $\mathsf{lsb}(x)$ denote the least-significant bit of x. Show that lsb is *not* a hard-core predicate for $f_{p,g}$.

Solution: Let g be a generator of $\mathbb{Z}_p^*$. We claim that $h \in \mathbb{Z}_p^*$ is a quadratic residue iff $\log_g h$ is even. One direction is easy: if $\log_g h$ is even, then

$$\left(g^{\log_g h/2}\right)^2 = g^{\log_g h} = h$$

and so h is a quadratic residue. For the other direction, let $h = y^2 \bmod p$ for some y. Then

$$h = \left(g^{\log_g y}\right)^2 = g^{2\log_g y},$$

and so $\log_g h = [2\log_g y \bmod (p-1)]$. (Recall that $p-1$ is the order of the group $\mathbb{Z}_p^*$.) But since $p-1$ is even, $[2\log_g y \bmod (p-1)]$ is even as well.

Focusing on the problem at hand, this means that $\mathsf{lsb}(x)$ is 0 iff $f_{p,q}(x)$ is a quadratic residue modulo p. Since quadratic residuosity modulo p can be decided easily, this implies that lsb is not a hard-core predicate for $f_{p,g}$.

13.17 Consider the plain Rabin encryption scheme where a message $m \in \mathcal{QR}_N$ is encrypted relative to a public key N (where N is a Blum integer) by computing the ciphertext $c := [m^2 \bmod N]$. Show a chosen-ciphertext attack on this scheme that recovers the entire private key.

Solution: Here is an attack: Choose uniform $x \in \mathbb{Z}_N^*$ (*not* necessarily in $\mathcal{QR}_N$), compute $c := [x^2 \bmod N]$, and request the decryption of c. Call this m. If $m \neq \pm x \bmod N$ (which occurs with probability 1/2) then N can be factored as in Lemma 13.35.

13.18 The plain Rabin signature scheme is like the plain RSA signature scheme, except using the Rabin trapdoor permutation. Show an attack on plain Rabin signatures by which the attacker learns the signer's private key.

Solution: In the plain Rabin signature scheme, a signature on $y \in \mathcal{QR}_N$ is given by $x = \sqrt{y} \bmod N$ with $x \in \mathcal{QR}_N$. In a forgery attack, the adversary $\mathcal{A}$ is allowed to ask for signatures on any value. Thus, $\mathcal{A}$ can choose a uniform $\hat{x} \in \mathbb{Z}_N^*$ and query its signing oracle with $y =$

$\hat{x}^2 \bmod N$. It will receive back $x = \sqrt{y} \bmod N$ with $x \in \mathcal{QR}_N$. If $\hat{x} \neq \pm x \bmod N$, then (as in Lemma 13.35) this will enable $\mathcal{A}$ to factor N. If $\hat{x} = \pm x \bmod N$ then $\mathcal{A}$ can try again until it succeeds.

13.19 Let N be a Blum integer.

(a) Define the set $S \stackrel{\text{def}}{=} \{x \in \mathbb{Z}_N^* \mid x < N/2 \text{ and } \mathcal{J}_N(x) = +1\}$. Define the function $f_N : S \to \mathbb{Z}_N^*$ by:

$$f_N(x) = \begin{cases} [x^2 \bmod N] & \text{if } [x^2 \bmod N] < N/2 \\ [-x^2 \bmod N] & \text{if } [x^2 \bmod N] > N/2 \end{cases}$$

Show that f_N is a permutation over S.

(b) Define a family of trapdoor permutations based on factoring using f_N as defined above.

Solution:

(a) Let $\mathsf{low} \stackrel{\text{def}}{=} \{x \in \mathbb{Z}_N^* \mid x < N/2\}$ and $\mathsf{high} \stackrel{\text{def}}{=} \{x \in \mathbb{Z}_N^* \mid x > N/2\}$. Note that $S = \mathcal{J}_N^{+1} \cap \mathsf{low}$. Observe that $x \in \mathsf{low}$ if and only if $[-x \bmod N] \in \mathsf{high}$.

We first show that $f_N(x) \in S$ for any $x \in \mathbb{Z}_N^*$. Clearly $f_N(x) \in \mathsf{low}$. If N is a Blum integer, then $[-1 \bmod N] \in \mathcal{QNR}_N^{+1}$ (see Exercise 13.11(a)) and so $\mathcal{J}_N(y) = \mathcal{J}_N(-y)$ for all $y \in \mathbb{Z}_N^*$. Since $\mathcal{J}_N([x^2 \bmod N]) = 1$, this shows that $f_N(x) \in \mathcal{J}_N^{+1}$.

We show that f_N is a permutation by showing that it is *surjective*; that is, for all $y \in S$ there is an $x \in S$ such that $f_N(x) = y$. If $y \in S$ is a quadratic residue, then y has a square root $\hat{x} \in \mathcal{QR}_N$ by Proposition 13.37. Both $\hat{x}$ and $[-\hat{x} \bmod N]$ have Jacobi symbol $+1$; both of these are square roots of y; and one of them, call it x, is in low. So $x \in S$ is the desired inverse in this case.

If $y \in S$ is a quadratic non-residue, then $[-y \bmod N] \in \mathcal{QR}_N$ (this uses the fact that $[-1 \bmod N]$ is a quadratic non-residue). Applying the same argument as above shows that there exists an $x \in S$ with $x^2 = -y \bmod N$, and this x is the desired inverse of y.

(b) Define a trapdoor permutation $(\mathsf{Gen}, \mathsf{Samp}, f)$ as follows. Gen is as in Section 13.5.2: on input 1^n it runs $\mathsf{GenModulus}(1^n)$ to obtain (N, p, q) and outputs $I = N$; the domain is just the set S as defined in this exercise. f is simply f_N as defined in this exercise. Samp repeatedly chooses elements of $\mathbb{Z}_N^*$ and outputs the first such element in S; this can be implemented in polynomial time (with negligible failure probability) since membership in S can be tested efficiently, and $|S| = |\mathcal{J}_N^{+1}|/2 = |\mathbb{Z}_N^*|/4$. (We did not prove the claim about the size of S, but it is not hard to prove given the solution to part (a).)

The proof that this is a trapdoor permutation if factoring is hard goes along the same lines as the proof of Theorem 13.36, using the observation that we can map $y \in \mathcal{QR}_N$ to either y or $-y$ (both of which have the same square root as y), one of which is in S.

13.20 (a) Let N be a Blum integer. Define the function $\mathsf{half}_N : \mathbb{Z}_N^* \to \{0,1\}$ as

$$\mathsf{half}_N(x) = \begin{cases} 0 \text{ if } x < N/2 \\ 1 \text{ if } x > N/2 \end{cases}$$

Show that the function $f : \mathbb{Z}_N^* \to \mathcal{QR}_N \times \{-1,+1\} \times \{0,1\}$ defined as

$$f(x) = [x^2 \bmod N], \mathcal{J}_N(x), \mathsf{half}_N(x)$$

is one-to-one.

(b) Suggest a "padded Rabin" encryption scheme that encrypts messages of length n. (All algorithms of your scheme should run in polynomial time, and the scheme should have correct decryption. Although a proof of security is unlikely, your scheme should not be susceptible to any obvious attacks.)

Solution:

(a) Since $|\mathcal{QR}_N \times \{-1,+1\} \times \{0,1\}| = |\mathbb{Z}_N^*|$, we prove f is one-to-one by showing that it is onto; that is, that for every $z \in \mathcal{QR}_N \times \{-1,+1\} \times \{0,1\}$ there is an $x \in \mathbb{Z}_N^*$ with $f(x) = z$.
Fix $z = \langle y, b_1, b_2 \rangle \in \mathcal{QR}_N \times \{-1,+1\} \times \{0,1\}$ where $y \in \mathcal{QR}_N$, $b_1 \in \{-1,+1\}$, and $b_2 \in \{0,1\}$. Let

$$(x_p, x_q), \ (-x_p, x_q), \ (x_p, -x_q), \ (-x_p, -x_q)$$

be the four square roots of y in the Chinese remaindering representation (where p and q are the prime factors of N). Since N is a Blum integer, $\mathcal{J}_p(-1) = -1$ (see Exercise 11.7(a)) and so one of the square roots of y, call it x, has Jacobi symbol b_1. (E.g., if $\mathcal{J}_N((x_p, x_q)) = -b_1$, then $\mathcal{J}_N((-x_p, x_q)) = b_1$.) Furthermore, $\mathcal{J}_N(-x) = \mathcal{J}_N(x)$ (using now the fact that $\mathcal{J}_q(-1) = -1$ also) and either $\mathsf{half}_N(x)$ or $\mathsf{half}_N([-x \bmod N])$ is equal to b_2.

(b) An answer is completely analogous to the padded RSA encryption scheme of Section 11.5.2.

Appendix B

Basic Algorithmic Number Theory – Solutions

B.1 Prove correctness of the extended Euclidean algorithm.

Solution: We prove correctness by induction on the second input b. When $b = 1$ then b always divides a and Algorithm B.10 returns $(b, 0, 1)$. This is correct, since $b = \gcd(a, b)$ and $0 \cdot a + 1 \cdot b = b$.

Assume correctness of Algorithm B.10 for all (positive) values of b up to some bound B; we prove that correctness holds for $b = B + 1$. Consider an execution of $\mathsf{eGCD}(a, b)$. If $b \mid a$ then the algorithm returns $(b, 0, 1)$ and this is a correct solution (as above). Otherwise, the algorithm makes a recursive call to $\mathsf{eGCD}(b, r)$ with $r = [a \bmod b]$. Note that $0 < r < b$. By our inductive assumption, we know that $\mathsf{eGCD}(b, r)$ outputs (d, X, Y) with $d = \gcd(b, r)$ and $Xb + Yr = d$; the final output of the algorithm is $(d, Y, X - Yq)$ where q is such that $a - r = qb$. We can verify correctness of this output as follows:

- Proposition B.6 shows that $d = \gcd(a, b)$.
- We have

$$Ya + (X - Yq)b = Ya + Xb - Yqb = Xb + Y(a - qb) = Xb + Yr = d,$$

as required.

B.2 Prove that the extended Euclidean algorithm runs in time polynomial in the lengths of its inputs.

Solution: For any given input (a, b), the inputs used in the recursive calls to eGCD in an execution of Algorithm B.10 are exactly the same as the inputs used in the recursive calls to GCD in an execution of Algorithm B.7. So the number of recursive calls is identical in each case. Since each recursive step (and, in particular, division-with-remainder) can be done in polynomial time, it follows from Corollary B.9 that the entire algorithm runs in polynomial time.

B.3 Show how to determine that an n-bit string is in $\mathbb{Z}_N^*$ in polynomial time.

Solution: See Algorithm B.1-S.

ALGORITHM B.1-S
Determining membership in $\mathbb{Z}_N^*$

Input: Modulus N; integer x
Output: Determine whether $x \in \mathbb{Z}_N^*$

if $x > N$ or $x = 0$, **return** "no"
if $\gcd(x, N) \neq 1$ **return** "no"
return "yes"