Cryptography: Theory and Practice Fourth Edition
Solution Manual

Douglas R. Stinson

Chapter 2

Classical Cryptography

2.1 Evaluate the following:

(a) 7503 mod 81.

Answer: $7503 \bmod 81 = 51$.

(b) $(-7503) \bmod 81$.

Answer: $(-7503) \bmod 81 = 45$.

(c) 81 mod 7503.

Answer: $81 \bmod 7503 = 81$.

(d) $(-81) \bmod 7503$.

Answer: $(-81) \bmod 7503 = 7422$.

2.2 Suppose that $a, m > 0$, and $a \not\equiv 0 \pmod{m}$. Prove that

$$(-a) \bmod m = m - (a \bmod m).$$

Answer: We have $a = qm + r$, where $1 \leq r \leq m-1$ and $r = a \bmod m$. Then $-a = -(q+1)m + (m-r)$, where $1 \leq m-r \leq m-1$. Therefore $(-a) \bmod m = m - r = m - (a \bmod m)$.

2.3 Prove that $a \bmod m = b \bmod m$ if and only if $a \equiv b \pmod{m}$.

Answer: $a \bmod m = b \bmod m$ implies that $a = q_1 m + r$ and $b = q_2 m + r$, where $0 \leq r \leq m-1$. Then $a - b = (q_1 - q_2)m$, so $a \equiv b \pmod{m}$. Conversely, suppose $a \equiv b \pmod{m}$. Then $a - b = qm$. Let $r = a \bmod m$. Then $a = q_1 m + r$ for some q_1, and hence $b = a - qm = (q_1 - q)m + r$, so $b \bmod m = r$.

2.4 Prove that $a \bmod m = a - \lfloor \frac{a}{m} \rfloor m$, where $\lfloor x \rfloor = \max\{y \in \mathbb{Z} : y \leq x\}$.

Answer: $(a - m + 1)/m \leq \lfloor \frac{a}{m} \rfloor \leq a/m$, so $a - m + 1 \leq m\lfloor \frac{a}{m} \rfloor \leq a$, and hence $0 \leq a - \lfloor \frac{a}{m} \rfloor m \leq m - 1$.

2.5 Use exhaustive key search to decrypt the following ciphertext, which was encrypted using a *Shift Cipher*:

```
BEEAKFYDJXUQYHYJIQRYHTYJIQFBQDUYJIIKFUHCQD.
```

Answer: The key is 16, and the plaintext is the following:

Look, up in the air, it's a bird, it's a plane, it's Superman!

2.6 If an encryption function e_K is identical to the decryption function d_K, then the key K is said to be an ***involutory key***. Find all the involutory keys in the *Shift Cipher* over $\mathbb{Z}_{26}$.

Answer: The involutory keys are 0 and 13.

2.7 Determine the number of keys in an *Affine Cipher* over $\mathbb{Z}_m$ for $m = 30, 100$ and 1225.

Answer: $30 = 2 \times 3 \times 5$, so $\phi(30) = 1 \times 2 \times 4 = 8$. The affine cipher over $\mathbb{Z}_{30}$ has $30 \times 8 = 240$ keys.
$100 = 2^2 \times 5^2$, so $\phi(100) = (2^2 - 2)(5^2 - 5) = 40$. The affine cipher over $\mathbb{Z}_{100}$ has $100 \times 40 = 4000$ keys.
$1225 = 5^2 \times 7^2$, so $\phi(1225) = (5^2 - 5)(7^2 - 7) = 840$. The affine cipher over $\mathbb{Z}_{1225}$ has $1225 \times 840 = 1029000$ keys.

2.8 List all the invertible elements in $\mathbb{Z}_m$ for $m = 28, 33$, and 35.

Answer: The invertible elements in $\mathbb{Z}_{28}$ are 1, 3, 5, 9, 11, 13, 15, 17, 19, 23, 25 and 27.
The invertible elements in $\mathbb{Z}_{33}$ are 1, 2, 4, 5, 7, 8, 10, 13, 14, 16, 17, 19, 20, 23, 25, 26, 28, 29, 31 and 32.
The invertible elements in $\mathbb{Z}_{35}$ are 1, 2, 3, 4, 6, 8, 9, 11, 12, 13, 16, 17, 18, 19, 22, 23, 24, 26, 27, 29, 31, 32, 33 and 34.

2.9 For $1 \leq a \leq 28$, determine $a^{-1} \bmod 29$ by trial and error.

Answer: $1^{-1} = 28$, $2^{-1} = 15$, $3^{-1} = 10$, $4^{-1} = 22$, $5^{-1} = 6$, $6^{-1} = 5$, $7^{-1} = 25$, $8^{-1} = 11$, $9^{-1} = 13$, $10^{-1} = 3$, $11^{-1} = 8$, $12^{-1} = 17$, $13^{-1} = 9$, $14^{-1} = 27$, $15^{-1} = 2$, $16^{-1} = 20$, $17^{-1} = 12$, $18^{-1} = 21$, $19^{-1} = 26$, $20^{-1} = 16$, $21^{-1} = 18$, $22^{-1} = 4$, $23^{-1} = 24$, $24^{-1} = 23$, $25^{-1} = 7$, $26^{-1} = 19$, $27^{-1} = 14$ and $28^{-1} = 28$.

2.10 Suppose that $K = (5, 21)$ is a key in an *Affine Cipher* over $\mathbb{Z}_{29}$.

(a) Express the decryption function $d_K(y)$ in the form $d_K(y) = a'y + b'$, where $a', b' \in \mathbb{Z}_{29}$.

Answer: $d_K(y) = 6y + 19$.

(b) Prove that $d_K(e_K(x)) = x$ for all $x \in \mathbb{Z}_{29}$.

Answer: $6(5x + 21) + 19 \equiv 30x + 145 \equiv x \pmod{29}$.

2.11 (a) Suppose that $K = (a, b)$ is a key in an *Affine Cipher* over $\mathbb{Z}_n$. Prove that K is an involutory key if and only if $a^{-1} \bmod n = a$ and $b(a + 1) \equiv 0 \pmod{n}$.

Answer: $K = (a, b)$ is an involutory key if and only if $a(ax + b) + b \equiv x \pmod{n}$ for all $x \in \mathbb{Z}_n$. Clearly $a(ax + b) + b \equiv a^2x + b(a + 1) \pmod{n}$, so we require that $a^2 \equiv 1 \pmod{n}$ and $b(a + 1) \equiv 0 \pmod{n}$.

(b) Determine all the involutory keys in the *Affine Cipher* over $\mathbb{Z}_{15}$.

Answer: $a^2 \equiv 1 \pmod{15}$ if and only if $a = 1, 4, 11$ or 14. If $a = 1$, then $b = 0$. If $a = 4$, then $b = 0, 3, 6, 9$ or 12. If $a = 11$, then $b = 0, 5$ or 10. Finally, if $a = 14$, then b can be any element of $\mathbb{Z}_{15}$.

(c) Suppose that $n = pq$, where p and q are distinct odd primes. Prove that the number of involutory keys in the *Affine Cipher* over $\mathbb{Z}_n$ is $n + p + q + 1$.

Answer: There are four possible values for a, namely, $a = 1$; $a = -1 \bmod n$; the solution to the system $a \equiv 1 \pmod{p}$, $a \equiv -1 \pmod{q}$; and the solution to the system $a \equiv -1 \pmod{p}$, $a \equiv 1 \pmod{q}$. If $a = 1$, then $b = 0$. If $a = -1$, then b can be any element in $\mathbb{Z}_n$. In the third case, we require that $b \equiv 0 \pmod{q}$, so there are p possible values for b. In the fourth case, we require that $b \equiv 0 \pmod{p}$, so there are q possible values for b. The total number of involutory keys is therefore $n + p + q + 1$.

2.12 (a) Let p be prime. Prove that the number of 2×2 matrices that are invertible over $\mathbb{Z}_p$ is $(p^2 - 1)(p^2 - p)$.

HINT Since p is prime, $\mathbb{Z}_p$ is a field. Use the fact that a matrix over a field is invertible if and only if its rows are linearly independent vectors (i.e., there does not exist a non-zero linear combination of the rows whose sum is the vector of all 0's).

Answer: The first row can be any non-zero vector, so there are $p^2 - 1$ possibilities. Given the first row, say r, the second row can be any vector that is not a scalar multiple of r. Therefore there are $p^2 - p$ possibilities for the second row, given the first row. Hence, the total number of 2×2 invertible matrices is $(p^2 - 1)(p^2 - p)$.

(b) For p prime and $m \geq 2$ an integer, find a formula for the number of $m \times m$ matrices that are invertible over $\mathbb{Z}_p$.

Answer: The number of invertible matrices is

$$(p^m - 1)(p^m - p)(p^m - p^2) \cdots (p^m - p^{m-1}).$$

2.13 For $n = 6, 9$, and 26, how many 2×2 matrices are there that are invertible over $\mathbb{Z}_n$?

Answer: For $n = 6$, there are $(2^2 - 1)(2^2 - 2)(3^2 - 1)(3^2 - 3) = 1728$ invertible matrices (use the Chinese remainder theorem and Exercise 2.12). Similarly, for $n = 26$, there are $(2^2 - 1)(2^2 - 2)(13^2 - 1)(13^2 - 13) = 157248$ invertible matrices. For $n = 9$, there are $3^4(3^2 - 1)(3^2 - 3) = 3888$ invertible matrices.

2.14 (a) Prove that $\det A \equiv \pm 1 \pmod{26}$ if A is a matrix over $\mathbb{Z}_{26}$ such that $A = A^{-1}$.

Answer: If $A = A^{-1}$, then $A^2 = I$ and hence $(\det A)^2 \equiv 1 \pmod{26}$. This implies that $\det A \equiv \pm 1 \pmod{26}$.

(b) Use the formula given in Corollary 2.4 to determine the number of involutory keys in the *Hill Cipher* (over $\mathbb{Z}_{26}$) in the case $m = 2$.

Answer: If $\det A \equiv 1 \pmod{26}$ then there are 8 involutory matrices, and if $\det A \equiv -1 \pmod{26}$ then there are 728 involutory matrices, for a total of 736 involutory matrices.

The eight involutory matrices with determinant 1 are as follows:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 25 & 0 \\ 0 & 25 \end{pmatrix}, \quad \begin{pmatrix} 1 & 13 \\ 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 25 & 13 \\ 0 & 25 \end{pmatrix},$$
$$\begin{pmatrix} 1 & 0 \\ 13 & 1 \end{pmatrix}, \quad \begin{pmatrix} 25 & 0 \\ 13 & 25 \end{pmatrix}, \quad \begin{pmatrix} 12 & 13 \\ 13 & 12 \end{pmatrix}, \quad \begin{pmatrix} 14 & 13 \\ 13 & 14 \end{pmatrix}.$$

The involutory matrices with determinant -1 have the following forms when reduced modulo 2:

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

When reduced modulo 13, an involutory matrix with determinant -1 has the following form:

$$\begin{pmatrix} x & y \\ z & -x \end{pmatrix},$$

where $x^2 + yz \equiv 1 \pmod{13}$. The number of triples $(x, y, z) \in (\mathbb{Z}_{13})^3$ that satisfy this congruence is easily computed: if $x = 2$ or 12, then there are 25 ordered pairs (y, z); and if $x \neq 2, 12$, then there are 12 ordered pairs (y, z). Hence, the total number of triples is $2 \times 25 + 11 \times 12 = 182$. Now we can use the Chinese remainder theorem to combine any solution modulo 2 with any solution modulo 13, so the total number of solutions modulo 26 is $4 \times 182 = 728$, as stated above.

2.15 Determine the inverses of the following matrices over $\mathbb{Z}_{26}$:

(a) $\begin{pmatrix} 11 & 15 \\ 1 & 20 \end{pmatrix}$

Answer: The inverse matrix is

$$\begin{pmatrix} 2 & 5 \\ 9 & 5 \end{pmatrix}.$$

(b) $\begin{pmatrix} 1 & 11 & 12 \\ 4 & 23 & 2 \\ 17 & 15 & 9 \end{pmatrix}$

Answer: The inverse matrix is

$$\begin{pmatrix} 25 & 11 & 22 \\ 10 & 13 & 4 \\ 17 & 24 & 1 \end{pmatrix}.$$

2.16 (a) Suppose that π is the following permutation of $\{1,\ldots,8\}$:

x	1	2	3	4	5	6	7	8
$\pi(x)$	4	1	6	2	7	3	8	5

Compute the permutation π^{-1}.

Answer: The permutation π^{-1} is as follows:

x	1	2	3	4	5	6	7	8
$\pi^{-1}(x)$	2	4	6	1	8	3	5	7

(b) Decrypt the following ciphertext, for a *Permutation Cipher* with $m = 8$, which was encrypted using the key π:

`TGEEMNELNNTDROEOAAHDOETCSHAEIRLM.`

Answer: The plaintext is the following:

Gentlemen do not read each other's mail.

2.17 (a) Prove that a permutation π in the *Permutation Cipher* is an involutory key if and only if $\pi(i) = j$ implies $\pi(j) = i$, for all $i, j \in \{1,\ldots,m\}$.

Answer: A permutation π is involutory if and only if $\pi(\pi(i)) = i$ for all i. Denoting $\pi(i) = j$, it must be the case that $\pi(j) = i$.

(b) Determine the number of involutory keys in the *Permutation Cipher* for $m = 2, 3, 4, 5$, and 6.

Answer: An involutory permutation must consist of fixed points and cycles of length two.
For $m = 2$, there are 2 involutory permutations.
For $m = 3$, there are 4 involutory permutations.
For $m = 4$, there are 3 permutations consisting of two cycles of length 2; 6 permutations having one cycle of length 2 and two fixed points; and 1 permutation consisting of 4 fixed points. The total number of involutory permutations is 10.
For $m = 5$, there are 15 permutations consisting of two cycles of length 2 and one fixed point; 10 permutations having one cycle of length 2 and three fixed points; and 1 permutation consisting of 5 fixed points. The total number of involutory permutations is 26.
For $m = 6$, there are 15 permutations consisting of three cycles of length 2; 45 permutations consisting of two cycles of length 2 and two fixed points; 15 permutations having one cycle of length 2 and four fixed points; and 1 permutation consisting of 5 fixed points. The total number of involutory permutations is 76.

2.18 Consider the following linear recurrence over $\mathbb{Z}_2$ of degree four:

$$z_{i+4} = (z_i + z_{i+1} + z_{i+2} + z_{i+3}) \bmod 2,$$

$i \geq 0$. For each of the 16 possible initialization vectors $(z_0, z_1, z_2, z_3) \in (\mathbb{Z}_2)^4$, determine the period of the resulting keystream.

Answer: $(0,0,0,0)$ produces a keystream with period 1, and all other initialization vectors produce a keystream with period 5.

2.19 Redo the preceding question, using the recurrence

$$z_{i+4} = (z_i + z_{i+3}) \bmod 2,$$

$i \geq 0$.

Answer: $(0,0,0,0)$ produces a keystream with period 1, and all other initialization vectors produce a keystream with period 15.

2.20 Suppose we construct a keystream in a synchronous stream cipher using the following method. Let $K \in \mathcal{K}$ be the key, let $\mathcal{L}$ be the keystream alphabet, and let Σ be a finite set of states. First, an initial state $\sigma_0 \in \Sigma$ is determined from K by some method. For all $i \geq 1$, the state σ_i is computed from the previous state σ_{i-1} according to the following rule:

$$\sigma_i = f(\sigma_{i-1}, K),$$

where $f : \Sigma \times \mathcal{K} \to \Sigma$. Also, for all $i \geq 1$, the keystream element z_i is computed using the following rule:

$$z_i = g(\sigma_i, K),$$

where $g : \Sigma \times \mathcal{K} \to \mathcal{L}$. Prove that any keystream produced by this method has period at most $|\Sigma|$.

Answer: For a fixed key K, each σ_i can be regarded as a function of σ_{i-1}. Define

$$t = \min\{i \geq 1 : \sigma_i \in \{\sigma_0, \ldots, \sigma_{i-1}\}\}.$$

It follows from the pigeon-hole principle that $t \leq |\Sigma|$, because $\sigma_i \in \Sigma$ for all $i \geq 0$. Suppose that $\sigma_t = \sigma_s$, where $0 \leq s < t$. Then it $\sigma_{i+t-s} = \sigma_i$ for all $i \geq s$. Hence, $z_{i+t-s} = z_i$ for all $i \geq s$, and the keystream has period $t - s \leq |\Sigma|$.

2.21 Below are given four examples of ciphertext, one obtained from a *Substitution Cipher*, one from a *Vigenère Cipher*, one from an *Affine Cipher*, and one unspecified. In each case, the task is to determine the plaintext.

Give a clearly written description of the steps you followed to decrypt each ciphertext. This should include all statistical analysis and computations you performed.

The first two plaintexts were taken from *The Diary of Samuel Marchbanks*, by Robertson Davies, Clarke Irwin, 1947; the fourth was taken from *Lake Wobegon Days*, by Garrison Keillor, Viking Penguin, Inc., 1985.

(a) *Substitution Cipher*:

```
EMGLOSUDCGDNCUSWYSFHNSFCYKDPUMLWGYICOXYSIPJCK
QPKUGKMGOLICGINCGACKSNISACYKZSCKXECJCKSHYSXCG
OIDPKZCNKSHICGIWYGKKGKGOLDSILKGOIUSIGLEDSPWZU
GFZCCNDGYYSFUSZCNXEOJNCGYEOWEUPXEZGACGNFGLKNS
ACIGOIYCKXCJUCIUZCFZCCNDGYYSFEUEKUZCSOCFZCCNC
IACZEJNCSHFZEJZEGMXCYHCJUMGKUCY
```

HINT F decrypts to w.

Answer: The plaintext is as follows:

> I may not be able to grow flowers, but my garden produces just as many dead leaves, old overshoes, pieces of rope, and bushels of dead grass as anybody's, and today I bought a wheelbarrow to help in clearing it up. I have always loved and respected the wheelbarrow. It is the one wheeled vehicle of which I am perfect master.

(b) *Vigenère Cipher*:

```
KCCPKBGUFDPHQTYAVINRRTMVGRKDNBVFDETDGILTXRGUD
DKOTFMBPVGEGLTGCKQRACQCWDNAWCRXIZAKFTLEWRPTYC
QKYVXCHKFTPONCQQRHJVAJUWETMCMSPKQDYHJVDAHCTRL
SVSKCGCZQQDZXGSFRLSWCWSJTBHAFSIASPRJAHKJRJUMV
GKMITZHFPDISPZLVLGWTFPLKKEBDPGCEBSHCTJRWXBAFS
PEZQNRWXCVYCGAONWDDKACKAWBBIKFTIOVKCGGHJVLNHI
FFSQESVYCLACNVRWBBIREPBBVFEXOSCDYGZWPFDTKFQIY
CWHJVLNHIQIBTKHJVNPIST
```

Answer: The keyword is *CRYPTO*, and the plaintext is as follows:

> I learned how to calculate the amount of paper needed for a room when I was at school. You multiply the square footage of the walls by the cubic contents of the floor and ceiling combined, and double it. You then allow half the total for openings such as windows and doors. Then you allow the other half for matching the pattern. Then you double the whole thing again to give a margin of error, and then you order the paper.

(c) *Affine Cipher*:

```
KQEREJEBCPPCJCRKIEACUZBKRVPKRBCIBQCARBJCVFCUP
KRIOFKPACUZQEPBKRXPEIIEABDKPBCPFCDCCAFIEABDKP
BCPFEQPKAZBKRHAIBKAPCCIBURCCDKDCCJCIDFUIXPAFF
ERBICZDFKABICBBENEFCUPJCVKABPCYDCCDPKBCOCPERK
IVKSCPICBRKIJPKABI
```

Answer: The key is $(19, 4)$. The plaintext consists of the French lyrics to "O Canada":

Ô Canada!
Terre de nos aïeux.
Ton front est ceint,
De fleurons glorieux.
Car ton bras
Sait porter l'épée,
Il sait porter la croix.
Ton histoire est une épopée,
des plus brillants exploits.
Et ta valeur,
de foi trempée,
protègera nos foyers et nos droits.

(d) unspecified cipher:

```
BNVSNSIHQCEELSSKKYERIFJKXUMBGYKAMQLJTYAVFBKVT
DVBPVVRJYYLAOKYMPQSCGDLFSRLLPROYGESEBUUALRWXM
MASAZLGLEDFJBZAVVPXWICGJXASCBYEHOSNMULKCEAHTQ
OKMFLEBKFXLRRFDTZXCIWBJSICBGAWDVYDHAVFJXZIBKC
GJIWEAHTTOEWTUHKRQVVRGZBXYIREMMASCSPBNLHJMBLR
FFJELHWEYLWISTFVVYFJCMHYUYRUFSFMGESIGRLWALSWM
NUHSIMYYITCCQPZSICEHBCCMZFEGVJYOCDEMMPGHVAAUM
ELCMOEHVLTIPSUYILVGFLMVWDVYDBTHFRAYISYSGKVSUU
HYHGGCKTMBLRX
```

Answer: This is a *Vigenère Cipher*. The keyword is *THEORY*, and the plaintext is as follows:

I grew up among slow talkers, men in particular, who dropped words a few at a time like beans in a hill, and when I got to Minneapolis where people took a Lake Wobegon comma to mean the end of a story, I couldn't speak a whole sentence in company and was considered not too bright. So I enrolled in a speech course taught by Orville Sand, the founder of reflexive relaxology, a self-hypnotic technique that enabled a person to speak up to three hundred words per minute.

2.22 (a) Suppose that $p_1, \ldots, p_n$ and $q_1, \ldots, q_n$ are both probability distributions, and $p_1 \geq \cdots \geq p_n$. Let $q'_1, \ldots, q'_n$ be any permutation of $q_1, \ldots, q_n$. Prove that the quantity

$$\sum_{i=1}^{n} p_i q'_i$$

is maximized when $q'_1 \geq \cdots \geq q'_n$.

Answer: Suppose that $q'_j < q'_k$ for some $j < k$. Define

$$q''_i = \begin{cases} q'_i & \text{if } i \notin \{j,k\} \\ q'_k & \text{if } i = j \\ q'_j & \text{if } i = k. \end{cases}$$

Then we have

$$\sum_{i=1}^{n} p_i q''_i - \sum_{i=1}^{n} p_i q'_i = (p_j - p_k)(q'_k - q'_j) \geq 0.$$

Therefore the desired sum is not decreased when q'_j and q'_k are exchanged. By a sequence of exchanges of this type, we see that the sum attains its maximum possible value when $q'_1 \geq \cdots \geq q'_n$.

(b) Explain why the expression in Equation (2.1) is likely to be maximized when $g = k_i$.

Answer: (Note: this equation is on page 47.) Suppose that π is a permutation of $\{0, \ldots, 25\}$ such that $p_{\pi(0)} \geq \cdots \geq p_{\pi(25)}$. Then it is "likely" that $f_{\pi(0)} \geq \cdots \geq f_{\pi(25)}$. Assuming that this is the case, we proceed. When $g = 0$, the following equation holds:

$$\sum_{i=0}^{25} \frac{p_i f_i}{n'} = \sum_{i=0}^{25} \frac{p_{\pi(i)} f_{\pi(i)}}{n'}.$$

By the result proven in part (a), this sum is at least as great as any sum

$$\sum_{i=0}^{25} \frac{p_i f_{i+g}}{n'},$$

where $g \neq 0$.

2.23 Suppose we are told that the plaintext

```
breathtaking
```

yields the ciphertext

```
RUPOTENTOIFV
```

where the *Hill Cipher* is used (but m is not specified). Determine the encryption matrix.

Answer: Using the first 9 plaintext and ciphertext characters, we compute

$$K = \begin{pmatrix} 1 & 17 & 4 \\ 0 & 19 & 7 \\ 19 & 0 & 10 \end{pmatrix}^{-1} \begin{pmatrix} 17 & 20 & 15 \\ 14 & 19 & 4 \\ 13 & 19 & 14 \end{pmatrix} = \begin{pmatrix} 3 & 21 & 20 \\ 4 & 15 & 23 \\ 6 & 14 & 5 \end{pmatrix}.$$

If desired, we can check this by verifying that the last 3 plaintext characters encrypt properly:

$$\begin{pmatrix} 8 & 13 & 6 \end{pmatrix} \begin{pmatrix} 3 & 21 & 20 \\ 4 & 15 & 23 \\ 6 & 14 & 5 \end{pmatrix} = \begin{pmatrix} 8 & 5 & 21 \end{pmatrix}.$$

2.24 An *Affine-Hill Cipher* is the following modification of a *Hill Cipher*: Let m be a positive integer, and define $\mathcal{P} = \mathcal{C} = (\mathbb{Z}_{26})^m$. In this cryptosystem, a key K consists of a pair (L, b), where L is an $m \times m$ invertible matrix over $\mathbb{Z}_{26}$, and $b \in (\mathbb{Z}_{26})^m$. For $x = (x_1, \dots, x_m) \in \mathcal{P}$ and $K = (L, b) \in \mathcal{K}$, we compute $y = e_K(x) = (y_1, \dots, y_m)$ by means of the formula $y = xL + b$. Hence, if $L = (\ell_{i,j})$ and $b = (b_1, \dots, b_m)$, then

$$(y_1, \dots, y_m) = (x_1, \dots, x_m) \begin{pmatrix} \ell_{1,1} & \ell_{1,2} & \dots & \ell_{1,m} \\ \ell_{2,1} & \ell_{2,2} & \dots & \ell_{2,m} \\ \vdots & \vdots & & \vdots \\ \ell_{m,1} & \ell_{m,2} & \dots & \ell_{m,m} \end{pmatrix} + (b_1, \dots, b_m).$$

Suppose Oscar has learned that the plaintext

`adisplayedequation`

is encrypted to give the ciphertext

`DSRMSIOPLXLJBZULLM`

and Oscar also knows that $m = 3$. Determine the key, showing all computations.

Answer: We are given the following:

$$\begin{aligned} x_1 &= (0, 3, 8) \\ x_2 &= (18, 15, 11) \\ x_3 &= (0, 24, 4) \\ x_4 &= (3, 4, 16) \\ x_5 &= (20, 0, 9) \\ x_6 &= (8, 14, 13) \end{aligned}$$

and

$$\begin{aligned} y_1 &= (3, 18, 17) \\ y_2 &= (12, 18, 8) \\ y_3 &= (14, 15, 11) \\ y_4 &= (23, 11, 9) \\ y_5 &= (1, 25, 20) \\ y_6 &= (11, 11, 12). \end{aligned}$$

For $1 \leq i \leq 6$, it holds that $y_i = x_i L + b$. Therefore, for $1 \leq i \leq 3$, we have $y_i - y_4 = (x_i - x_4)L$. We form the 3×3 matrix X' having rows $x_i - x_4$ ($1 \leq i \leq 3$) and the 3×3 matrix Y' having rows $y_i - y_4$ ($1 \leq i \leq 3$); then $L = (X')^{-1}Y'$. Once we have found L, we can determine b from the equation $b = y_1 - x_1 L$.

In the given example, we have

$$X' = \begin{pmatrix} 23 & 25 & 18 \\ 15 & 11 & 21 \\ 23 & 20 & 14 \end{pmatrix},$$

$$Y' = \begin{pmatrix} 6 & 7 & 8 \\ 15 & 7 & 25 \\ 17 & 4 & 2 \end{pmatrix},$$

and L can be computed to be

$$L = \begin{pmatrix} 3 & 6 & 4 \\ 5 & 15 & 18 \\ 17 & 8 & 5 \end{pmatrix}.$$

Then

$$b = \begin{pmatrix} 8 & 13 & 1 \end{pmatrix}.$$

If desired, it can be checked that $y_i = x_i L + b$, for $1 \leq i \leq 6$.

2.25 Here is how we might cryptanalyze the *Hill Cipher* using a ciphertext-only attack. Suppose that we know that $m = 2$. Break the ciphertext into blocks of length two letters (digrams). Each such digram is the encryption of a plaintext digram using the unknown encryption matrix. Pick out the most frequent ciphertext digram and assume it is the encryption of a common digram in the list following Table 2.1 (for example, TH or ST). For each such guess, proceed as in the known-plaintext attack, until the correct encryption matrix is found.

Here is a sample of ciphertext for you to decrypt using this method:

```
LMQETXYEAGTXCTUIEWNCTXLZEWUAISPZYVAPEWLMGQWYA
XFTCJMSQCADAGTXLMDXNXSNPJQSYVAPRIQSMHNOCVAXFV
```

Answer: The key is

$$\begin{pmatrix} 4 & 11 \\ 13 & 9 \end{pmatrix}.$$

The plaintext is the following:

> The king was in his counting house, counting out his money. The queen was in the parlour, eating bread and honey.

2.26 We describe a special case of a *Permutation Cipher*. Let m, n be positive integers. Write out the plaintext, by rows, in $m \times n$ rectangles. Then form the ciphertext by taking the columns of these rectangles. For example, if $m = 3$, $n = 4$, then we would encrypt the plaintext "*cryptography*" by forming the following rectangle:

```
cryp
togr
aphy
```

The ciphertext would be "*CTAROPYGHPRY*."

(a) Describe how Bob would decrypt a ciphertext string (given values for m and n).

Answer: Bob can write out the ciphertext string by rows, in $n \times m$ rectangles. The plaintext is formed by taking the columns of these rectangles.

(b) Decrypt the following ciphertext, which was obtained by using this method of encryption:

```
MYAMRARUYIQTENCTORAHROYWDSOYEOUARRGDERNOGW
```

Answer: Here $m = 2$ and $n = 3$. The plaintext is the following:

Mary, Mary, quite contrary, how does your garden grow?

2.27 The purpose of this exercise is to prove the statement made in Section 2.2.5 that the $m \times m$ coefficient matrix is invertible. This is equivalent to saying that the rows of this matrix are linearly independent vectors over $\mathbb{Z}_2$.

Suppose that the recurrence has the form

$$z_{m+i} = \sum_{j=0}^{m-1} c_j z_{i+j} \bmod 2,$$

where $(z_1, \dots, z_m)$ comprises the initialization vector. For $i \geq 1$, define

$$v_i = (z_i, \dots, z_{i+m-1}).$$

Note that the coefficient matrix has the vectors $v_1, \dots, v_m$ as its rows, so our objective is to prove that these m vectors are linearly independent.

Prove the following assertions:

(a) For any $i \geq 1$,

$$v_{m+i} = \sum_{j=0}^{m-1} c_j v_{i+j} \bmod 2.$$

Answer: This is immediate.

(b) Choose h to be the minimum integer such that there exists a non-trivial linear combination of the vectors $v_1, \ldots, v_h$ which sums to the vector $(0, \ldots, 0)$ modulo 2. Then

$$v_h = \sum_{j=0}^{h-2} \alpha_j v_{j+1} \bmod 2,$$

and not all the α_j's are zero. Observe that $h \leq m+1$, since any $m+1$ vectors in an m-dimensional vector space are dependent.

Answer: A dependence relation has the form

$$\sum_{j=0}^{h-1} \alpha_j v_{j+1} \bmod 2 = (0, \ldots, 0),$$

where $\alpha_0, \ldots, \alpha_{h-1} \in \{0,1\}$. Clearly $h \leq m+1$, because any $m+1$ vectors are linearly dependent. Also, we note that $\alpha_{h-1} = 1$ by the minimality of h. Therefore

$$v_h = \sum_{j=0}^{h-2} \alpha_j v_{j+1} \bmod 2.$$

Now, could it be the case that $\alpha_0 = \cdots = \alpha_{h-2} = 0$? If so, then we have $v_h = (0, \ldots, 0)$. But $v_h = (z_h, \ldots, z_{h+m-1})$, so $z_h = \cdots = z_{h+m-1} = 0$. Using the fact that $c_0 = 1$ (as discussed in Section 2.1.7), we can rewrite the recurrence

$$z_{m+i} = \sum_{j=0}^{m-1} c_j z_{i+j} \bmod 2$$

"backwards", as follows:

$$z_i = \sum_{j=1}^{m} c_j z_{i+j} \bmod 2,$$

where we define $c_m = 1$. Then we see that $z_1 = \cdots = z_m = 0$, which generates a keystream consisting entirely of "0"s. We do not allow this case to occur (as discussed on page 36), which proves the desired result.

(c) Prove that the keystream must satisfy the recurrence

$$z_{h-1+i} = \sum_{j=0}^{h-2} \alpha_j z_{j+i} \bmod 2$$

for any $i \geq 1$.

Answer: This is immediate.

(d) If $h \leq m$, then the keystream satisfies a linear recurrence of degree less than m. Show that this is impossible, by considering the initialization vector $(0, \dots, 0, 1)$. Hence, conclude that $h = m + 1$, and therefore the matrix must be invertible.

Answer: Suppose $h \leq m$. Let $i = m - h + 1$; then $i \geq 1$. Then, from part (c), we have

$$z_m = \sum_{j=0}^{h-2} \alpha_j z_{j+m-h+1} \bmod 2 = 0.$$

This contradicts the assumption that $z_m = 1$ and hence $h \geq m + 1$.

2.28 Decrypt the following ciphertext, obtained from the *Autokey Cipher*, by using exhaustive key search:

```
MALVVMAFBHBUQPTSOXALTGVWWRG
```

Answer: The key is 19, and the plaintext is the following:

There is no time like the present.

2.29 We describe a stream cipher that is a modification of the *Vigenère Cipher*. Given a keyword $(K_1, \dots, K_m)$ of length m, construct a keystream by the rule $z_i = K_i$ $(1 \leq i \leq m)$, $z_{i+m} = (z_i + 1) \bmod 26$ $(i \geq 1)$. In other words, each time we use the keyword, we replace each letter by its successor modulo 26. For example, if *SUMMER* is the keyword, we use *SUMMER* to encrypt the first six letters, we use *TVNNFS* for the next six letters, and so on.

(a) Describe how you can use the concept of index of coincidence to first determine the length of the keyword, and then actually find the keyword.

Answer: Suppose we hypothesize that the keyword length is m. Define the following modified ciphertext:

$$y'_j = y_j - \left\lfloor \frac{j-1}{m} \right\rfloor,$$

$j = 1, 2, \dots$. Then the string $y'_1 y'_2 \cdots$ is the encryption of the same plaintext, using the usual *Vigenère Cipher* with the same keyword. Hence the methods used to cryptanalyze the *Vigenère Cipher* can be applied to this modified ciphertext string to determine the keyword length and the actual keyword.

(b) Test your method by cryptanalyzing the following ciphertext:

```
IYMYSILONRFNCQXQJEDSHBUIBCJUZBOLFQYSCHATPEQGQ
JEJNGNXZWHHGWFSUKULJQACZKKJOAAHGKEMTAFGMKVRDO
PXNEHEKZNKFSKIFRQVHHOVXINPHMRTJPYWQGJWPUUVKFP
OAWPMRKKQZWLQDYAZDRMLPBJKJOBWIWPSEPVVQMBCRYVC
RUZAAOUMBCHDAGDIEMSZFZHALIGKEMJJFPCIWKRMLMPIN
AYOFIREAOLDTHITDVRMSE
```

Answer: Tke keyword is *PRIME*. The plaintext is from page 351 of "The Codebreakers", by D. Kahn, Macmillan, 1967.

> The most famous cryptologist in history owes his fame less to what he did than to what he said, and to the sensational way in which he said it, and this was most perfectly in character, for Herbert Osborne Yardley was perhaps the most engaging, articulate, and technicolored personality in the business.

2.30 We describe another stream cipher, which incorporates one of the ideas from the *Enigma* machime used by Germany in World War II. Suppose that π is a fixed permutation of $\mathbb{Z}_{26}$. The key is an element $K \in \mathbb{Z}_{26}$. For all integers $i \geq 1$, the keystream element $z_i \in \mathbb{Z}_{26}$ is defined according to the rule $z_i = (K + i - 1) \bmod 26$. Encryption and decryption are performed using the permutations π and π^{-1}, respectively, as follows:

$$e_z(x) = \pi(x) + z \bmod 26$$

and

$$d_z(y) = \pi^{-1}(y - z \bmod 26),$$

where $z \in \mathbb{Z}_{26}$.

Suppose that π is the following permutation of $\mathbb{Z}_{26}$:

x	0	1	2	3	4	5	6	7	8	9	10	11	12
$\pi(x)$	23	13	24	0	7	15	14	6	25	16	22	1	19

x	13	14	15	16	17	18	19	20	21	22	23	24	25
$\pi(x)$	18	5	11	17	2	21	12	20	4	10	9	3	8

The following ciphertext has been encrypted using this stream cipher; use exhaustive key search to decrypt it:

```
WRTCNRLDSAFARWKXFTXCZRNHNYPDTZUUKMPLUSOXNEUDO
KLXRMCBKGRCCURR
```

Answer: The key is $K = 10$, and the decrypted plaintext is the following:

> The first deposit consisted of one thousand and fourteen pounds of gold.

Chapter 3

Shannon's Theory, Perfect Secrecy, and the One-Time Pad

3.1 Referring to Example 3.1, suppose we define the event

$$T_d\{(i,j) \in Z : |i-j| = d\},$$

for $0 \leq d \leq 5$. (That is, the event T_d corresponds to the situation where the difference of a pair of dice is equal to d.) Compute the probabilities $\mathbf{Pr}[T_d]$, $0 \leq d \leq 5$.

Answer:

$$\begin{aligned}
\mathbf{Pr}[T_0] &= \frac{6}{36} \\
\mathbf{Pr}[T_1] &= \frac{10}{36} \\
\mathbf{Pr}[T_2] &= \frac{8}{36} \\
\mathbf{Pr}[T_3] &= \frac{6}{36} \\
\mathbf{Pr}[T_4] &= \frac{4}{36} \\
\mathbf{Pr}[T_5] &= \frac{2}{36}.
\end{aligned}$$

3.2 Referring to Example 3.2, determine all the joint and conditional probabilities, $\mathbf{Pr}[x,y]$, $\mathbf{Pr}[x|y]$, and $\mathbf{Pr}[y|x]$, where $x \in \{2,\ldots,12\}$ and $y \in \{D,N\}$.

Answer: The probabilities are as follows:

x	y	$\mathbf{Pr}[x\vert y]$	$\mathbf{Pr}[y\vert x]$	$\mathbf{Pr}[x,y]$
2	D	1/6	1	1/36
3	D	0	0	0
4	D	1/6	1/3	1/36
5	D	0	0	0
6	D	1/6	1/5	1/36
7	D	0	0	0
8	D	1/6	1/5	1/36
9	D	0	0	0
10	D	1/6	1/3	1/36
11	D	0	0	0
12	D	1/6	1	1/36
2	N	0	0	0
3	N	2/30	1	2/36
4	N	2/30	2/3	2/36
5	N	4/30	1	4/36
6	N	4/30	4/5	4/36
7	N	6/30	1	6/36
8	N	4/30	4/5	4/36
9	N	4/30	1	4/36
10	N	2/30	2/3	2/36
11	N	2/30	1	2/36
12	N	0	0	0

3.3 Let n be a positive integer. A ***Latin square*** of order n is an $n \times n$ array L of the integers $1, \ldots, n$ such that every one of the n integers occurs exactly once in each row and each column of L. An example of a Latin square of order 3 is as follows:

1	2	3
3	1	2
2	3	1

Given any Latin square L of order n, we can define a related *Latin Square Cryptosystem*. Take $\mathcal{P} = \mathcal{C} = \mathcal{K} = \{1, \ldots, n\}$. For $1 \leq i \leq n$, the encryption rule e_i is defined to be $e_i(j) = L(i, j)$. (Hence each row of L gives rise to one encryption rule.)

Give a complete proof that this *Latin Square Cryptosystem* achieves perfect secrecy provided that every key is used with equal probability.

Answer: For each $x, y \in \{1, \ldots, n\}$, there exists a unique key $K_{x,y}$ such that $e_{K_{x,y}}(x) = y$. Therefore, $C(K) = \{1, \ldots, n\}$ for all $K \in \mathcal{K}$. For any

$y \in \{1, \ldots, n\}$, we have

$$\begin{aligned}\mathbf{Pr}[\mathbf{y} = y] &= \sum_{x \in \{1,\ldots,n\}} \mathbf{Pr}[\mathbf{K} = K_{x,y}]\mathbf{Pr}[\mathbf{x} = x] \\ &= \sum_{x \in \{1,\ldots,n\}} (1/n) \times \mathbf{Pr}[\mathbf{x} = x] \\ &= \frac{1}{n}.\end{aligned}$$

Then, for any $x, y \in \{1, \ldots, n\}$, we compute

$$\mathbf{Pr}[\mathbf{y} = y | \mathbf{x} = x] = \mathbf{Pr}[\mathbf{K} = K_{x,y}] = \frac{1}{n}.$$

Finally, using Bayes' Theorem, we see that

$$\mathbf{Pr}[\mathbf{x} = x | \mathbf{y} = y] = \mathbf{Pr}[\mathbf{x} = x]$$

for all x, y.

3.4 Let $\mathcal{P} = \{a, b\}$ and let $\mathcal{K} = \{K_1, K_2, K_3, K_4, K_5\}$. Let $\mathcal{C} = \{1, 2, 3, 4, 5\}$, and suppose the encryption functions are represented by the following encryption matrix:

	a	b
K_1	1	2
K_2	2	3
K_3	3	1
K_4	4	5
K_5	5	4

Now choose two positive real numbers α and β such that $\alpha + \beta = 1$, and define $\mathbf{Pr}[K_1] = \mathbf{Pr}[K_2] = \mathbf{Pr}[K_3] = \alpha/3$ and $\mathbf{Pr}[K_4] = \mathbf{Pr}[K_5] = \beta/2$.

Prove that this cryptosystem achieves perfect secrecy.

Answer: We prove that $\mathbf{Pr}[y|x] = \mathbf{Pr}[y]$ for $x = a, b$ and $y = 1, \ldots, 5$. First, we compute

$$\begin{aligned} \mathbf{Pr}[1|a] &= \tfrac{\alpha}{3} & \mathbf{Pr}[2|b] &= \tfrac{\alpha}{3} \\ \mathbf{Pr}[2|a] &= \tfrac{\alpha}{3} & \mathbf{Pr}[3|b] &= \tfrac{\alpha}{3} \\ \mathbf{Pr}[3|a] &= \tfrac{\alpha}{3} & \mathbf{Pr}[1|b] &= \tfrac{\alpha}{3} \\ \mathbf{Pr}[4|a] &= \tfrac{\beta}{2} & \mathbf{Pr}[5|b] &= \tfrac{\beta}{2} \\ \mathbf{Pr}[5|a] &= \tfrac{\beta}{2} & \mathbf{Pr}[4|b] &= \tfrac{\beta}{2} \end{aligned}$$

It then follows that

$$\begin{aligned} \mathbf{Pr}[1] &= \mathbf{Pr}[a]\mathbf{Pr}[1|a] + \mathbf{Pr}[b]\mathbf{Pr}[1|b] = (\mathbf{Pr}[a] + \mathbf{Pr}[b])\tfrac{\alpha}{3} = \tfrac{\alpha}{3} \\ \mathbf{Pr}[2] &= \mathbf{Pr}[a]\mathbf{Pr}[2|a] + \mathbf{Pr}[b]\mathbf{Pr}[2|b] = (\mathbf{Pr}[a] + \mathbf{Pr}[b])\tfrac{\alpha}{3} = \tfrac{\alpha}{3} \\ \mathbf{Pr}[3] &= \mathbf{Pr}[a]\mathbf{Pr}[3|a] + \mathbf{Pr}[b]\mathbf{Pr}[3|b] = (\mathbf{Pr}[a] + \mathbf{Pr}[b])\tfrac{\alpha}{3} = \tfrac{\alpha}{3} \\ \mathbf{Pr}[4] &= \mathbf{Pr}[a]\mathbf{Pr}[4|a] + \mathbf{Pr}[b]\mathbf{Pr}[4|b] = (\mathbf{Pr}[a] + \mathbf{Pr}[b])\tfrac{\beta}{2} = \tfrac{\beta}{2} \\ \mathbf{Pr}[5] &= \mathbf{Pr}[a]\mathbf{Pr}[5|a] + \mathbf{Pr}[b]\mathbf{Pr}[5|b] = (\mathbf{Pr}[a] + \mathbf{Pr}[b])\tfrac{\beta}{2} = \tfrac{\beta}{2}. \end{aligned}$$

3.5 (a) Prove that the *Affine Cipher* achieves perfect secrecy if every key is used with equal probability $1/312$.

Answer: For each $x, y \in \mathbb{Z}_{26}$, and for each $a \in {\mathbb{Z}_{26}}^*$, there exists a unique $b(x,y,a) \in \mathbb{Z}_{26}$ such that $e_{(a,b(x,y,a))}(x) = y$. Also, $C(K) = \{1, \dots, n\}$ for all $K \in \mathcal{K}$. For any $y \in \{1, \dots, n\}$, we have

$$\begin{aligned}\mathbf{Pr}[\mathbf{y} = y] &= \sum_{x \in \{1,\dots,n\}} \sum_{a \in {\mathbb{Z}_{26}}^*} \mathbf{Pr}[\mathbf{K} = (a, b(x,y,a))]\mathbf{Pr}[\mathbf{x} = x] \\ &= \sum_{x \in \{1,\dots,n\}} (12/312) \times \mathbf{Pr}[\mathbf{x} = x] \\ &= \frac{1}{26}.\end{aligned}$$

Then, for any $x, y \in \mathbb{Z}_{26}$, we compute

$$\begin{aligned}\mathbf{Pr}[\mathbf{y} = y|\mathbf{x} = x] &= \sum_{a \in {\mathbb{Z}_{26}}^*} \mathbf{Pr}[\mathbf{K} = (a, b(x,y,a))] \\ &= \frac{12}{312} \\ &= \frac{1}{26}.\end{aligned}$$

Finally, using Bayes' Theorem, we see that

$$\mathbf{Pr}[\mathbf{x} = x|\mathbf{y} = y] = \mathbf{Pr}[\mathbf{x} = x]$$

for all x, y.

(b) More generally, suppose we are given a probability distribution on the set

$$\{a \in \mathbb{Z}_{26} : \gcd(a, 26) = 1\}.$$

Suppose that every key (a, b) for the *Affine Cipher* is used with probability $\mathbf{Pr}[a]/26$. Prove that the *Affine Cipher* achieves perfect secrecy when this probability distribution is defined on the keyspace.

Answer: Proceeding as in part (a), for any $y \in \{1, \dots, n\}$, we have

$$\begin{aligned}\mathbf{Pr}[\mathbf{y} = y] &= \sum_{x \in \{1,\dots,n\}} \sum_{a \in {\mathbb{Z}_{26}}^*} \mathbf{Pr}[\mathbf{K} = (a, b(x,y,a))]\mathbf{Pr}[\mathbf{x} = x] \\ &= \sum_{x \in \{1,\dots,n\}} \sum_{a \in {\mathbb{Z}_{26}}^*} (\mathbf{Pr}[a]/26) \times \mathbf{Pr}[\mathbf{x} = x] \\ &= \sum_{x \in \{1,\dots,n\}} (1/26) \times \mathbf{Pr}[\mathbf{x} = x] \\ &= \frac{1}{26}.\end{aligned}$$

Then, for any $x, y \in \mathbb{Z}_{26}$, we compute

$$\begin{aligned}\mathbf{Pr}[\mathbf{y} = y|\mathbf{x} = x] &= \sum_{a \in \mathbb{Z}_{26}{}^*} \mathbf{Pr}[\mathbf{K} = (a, b(x,y,a))] \\ &= \sum_{a \in \mathbb{Z}_{26}{}^*} \frac{\mathbf{Pr}[a]}{26} \\ &= \frac{1}{26}.\end{aligned}$$

Finally, using Bayes' Theorem, we see that

$$\mathbf{Pr}[\mathbf{x} = x|\mathbf{y} = y] = \mathbf{Pr}[\mathbf{x} = x]$$

for all x, y.

3.6 Suppose a cryptosystem achieves perfect secrecy for a particular plaintext probability distribution. Prove that perfect secrecy is maintained for any plaintext probability distribution.

Answer: Let $\mathsf{p} = p_{x_1}, \dots, p_{x_n}$ be a probability distribution on the plaintext space $\mathcal{P} = \{x_1, \dots, x_n\}$, and suppose that the cryptosystem achieves perfect secrecy when the plaintext is chosen using this plaintext probability distribution. Let $\mathsf{q} = q_{x_1}, \dots, q_{x_n}$ be an arbitrary probability distribution on $\mathcal{P}$. It should be clear that $\mathbf{Pr}[\mathbf{y} = y|\mathbf{x} = x]$ does not depend on the plaintext probability distribution.

Because the perfect secrecy property holds with respect to p, we have that

$$\mathbf{Pr}_{\mathsf{p}}[\mathbf{y} = y] = \mathbf{Pr}[\mathbf{y} = y|\mathbf{x} = x]$$

for all $x \in \mathcal{P}$, $y \in \mathcal{Y}$. Therefore it holds that

$$\sum_{\{K : y \in C(K)\}} (\mathbf{Pr}[\mathbf{K} = K] \times p_{d_K(y)}) = \sum_{\{K : x = d_K(y)\}} \mathbf{Pr}[\mathbf{K} = K]$$

for all $x \in \mathcal{P}$, $y \in \mathcal{Y}$. Now, we compute $\mathbf{Pr}_{\mathsf{q}}[\mathbf{y} = y]$:

$$\begin{aligned}\mathbf{Pr}_{\mathsf{q}}[\mathbf{y} = y] &= \sum_{\{K : y \in C(K)\}} (\mathbf{Pr}[\mathbf{K} = K] \times q_{d_K(y)}) \\ &= \sum_{x_i \in \mathcal{P}} q_{x_i} \sum_{\{K : x_i = d_K(y)\}} \mathbf{Pr}[\mathbf{K} = K] \\ &= \sum_{x_i \in \mathcal{P}} q_{x_i} \sum_{\{K : y \in C(K)\}} (\mathbf{Pr}[\mathbf{K} = K] \times p_{d_K(y)}) \\ &= \left(\sum_{x_i \in \mathcal{P}} q_{x_i}\right) \times \left(\sum_{\{K : y \in C(K)\}} (\mathbf{Pr}[\mathbf{K} = K] \times p_{d_K(y)})\right) \\ &= \sum_{\{K : y \in C(K)\}} (\mathbf{Pr}[\mathbf{K} = K] \times p_{d_K(y)}) \\ &= \mathbf{Pr}[\mathbf{y} = y|\mathbf{x} = x],\end{aligned}$$

as desired.

3.7 Prove that if a cryptosystem has perfect secrecy and $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{P}|$, then every ciphertext is equally probable.

Answer: This follows from the proof of Theorem 3.4.

3.8 Suppose that y and y' are two ciphertext elements (i.e., binary n-tuples) in the *One-time Pad* that were obtained by encrypting plaintext elements x and x', respectively, using the same key, K. Prove that $x + x' \equiv y + y' \pmod 2$.

Answer: We have $y = x + K \bmod 2$ and $y' = x' + K \bmod 2$. Adding, we see that

$$y + y' = x + K + x' + K = x + x' \bmod 2.$$

3.9 (a) Construct the encryption matrix (as defined in Example 3.3) for the *One-time Pad* with $n = 3$.

Answer:

K	$x =$ 000	001	010	011	100	101	110	111
000	000	001	010	011	100	101	110	111
001	001	000	011	010	101	100	111	110
010	010	011	000	001	110	111	100	101
011	011	010	001	000	111	110	101	100
100	100	101	110	111	000	001	010	011
101	101	100	111	110	001	000	011	010
110	110	111	100	101	010	011	000	001
111	111	110	101	100	011	010	001	000

(b) For any positive integer n, give a direct proof that the encryption matrix of a *One-time Pad* defined over $(\mathbb{Z}_2)^n$ is a Latin square of order 2^n, in which the symbols are the elements of $(\mathbb{Z}_2)^n$.

Answer: Suppose that $x, y, K \in (\mathbb{Z}_2)^n$. We have that $e_K(x) = y$ if and only if $x + K = y$ (in $(\mathbb{Z}_2)^n$). Given K and y, we can solve for x uniquely: $x = y + K$. Therefore every row of the encryption matrix contains every symbol in exactly one cell. Given x and y, we can solve for K uniquely: $K = x + y$. Therefore every column of the encryption matrix contains every symbol in exactly one cell.

3.10 Suppose that $\mathbf{S}$ is a random variable representing the sum of a pair of dice (see Example 3.1). Compute $H(\mathbf{S})$.

Answer:

$$\begin{aligned} H(\mathbf{S}) &= -2\left(\frac{1}{36}\log_2\frac{1}{36} + \frac{1}{18}\log_2\frac{1}{18} + \frac{1}{12}\log_2\frac{1}{12} + \frac{1}{9}\log_2\frac{1}{9} + \frac{5}{36}\log_2\frac{5}{36}\right) \\ &\quad -\frac{1}{6}\log_2\frac{1}{6} \\ &\approx 3.274401919. \end{aligned}$$

3.11 Prove from first principles (i.e., using the definition) that the function $f(x) = x^2$ is concave over the interval $(-\infty, \infty)$.

Note: This question is incorrect. The function $f(x) = x^2$ is a *convex function*, which means that

$$f\left(\frac{x+y}{2}\right) \leq \frac{f(x)+f(y)}{2}.$$

Equivalently, the function $f(x) = -x^2$ is concave.

Answer: We prove that $f(x)$ is convex as follows. We have

$$f\left(\frac{x+y}{2}\right) = \frac{(x+y)^2}{4} = \frac{x^2+2xy+y^2}{4}$$

and

$$\frac{f(x)+f(y)}{2} = \frac{x^2+y^2}{2}.$$

Therefore,

$$\begin{aligned}
\frac{f(x)+f(y)}{2} - f\left(\frac{x+y}{2}\right) &= \frac{x^2+y^2}{2} - \frac{x^2+2xy+y^2}{4}\\
&= \frac{x^2-2xy+y^2}{4}\\
&= \frac{(x-y)^2}{4}\\
&\geq 0.
\end{aligned}$$

3.12 Prove that $H(\mathbf{X},\mathbf{Y}) = H(\mathbf{Y}) + H(\mathbf{X}|\mathbf{Y})$. Then show as a corollary that $H(\mathbf{X}|\mathbf{Y}) \leq H(\mathbf{X})$, with equality if and only if $\mathbf{X}$ and $\mathbf{Y}$ are independent.

Answer: First, we observe that

$$\mathbf{Pr}[y]\mathbf{Pr}[x|y] = \mathbf{Pr}[x,y]$$

and

$$\log_2(\mathbf{Pr}[x|y]) = \log_2(\mathbf{Pr}[x,y]/\mathbf{Pr}[y]) = \log_2 \mathbf{Pr}[x,y] - \log_2 \mathbf{Pr}[y].$$

Therefore

$$\begin{aligned}
H(\mathbf{X}|\mathbf{Y}) &= -\sum_y \sum_x \mathbf{Pr}[y]\mathbf{Pr}[x|y] \log_2 \mathbf{Pr}[x|y]\\
&= -\sum_y \sum_x \mathbf{Pr}[x,y](\log_2 \mathbf{Pr}[x,y] - \log_2 \mathbf{Pr}[y])\\
&= H(\mathbf{X},\mathbf{Y}) + \sum_y \sum_x \mathbf{Pr}[x,y] \log_2 \mathbf{Pr}[y]\\
&= H(\mathbf{X},\mathbf{Y}) + \sum_y \log_2 \mathbf{Pr}[y] \left(\sum_x \mathbf{Pr}[x,y]\right)\\
&= H(\mathbf{X},\mathbf{Y}) + \sum_y (\log_2 \mathbf{Pr}[y] \times \mathbf{Pr}[y])\\
&= H(\mathbf{X},\mathbf{Y}) - H(\mathbf{Y}),
\end{aligned}$$

as desired.

Theorem 3.7 says that $H(\mathbf{X},\mathbf{Y}) \leq H(\mathbf{X}) + H(\mathbf{Y})$, with equality if and only if $\mathbf{X}$ and $\mathbf{Y}$ are independent. Therefore we have

$$\begin{aligned} H(\mathbf{X}) + H(\mathbf{Y}) &\geq H(\mathbf{X},\mathbf{Y}) \\ &= H(\mathbf{Y}) + H(\mathbf{X}|\mathbf{Y}), \end{aligned}$$

which implies that $H(\mathbf{X}) \geq H(\mathbf{X}|\mathbf{Y})$. Further, equality occurs if and only if $\mathbf{X}$ and $\mathbf{Y}$ are independent.

3.13 Prove that a cryptosystem has perfect secrecy if and only if $H(\mathbf{P}|\mathbf{C}) = H(\mathbf{P})$.
Answer: From Exercise 3.12, we have that $H(\mathbf{P}|\mathbf{C}) = H(\mathbf{P})$ if and only if $\mathbf{P}$ and $\mathbf{C}$ are independent. This is true if and only if $\mathbf{Pr}[x,y] = \mathbf{Pr}[x]\mathbf{Pr}[y]$ for all $x \in \mathcal{P}$ and all $y \in \mathcal{C}$. Writing $\mathbf{Pr}[x,y] = \mathbf{Pr}[x|y]\mathbf{Pr}[y]$, the condition becomes $\mathbf{Pr}[x|y]\mathbf{Pr}[y] = \mathbf{Pr}[x]\mathbf{Pr}[y]$, which simplifies to $\mathbf{Pr}[x|y] = \mathbf{Pr}[x]$. This is precisely the perfect secrecy condition.

3.14 Prove that, in any cryptosystem, $H(\mathbf{K}|\mathbf{C}) \geq H(\mathbf{P}|\mathbf{C})$. (Intuitively, this result says that, given a ciphertext, the opponent's uncertainty about the key is at least as great as his uncertainty about the plaintext.)

Answer: Theorem 3.10 says that

$$H(\mathbf{K}|\mathbf{C}) = H(\mathbf{K}) + H(\mathbf{P}) - H(\mathbf{C}).$$

Then we compute a bound on $H(\mathbf{P}|\mathbf{C})$ as follows:

$$\begin{aligned} H(\mathbf{P}|\mathbf{C}) &= H(\mathbf{P},\mathbf{C}) - H(\mathbf{C}) \\ &= H(\mathbf{K},\mathbf{P},\mathbf{C}) - H(\mathbf{K}|\mathbf{P},\mathbf{C}) - H(\mathbf{C}) \\ &\leq H(\mathbf{K},\mathbf{P},\mathbf{C}) - H(\mathbf{C}) \\ &= H(\mathbf{K},\mathbf{P}) - H(\mathbf{C}) \\ &\leq H(\mathbf{K}) + H(\mathbf{P}) - H(\mathbf{C}) \\ &= H(\mathbf{K}|\mathbf{C}). \end{aligned}$$

3.15 Consider a cryptosystem in which $\mathcal{P} = \{a,b,c\}$, $\mathcal{K} = \{K_1,K_2,K_3\}$ and $\mathcal{C} = \{1,2,3,4\}$. Suppose the encryption matrix is as follows:

	a	b	c
K_1	1	2	3
K_2	2	3	4
K_3	3	4	1

Given that keys are chosen equiprobably, and the plaintext probability distribution is $\mathbf{Pr}[a] = 1/2$, $\mathbf{Pr}[b] = 1/3$, $\mathbf{Pr}[c] = 1/6$, compute $H(\mathbf{P})$, $H(\mathbf{C})$, $H(\mathbf{K})$, $H(\mathbf{K}|\mathbf{C})$, and $H(\mathbf{P}|\mathbf{C})$.

Answer: From the given probability distributions on $\mathcal{K}$ and $\mathcal{P}$, we have

$H(\mathbf{K}) = 1.585$ and $H(\mathbf{P}) = 1.459$. We next compute the probability distribution on $\mathcal{C}$ to be $\mathbf{Pr}[1] = 4/18$, $\mathbf{Pr}[2] = 5/18$, $\mathbf{Pr}[3] = 6/18$ and $\mathbf{Pr}[4] = 3/18$. Then $H(\mathbf{C}) = 1.955$. Next, we compute

$$H(\mathbf{K}|\mathbf{C}) = H(\mathbf{K}) + H(\mathbf{P}) - H(\mathbf{C}) = 1.089.$$

In order to compute $H(\mathbf{P}|\mathbf{C})$, we first compute $\mathbf{Pr}[x|y]$ for all $x \in \mathcal{P}$ and all $y \in \mathcal{C}$:

	a	b	c
1	3/4	0	1/4
2	3/5	2/5	0
3	1/2	1/3	1/6
4	0	2/3	1/3

From this, we compute $H(\mathbf{P}|1) = .8113$, $H(\mathbf{P}|2) = .9710$, $H(\mathbf{P}|3) = 1.459$ and $H(\mathbf{P}|4) = .9183$. Finally,

$$H(\mathbf{P}|\mathbf{C}) = \left(\frac{4}{18}, \frac{5}{18}, \frac{6}{18}, \frac{3}{18}\right) \cdot (.8113, .9710, 1.459, 9183) = 1.062.$$

3.16 Compute $H(\mathbf{K}|\mathbf{C})$ and $H(\mathbf{K}|\mathbf{P},\mathbf{C})$ for the *Affine Cipher*, assuming that keys are used equiprobably and the plaintexts are equiprobable.

Answer: $H(\mathbf{K}|\mathbf{C}) = \log_2 312$ and $H(\mathbf{K}|\mathbf{P},\mathbf{C}) = \log_2 12$.

3.17 Suppose that *APNDJI* or *XYGROBO* are ciphertexts that are obtained from encryption using the *Shift Cipher*. Show in each case that there are two "meaningful" plaintexts that could encrypt to the given ciphertext. (Thanks to John van Rees for these examples.)

Answer: *APNDJI* could decrypt to *FUSION* or *LAYOUT*. *XYGROBO* could decrypt to *NOWHERE* or *ABJURER*.

3.18 Consider a *Vigenère Cipher* with keyword length m. Show that the unicity distance is $1/R_L$, where R_L is the redundancy of the underlying language. (This result is interpreted as follows. If n_0 denotes the number of alphabetic characters being encrypted, then the "length" of the plaintext is n_0/m, since each plaintext element consists of m alphabetic characters. So, a unicity distance of $1/R_L$ corresponds to a plaintext consisting of m/R_L alphabetic characters.)

Answer: In the *Vigenère Cipher*, we have $|\mathcal{P}| = |\mathcal{K}| = 26^m$, so the estimate for the unicity distance is

$$\frac{\log_2 |\mathcal{K}|}{R_L \log_2 |\mathcal{P}|} = \frac{1}{R_L}.$$

3.19 Show that the unicity distance of the *Hill Cipher* (with an $m \times m$ encryption matrix) is less than m/R_L. (Note that the number of alphabetic characters in a plaintext of this length is m^2/R_L.)

Answer: The number of $m \times m$ matrices with entries from $\mathbb{Z}_{26}$ is 26^{m^2}, but not all of these matrices are invertible. Therefore $|\mathcal{K}| < 26^{m^2}$. Also, $|\mathcal{P}| = 26^m$. The estimate for the unicity distance is

$$\frac{\log_2 |\mathcal{K}|}{R_L \log_2 |\mathcal{P}|} < \frac{m^2(\log_2 26)}{m(\log_2 26)R_L} = \frac{m}{R_L}.$$

3.20 A *Substitution Cipher* over a plaintext space of size n has $|\mathcal{K}| = n!$ ***Stirling's formula*** gives the following estimate for $n!$:

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n.$$

(a) Using Stirling's formula, derive an estimate of the unicity distance of the *Substitution Cipher*.

Answer: We have that

$$\log_2 |\mathcal{K}| \approx \log_2\left(\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\right) = n(\log_2 n - c_1) + 0.5\log_2 n + c_2,$$

where c_1, c_2 are small positive constants. $\log_2 |\mathcal{P}| = \log_s n$, so an estimate for the unicity distance is

$$\frac{1}{r_L}\left(n - \frac{c_1 n}{\log_2 n} + \frac{1}{2}\right) < \frac{n}{r_L}.$$

(b) Let $m \geq 1$ be an integer. The *m-gram Substitution Cipher* is the *Substitution Cipher* where the plaintext (and ciphertext) spaces consist of all 26^m m-grams. Estimate the unicity distance of the *m-gram Substitution Cipher* if $R_L = 0.75$.

Answer: To simplify things, we will use the estimate $\log_2(n!) \approx n\log_2 n$. Setting $n = 26^m$, we get

$$\log_2 |\mathcal{K}| \approx 26^m \log_2(26^m) \approx (\log_2 26)m26^m.$$

$\log_2 |\mathcal{P}| = (\log_2 26)m$, so the estimate for the unicity distance is

$$\frac{(\log_2 26)m26^m}{r_L(\log_2 26)m} \approx 1.33 \times 26^m.$$

Chapter 4

Block Ciphers and Stream Ciphers

4.1 Let y be the output of Algorithm 4.1 on input x, where π_S and π_P are defined as in Example 4.1. In other words,

$$y = \text{SPN}\left(x, \pi_S, \pi_P, (K^1, \ldots, K^{\mathcal{N}+1})\right),$$

where $(K^1, \ldots, K^{\mathcal{N}+1})$ is the key schedule. Find a substitution π_{S^*} and a permutation π_{P^*} such that

$$x = \text{SPN}\left(y, \pi_{S^*}, \pi_{P^*}, (L^{\mathcal{N}+1}, \ldots, L^1)\right),$$

where each L^i is a permutation of K^i.

Answer: The decryption algorithm is as follows:

$$x = \text{SPN}\left(y, (\pi_S)^{-1}, (\pi_P)^{-1}, ((\pi_P)^{-1}(K^{\mathcal{N}+1}), \ldots, (\pi_P)^{-1}(K^1))\right).$$

4.2 Prove that decryption in a Feistel cipher can be done by applying the encryption algorithm to the ciphertext, with the key schedule reversed.

Answer: We illustrate the idea with *DES*, which also includes an initial permutation. A similar analysis applies to any Feistel cipher. *DES* encryption proceeds as follows:

$$\begin{aligned}
&L^0R^0 = \mathsf{IP}(x)\\
&L^1 = R^0\\
&R^1 = L^0 \oplus f(R^0, K^1)\\
&L^2 = R^1\\
&R^2 = L^1 \oplus f(R^1, K^2)\\
&\vdots\\
&L^{15} = R^{14}\\
&R^{15} = L^{14} \oplus f(R^{14}, K^{15})\\
&L^{16} = R^{15}\\
&R^{16} = L^{15} \oplus f(R^{15}, K^{16})\\
&y = \mathsf{IP}^{-1}(R^{16}L^{16})
\end{aligned}$$

Now, we proceed to decrypt the ciphertext y in a step-by-step fashion. We use prime markings ($'$) to denote the left and right halves of the partially decrypted ciphertext:

$$\begin{aligned}
&(L')^0(R')^0 = \mathsf{IP}(y) = R^{16}L^{16}\\
&(L')^1 = (R')^0 = L^{16} = R^{15}\\
&(R')^1 = (L')^0 \oplus f((R')^0, K^{16}) = R^{16} \oplus f(R^{15}, K^{16}) = L^{15}\\
&(L')^2 = (R')^1 = L^{15} = R^{14}\\
&(R')^2 = (L')^1 \oplus f((R')^1, K^{15}) = R^{15} \oplus f(R^{14}, K^{15}) = L^{14}\\
&\vdots\\
&(L')^{15} = (R')^{14} = L^2 = R^1\\
&(R')^{15} = (L')^{14} \oplus f((R')^{14}, K^2) = R^2 \oplus f(R^1, K^2) = L^1\\
&(L')^{16} = (R')^{15} = L^1 = R^0\\
&(R')^{16} = (L')^{15} \oplus f((R')^{15}, K^1) = R^1 \oplus f(R^0, K^1) = L^0\\
&y = \mathsf{IP}^{-1}((R')^{16}(L')^{16}) = \mathsf{IP}^{-1}(L^0R^0) = x.
\end{aligned}$$

In general, we have $(L')^j = R^{16-j}$ and $(R')^j = L^{16-j}$ for $0 \leq j \leq 16$. This can be proven formally by induction, if desired.

4.3 Let $\mathit{DES}(x, K)$ represent the encryption of plaintext x with key K using the *DES* cryptosystem. Suppose $y = \mathit{DES}(x, K)$ and $y' = \mathit{DES}(c(x), c(K))$, where $c(\cdot)$ denotes the bitwise complement of its argument. Prove that $y' = c(y)$ (i.e., if we complement the plaintext and the key, then the ciphertext is also complemented). Note that this can be proved using only the "high-level" description of *DES*—the actual structure of S-boxes and other components of the system are irrelevant.

Answer: The key fact is that $f(c(A), c(J)) = f(A, J)$, which is easily seen from the description of f. Then, as usual, let the partial encryptions of $\mathit{DES}(x, K)$ be denoted L^jR^j, $0 \leq j \leq 16$. Then it is easy to see that the partial encryptions of $\mathit{DES}(c(x), c(K))$ are $c(L^j)c(R^j)$, $0 \leq j \leq 16$. This can be proven formally by induction, if desired.

4.4 Suppose that we have the following 128-bit *AES* key, given in hexadecimal notation:

```
2B7E151628AED2A6ABF7158809CF4F3C
```

Construct the complete key schedule arising from this key.

Answer: This example is worked out in detail, starting on page 27 of the official FIPS 197 description, which can be found at the following web page:

```
csrc.nist.gov/publications/fips/fips197/fips-197.pdf
```

4.5 Compute the encryption of the following plaintext (given in hexadecimal notation) using the 10-round *AES*:

```
3243F6A8885A308D313198A2E0370734
```

Use the 128-bit key from the previous exercise.

Answer: This example is worked out in detail, starting on page 33 of the official FIPS 197 description, which can be found at the following web page:

`csrc.nist.gov/publications/fips/fips197/fips-197.pdf`

4.6 Prove that decryption in CBC mode or CFB mode can be parallelized efficiently. More precisely, suppose we have n ciphertext blocks and n processors. Show that it is possible to decrypt all n ciphertext blocks in constant time.

Answer: Suppose we are given n ciphertext blocks, say $y_1, \ldots, y_n$, and we have n processors, say $P_1, \ldots, P_n$.

For CBC mode, in the first step, each P_i decrypts y_i, obtaining z_i. In the second step, each P_i computes $x_i = z_i \oplus y_{i-1}$.

For CFB mode, in the first step, each P_i encrypts y_{i-1}, obtaining z_i. In the second step, each P_i computes $x_i = z_i \oplus y_i$.

4.7 Describe in detail how both encryption and decryption in CTR mode can be parallelized efficiently.

Answer: Suppose we are given n plaintexts, say $x_1, \ldots, x_n$, and we have n processors, say $P_1, \ldots, P_n$. Each P_i computes T_i, encrypts it, and then xors the result with x_i.

For decryption, suppose we are given n ciphertexts, say $y_1, \ldots, y_n$. Each P_i computes T_i, encrypts it, and then xors the result with y_i.

4.8 Suppose that $X = (x_1, \ldots, x_n)$ and $X' = (x'_1, \ldots, x'_n)$ are two sequences of n plaintext blocks. Define

$$\mathbf{same}(X, X') = \max\{j : x_i = x'_i \text{ for all } i \leq j\}.$$

Suppose X and X' are encrypted in CBC or CFB mode using the same key and the same IV. Show that it is easy for an adversary to compute $\mathbf{same}(X, X')$.

Answer: The same approach works for both CBC mode and CFB mode. Let $Y = (y_1, \ldots, y_n)$ and $Y' = (y'_1, \ldots, y'_n)$ be the ciphertexts corresponding to X and X', respectively. Suppose that $x_i = x'_i$ for $1 \leq i \leq j$, and suppose $x_{i+1} \neq x'_{i+1}$ Then it follows that $y_i = y'_i$ for $1 \leq i \leq j$ and $y_{i+1} \neq y'_{i+1}$. Hence, the adversary can compute

$$\mathbf{same}(X, X') = \max\{j : y_i = y'_i \text{ for all } i \leq j\}.$$

This computation only requires knowledge of the ciphertext blocks.

4.9 Suppose that $X = (x_1, \ldots, x_n)$ and $X' = (x'_1, \ldots, x'_n)$ are two sequences of n plaintext blocks. Let Suppose X and X' are encrypted in OFB mode using the same key and the same IV. Show that it is easy for an adversary to compute $X \oplus X'$. Show that a similar result holds for CTR mode if *ctr* is reused.

Answer: Let Y and Y' be the encryptions of X and X', respectively. In both scenarios, the two encryptions are performed using identical keystreams.

Since plaintexts are xor-ed with the keysream to construct the ciphertext, it is immediate that $Y \oplus Y' = X \oplus X'$. Thus the computation of $X \oplus X'$ only requires knowledge of the ciphertext blocks.

4.10 Suppose a sequence of plaintext blocks, $x_1 \ldots x_n$, yields the ciphertext sequence $y_1 \ldots y_n$. Suppose that one ciphertext block, say y_i, is transmitted incorrectly (i.e., some 1's are changed to 0's and vice versa). Show that the number of plaintext blocks that will be decrypted incorrectly is equal to one if ECB or OFB modes are used for encryption; and equal to two if CBC or CFB modes are used.

Answer: It is immediate that there is only one incorrectly decrypted ciphertext block when ECB or OFB modes are used for encryption.

Suppose that CBC mode is used, and the ciphertext block y_i is transmitted incorrectly as y_i^*. $x_1, \ldots, x_{i-1}$ are decrypted correctly. The next two ciphertext blocks are decrypted incorrectly:

$$\begin{aligned} x_i^* &= d_K(y_i^*) \oplus y_{i-1} \quad \text{and} \\ x_{i+1}^* &= d_K(y_{i+1}) \oplus y_i^*. \end{aligned}$$

Then all subsequent ciphertext blocks are decrypted correctly.

Suppose that CFB mode is used, and the ciphertext block y_i is transmitted incorrectly as y_i^*. $x_1, \ldots, x_{i-1}$ are decrypted correctly. The next two ciphertext blocks are decrypted incorrectly:

$$\begin{aligned} x_i^* &= e_K(y_{i-1}) \oplus y_i^* \quad \text{and} \\ x_{i+1}^* &= e_K(y_i^*) \oplus y_{i+1}. \end{aligned}$$

Then all subsequent ciphertext blocks are decrypted correctly.

4.11 The purpose of this question is to investigate a time-memory trade-off for a chosen plaintext attack on a certain type of cipher. Suppose we have a cryptosystem in which $\mathcal{P} = \mathcal{C} = \mathcal{K}$, which attains perfect secrecy. Then it must be the case that $e_K(x) = e_{K_1}(x)$ implies $K = K_1$. Denote $\mathcal{P} = Y = \{y_1, \ldots, y_N\}$. Let x be a fixed plaintext. Define the function $g : Y \to Y$ by the rule $g(y) = e_y(x)$. Define a directed graph G having vertex set Y, in which the edge set consists of all the directed edges of the form $(y_i, g(y_i))$, $1 \leq i \leq N$.

(a) Prove that G consists of the union of disjoint directed cycles.

Answer: $g(y) = g(y')$ implies $e_y(x) = e_{y'}(x)$, which implies $y = y'$ (as remarked above). Therefore g is a permutation of the set Y, and its representation as a directed graph is a union of disjoint directed cycles.

(b) Let T be a desired time parameter. Suppose we have a set of elements $Z = \{z_1, \ldots, z_m\} \subseteq Y$ such that, for every element $y_i \in Y$, either y_i is contained in a cycle of length at most T, or there exists an element

$z_j \neq y_i$ such that the distance from y_i to z_j (in G) is at most T. Prove that there exists such a set Z such that

$$|Z| \leq \frac{2N}{T},$$

so $|Z|$ is $O(N/T)$.

Answer: Let the cycles in T be denoted $C_1, C_2, \ldots, C_r$. Note that $\sum |C_i| = N$. It is easy to construct a set Z, satisfying the desired properties, such that every cycle C_i contains exactly $\left\lceil \frac{|C_i|}{T} \right\rceil$ points of Z. It can be verified that $\lceil x \rceil \leq 2x$ for all $x \geq 1$. Hence we have that

$$\sum_{i=1}^{r} \left\lceil \frac{|C_i|}{T} \right\rceil \leq \sum_{i=1}^{r} \frac{2|C_i|}{T} = \frac{2N}{T}.$$

(c) For each $z_j \in Z$, define $g^{-T}(z_j)$ to be the element y_i such that $g^T(y_i) = z_j$, where g^T is the function that consists of T iterations of g. Construct a table X consisting of the ordered pairs $(z_j, g^{-T}(z_j))$, sorted with respect to their first coordinates.

A pseudo-code description of an algorithm to find K, given $y = e_K(x)$, is presented.

```
Algorithm 4.7: TIME-MEMORY TRADE-OFF($y$)

$y_0 \leftarrow y$
$backup \leftarrow$ false
while $g(y) \neq y_0$
  do if $y = z_j$ for some $j$ and not $backup$
       then $y \leftarrow g^{-T}(z_j)$
            $backup \leftarrow$ true
       else $y \leftarrow g(y)$
            $K \leftarrow y$
```

Prove that this algorithm finds K in at most T steps. (Hence the time-memory trade-off is $O(N)$.)

Answer: The algorithm requires at most T iterations of the while loop to find $y = z_j$, and then at most T further iterations until $g(y) = y_0$. Therefore the total number of iterations is $O(T)$. Each iteration requires time $O(1) + O(\log N)$ (assuming we do a binary search of the z_j's), so the total time is $O(T \log N)$. The memory requirement is $O(N \log N/T)$ bits. Therefore the product of time and memory is $O(N(\log N)^2)$. If we ignore the logarithmic factor (as is usually done in analyses of this type), the product is $O(N)$.

(d) Describe a pseudo-code algorithm to construct the desired set Z in time $O(NT)$ without using an array of size N.

Answer: We construct Z, as well as the set X of ordered pairs of the form $(z, g^{-T}(z))$, as shown in the following algorithm.

```
Algorithm: CONSTRUCTXANDZ ($x$)

$Z \leftarrow \emptyset$
for $i \leftarrow 1$ to $N$
  do
    $newcycle \leftarrow true$
    $y_0 \leftarrow y_i$
    $y \leftarrow y_i$
    for $j \leftarrow 1$ to $T$
      do
        $y_0 \leftarrow g(y_0)$
        if $y_0 \in Z$
          then $newcycle \leftarrow false$
    if $newcycle$
      then
        $endcycle \leftarrow false$
        while not $endcycle$
          do
            $Z \leftarrow Z \cup \{y_0\}$
            $X \leftarrow X \cup \{(y_0, y)\}$
            $y \leftarrow y_0$
            for $j \leftarrow 1$ to $T$
              do
                $y_0 \leftarrow g(y_0)$
                if $y_0 \in Z$
                  then $endcycle \leftarrow true$
```

4.12 Suppose that $\mathbf{X_1}, \mathbf{X_2}$, and $\mathbf{X_3}$ are independent discrete random variables defined on the set $\{0, 1\}$. Let ϵ_i denote the bias of $\mathbf{X_i}$, for $i = 1, 2, 3$. Prove that $\mathbf{X_1} \oplus \mathbf{X_2}$ and $\mathbf{X_2} \oplus \mathbf{X_3}$ are independent if and only if $\epsilon_1 = 0$, $\epsilon_3 = 0$, or $\epsilon_2 = \pm 1/2$.

Answer: $\mathbf{X_1} \oplus \mathbf{X_2}$ has bias $2\epsilon_1\epsilon_2$ and $\mathbf{X_2} \oplus \mathbf{X_3}$ has bias $2\epsilon_2\epsilon_3$. Suppose that $\mathbf{X_1} \oplus \mathbf{X_2}$ and $\mathbf{X_2} \oplus \mathbf{X_3}$ are independent. Then the bias of $(\mathbf{X_1} \oplus \mathbf{X_2}) \oplus (\mathbf{X_2} \oplus \mathbf{X_3})$ would be $2(2\epsilon_1\epsilon_2)(2\epsilon_2\epsilon_3)$. However,

$$(\mathbf{X_1} \oplus \mathbf{X_2}) \oplus (\mathbf{X_2} \oplus \mathbf{X_3}) = \mathbf{X_1} \oplus \mathbf{X_3}$$

has bias $2\epsilon_1\epsilon_3$. Therefore

$$4\epsilon_1(\epsilon_2)^2\epsilon_3 = \epsilon_1\epsilon_3.$$

This implies that $\epsilon_1 = 0$, $\epsilon_3 = 0$ or $\epsilon_2 = \pm 1/2$.

Conversely, suppose that $\epsilon_1 = 0$, $\epsilon_3 = 0$ or $\epsilon_2 = \pm 1/2$. The two random variables $\mathbf{X_1} \oplus \mathbf{X_2}$ and $\mathbf{X_2} \oplus \mathbf{X_2}$ are independent if and only if

$$\mathbf{Pr}[(\mathbf{X_1} \oplus \mathbf{X_2} = a) \text{ and } (\mathbf{X_2} \oplus \mathbf{X_3} = b)] = \mathbf{Pr}[\mathbf{X_1} \oplus \mathbf{X_2} = a] \times \mathbf{Pr}[\mathbf{X_2} \oplus \mathbf{X_3} = b]$$

for $a, b \in \{0,1\}$. These four conditions are as follows:

$$\begin{aligned}
&p_1p_2p_3 + (1-p_1)(1-p_2)(1-p_3)\\
&\quad = \quad (p_1p_2 + (1-p_1)(1-p_2))(p_2p_3 + (1-p_2)(1-p_3)),\\
&p_1p_2(1-p_3) + (1-p_1)(1-p_2)p_3\\
&\quad = \quad (p_1p_2 + (1-p_1)(1-p_2))(p_2(1-p_3) + (1-p_2)p_3),\\
&p_1(1-p_2)(1-p_3) + (1-p_1)p_2p_3\\
&\quad = \quad (p_1(1-p_2) + (1-p_1)p_2)(p_2p_3 + (1-p_2)(1-p_3)), \quad \text{and}\\
&p_1(1-p_2)p_3 + (1-p_1)p_2(1-p_3)\\
&\quad = \quad (p_1(1-p_2) + (1-p_1)p_2)(p_2(1-p_3) + (1-p_2)p_3).
\end{aligned}$$

It is straightforward to verify that these four conditions are satisfied when $p_1 = 1/2$, when $p_3 = 1/2$, when $p_2 = 0$ and when $p_2 = 1$.

4.13 Suppose that $\pi_S : \{0,1\}^m \to \{0,1\}^n$ is an S-box. Prove the following facts about the function N_L (as defined in Definition 4.1).

(a) $N_L(0,0) = 2^m$.

Answer: This is trivial.

(b) $N_L(a,0) = 2^{m-1}$ for all integers a such that $0 < a \leq 2^m - 1$.

Answer: For $a \in \{0,1\}^m$, $a \neq (0,\ldots,0)$, there are exactly 2^{m-1} bitstrings $x \in \{0,1\}^m$ such that $\sum a_i x_i \equiv 0 \pmod 2$.

(c) For all integers b such that $0 \leq b \leq 2^n - 1$, it holds that

$$\sum_{a=0}^{2^m-1} N_L(a,b) = 2^{2m-1} \pm 2^{m-1}.$$

Answer: Suppose x is fixed; then $y = \pi_S(x)$ and $c = \sum b_i y_i \bmod 2$ is determined. If $x \neq 0$, then there are 2^{m-1} choices for a such that $\sum a_i x_i \bmod 2 = c$ (by part (b)). If $x = 0$, then there are either 0 or 2^m choices for a such that $\sum a_i x_i \bmod 2 = c$ (by part (a), depending on whether $c = 0$ or 1, respectively). Therefore it follows that

$$\sum_{a=0}^{2^m-1} N_L(a,b) = (2^m - 1)2^{m-1} + (0 \text{ or } 2^m) = 2^{2m-1} \pm 2^{m-1}.$$

(d) It holds that

$$\sum_{a=0}^{2^m-1} \sum_{b=0}^{2^n-1} N_L(a,b) \in \{2^{n+2m-1}, 2^{n+2m-1} + 2^{n+m-1}\}.$$

Answer: If $x \neq 0$, then there are 2^{m-1} choices for a for each b (by part (c)). Therefore we obtain $2^{n+m-1}(2^m - 1)$ quadruples (a, b, x, y) with $x \neq 0$ such that $\sum a_i x_i + \sum b_i y_i \bmod 2 = 0$.
Now we consider $x = 0$. Define $y_0 = \pi_S(0, \ldots, 0)$. If $y_0 = 0$, then all possible a and b work, so the number of quadruples $(a, b, x = 0, y = 0)$ is 2^{n+m}. If $y_0 \neq 0$, then for each a, there are 2^{n-1} choices for b, and the number of quadruples $(a, b, x = 0, y = y_0)$ is 2^{n+m-1}.
In total, the number of quadruples is

$$2^{n+m-1}(2^m - 1) + (2^{n+m} \text{ or } 2^{n+m-1}) = 2^{n+2m-1} \text{ or } 2^{n+2m-1} + 2^{n+m-1}.$$

4.14 An S-box $\pi_S : \{0,1\}^m \to \{0,1\}^n$ is said to be ***balanced*** if

$$|{\pi_S}^{-1}(y)| = 2^{n-m}$$

for all $y \in \{0,1\}^n$. Prove the following facts about the function N_L for a balanced S-box.

(a) $N_L(0, b) = 2^{m-1}$ for all integers b such that $0 < b \leq 2^n - 1$.

Answer: When $b \neq 0$, there are 2^{n-1} y's such that $\sum b_i y_i \bmod 2 = 0$. For each such y, there are exactly 2^{n-m} x's such that $\pi_S(x) = y$. Therefore, $N_L(0, b) = 2^{n-1} \times 2^{n-m} = 2^{m-1}$.

(b) For all integers a such that $0 \leq a \leq 2^m - 1$, it holds that

$$\sum_{b=0}^{2^n-1} N_L(a, b) = 2^{m+n-1} - 2^{m-1} + i2^n,$$

where i is an integer such that $0 \leq i \leq 2^{m-n}$.

Answer: When $y \neq 0$, thre are 2^{m-n} x's such that $y = \pi_S(x)$. For each such x, there are 2^{n-1} b's such that $\sum b_i y_i \equiv \sum a_i x_i \bmod 2$. Thus we obtain $2^{m-1}(2^n - 1)$ triples (b, x, y) with $y \neq 0$ such that $\sum a_i x_i + \sum b_i y_i \bmod 2 = 0$.
Now consider $y = 0$. Define

$$X_0 = \{x \in {\pi_S}^{-1}(0) : \sum a_i x_i \bmod 2 = 0\}$$

and denote $i = |X_0|$. Note that $0 \leq i \leq 2^{n-m}$. For each $x \in X_0$ and for every b, it holds that $\sum a_i x_i + \sum b_i y_i \bmod 2 = 0$. on the other hand, if $x \in {\pi_S}^{-1}(0) \backslash X_0$, then the condition holds for no b. Hence, we get $i2^n$ triples $(b, x, y = 0)$ with $y = 0$ such that $\sum a_i x_i + \sum b_i y_i \bmod 2 = 0$. Hence, the total number of triples is $2^{m+n-1} - 2^{m-1} + i2^n$, where $0 \leq i \leq 2^{n-m}$.

4.15 Suppose that the S-box of Example 4.1 is replaced by the S-box defined by the following substitution $\pi_{S'}$:

z	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\pi_{S'}(z)$	8	4	2	1	C	6	3	D	A	5	E	7	F	B	9	0

(a) Compute the linear approximation table for this S-box.

Answer: The table is as follows:

	b															
a	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
1	8	10	6	8	10	8	8	6	4	6	6	8	10	8	4	10
2	8	10	8	10	6	8	6	8	6	8	10	4	4	6	8	10
3	8	8	10	10	8	12	10	6	6	6	8	8	10	6	12	8
4	8	10	8	6	8	10	8	6	10	4	10	8	6	8	6	4
5	8	12	6	6	10	10	8	12	6	10	8	8	8	8	10	6
6	8	8	12	8	10	10	6	10	8	8	8	12	6	6	6	10
7	8	6	6	8	12	6	10	8	8	6	6	8	4	6	10	8
8	8	10	10	8	8	6	6	8	10	8	4	6	10	4	8	6
9	8	8	8	12	10	10	6	10	10	6	6	6	8	12	8	8
A	8	12	10	10	6	6	12	8	8	8	6	10	6	10	8	8
B	8	6	12	6	8	6	8	10	4	6	8	6	8	10	8	6
C	8	8	10	10	12	8	10	6	8	12	10	6	8	8	6	6
D	8	6	8	6	6	12	10	8	8	10	4	6	6	8	6	8
E	8	6	6	12	6	8	8	10	6	8	8	10	8	6	6	4
F	8	8	8	8	8	8	12	12	10	6	10	6	10	6	6	10

(b) Find a linear approximation using three active S-boxes, and use the piling-up lemma to estimate the bias of the random variable

$$\mathbf{X_{16}} \oplus \mathbf{U_1^4} \oplus \mathbf{U_9^4}.$$

Answer: The approximation incorporates the following three active S-boxes:

- In S_4^1, the random variable $\mathbf{T_1} = \mathbf{U_{16}^1} \oplus \mathbf{V_{13}^1}$ has bias $-1/4$
- In S_1^2, the random variable $\mathbf{T_2} = \mathbf{U_4^2} \oplus \mathbf{V_1^2}$ has bias $-1/4$
- In S_1^3, the random variable $\mathbf{T_3} = \mathbf{U_1^3} \oplus \mathbf{V_1^3} \oplus \mathbf{V_3^3}$ has bias $-1/4$

Using the piling-up lemma, the bias of the random variable $\mathbf{T_1} \oplus \mathbf{T_2} \oplus \mathbf{T_3}$ is estimated to be $-1/16$. Now, use the following relations:

$$\begin{aligned}
\mathbf{U_{16}^1} &= \mathbf{X_{16}} \oplus \mathbf{K_{16}^1}\\
\mathbf{U_4^2} &= \mathbf{V_{13}^1} \oplus \mathbf{K_4^2}\\
\mathbf{U_1^3} &= \mathbf{V_1^2} \oplus \mathbf{K_1^3}\\
\mathbf{U_1^4} &= \mathbf{V_1^3} \oplus \mathbf{K_1^4}\\
\mathbf{U_9^4} &= \mathbf{V_3^3} \oplus \mathbf{K_9^4}
\end{aligned}$$

to show that

$$\mathbf{T_1} \oplus \mathbf{T_2} \oplus \mathbf{T_3} = \mathbf{X_{16}} \oplus \mathbf{U_1^4} \oplus \mathbf{U_9^4} \oplus \text{key bits}.$$

Therefore we estimate that the bias of $\mathbf{X_{16}} \oplus \mathbf{U_1^4} \oplus \mathbf{U_9^4}$ is $\pm 1/16$.

(c) Describe a linear attack, analogous to Algorithm 4.3, that will find eight subkey bits in the last round.

Answer:

```
Algorithm: LINEARATTACK ($\mathcal{T}, T, {\pi_{S'}}^{-1}$)

for $(L_1, L_2) \leftarrow (0,0)$ to $(F,F)$
  do $Count[L_1, L_2] \leftarrow 0$
for each $(x, y) \in \mathcal{T}$
  do for $(L_1, L_2) \leftarrow (0,0)$ to $(F,F)$
       do $v^4_{(1)} \leftarrow L_1 \oplus y_{(1)}$
          $v^4_{(3)} \leftarrow L_2 \oplus y_{(3)}$
          $u^4_{(1)} \leftarrow {\pi_{S'}}^{-1}(v^4_{(1)})$
          $u^4_{(3)} \leftarrow {\pi_{S'}}^{-1}(v^4_{(3)})$
          $z \leftarrow x_{16} \oplus u^4_1 \oplus u^4_9$
          if $z = 0$
            then $Count[L_1, L_2] \leftarrow Count[L_1, L_2] + 1$
$max \leftarrow -1$
for $(L_1, L_2) \leftarrow (0,0)$ to $(F,F)$
  do $Count[L_1, L_2] \leftarrow |Count[L_1, L_2] - T/2|$
     if $Count[L_1, L_2] > max$
       then $max \leftarrow Count[L_1, L_2]$
            $maxkey \leftarrow (L_1, L_2)$
output $(maxkey)$
```

(d) Implement your attack and test it to see how many plaintexts are required in order for the algorithm to find the correct subkey bits (approximately 1000–1500 plaintexts should suffice; this attack is more efficient than Algorithm 4.3 because the bias is larger by a factor of 2, which means that the number of plaintexts can be reduced by a factor of about 4).

4.16 Suppose that the S-box of Example 4.1 is replaced by the S-box defined by the following substitution $\pi_{S''}$:

z	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\pi_{S''}(z)$	E	2	1	3	D	9	0	6	F	4	5	A	8	C	7	B

(a) Compute the table of values N_D (as defined in Definition 4.3) for this S-box.

Answer: The table of values is as follows:

	b'															
a'	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	2	0	4	0	2	0	0	0	0	2	4	0	0	2
2	0	2	0	0	0	0	0	2	0	0	2	0	0	2	2	6
3	0	2	0	4	0	2	0	0	0	2	0	4	0	2	0	0
4	0	4	2	2	0	2	0	2	2	0	0	2	0	0	0	0
5	0	0	0	4	0	0	0	4	0	0	0	0	2	2	2	2
6	0	0	0	0	2	0	2	0	2	0	2	0	2	2	2	2
7	0	0	4	2	2	0	0	0	4	2	0	0	0	0	2	0
8	0	2	0	0	2	4	2	2	0	2	0	0	0	2	0	0
9	0	6	0	0	0	0	2	0	0	0	2	4	0	2	0	0
A	0	0	2	0	0	0	0	2	4	0	4	2	0	0	2	0
B	0	0	0	0	2	2	2	2	0	0	0	0	4	0	4	0
C	0	0	2	0	0	2	4	0	2	0	0	0	2	2	2	0
D	0	0	2	2	2	0	2	0	0	2	6	0	0	0	0	0
E	0	0	2	2	0	0	0	0	2	6	0	0	0	0	0	4
F	0	0	0	0	2	4	0	2	0	2	0	2	2	2	0	0

(b) Find a differential trail using four active S-boxes, namely, S_1^1, S_4^1, S_4^2, and S_4^3, that has propagation ratio $27/2048$.

Answer: The following propagation ratios of differentials can be verified from the table computed in part (a):

- In S_1^1, $R_p(1001, 0001) = 3/8$
- In S_4^1, $R_p(1001, 0001) = 3/8$
- In S_4^2, $R_p(1001, 0001) = 3/8$
- In S_4^3, $R_p(0001, 1100) = 1/4$

These differentials can be combined to form a differential trail of the first three rounds of the SPN:

$$R_p(\mathtt{1001\ 0000\ 0000\ 1001}, \mathtt{0000\ 0000\ 0000\ 1100}) = \frac{27}{2048}.$$

Hence, it can be verified that

$$x' = \mathtt{1001\ 0000\ 0000\ 1001} \Rightarrow (u^4)' = \mathtt{0001\ 0001\ 0000\ 0000}$$

with probability $27/2048$.

(c) Describe a differential attack, analogous to Algorithm 4.3, that will find eight subkey bits in the last round.

Answer: The algorithm is presented on the next page.

(d) Implement your attack and test it to see how many plaintexts are required in order for the algorithm to find the correct subkey bits (approximately 100–200 plaintexts should suffice; this attack is not as efficient as Algorithm 4.3 because the propagation ratio is smaller by a factor of 2).

```
Algorithm: DIFFERENTIALATTACK ($\mathcal{T}, T, {\pi_{S''}}^{-1}$)

for $(L_1, L_2) \leftarrow (0,0)$ to $(F, F)$
  do $Count[L_1, L_2] \leftarrow 0$
for each $(x, y, x^*, y^*) \in \mathcal{T}$
  do if $(y_{(3)} = (y_{(3)})^*)$ and $(y_{(4)} = (y_{(4)})^*)$
       then for $(L_1, L_2) \leftarrow (0,0)$ to $(F, F)$
              do $v^4_{(1)} \leftarrow L_1 \oplus y_{(1)}$
                 $v^4_{(2)} \leftarrow L_2 \oplus y_{(2)}$
                 $u^4_{(1)} \leftarrow {\pi_{S''}}^{-1}(v^4_{(1)})$
                 $u^4_{(2)} \leftarrow {\pi_{S''}}^{-1}(v^4_{(2)})$
                 $(v^4_{(1)})^* \leftarrow L_1 \oplus (y_{(1)})^*$
                 $(v^4_{(2)})^* \leftarrow L_2 \oplus (y_{(2)})^*$
                 $(u^4_{(1)})^* \leftarrow {\pi_{S''}}^{-1}((v^4_{(1)})^*)$
                 $(u^4_{(2)})^* \leftarrow {\pi_{S''}}^{-1}((v^4_{(2)})^*)$
                 $(u^4_{(1)})' \leftarrow u^4_{(1)} \oplus (u^4_{(1)})^*$
                 $(u^4_{(2)})' \leftarrow u^4_{(2)} \oplus (u^4_{(2)})^*$
                 if $((u^4_{(1)})' = 0001)$ and $((u^4_{(2)})' = 0001)$
                   then $Count[L_1, L_2] \leftarrow Count[L_1, L_2] + 1$
$max \leftarrow -1$
for $(L_1, L_2) \leftarrow (0,0)$ to $(F, F)$
  do if $Count[L_1, L_2] > max$
       then $max \leftarrow Count[L_1, L_2]$
            $maxkey \leftarrow (L_1, L_2)$
output ($maxkey$)
```

4.17 Suppose that we use the SPN presented in Example 4.1, but the S-box is replaced by a function π_T that is not a permutation. This means, in particular, that π_T is not surjective. Use this fact to derive a ciphertext-only attack that can be used to determine the key bits in the last round, given a sufficient number of ciphertexts that all have been encrypted using the same key.

Answer: Suppose that ${\pi_T}^{-1}(z) = \emptyset$ for some $z \in \{0,1\}^4$. Suppose we are given a set of ciphertexts $\mathcal{T}$, all of which are encrypted using the same unknown key, K. For each $y = y_{(1)} \parallel y_{(2)} \parallel y_{(3)} \parallel y_{(4)} \in T$, and for each i, $1 \leq i \leq 4$, it must be the case that $y_{(i)} \oplus K_{(i)} \neq z$. For $1 \leq i \leq 4$, define

$$\mathcal{K}_i = \{0,1\}^4 \backslash \{z \oplus y_{(i)} : y \in \mathcal{T}\}.$$

Then $K_{(i)} \in \mathcal{K}_i$, $1 \leq i \leq 4$. If $|\mathcal{T}|$ is reasonably large, then we expect that $|\mathcal{K}_i| = 1$ for $1 \leq i \leq 4$, and hence the key K can be determined.

4.18 The *Geffe Generator* is the combining function $F : (\mathbb{Z}_2)^3 \rightarrow \mathbb{Z}_2$ defined by the following formula:

$$F(z_1, z_2, z_3) = (z_1 \wedge z_2) \oplus (\neg z_1 \wedge z_3).^1$$

Determine the correlations between the inputs and output of this function, as was done in Section 4.8.1 for the majority function.

Answer: We tabulate the values of the function F:

z_1	z_2	z_3	$F(z_1, z_2, z_3)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

From this, we see that

$$\mathbf{Pr}[\mathbf{z} = \mathbf{z_1}] = \frac{1}{2},$$

$$\mathbf{Pr}[\mathbf{z} = \mathbf{z_2}] = \frac{3}{4},$$

and

$$\mathbf{Pr}[\mathbf{z} = \mathbf{z_3}] = \frac{3}{4}.$$

4.19 Describe how the correlations computed in the previous exercise can be used to mount a correlation attack against the *Geffe Generator*. Note that this is a

[1]The notation $\neg z$ denotes the negation of a boolean variable z.

bit more complicated than the attack against the majority function generator because not all three correlations are bounded away from 1/2.

Answer: Suppose that the three LFSRs have lengths L_1, L_2 and L_3, resp. We begin by using the approach described in Section 4.8.1. Given a sufficient number of keystream bits, we can find the initial states of the second and third LFSRs. This requires testing $2^{L_2} + 2^{L_3}$ sequences. We cannot independently determine the initial state for the first LFSR because $\mathbf{Pr}[\mathbf{z} = \mathbf{z_1}] = 1/2$. However, since we have already determined the initial states of the other two LFSRs, we can test each possible initial state for the first LFSR by generating the keystream using the initial states for the second and third LFSRs that we have already determined. So the total number of sequences generated is $2^{L_1} + 2^{L_2} + 2^{L_3}$.

4.20 A function is ***balanced*** if it takes on the values 0 and 1 equally often. Construct a balanced combining function $F : (\mathbb{Z}_2)^3 \rightarrow \mathbb{Z}_2$ such that

$$\mathbf{Pr}[\mathbf{z_j} = \mathbf{z}] = \frac{1}{2},$$

for $j = 1, 2, 3$.

Answer: One simple solution is $F(z_1, z_2, z_3) = z_1 \oplus z_2 \oplus z_3$.

Chapter 5

Hash Functions and Message Authentication

5.1 Define a toy hash function $h : (\mathbb{Z}_2)^7 \rightarrow (\mathbb{Z}_2)^4$ by the rule $h(x) = xA$ where all operations are modulo 2 and

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Find all preimages of $(0, 1, 0, 1)$.

Answer: This is equivalent to solving the following system of equations:

$$\begin{aligned} x_1 + x_2 + x_3 + x_4 &= 0 \\ x_2 + x_3 + x_4 + x_5 &= 1 \\ x_3 + x_4 + x_5 + x_6 &= 0 \\ x_4 + x_5 + x_6 + x_7 &= 1 \end{aligned}$$

There are eight solutions for $(x_1, \ldots, x_7)$: $(1,1,1,1,0,0,0)$, $(1,1,0,0,0,0,1)$, $(1,0,1,0,0,1,0)$, $(1,0,0,1,0,1,1)$, $(0,1,1,0,1,0,0)$, $(0,1,0,1,1,0,1)$, $(0,0,1,1,1,1,0)$, $(0,0,0,0,1,1,1)$.

5.2 Suppose $h : \mathcal{X} \rightarrow \mathcal{Y}$ is an (N, M)-hash function. For any $y \in \mathcal{Y}$, let

$$h^{-1}(y) = \{x : h(x) = y\}$$

and denote $s_y = |h^{-1}(y)|$. Define

$$S = |\{\{x_1, x_2\} : h(x_1) = h(x_2)\}|.$$

Note that S counts the number of unordered pairs in $\mathcal{X}$ that collide under h.

(a) Prove that

$$\sum_{y \in \mathcal{Y}} s_y = N,$$

so the mean of the s_y's is '

$$\bar{s} = \frac{N}{M}.$$

Answer: Clearly the sets $h^{-1}(y)$, $y \in \mathcal{Y}$, form a partition of $\mathcal{X}$. Hence, $\sum_{y \in \mathcal{Y}} s_y = |\mathcal{X}| = N$. Then, because $|\mathcal{Y}| = M$, it is immediate that the mean of the s_y's is N/M.

(b) Prove that

$$S = \sum_{y \in \mathcal{Y}} \binom{s_y}{2} = \frac{1}{2} \sum_{y \in \mathcal{Y}} {s_y}^2 - \frac{N}{2}.$$

Answer: We have the following:

$$\begin{aligned} \sum_{y \in \mathcal{Y}} \binom{s_y}{2} &= \frac{1}{2} \sum_{y \in \mathcal{Y}} {s_y}^2 - \frac{1}{2} \sum_{y \in \mathcal{Y}} s_y \\ &= \frac{1}{2} \sum_{y \in \mathcal{Y}} {s_y}^2 - \frac{N}{2}, \end{aligned}$$

using the result proven in part (a).

(c) Prove that

$$\sum_{y \in \mathcal{Y}} (s_y - \bar{s})^2 = 2S + N - \frac{N^2}{M}.$$

Answer: We have the following:

$$\begin{aligned} \sum_{y \in \mathcal{Y}} (s_y - \bar{s})^2 &= \sum_{y \in \mathcal{Y}} {s_y}^2 - 2\bar{s} \sum_{y \in \mathcal{Y}} s_y + M\bar{s}^2 \\ &= 2\left(S + \frac{N}{2}\right) - \frac{2N}{M} \times N + M\left(\frac{N}{M}\right)^2 \\ &= 2S + N - \frac{N^2}{M}. \end{aligned}$$

(d) Using the result proved in part (c), prove that

$$S \geq \frac{1}{2}\left(\frac{N^2}{M} - N\right).$$

Further, show that equality is attained if and only if

$$s_y = \frac{N}{M}$$

for every $y \in \mathcal{Y}$.

Answer: Clearly

$$\sum_{y \in \mathcal{Y}} (s_y - \bar{s})^2 \geq 0,$$

and this sum is zero if and only if $s_y = \bar{s}$ for all $y \in \mathcal{Y}$. In other words,

$$0 \leq 2S + N - \frac{N^2}{M},$$

and equality occurs if and only if $s_y = N/M$ for all $y \in \mathcal{Y}$. Finally, note that

$$0 \leq 2S + N - \frac{N^2}{M} \Leftrightarrow S \geq \frac{1}{2}\left(\frac{N^2}{M} - N\right).$$

5.3 As in Exercise 5.2, suppose $h : \mathcal{X} \to \mathcal{Y}$ is an (N, M)-hash function, and let

$$h^{-1}(y) = \{x : h(x) = y\}$$

for any $y \in \mathcal{Y}$. Let ϵ denote the probability that $h(x_1) = h(x_2)$, where x_1 and x_2 are random (not necessarily distinct) elements of $\mathcal{X}$. Prove that

$$\epsilon \geq \frac{1}{M},$$

with equality if and only if

$$|h^{-1}(y)| = \frac{N}{M}$$

for every $y \in \mathcal{Y}$.

Answer: Define

$$T = |\{(x_1, x_2) : h(x_1) = h(x_2)\}|.$$

Then $T = 2S + N$, where S is defined as in Exercise 5.2. (The term "$+N$" accounts for the collisions where $x_1 = x_2$; and each unordered pair $\{x_1, x_2\}$ with $h(x_1) = h(x_2)$ accounts for two ordered pairs, namely, (x_1, x_2) and (x_2, x_1).) Using the result proven in Exercise 5.1, part (d), we have that

$$\epsilon = \frac{2S + N}{N^2} \geq \frac{2\left(\frac{1}{2}\left(\frac{N^2}{M} - N\right)\right) + N}{N^2} = \frac{1}{M}.$$

Further, equality occurs if and only if $s_y = N/M$ for all $y \in \mathcal{Y}$ (as in Exercise 5.1, part (d)).

5.4 Suppose that $h : \mathcal{X} \to \mathcal{Y}$ is an (N, M)-hash function, let

$$h^{-1}(y) = \{x : h(x) = y\}$$

and let $s_y = |h^{-1}(y)|$ for any $y \in \mathcal{Y}$. Suppose that we try to solve **Preimage** for the function h, using Algorithm 5.1, assuming that we have only oracle access for h. For a given $y \in \mathcal{Y}$, suppose that $\mathcal{X}_0$ is chosen to be a random subset of $\mathcal{X}$ having cardinality q.

(a) Prove that the success probability of Algorithm 5.1, given y, is

$$1 - \frac{\binom{N-s_y}{q}}{\binom{N}{q}}.$$

Answer: The total number of subsets $\mathcal{X}_0 \subseteq \mathcal{X}$ such that $|\mathcal{X}_0| = q$ is $\binom{N}{q}$. The number of subsets $\mathcal{X}_0 \subseteq \mathcal{X}$ such that $|\mathcal{X}_0| = q$ and $\mathcal{X}_0 \cap h^{-1}(y) = \emptyset$ is $\binom{N-s_y}{q}$. Therefore the failure probability of Algorithm 5.1 is $\binom{N-s_y}{q}/\binom{N}{q}$, and the result follows.

(b) Prove that the average success probability of Algorithm 5.1 (over all $y \in \mathcal{Y}$) is

$$1 - \frac{1}{M}\sum_{y\in\mathcal{Y}} \frac{\binom{N-s_y}{q}}{\binom{N}{q}}.$$

Answer: The average success probability is

$$\frac{1}{M}\sum_{y\in\mathcal{Y}} \left(1 - \frac{\binom{N-s_y}{q}}{\binom{N}{q}}\right) = 1 - \frac{1}{M}\sum_{y\in\mathcal{Y}} \frac{\binom{N-s_y}{q}}{\binom{N}{q}}.$$

(c) In the case $q = 1$, show that the success probability in part (b) is $1/M$.

Answer: We compute as follows:

$$\begin{aligned} 1 - \frac{1}{M}\sum_{y\in\mathcal{Y}} \frac{\binom{N-s_y}{1}}{\binom{N}{1}} &= 1 - \frac{1}{M}\sum_{y\in\mathcal{Y}} \frac{N-s_y}{N} \\ &= 1 - \frac{1}{M}\left(M - \sum_{y\in\mathcal{Y}} \frac{s_y}{N}\right) \\ &= 1 - \frac{1}{M}(M-1) \\ &= \frac{1}{M}, \end{aligned}$$

where we use the fact that $\sum_{y\in\mathcal{Y}} s_y = N$, which was proven in Exercise 5.2(a).

5.5 Suppose that $h : \mathcal{X} \to \mathcal{Y}$ is an (N, M)-hash function, let

$$h^{-1}(y) = \{x : h(x) = y\}$$

and let $s_y = |h^{-1}(y)|$ for any $y \in \mathcal{Y}$. Suppose that we try to solve **Second Preimage** for the function h, using Algorithm 5.2, assuming that we have only oracle access for h. For a given $x \in \mathcal{Y}$, suppose that $\mathcal{X}_0$ is chosen to be a random subset of $\mathcal{X}\backslash\{x\}$ having cardinality $q - 1$.

(a) Prove that the success probability of Algorithm 5.2, given x, is

$$1 - \frac{\binom{N-s_y}{q-1}}{\binom{N-1}{q-1}}.$$

Answer: The total number of subsets $\mathcal{X}_0 \subseteq \mathcal{X} \backslash \{x\}$ such that $|\mathcal{X}_0| = q - 1$ is $\binom{N-1}{q-1}$. Denote $h(x) = y$; then the number of subsets $\mathcal{X}_0 \subseteq \mathcal{X} \backslash \{x\}$ such that $|\mathcal{X}_0| = q - 1$ and $\mathcal{X}_0 \cap h^{-1}(y) = \emptyset$ is $\binom{N-s_y}{q-1}$. Therefore the failure probability of Algorithm 5.2 is $\binom{N-s_y}{q-1} / \binom{N-1}{q-1}$, and the result follows.

(b) Prove that the average success probability of Algorithm 5.2 (over all $x \in \mathcal{X}$) is

$$1 - \frac{1}{N} \sum_{y \in \mathcal{Y}} \frac{s_y \binom{N-s_y}{q-1}}{\binom{N-1}{q-1}}.$$

Answer: The average success probability is

$$\frac{1}{N} \sum_{x \in \mathcal{X}} \left(1 - \frac{\binom{N-s_{h(x)}}{q-1}}{\binom{N-1}{q-1}} \right) = 1 - \frac{1}{N} \sum_{y \in \mathcal{Y}} \frac{s_y \binom{N-s_y}{q-1}}{\binom{N-1}{q-1}}.$$

(c) In the case $q = 2$, show that the success probability in part (b) is

$$\frac{\sum_{y \in \mathcal{Y}} {s_y}^2}{N(N-1)} - \frac{1}{N-1}.$$

Answer: We compute as follows:

$$\begin{aligned} 1 - \frac{1}{N} \sum_{y \in \mathcal{Y}} \frac{s_y \binom{N-s_y}{1}}{\binom{N-1}{1}} &= 1 - \frac{1}{N(N-1)} \left(\sum_{y \in \mathcal{Y}} N s_y - \sum_{y \in \mathcal{Y}} {s_y}^2 \right) \\ &= 1 - \frac{N}{N-1} + \frac{\sum_{y \in \mathcal{Y}} {s_y}^2}{N(N-1)} \\ &= \frac{\sum_{y \in \mathcal{Y}} {s_y}^2}{N(N-1)} - \frac{1}{N-1}, \end{aligned}$$

where we use the fact that $\sum_{y \in \mathcal{Y}} s_y = N$, which was proven in Exercise 5.2(a).

5.6 (This exercise is based on an example from the *Handbook of Applied Cryptography* by A.J. Menezes, P.C. Van Oorschot, and S.A. Vanstone.) Suppose g is a collision resistant hash function that takes an arbitrary bitstring as input and produces an n-bit message digest. Define a hash function h as follows:

$$h(x) = \begin{cases} 0 \parallel x & \text{if } x \text{ is a bitstring of length } n \\ 1 \parallel g(x) & \text{otherwise.} \end{cases}$$

(a) Prove that h is collision resistant.

Answer: Suppose $h(x) = h(x')$ for some $x \neq x'$. The proof is divided into cases:

case 1 Suppose $|x| = |x'| = n$. Then $h(x) = 0 \parallel x$ and $h(x') = 0 \parallel x'$. Since $h(x) = h(x')$, it follows that $x = x'$, which is a contradiction.

case 2 Suppose $|x| = n$ and $|x'| \neq n$. Then the first bit of $h(x)$ is 0 and the first bit of $h(x')$ is 1, so $h(x) \neq h(x')$.

case 3 Suppose $|x| \neq n$ and $|x'| \neq n$. Then $h(x) = 1 \parallel g(x)$ and $h(x') = 1 \parallel g(x')$. Since $h(x) = h(x')$, it follows that $g(x) = g(x')$ and we have found a collision for g. This contradicts the assumption that g is collision resistant.

(b) Prove that h is not preimage resistant. More precisely, show that preimages (for the function h) can easily be found for half of the possible message digests.

Answer: Suppose that y has $n+1$ bits and the first bit of y is 0. Write $y = 0 \parallel y'$. Then y' has n bits and $h(y') = 0 \parallel y' = y$. Because exactly half of the possible message digests have 0 as their first bit, we have proven the desired result.

5.7 If we define a hash function (or compression function) h that will hash an n-bit binary string to an m-bit binary string, we can view h as a function from $\mathbb{Z}_{2^n}$ to $\mathbb{Z}_{2^m}$. It is tempting to define h using integer operations modulo 2^m. We show in this exercise that some simple constructions of this type are insecure and should therefore be avoided.

(a) Suppose that $n = m > 1$ and $h : \mathbb{Z}_{2^m} \to \mathbb{Z}_{2^m}$ is defined as

$$h(x) = x^2 + ax + b \bmod 2^m.$$

Prove that it is (usually) easy to solve **Second Preimage** for any $x \in \mathbb{Z}_{2^m}$ without having to solve a quadratic equation.

HINT Show that it is possible to find a linear function $g(x)$ such that $h(g(x)) = h(x)$ for all x. This solves **Second Preimage** for any x such that $g(x) \neq x$.

Answer: Suppose that a is even; then $a2^{m-1} \equiv 0 \pmod{2^m}$. Also, $2^{2m-2} \equiv 0 \pmod{2^m}$ because $m \geq 2$. Define $x' = x + 2^{m-1} \bmod 2^m$; then

$$\begin{aligned} h(x') &= (x + 2^{m-1})^2 + a(x + 2^{m-1}) + b \bmod 2^m \\ &= x^2 + 2^m x + 2^{2m-2} + ax + a2^{m-1} + b \bmod 2^m \\ &= x^2 + ax + b \bmod 2^m \\ &= f(x). \end{aligned}$$

Now suppose that a is odd. Define $x' = -x - a \bmod 2^m$; note that $x' \neq x$ because $2x + a$ is odd. Now, we have that

$$\begin{aligned} h(x') &= (-x-a)(-x) + b \bmod 2^m \\ &= (x+a)x + b \bmod 2^m \\ &= h(x). \end{aligned}$$

Therefore, given any x, we can find $x' \neq x$ such that $h(x') = h(x)$.

(b) Suppose that $n > m$ and $h : \mathbb{Z}_{2^n} \rightarrow \mathbb{Z}_{2^m}$ is defined to be a polynomial of degree d:

$$h(x) = \sum_{i=0}^{d} a_i x^i \bmod 2^m,$$

where $a_i \in \mathbb{Z}$ for $0 \leq i \leq d$. Prove that it is easy to solve **Second Preimage** for any $x \in \mathbb{Z}_{2^n}$ without having to solve a polynomial equation.

HINT Make use of the fact that $h(x)$ is defined using reduction modulo 2^m, but the domain of h is $\mathbb{Z}_{2^n}$, where $n > m$.

Answer: Define $x' = x + 2^m \bmod 2^n$. Then $x' \neq x$ and $h(x') = h(x)$.

5.8 Suppose that $f : \{0,1\}^m \rightarrow \{0,1\}^m$ is a preimage resistant bijection. Define $h : \{0,1\}^{2m} \rightarrow \{0,1\}^m$ as follows. Given $x \in \{0,1\}^{2m}$, write

$$x = x' \parallel x''$$

where $x', x'' \in \{0,1\}^m$. Then define

$$h(x) = f(x' \oplus x'').$$

Prove that h is not second preimage resistant.

Answer: We are given $x = x' \parallel x''$. Let $x_0 \in \{0,1\}^m$, $x_0 \neq 0\,0 \cdots 0$. Define $x_1' = x' \oplus x_0$, $x_1'' = x'' \oplus x_0$ and $x_1 = x_1' \parallel x_1''$. Then $x \neq x_1$ and $h(x) = h(x_1)$.

5.9 For $M = 365$ and $15 \leq q \leq 30$, compare the exact value of ϵ given by the formula in the statement of Theorem 5.4 with the estimate for ϵ derived after the proof of that theorem.

Answer: Define ϵ_1 to denote the exact probability, as computed in Theorem 4.4; and define $\epsilon_2 = 1 - e^{-q(q-1)/(2M)}$. Values of ϵ_1 and ϵ_2 are tabulated as follows:

q	ϵ_1	ϵ_2
15	.2529013198	.2499918703
16	.2836040053	.2801893756
17	.3150076653	.3110611335
18	.3469114179	.3424129197
19	.3791185260	.3740552376
20	.4114383836	.4058051275
21	.4436883352	.4374878054
22	.4756953077	.4689381108
23	.5072972343	.5000017522
24	.5383442579	.5305363394
25	.5686997040	.5604121995
26	.5982408201	.5895129752
27	.6268592823	.6177360099
28	.6544614723	.6449925266
29	.6809685375	.6712076120
30	.7063162427	.6963200177

5.10 Suppose that messages are designated as "safe" or "dangerous" and an adversary is trying to find a collision of one safe and one dangerous message under a hash function h. That is, the adversary is trying to find a safe message x and a dangerous message x' such that $h(x) = h(x')$. An obvious attack would be to choose a set $\mathcal{X}_0$ of Q safe messages and a set $\mathcal{X}_0'$ of Q' dangerous messages, and test the QQ' resulting ordered pairs $(x, x') \in \mathcal{X}_0 \times \mathcal{X}_0'$ to see if a collision occurs. We analyze the success of this approach in the random oracle model, assuming that there are M possible message digests.

(a) For a fixed value $x \in \mathcal{X}_0$, determine an upper bound on the probability that $h(x) \neq h(x')$ for all $x' \in \mathcal{X}_0'$.

Answer: The probability of collision for two given inputs is $1/M$. Since x is fixed and there are Q' possibilities for x', the probability that $h(x) \neq h(x')$ for all $x' \in \mathcal{X}_0'$ is $(1 - 1/M)^{Q'}$.

(b) Using the result from (a), determine an upper bound on the probability that $h(x) \neq h(x')$ for all $x \in \mathcal{X}_0$ and all $x' \in \mathcal{X}_0'$.

Answer: There are Q possibilities for x. Using the result from part (a), the probability that $h(x) \neq h(x')$ for all $x \in \mathcal{X}_0$ and all $x' \in \mathcal{X}_0'$ is $((1-1/M)^{Q'})^Q = (1-1/M)^{QQ'}$.

(c) Show that there is a 50% probability of finding at least one collision using this method if $QQ' \approx cM$, for a suitable positive constant c.

Answer: We want $(1 - 1/M)^{QQ'} \approx 1/2$. We use the estimate $1 - x \approx e^{-x}$ (see page 145). Thus we want $(e^{-1/M})^{QQ'} = e^{-QQ'/M} \approx 1/2$. This simplifies to $-QQ'/M \approx \ln(1/2)$, so $QQ'/M \approx \ln 2$ and finally, $QQ' \approx M \ln 2$. So the desired constant $c \approx \ln 2$.

5.11 Suppose $h : \mathcal{X} \rightarrow \mathcal{Y}$ is a hash function where $|\mathcal{X}|$ and $|\mathcal{Y}|$ are finite and $|\mathcal{X}| \geq 2|\mathcal{Y}|$. Suppose that h is a ***balanced hash function*** (i.e.,

$$|h^{-1}(y)| = \frac{|\mathcal{X}|}{|\mathcal{Y}|}$$

for all $y \in \mathcal{Y}$). Finally, suppose ORACLE-PREIMAGE is an (ϵ, Q)-algorithm for **Preimage**, for the fixed hash function h. Prove that COLLISION-TO-PREIMAGE is an $(\epsilon/2, Q+1)$-algorithm for **Collision**, for the fixed hash function h.

Answer: We compute as follows:

$$\begin{aligned}
&\mathbf{Pr}[\text{COLLISIONTOPREIMAGE succeeds}] \\
&= \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \mathbf{Pr}[\text{COLLISIONTOPREIMAGE succeeds}|x] \\
&= \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \left(\mathbf{Pr}[\text{ORACLEPREIMAGE succeeds}|h(x)] \times \frac{\frac{|\mathcal{X}|}{|\mathcal{Y}|} - 1}{\frac{|\mathcal{X}|}{|\mathcal{Y}|}} \right) \\
&= \frac{1}{|\mathcal{X}|} \left(1 - \frac{|\mathcal{Y}|}{|\mathcal{X}|}\right) \sum_{x \in \mathcal{X}} \mathbf{Pr}[\text{ORACLEPREIMAGE succeeds}|h(x)] \\
&= \frac{1}{|\mathcal{X}|} \left(1 - \frac{|\mathcal{Y}|}{|\mathcal{X}|}\right) \frac{|\mathcal{X}|}{|\mathcal{Y}|} \sum_{y \in \mathcal{Y}} \mathbf{Pr}[\text{ORACLEPREIMAGE succeeds}|y] \\
&= \frac{1}{|\mathcal{Y}|} \left(1 - \frac{|\mathcal{Y}|}{|\mathcal{X}|}\right) \sum_{y \in \mathcal{Y}} \mathbf{Pr}[\text{ORACLEPREIMAGE succeeds}|y] \\
&\geq \frac{1}{|\mathcal{Y}|} \left(1 - \frac{|\mathcal{Y}|}{|\mathcal{X}|}\right) |\mathcal{Y}| \, \epsilon \\
&\geq \frac{\epsilon}{2}.
\end{aligned}$$

5.12 Suppose $h_1 : \{0,1\}^{2m} \rightarrow \{0,1\}^m$ is a collision resistant hash function.

(a) Define $h_2 : \{0,1\}^{4m} \rightarrow \{0,1\}^m$ as follows:

1. Write $x \in \{0,1\}^{4m}$ as $x = x_1 \parallel x_2$, where $x_1, x_2 \in \{0,1\}^{2m}$.
2. Define $h_2(x) = h_1(h_1(x_1) \parallel h_1(x_2))$.

Prove that h_2 is collision resistant (i.e., given a collision for h_2, show how to find a collision for h_1).

Answer: Suppose that we have found a collision for h_2, say $h_2(x) = h_2(x')$ where $x \neq x'$. Denote $x = x_1 \parallel x_2$ and $x' = x'_1 \parallel x'_2$.
First, suppose that $h_1(x_1) \neq h_1(x'_1)$. Then

$$h_1(x_1) \parallel h_1(x_2) \neq h_1(x'_1) \parallel h_1(x'_2)$$

and

$$h_1(h_1(x_1) \parallel h_1(x_2)) = h_1(h_1(x'_1) \parallel h_1(x'_2)).$$

Therefore we have found a collision for h_1.

If $h_1(x_2) \neq h_1(x_2')$, then we have a collision for h_1 by a similar argument.

Therefore we can assume that $h_1(x_1) = h_1(x_1')$ and $h_1(x_2) = h_1(x_2')$. Because $x \neq x'$, it follows that $(x_1, x_2) \neq (x_1', x_2')$. Therefore $x_1 \neq x_1'$ or $x_2 \neq x_2'$. In either of these two cases, we have a collision for h_1.

We conclude that we can always find a collision for h_1, given a collision for h_2.

(b) For an integer $i \geq 2$, define a hash function $h_i : \{0,1\}^{2^i m} \rightarrow \{0,1\}^m$ recursively from h_{i-1}, as follows:

1. Write $x \in \{0,1\}^{2^i m}$ as $x = x_1 \parallel x_2$, where $x_1, x_2 \in \{0,1\}^{2^{i-1} m}$.
2. Define $h_i(x) = h_1(h_{i-1}(x_1) \parallel h_{i-1}(x_2))$.

Prove that h_i is collision resistant.

Answer: Suppose that we have found a collision for h_i, say $h_i(x) = h_i(x')$ where $x \neq x'$. Denote $x = x_1 \parallel x_2$ and $x' = x_1' \parallel x_2'$.

First, suppose that $h_{i-1}(x_1) \neq h_{i-1}(x_1')$. Then

$$h_{i-1}(x_1) \parallel h_{i-1}(x_2) \neq h_{i-1}(x_1') \parallel h_{i-1}(x_2')$$

and

$$h_1(h_{i-1}(x_1) \parallel h_{i-1}(x_2)) = h_1(h_{i-1}(x_1') \parallel h_{i-1}(x_2')).$$

Therefore we have found a collision for h_1.

If $h_{i-1}(x_2) \neq h_{i-1}(x_2')$, then we have a collision for h_1 by a similar argument.

Therefore we can assume that $h_{i-1}(x_1) = h_{i-1}(x_1')$ and $h_{i-1}(x_2) = h_{i-1}(x_2')$. Because $x \neq x'$, it follows that $(x_1, x_2) \neq (x_1', x_2')$. Therefore $x_1 \neq x_1'$ or $x_2 \neq x_2'$. In either of these two cases, we have a collision for h_{i-1}.

We conclude that we can always find a collision for at least one of h_1 or h_{i-1}, given a collision for h_i.

5.13 In this exercise, we consider a simplified version of the Merkle-Damgård construction. Suppose

$$\mathbf{compress} : \{0,1\}^{m+t} \rightarrow \{0,1\}^m,$$

where $t \geq 1$, and suppose that

$$x = x_1 \parallel x_2 \parallel \cdots \parallel x_k,$$

where

$$|x_1| = |x_2| = \cdots = |x_k| = t.$$

We study the following iterated hash function:

Algorithm 5.8: SIMPLIFIED MERKLE-DAMGÅRD (x, k, t)

```
external compress
$z_1 \leftarrow 0^m \parallel x_1$
$g_1 \leftarrow$ compress$(z_1)$
for $i \leftarrow 1$ to $k-1$
    do $\begin{cases} z_{i+1} \leftarrow g_i \parallel x_{i+1} \\ g_{i+1} \leftarrow \textbf{compress}(z_{i+1}) \end{cases}$
$h(x) \leftarrow g_k$
return $(h(x))$
```

Suppose that **compress** is collision resistant, and suppose further that **compress** is ***zero preimage resistant***, which means that it is hard to find $z \in \{0,1\}^{m+t}$ such that **compress**$(z) = 0^m$. Under these assumptions, prove that h is collision resistant.

Answer: Suppose that $h(x) = h(x')$ where $x \neq x'$. We consider two cases:

(a) $|x| = |x'| = kt$ for some positive integer k, and

(b) $|x| = kt$ and $|x'| = \ell t$, where k and ℓ are positive integers such that $\ell > k$.

We consider the two cases in turn.

(a) We have $g_k = g'_k$. If $z_k \neq z'_k$, then we have a collision for **compress** and we're done, so we assume that $z_k = z'_k$. This implies that $g_{k-1} = g'_{k-1}$ and $x_k = x'_k$.
Now if $z_{k-1} \neq z'_{k-1}$, then we have a collision, so we assume $z_{k-1} = z'_{k-1}$, which implies that $g_{k-2} = g'_{k-2}$ and $x_{k-1} = x'_{k-1}$.
Continuing to work backwards, either we find a collision for **compress**, or we have $x_i = x'_i$ for $i = k, k-1, \dots, 1$. But then $x = x'$, a contradiction. We conclude that we always find a collision for **compress** in this case.

(b) We have $g_k = g'_\ell$. If $z_k \neq z'_\ell$, then we have a collision for **compress** and we're done, so we assume that $z_k = z'_\ell$. This implies that $g_{k-1} = g'_{\ell-1}$ and $x_k = x'_\ell$.
Now if $z_{k-1} \neq z'_{\ell-1}$, then we have a collision, so we assume $z_{k-1} = z'_{\ell-1}$, which implies that $g_{k-2} = g'_{\ell-2}$ and $x_{k-1} = x'_{\ell-1}$.
Continuing to work backwards, either we find a collision for **compress**, or we eventually reach the situation where $z_1 = z'_{\ell-k+1}$. Then $0^m = g'_{\ell-k} =$ **compress**$(z'_{\ell-k})$, so **compress** is not zero preimage resistant. Therefore we either find a collision or a zero preimage for **compress** in this case.

5.14 Message authentication codes are often constructed using block ciphers in

CBC mode. Here we consider the construction of a message authentication code using a block cipher in CFB mode. Given a sequence of plaintext blocks, $x_1, \ldots, x_n$, suppose we define the initialization vector IV to be x_1. Then encrypt the sequence $x_2, \ldots, x_n$ using key K in CFB mode, obtaining the ciphertext sequence $y_1, \ldots, y_{n-1}$ (note that there are only $n-1$ ciphertext blocks). Finally, define the MAC to be $e_K(y_{n-1})$. Prove that this MAC actually turns out to be identical to CBC-MAC, as presented in Section 5.5.2.

Answer: Using CFB mode, we obtain the following:

$$\begin{aligned}
\text{IV} &= x_1 \\
y_1 &= e_K(x_1) \oplus x_2 \\
y_2 &= e_K(y_1) \oplus x_3 \\
y_3 &= e_K(y_2) \oplus x_4 \\
\vdots \;&\;\vdots\;\; \vdots \\
y_{n-1} &= e_K(y_{n-2}) \oplus x_n \\
\text{MAC} &= e_K(y_{n-1}).
\end{aligned}$$

Using CBC mode with IV $= 0\,0 \cdots 0$, we obtain the following:

$$\begin{aligned}
\text{IV} &= 0\,0 \cdots 0 \\
y'_1 &= e_K(x_1) \\
y'_2 &= e_K(y'_1 \oplus x_2) \\
y'_3 &= e_K(y'_2 \oplus x_3) \\
\vdots \;&\;\vdots\;\; \vdots \\
y'_n &= e_K(y'_{n-1} \oplus x_n) \\
\text{MAC}' &= y'_n.
\end{aligned}$$

It is easy to prove by induction on i that $y_i = y'_i \oplus x_{i+1}$, $1 \leq i \leq n-1$. Finally, we have

$$\begin{aligned}
\text{MAC} &= e_K(y_{n-1}) \\
&= e_K(y'_{n-1} \oplus x_n) \\
&= y'_n \\
&= \text{MAC}'.
\end{aligned}$$

Therefore the same MAC is produced by both methods.

5.15 Suppose that $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ is a cryptosystem with $\mathcal{P} = \mathcal{C} = \{0,1\}^m$. Let $n \geq 2$ be a fixed integer, and define a hash family $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$, where $\mathcal{X} = (\{0,1\}^m)^n$ and $\mathcal{Y} = \{0,1\}^m$, as follows:

$$h_K(x_1, \ldots, x_n) = e_K(x_1) \oplus \cdots \oplus e_K(x_n).$$

Suppose that $(x_1, \dots, x_n)$ is an arbitrary message. Show how an adversary can then determine $h_K(x_1, \dots, x_n) = e_K(x_1)$ by using at most one oracle query. (This is called a ***selective forgery***, because a specific message is given to the adversary and the adversary is then required to find the tag for the given message.)

HINT The proof is divided into three mutually exclusive cases as follows:

case 1 In this case, we assume that not all of the x_i's are identical. Here, one oracle query suffices.

case 2 In this case, we assume n is even and $x_1 = \cdots = x_n$. Here, no oracle queries are required.

case 3 In this case, we assume $n \geq 3$ is odd and $x_1 = \cdots = x_n$. Here, one oracle query suffices.

Answer: First, suppose that $x_i \neq x_j$ for some i, j. Define

$$x'_k = \begin{cases} x_k & \text{if } k \neq i, j \\ x_i & \text{if } k = j \\ x_j & \text{if } k = i. \end{cases}$$

Request the MAC of $(x'_1, \dots, x'_n)$, say y_0. Then y_0 is a forged MAC for the message $(x_1, \dots, x_n)$.

Now, suppose that $x_1 = \cdots = x_n$. If n is even, then $h_K(x_1, \dots, x_1) = 0$ (i.e., we have a $(1, 0)$-forgery). If n is odd, then let $x' \neq x_1$ and request the MAC of $(x', \dots, x', x_1)$, say y_0. Then y_0 is a forged MAC for the message $(x_1, \dots, x_1)$.

5.16 (a) Suppose that the hash family $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ is a secure MAC algorithm. The tag for a message $x \in \mathcal{X}$ is $h_K(x)$. Suppose we instead computed the tag to be $x \parallel h_K(x)$. Would the resulting MAC algorithm still be considered secure? Explain.

Answer: The modified MAC is secure. Suppose an adversary can construct a valid tag $x \parallel h_K(x)$ after observing various valid tags $x' \parallel h_K(x')$ for $x' \neq x$. Then $h_K(x)$ is a valid tag for x in the original MAC, so the adversary can break the original MAC, which is a contradiction.

(b) Discuss why the general strategy of MAC-and-encrypt should be avoided.

HINT Consider modifying a secure MAC algorithm as described in part (a) and examine the impact of this change in the context of MAC-and-encrypt.

Answer: Suppose $h_K(x)$ is the tag in a secure MAC. Then, from part (a), the modified MAC, in which a tag is $x \parallel h_K(x)$, is also secure. Now suppose we also have a secure encryption scheme. Using MAC-and-encrypt, we would choose a MAC key K_1 and an encryption key K_2

and then transmit the pair $(e_{K_2}(x), x \parallel h_{K_1}(x))$. However, the plaintext x can trivially be extracted from this pair, so MAC-and-encrypt is not secure.

5.17 Suppose that $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ is a strongly universal (N, M)-hash family.

(a) If $|\mathcal{K}| = M^2$, show that there exists a $(1, 2)$-forger for this hash family (i.e., $Pd_2 = 1$).

Answer: Choose any $x, x' \in \mathcal{X}$ such that $x \neq x'$. Request the MACs of x and x', which we denote y, y', respectively. There is a unique key K such that $h_K(x) = y$ and $h_K(x') = y'$. Now given any $x'' \neq x, x'$, it is possible to compute the forged MAC $h_K(x'')$ because the key K is known.

(b) (This generalizes the result proven in part (a).) Denote $\lambda = |\mathcal{K}|/M^2$. Prove there exists a $(1/\lambda, 2)$-forger for this hash family (i.e., $Pd_2 \geq 1/\lambda$).

Answer: Choose any $x, x' \in \mathcal{X}$ such that $x \neq x'$. Request the MACs of x and x', which we denote y, y', respectively. There are exactly λ keys, say $K_1, \ldots, K_\lambda$ such that $h_{K_i}(x) = y$ and $h_{K_i}(x') = y'$ for $1 \leq i \leq \lambda$. Choose $K \in \{K_1, \ldots, K_\lambda\}$ randomly. Now given any $x'' \neq x, x'$, the MAC $h_K(x'')$ is valid with probability at least $1/\lambda$, because the probability that K is the correct key is $1/\lambda$.

5.18 Compute Pd_0 and Pd_1 for the following authentication code, represented in matrix form:

key	1	2	3	4
1	1	1	2	3
2	1	2	3	1
3	2	1	3	1
4	2	3	1	2
5	3	2	1	3
6	3	3	2	1

Answer: $Pd_0 = 1/2$; the pair $(4, 1)$ will be valid with this probability.

Define a_{ij} to denote the probability of forging a MAC for a new message, given that the MAC of i is j (where $1 \leq i \leq 4, 1 \leq j \leq 3$). It is easy to verify the following:

i	j	a_{ij}	optimal forgery
1	1	1/2	(2,1)
1	2	1/2	(2,1)
1	3	1/2	(2,2)
2	1	1/2	(1,1)
2	2	1/2	(1,1)
2	3	1/2	(1,2)
3	1	1/2	(1,2)
3	2	1/2	(1,1)
3	3	1	(4,1)
4	1	2/3	(3,3)
4	2	1	(1,2)
4	3	1/2	(1,1)

Then

$$Pd_1 = \max\{\min\{a_{ij} : 1 \leq j \leq 3\} : 1 \leq i \leq 4\} = \frac{1}{2}.$$

5.19 Let p be an odd prime. For $a, b \in \mathbb{Z}_p$, define $f_{(a,b)} : \mathbb{Z}_p \rightarrow \mathbb{Z}_p$ by the rule

$$f_{(a,b)}(x) = (x+a)^2 + b \bmod p.$$

Prove that $(\mathbb{Z}_p, \mathbb{Z}_p, \mathbb{Z}_p \times \mathbb{Z}_p, \{f_{(a,b)} : a, b \in \mathbb{Z}_p\})$ is a strongly universal (p, p)-hash family.

Answer: Suppose that $x, x', y, y' \in \mathbb{Z}_p$, where $x \neq x'$. We will show that there is a unique key $(a, b) \in \mathbb{Z}_p \times \mathbb{Z}_p$ such that $(x+a)^2 + b \equiv y \pmod{p}$ and $(x'+a)^2 + b \equiv y' \pmod{p}$. Subtracting these two equations, we have

$$\begin{aligned}
(x+a)^2 - (x'+a)^2 &\equiv y - y' \pmod{p}\\
x^2 - (x')^2 + 2a(x - x') &\equiv y - y' \pmod{p}\\
x + x' + 2a &\equiv (y - y')(x - x')^{-1} \pmod{p}\\
a &= 2^{-1}((y-y')(x-x')^{-1} - (x + x')) \bmod p.
\end{aligned}$$

Now that a has been determined uniquely (modulo p), we can solve for b, because $b = y - (x+a)^2 \bmod p$.

5.20 Let $k \geq 1$ be an integer. An (N, M) hash family, $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$, is ***strongly** k**-universal*** provided that the following condition is satisfied for all choices of k distinct elements $x_1, x_2, \ldots, x_k \in \mathcal{X}$ and for all choices of k (not necessarily distinct) elements $y_1, \ldots, y_k \in \mathcal{Y}$:

$$|\{K \in \mathcal{K} : h_K(x_i) = y_i \text{ for } 1 \leq i \leq k\}| = \frac{|\mathcal{K}|}{M^k}.$$

(a) Prove that a strongly k-universal hash family is strongly ℓ-universal for all ℓ such that $1 \leq \ell \leq k$.

Answer: Without loss of generality, suppose that $\ell < k$. Suppose that $x_1, x_2, \ldots, x_\ell \in \mathcal{X}$ are distinct, and suppose that $y_1, \ldots, y_\ell \in \mathcal{Y}$. Let $x_{\ell+1}, \ldots, x_k \in \mathcal{X}$ be chosen such that $x_1, x_2, \ldots, x_k$ are all distinct. Now, for any $(k-\ell)$-tuple $(y_{\ell+1}, \ldots, y_k) \in \mathcal{Y}^{k-\ell}$, it holds that

$$|\{K \in \mathcal{K} : h_K(x_i) = y_i \text{ for } 1 \le i \le k\}| = \frac{|\mathcal{K}|}{M^k}.$$

Then it is clear that

$$\begin{aligned} &|\{K \in \mathcal{K} : h_K(x_i) = y_i \text{ for } 1 \le i \le \ell\}| \\ &= \sum_{(y_{\ell+1},\ldots,y_k) \in \mathcal{Y}^{k-\ell}} |\{K \in \mathcal{K} : h_K(x_i) = y_i \text{ for } 1 \le i \le k\}| \\ &= M^{k-\ell} \times \frac{|\mathcal{K}|}{M^k} \\ &= \frac{|\mathcal{K}|}{M^\ell}, \end{aligned}$$

as desired.

(b) Let p be prime and let $k \ge 1$ be an integer. For all k-tuples $(a_0, \ldots, a_{k-1}) \in (\mathbb{Z}_p)^k$, define $f_{(a_0,\ldots,a_{k-1})} : \mathbb{Z}_p \to \mathbb{Z}_p$ by the rule

$$f_{(a_0,\ldots,a_{k-1})}(x) = \sum_{i=0}^{k-1} a_i x^i \bmod p.$$

Prove that $\left(\mathbb{Z}_p, \mathbb{Z}_p, (\mathbb{Z}_p)^k, \{f_{(a_0,\ldots,a_{k-1})} : (a_0, \ldots, a_{k-1}) \in (\mathbb{Z}_p)^k\}\right)$ is a strongly k-universal (p, p) hash family.

HINT Use the fact that any degree d polynomial over a field has at most d roots.

Answer: Let $x_1, x_2, \ldots, x_k \in \mathcal{X}$ be k distinct elements. There are p^k possible keys, and p^k possible k-tuples $(y_1, \ldots, y_k) \in (\mathbb{Z}_p)^k$. We will show that, given any k-tuple $(y_1, \ldots, y_k) \in (\mathbb{Z}_p)^k$, there is exactly one key $(a_0, \ldots, a_{k-1}) \in (\mathbb{Z}_p)^k$ such that $f_{(a_0,\ldots,a_{k-1})}(x_i) = y_i$ for $1 \le i \le k$. Suppose this is not the case. Then there must exist two different keys $(a_0, \ldots, a_{k-1}) \ne (a'_0, \ldots, a'_{k-1})$ such that $f_{(a_0,\ldots,a_{k-1})}(x_i) = f_{(a'_0,\ldots,a'_{k-1})}(x_i) = y_i$ for $1 \le i \le k$. This implies that

$$\sum_{i=0}^{k-1} (a_i - a'_i) x^i \equiv 0 \pmod{p}$$

has at least k solutions in $\mathbb{Z}_p$, namely $x_1, x_2, \ldots, x_k$. In other words, the polynomial

$$g(x) = \sum_{i=0}^{k-1} (a_i - a'_i) x^i$$

has at least k distinct roots in the field $\mathbb{Z}_p$. The two k-tuples $(a_0, \ldots, a_{k-1})$ and $(a'_0, \ldots, a'_{k-1})$ are different, so the polynomial $g(x)$ is not the zero polynomial. But a non-zero polynomial of degree at most $k-1$ cannot have k distinct roots in a field, so we have a contradiction. This contradiction establishes the desired result.

Chapter 6

The RSA Cryptosystem and Factoring Integers

6.1 In Algorithm 6.1, prove that

$$\gcd(r_0, r_1) = \gcd(r_1, r_2) = \cdots = \gcd(r_{m-1}, r_m) = r_m$$

and, hence, $r_m = \gcd(a, b)$.

Answer: Suppose that $0 \leq i \leq m-2$. Then have that $r_i = q_{i+1}r_{i+1} + r_{i+2}$. If $d|r_i$ and $d|r_{i+1}$, then $d|r_{i+2}$. Also, if $d|r_{r+1}$ and $d|r_{i+2}$, then $d|r_i$. This proves that

$$\gcd(r_0, r_1) = \gcd(r_1, r_2) = \cdots = \gcd(r_{m-1}, r_m).$$

Now, using the equation $r_{m-1} = q_m r_m$, we have that $\gcd(r_{m-1}, r_m) = r_m$, and the result is proven.

6.2 Suppose that $a > b$ in Algorithm 6.1.

(a) Prove that $r_i \geq 2r_{i+2}$ for all i such that $0 \leq i \leq m-2$.

Answer: $r_i = q_{i+1}r_{i+1} + r_{i+2} \geq r_{i+1} + r_{i+2} > 2r_{i+2}$ for $0 \leq i \leq m-2$.

(b) Prove that m is $O(\log a)$.

Answer: Suppose first that m is even. Then $a = r_0 > 2r_2 > \cdots > 2^{m/2}r_m \geq 2^{m/2}$. Therefore $m < 2\log_2 a$. If m is odd, then it can be shown in a similar fashion that $m < 2\log_2 a + 1$. In either case, m is $O(\log a)$.

(c) Prove that m is $O(\log b)$.

Answer: Suppose first that m is odd. Then $b = r_1 > 2r_3 > \cdots > 2^{(m-1)/2}r_m \geq 2^{(m-1)/2}$. Therefore $m < 2\log_2 b + 1$. If m is even, then it can be shown in a similar fashion that $m < 2\log_2 a + 2$. In either case, m is $O(\log b)$.

6.3 Use the EXTENDED EUCLIDEAN ALGORITHM to compute the following multiplicative inverses:

(a) $17^{-1} \bmod 101$

Answer: $17^{-1} \bmod 101 = 6$.

(b) $357^{-1} \bmod 1234$

Answer: $357^{-1} \bmod 1234 = 1075$.

(c) $3125^{-1} \bmod 9987$.

Answer: $3125^{-1} \bmod 9987 = 1844$.

6.4 Compute $\gcd(57, 93)$, and find integers s and t such that $57s + 93t = \gcd(57, 93)$.

Answer: $\gcd(57, 93) = 3 = 18 \times 57 - 11 \times 93$.

6.5 Suppose $\chi : \mathbb{Z}_{105} \rightarrow \mathbb{Z}_3 \times \mathbb{Z}_5 \times \mathbb{Z}_7$ is defined as

$$\chi(x) = (x \bmod 3, x \bmod 5, x \bmod 7).$$

Give an explicit formula for the function χ^{-1}, and use it to compute $\chi^{-1}(2, 2, 3)$.

Answer: $\chi^{-1}(a_1, a_2, a_3) = 70a_1 + 21a_2 + 15a_3 \bmod 105$, and $\chi^{-1}(2, 2, 3) = 17$.

6.6 Solve the following system of congruences:

$$\begin{aligned} x &\equiv 12 \pmod{25} \\ x &\equiv 9 \pmod{26} \\ x &\equiv 23 \pmod{27}. \end{aligned}$$

Answer: $x = 14387$.

6.7 Solve the following system of congruences:

$$\begin{aligned} 13x &\equiv 4 \pmod{99} \\ 15x &\equiv 56 \pmod{101}. \end{aligned}$$

HINT First use the EXTENDED EUCLIDEAN ALGORITHM, and then apply the Chinese remainder theorem.

Answer: $x = 7471$.

6.8 Use Theorem 6.8 to find the smallest primitive element modulo 97.

Answer: $2^{48} \bmod 97 = 1$, $3^{48} \bmod 97 = 1$, $4^{48} \bmod 97 = 1$, $5^{48} \bmod 97 = 96$ and $5^{32} \bmod 97 = 35$. Therefore the smallest primitive root modulo 97 is 5.

6.9 How many primitive elements are there modulo 1041817?

Answer: There are 230400 primitive elements modulo 1041817.

6.10 Suppose that $p = 2q + 1$, where p and q are odd primes. Suppose further that $\alpha \in \mathbb{Z}_p{}^*$, $\alpha \not\equiv \pm 1 \pmod{p}$. Prove that α is a primitive element modulo p if and only if $\alpha^q \equiv -1 \pmod{p}$.

Answer: This follows immediately from Theorem 6.8, which (in this case) states that α is a primitive element modulo p if and only if $\alpha^q \not\equiv -1 \pmod{p}$ and $\alpha^2 \not\equiv 1 \pmod{p}$. But $\alpha^2 \equiv 1 \pmod{p}$ if and only if $\alpha \equiv \pm 1 \pmod{p}$. We have assumed that $\alpha \not\equiv \pm 1 \pmod{p}$, so the result follows.

6.11 Suppose that $n = pq$, where p and q are distinct odd primes and $ab \equiv 1 \pmod{(p-1)(q-1)}$. The RSA encryption operation is $e(x) = x^b \bmod n$ and the decryption operation is $d(y) = y^a \bmod n$. We proved that $d(e(x)) = x$ if $x \in \mathbb{Z}_n{}^*$. Prove that the same statement is true for any $x \in \mathbb{Z}_n$.

HINT Use the fact that $x_1 \equiv x_2 \pmod{pq}$ if and only if $x_1 \equiv x_2 \pmod{p}$ and $x_1 \equiv x_2 \pmod{q}$. This follows from the Chinese remainder theorem.

Answer: Suppose $x \not\equiv 0 \pmod{p}$. Then, for some integer $k > 0$, it holds that

$$x^{ab} = x^{1+k(p-1)(q-1)} \equiv x \times x^{k(p-1)(q-1)} \equiv x \pmod{p}.$$

If $x \equiv 0 \pmod{p}$, then $x^{ab} \equiv x \equiv 0 \pmod{p}$. Therefore $x^{ab} \equiv x \pmod{p}$ for any $x \in \mathbb{Z}_p$. Similarly, $x^{ab} \equiv x \pmod{q}$ for any $x \in \mathbb{Z}_q$. Now, applying the hint, $x^{ab} \equiv x \pmod{n}$ for any $x \in \mathbb{Z}_n$.

6.12 For $n = pq$, where p and q are distinct odd primes, define

$$\lambda(n) = \frac{(p-1)(q-1)}{\gcd(p-1, q-1)}.$$

Suppose that we modify the *RSA Cryptosystem* by requiring that $ab \equiv 1 \pmod{\lambda(n)}$.

(a) Prove that encryption and decryption are still inverse operations in this modified cryptosystem.

Answer: Denote $d = \gcd(p-1, q-1)$, $p - 1 = p'd$ and $q - 1 = q'd$. Then

$$\lambda(n) = p'q'd = (p-1)q' = p'(q-1).$$

We have that $ab \equiv 1 \pmod{\lambda(n)}$, so

$$ab = k\lambda(n) + 1 = k(p-1)q' + 1$$

for some positive integer k. Then

$$x^{ab} \equiv x^{k(p-1)q'+1} \pmod{p} \equiv x \pmod{p}.$$

Similarly,

$$x^{ab} \equiv x^{k(q-1)p'+1} \pmod{q} \equiv x \pmod{q}.$$

Since $x^{ab} \equiv x \pmod{p}$ and $x^{ab} \equiv \pmod{q}$, it follows immediately that $x^{ab} \equiv x \pmod{n}$.

(b) If $p = 37$, $q = 79$, and $b = 7$, compute a in this modified cryptosystem, as well as in the original *RSA Cryptosystem*.

Answer: $d = 6$, $\lambda(n) = 468$ and $\phi(n) = 2808$. $b^{-1} \bmod \lambda(n) = 67$ and $b^{-1} \bmod \phi(n) = 2407$.

6.13 Two samples of RSA ciphertext are presented in Tables 6.2 and 6.3. Your task is to decrypt them. The public parameters of the system are $n = 18923$ and $b = 1261$ (for Table 6.2) and $n = 31313$ and $b = 4913$ (for Table 6.3). This can be accomplished as follows. First, factor n (which is easy because it is so small). Then compute the exponent a from $\phi(n)$, and, finally, decrypt the ciphertext. Use the SQUARE-AND-MULTIPLY ALGORITHM to exponentiate modulo n.

In order to translate the plaintext back into ordinary English text, you need to know how alphabetic characters are "encoded" as elements in $\mathbb{Z}_n$. Each element of $\mathbb{Z}_n$ represents three alphabetic characters as in the following examples:

$$\begin{array}{lcrcr} DOG & \rightarrow & 3 \times 26^2 + 14 \times 26 + 6 & = & 2398 \\ CAT & \rightarrow & 2 \times 26^2 + 0 \times 26 + 19 & = & 1371 \\ ZZZ & \rightarrow & 25 \times 26^2 + 25 \times 26 + 25 & = & 17575. \end{array}$$

You will have to invert this process as the final step in your program.

The first plaintext was taken from *The Diary of Samuel Marchbanks*, by Robertson Davies, 1947, and the second was taken from *Lake Wobegon Days*, by Garrison Keillor, 1985.

Answer: The first plaintext was encrypted using the values $n = 18923 = 127 \times 149$ and $b = 1261$. Hence, $\phi(n) = 126 \times 148 = 18648$ and $a = 1261^{-1} \bmod 18648 = 5797$.

The first ciphertext element, $y = 12423$, is decrypted to $x = 5438$. We convert this to three letters as follows:

$$\begin{array}{rcl} 5438 \bmod 26 & = & 4 \\ (5438 - 4)/26 & = & 209 \\ 209 \bmod 26 & = & 1 \\ (209 - 1)/26 & = & 8 \\ 8 \bmod 26 & = & 8. \end{array}$$

Therefore, the triple $8, 1, 4$ corresponds to the three letters i, b, e.

The complete plaintext is as follows:

> I became involved in an argument about modern painting, a subject upon which I am spectacularly ill-informed. However, many of my friends can become heated and even violent on the subject, and I enjoy their wrangles in a modest way. I am an artist myself and I have some sympathy with the abstractionists, although I have gone beyond them in my own approach to art. I am a lumpist. Two or three decades ago it was quite fashionable to be a cubist and to draw everything in cubes. Then there was a revolt by the vorticists who drew everything in whirls. We now have the abstractionists who paint everything in a very abstracted manner, but my own small works done on my telephone pad are composed of carefully shaded, strangely shaped lumps with traces of cubism, vorticism, and abstractionism in them. For those who possess the seeing eye, as a lumpist, I stand alone.

TABLE 6.2: RSA ciphertext

12423	11524	7243	7459	14303	6127	10964	16399
9792	13629	14407	18817	18830	13556	3159	16647
5300	13951	81	8986	8007	13167	10022	17213
2264	961	17459	4101	2999	14569	17183	15827
12693	9553	18194	3830	2664	13998	12501	18873
12161	13071	16900	7233	8270	17086	9792	14266
13236	5300	13951	8850	12129	6091	18110	3332
15061	12347	7817	7946	11675	13924	13892	18031
2620	6276	8500	201	8850	11178	16477	10161
3533	13842	7537	12259	18110	44	2364	15570
3460	9886	8687	4481	11231	7547	11383	17910
12867	13203	5102	4742	5053	15407	2976	9330
12192	56	2471	15334	841	13995	17592	13297
2430	9741	11675	424	6686	738	13874	8168
7913	6246	14301	1144	9056	15967	7328	13203
796	195	9872	16979	15404	14130	9105	2001
9792	14251	1498	11296	1105	4502	16979	1105
56	4118	11302	5988	3363	15827	6928	4191
4277	10617	874	13211	11821	3090	18110	44
2364	15570	3460	9886	9988	3798	1158	9872
16979	15404	6127	9872	3652	14838	7437	2540
1367	2512	14407	5053	1521	297	10935	17137
2186	9433	13293	7555	13618	13000	6490	5310
18676	4782	11374	446	4165	11634	3846	14611
2364	6789	11634	4493	4063	4576	17955	7965
11748	14616	11453	17666	925	56	4118	18031
9522	14838	7437	3880	11476	8305	5102	2999
18628	14326	9175	9061	650	18110	8720	15404
2951	722	15334	841	15610	2443	11056	2186

TABLE 6.3: RSA ciphertext

6340	8309	14010	8936	27358	25023	16481	25809
23614	7135	24996	30590	27570	26486	30388	9395
27584	14999	4517	12146	29421	26439	1606	17881
25774	7647	23901	7372	25774	18436	12056	13547
7908	8635	2149	1908	22076	7372	8686	1304
4082	11803	5314	107	7359	22470	7372	22827
15698	30317	4685	14696	30388	8671	29956	15705
1417	26905	25809	28347	26277	7897	20240	21519
12437	1108	27106	18743	24144	10685	25234	30155
23005	8267	9917	7994	9694	2149	10042	27705
15930	29748	8635	23645	11738	24591	20240	27212
27486	9741	2149	29329	2149	5501	14015	30155
18154	22319	27705	20321	23254	13624	3249	5443
2149	16975	16087	14600	27705	19386	7325	26277
19554	23614	7553	4734	8091	23973	14015	107
3183	17347	25234	4595	21498	6360	19837	8463
6000	31280	29413	2066	369	23204	8425	7792
25973	4477	30989					

The second plaintext was encrypted using the values $n = 31313 = 173 \times 181$ and $b = 4913$. Hence, $\phi(n) = 172 \times 180 = 30960$ and $a = 4913^{-1} \bmod 30960 = 6497$.

The second plaintext is as follows:

> Lake Wobegon is mostly poor sandy soil, and every spring the earth heaves up a new crop of rocks. Piles of rocks ten feet high in the corners of fields, picked by generations of us, monuments to our industry. Our ancestors chose the place, tired from their long journey, sad for having left the motherland behind, and this place reminded them of there, so they settled here, forgetting that they had left there because the land wasn't so good. So the new life turned out to be a lot like the old, except the winters are worse.

6.14 A common way to speed up RSA decryption incorporates the Chinese remainder theorem, as follows. Suppose that $d_K(y) = y^d \bmod n$ and $n = pq$. Define $d_p = d \bmod (p-1)$ and $d_q = d \bmod (q-1)$; and let $M_p = q^{-1} \bmod p$ and $M_q = p^{-1} \bmod q$. Then consider the following algorithm:

```
Algorithm 6.15: CRT-OPTIMIZED RSA DECRYPTION($n, d_p, d_q, M_p, M_q, y$)

  $x_p \leftarrow y^{d_p} \bmod p$
  $x_q \leftarrow y^{d_q} \bmod q$
  $x \leftarrow M_p q x_p + M_q p x_q \bmod n$
  return ($x$)
```

Algorithm 6.15 replaces an exponentiation modulo n by modular exponentiations modulo p and q. If p and q are ℓ-bit integers and exponentiation modulo an ℓ-bit integer takes time $c\ell^3$, then the time to perform the required exponentiation(s) is reduced from $c(2\ell)^3$ to $2c\ell^3$, a savings of 75%. The final step, involving the Chinese remainder theorem, requires time $O(\ell^2)$ if d_p, d_q, M_p, and M_q have been pre-computed.

(a) Prove that the value x returned by Algorithm 6.15 is, in fact, $y^d \bmod n$.

Answer:

$$\begin{aligned} x &\equiv M_p q x_p \pmod p \\ &\equiv q^{-1} q x_p \pmod p \\ &\equiv x_p \pmod p \\ &\equiv y^{d_p} \pmod p \\ &\equiv y^d \pmod p, \end{aligned}$$

because $d_p \equiv d \pmod{p-1}$. Similarly, $x \equiv y^d \pmod q$. Therefore $x \equiv y^d \pmod n$.

(b) Given that $p = 1511$, $q = 2003$, and $d = 1234577$, compute d_p, d_q, M_p, and M_q.

Answer: We have that $d_p = 907$, $d_q = 1345$, $M_p = 777$ and $M_q = 973$.

(c) Given the above values of p, q, and d, decrypt the ciphertext $y = 152702$ using Algorithm 6.15.

Answer: We obtain $x_p = 242$, $x_q = 1087$ and $x = 1443247$.

6.15 Prove that the *RSA Cryptosystem* is insecure against a chosen ciphertext attack. In particular, given a ciphertext y, describe how to choose a ciphertext $\hat{y} \neq y$, such that knowledge of the plaintext $\hat{x} = d_K(\hat{y})$ allows $x = d_K(y)$ to be computed.

HINT Use the multiplicative property of the *RSA Cryptosystem*, i.e., that

$$e_K(x_1)e_K(x_2) \bmod n = e_K(x_1 x_2 \bmod n).$$

Answer: Choose a random x_0 and compute $y_0 = e_K(x_0)$. Define $\hat{y} = y_0 y \bmod n$, and obtain the decryption $\hat{x} = d_K(\hat{y})$. Then compute $x = \hat{x}{x_0}^{-1} \bmod n$.

6.16 This exercise exhibits what is called a ***protocol failure***. It provides an example where ciphertext can be decrypted by an opponent, without determining the key, if a cryptosystem is used in a careless way. The moral is that it is not sufficient to use a "secure" cryptosystem in order to guarantee "secure" communication.

Suppose Bob has an *RSA Cryptosystem* with a large modulus n for which the factorization cannot be found in a reasonable amount of time. Suppose Alice sends a message to Bob by representing each alphabetic character as an integer between 0 and 25 (i.e., $A \leftrightarrow 0$, $B \leftrightarrow 1$, etc.), and then encrypting each residue modulo 26 as a separate plaintext character.

(a) Describe how Oscar can easily decrypt a message that is encrypted in this way.

Answer: Oscar can encrypt each of the 26 possible plaintexts, and record the values of the corresponding 26 ciphertexts in a table. Then any ciphertext string can be decrypted by referring to the precomputed table.

(b) Illustrate this attack by decrypting the following ciphertext (which was encrypted using an *RSA Cryptosystem* with $n = 18721$ and $b = 25$) without factoring the modulus:

$$365, 0, 4845, 14930, 2608, 2608, 0.$$

Answer: The plaintext is *vanilla*.

6.17 This exercise illustrates another example of a protocol failure (due to Simmons) involving the *RSA Cryptosystem*; it is called the ***common modulus protocol failure***. Suppose Bob has an *RSA Cryptosystem* with modulus n and encryption exponent b_1, and Charlie has an *RSA Cryptosystem* with (the same) modulus n and encryption exponent b_2. Suppose also that $\gcd(b_1, b_2) = 1$. Now, consider the situation that arises if Alice encrypts the same plaintext x to send to both Bob and Charlie. Thus, she computes $y_1 = x^{b_1} \bmod n$ and $y_2 = x^{b_2} \bmod n$, and then she sends y_1 to Bob and y_2 to Charlie. Suppose Oscar intercepts y_1 and y_2, and performs the computations indicated in Algorithm 6.16.

```
**Algorithm 6.16:** RSA COMMON MODULUS DECRYPTION($n, b_1, b_2, y_1, y_2$)

$c_1 \leftarrow {b_1}^{-1} \bmod b_2$
$c_2 \leftarrow (c_1 b_1 - 1)/b_2$
$x_1 \leftarrow {y_1}^{c_1} ({y_2}^{c_2})^{-1} \bmod n$
**return** $(x_1)$
```

(a) Prove that the value x_1 computed in Algorithm 6.16 is in fact Alice's plaintext, x. Thus, Oscar can decrypt the message Alice sent, even though the cryptosystem may be "secure."

Answer: We use the fact that $b_2c_2 = b_1c_1 - 1$. Working in $\mathbb{Z}_n$, we have that

$$y_1{}^{c_1}(y_2{}^{c_2})^{-1} = x^{b_1c_1}(x^{b_2c_2})^{-1} = x^{b_1c_1 - b_1c_2} = x.$$

(b) Illustrate the attack by computing x by this method if $n = 18721$, $b_1 = 43$, $b_2 = 7717$, $y_1 = 12677$, and $y_2 = 14702$.

Answer: $c_1 = 2692$, $c_2 = 15$, and $x = 15001$.

6.18 We give yet another protocol failure involving the *RSA Cryptosystem*. Suppose that three users in a network, say Bob, Bart, and Bert, all have public encryption exponents $b = 3$. Let their moduli be denoted by n_1, n_2, n_3, and assume that n_1, n_2, and n_3, are pairwise relatively prime. Now suppose Alice encrypts the same plaintext x to send to Bob, Bart, and Bert. That is, Alice computes $y_i = x^3 \bmod n_i$, $1 \leq i \leq 3$. Describe how Oscar can compute x, given y_1, y_2, and y_3, without factoring any of the moduli.

Answer: Consider the following system of three congruences:

$$\begin{aligned} z &\equiv y_1 \bmod n_1 \\ z &\equiv y_2 \bmod n_2 \\ z &\equiv y_3 \bmod n_3. \end{aligned}$$

Using the Chinese remainder theorem, it is easy to find the unique solution z to this system such that $0 \leq z < n_1n_2n_3$. However, the integer x^3 is a solution to the same system, and $0 \leq x^3 < n_1n_2n_3$. Since the system has a unique solution modulo $n_1n_2n_3$, it must be the case that $x^3 = z$. Therefore $x = z^{1/3}$.

6.19 A plaintext x is said to be a ***fixed plaintext*** if $e_K(x) = x$. Show that, for the *RSA Cryptosystem*, the number of fixed plaintexts $x \in \mathbb{Z}_n{}^*$ is equal to

$$\gcd(b-1, p-1) \times \gcd(b-1, q-1).$$

HINT Consider the following system of two congruences:

$$\begin{aligned} e_K(x) &\equiv x \pmod{p}, \\ e_K(x) &\equiv x \pmod{q}. \end{aligned}$$

Answer: $e_K(x) = x$ if and only if

$$\begin{aligned} x^b &\equiv x \pmod{p} \quad \text{and} \\ x^b &\equiv x \pmod{q}. \end{aligned}$$

First, we determine the number of solutions $x \in \mathbb{Z}_p{}^*$ to the congruence $x^b \equiv$

$x \pmod p$. Let $\alpha \in \mathbb{Z}_p{}^*$ be a primitive element. Then any $x \in \mathbb{Z}_p{}^*$ can be written uniquely in the form $x = \alpha^i \bmod p$, where $0 \le i \le p-2$. Further, $x^b \equiv x \pmod p$ if and only if $ib \equiv i \pmod{p-1}$, or $i(b-1) \equiv 0 \pmod{p-1}$. This congruence has $\gcd(b-1, p-1)$ solutions, namely

$$i = \frac{j(p-1)}{\gcd(b-1,p-1)},$$

$0 \le j < \gcd(b-1,p-1)$. Therefore $x^b \equiv x \pmod p$ has $\gcd(b-1,p-1)$ solutions $x \in \mathbb{Z}_p{}^*$.

Similarly, $x^b \equiv x \pmod q$ has $\gcd(b-1,q-1)$ solutions $x \in \mathbb{Z}_q{}^*$. Using the Chinese reaminder theorem, it is clear that the number of solutions $x \in \mathbb{Z}_n{}^*$ to the system

$$\begin{aligned} x^b &\equiv x \pmod p, \\ x^b &\equiv x \pmod q. \end{aligned}$$

is exactly $\gcd(b-1,p-1) \times \gcd(b-1,q-1)$.

6.20 Suppose **A** is a deterministic algorithm that is given as input an RSA modulus n, an encryption exponent b, and a ciphertext y. **A** will either decrypt y or return no answer. Supposing that there are $\epsilon(n-1)$ nonzero ciphertexts which **A** is able to decrypt, show how to use **A** as an oracle in a Las Vegas decryption algorithm having success probability ϵ.

Answer: Suppose we are given n, b and a ciphertext $y \in \mathbb{Z}_n$. If $y = 0$, then its decryption is $x = 0$. If $\gcd(y,n) > 1$, then it is possible to factor n, in which case y can easily be decrypted. Therefore we suppose that $\gcd(y,n) = 1$.

Now, the algorithm **B** should choose a random $x_1 \in \mathbb{Z}_n$, $x_1 \ne 0$; compute $y_1 = x_1{}^b \bmod n$; and compute $y' = yy_1 \bmod n$. Then call the algorithm **A** with input n, b, y'. If **A** returns a decryption of y', say x', then $x = x'/x_1 \bmod n$.

We need to analyze the success probability of **B**. If $\gcd(y,n) > 1$, then **B** has success probability equal to 1. If $\gcd(y,n) = 1$, then y' is a random non-zero element of $\mathbb{Z}_n$, so the success probability is $\epsilon(n-1)/(n-1) = \epsilon$. Therefore, for any input y, the success probability of **B** is greater than ϵ.

6.21 Write a program to evaluate Jacobi symbols using the four properties presented in Section 6.4. The program should not do any factoring, other than dividing out powers of two. Test your program by computing the following Jacobi symbols:

$$\left(\frac{610}{987}\right), \left(\frac{20964}{1987}\right), \left(\frac{1234567}{11111111}\right).$$

Answer: The three Jacobi symbols are -1, 1 and -1, respectively.

6.22 For $n = 837, 851$, and 1189, find the number of bases b such that n is an Euler pseudo-prime to the base b.

Answer: The number of bases b is 10, 2 and 8 respectively.

6.23 The purpose of this question is to prove that the error probability of the Solovay-Strassen primality test is at most 1/2. Let $\mathbb{Z}_n{}^*$ denote the group of units modulo n. Define

$$G(n) = \left\{ a : a \in \mathbb{Z}_n{}^*, \left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \pmod{n} \right\}.$$

(a) Prove that $G(n)$ is a subgroup of $\mathbb{Z}_n{}^*$. Hence, by Lagrange's theorem, if $G(n) \neq \mathbb{Z}_n{}^*$, then

$$|G(n)| \leq \frac{|\mathbb{Z}_n{}^*|}{2} \leq \frac{n-1}{2}.$$

Answer: Suppose that $a, b \in G(n)$. Then

$$\left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \pmod{n}$$

and

$$\left(\frac{b}{n}\right) \equiv b^{(n-1)/2} \pmod{n}.$$

It follows from the multiplicative rule of Jacobi symbols (page 206, property 3) that

$$\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right) \equiv a^{(n-1)/2} b^{(n-1)/2} \pmod{n} \equiv (ab)^{(n-1)/2} \pmod{n}.$$

Therefore $ab \in G(n)$. Since $G(n)$ is a subset of a multiplicative finite group that is closed under the operation of multiplication, it must be a subgroup.

(b) Suppose $n = p^k q$, where p and q are odd, p is prime, $k \geq 2$, and $\gcd(p, q) = 1$. Let $a = 1 + p^{k-1}q$. Prove that

$$\left(\frac{a}{n}\right) \not\equiv a^{(n-1)/2} \pmod{n}.$$

HINT Use the binomial theorem to compute $a^{(n-1)/2}$.

Answer: We have that

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p}\right)^k \left(\frac{a}{q}\right) = 1.$$

On the other hand,

$$\begin{aligned} a^{(n-1)/2} &= \sum_{i=0}^{(n-1)/2} \binom{(n-1)/2}{i} (p^{k-1}q)^i \\ &\equiv 1 + \frac{n-1}{2} p^{k-1} q \pmod{n}. \end{aligned}$$

Suppose that $\left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \pmod{n}$. Then

$$\frac{n-1}{2} p^{k-1} q \equiv 0 \pmod{n}.$$

This implies that

$$p^k q | \frac{n-1}{2} p^{k-1} q,$$

$$p | \frac{n-1}{2},$$

and hence $n \equiv 1 \pmod{p}$. But $n \equiv 0 \pmod{p}$, so we have a contradiction.

(c) Suppose $n = p_1 \dots p_s$, where the p_i's are distinct odd primes. Suppose $a \equiv u \pmod{p_1}$ and $a \equiv 1 \pmod{p_2 p_3 \cdots p_s}$, where u is a quadratic non-residue modulo p_1 (note that such an a exists by the Chinese remainder theorem). Prove that

$$\left(\frac{a}{n}\right) \equiv -1 \pmod{n},$$

but

$$a^{(n-1)/2} \equiv 1 \pmod{p_2 p_3 \cdots p_s},$$

so

$$a^{(n-1)/2} \not\equiv -1 \pmod{n}.$$

Answer: On one hand, we have

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right) \left(\frac{a}{p_2 p_3 \dots p_s}\right) = (-1)(1) = -1.$$

If $a^{(n-1)/2} \equiv -1 \pmod{n}$, then $a^{(n-1)/2} \equiv -1 \pmod{p_2 p_3 \dots p_s}$. But $a^{(n-1)/2} \equiv 1 \pmod{p_2 p_3 \dots p_s}$, so we conclude that $a^{(n-1)/2} \not\equiv -1 \pmod{n}$, and hence

$$\left(\frac{a}{n}\right) \not\equiv a^{(n-1)/2} \pmod{n}.$$

(d) If n is odd and composite, prove that $|G(n)| \leq (n-1)/2$.

Answer: This follows immediately from the results proven in parts (a), (b) and (c). If n is the product of distinct primes, then (c) shows that $G(n) \neq \mathbb{Z}_n{}^*$. Otherwise, (b) establishes the same result. Then the result shown in (a) can be applied.

(e) Summarize the above: prove that the error probability of the Solovay-Strassen primality test is at most $1/2$.

Answer: Suppose n is composite. If $\gcd(a, n) \neq 1$, then the Solovay-Strassen test returns the correct answer. If $\gcd(a, n) = 1$, then the

Solovay-Strassen test returns the wrong answer if and only if $a \in G(n)$. We proved in part (d) that $|G(n)| \leq (n-1)/2$, so the probability of a wrong answer is at most

$$\frac{n-1-|\mathbb{Z}_n{}^*|}{n-1} \times 0 + \frac{|\mathbb{Z}_n{}^*|}{n-1} \times \frac{|G(n)|}{|\mathbb{Z}_n{}^*|} = \frac{|G(n)|}{n-1} \leq \frac{1}{2}.$$

6.24 Suppose we have a Las Vegas algorithm with failure probability ϵ.

(a) Prove that the probability of first achieving success on the nth trial is $p_n = \epsilon^{n-1}(1-\epsilon)$.

Answer: The probability of $n-1$ failures followed by a success is

$$\epsilon^{n-1}(1-\epsilon).$$

(b) The average (expected) number of trials to achieve success is

$$\sum_{n=1}^{\infty}(n \times p_n).$$

Show that this average is equal to $1/(1-\epsilon)$.

Answer:

$$\begin{aligned}
\sum_{n=1}^{\infty}(n \times p_n) &= \sum_{n=1}^{\infty} n\epsilon^{n-1}(1-\epsilon) \\
&= (1-\epsilon)\sum_{n=1}^{\infty}\sum_{j=1}^{n}\epsilon^{n-1} \\
&= (1-\epsilon)\sum_{j=1}^{\infty}\sum_{n=j}^{\infty}\epsilon^{n-1} \\
&= (1-\epsilon)\sum_{j=1}^{\infty}\frac{\epsilon^{j-1}}{1-\epsilon} \\
&= \sum_{j=1}^{\infty}\epsilon^{j-1} \\
&= \frac{1}{1-\epsilon}.
\end{aligned}$$

(c) Let δ be a positive real number less than 1. Show that the number of iterations required in order to reduce the probability of failure to at most δ is

$$\left\lceil \frac{\log_2 \delta}{\log_2 \epsilon} \right\rceil.$$

Answer: The probability of success after at most m trials is

$$\sum_{j=1}^{m} p_j = 1 - \epsilon^m.$$

Therefore, the probability of failure after m trials is ϵ^m. We want to have $\epsilon^m \leq \delta$, which is equivalent to $m \log_2 \epsilon \leq \log_2 \delta$. Because $\log_2 \epsilon < 0$, this is the same as

$$m \geq \frac{\log_2 \delta}{\log_2 \epsilon}.$$

Since m is an integer, we require

$$m \geq \left\lceil \frac{\log_2 \delta}{\log_2 \epsilon} \right\rceil.$$

6.25 Suppose throughout this question that p is an odd prime and $\gcd(a, p) = 1$.

(a) Suppose that $i \geq 2$ and $b^2 \equiv a \pmod{p^{i-1}}$. Prove that there is a unique $x \in \mathbb{Z}_{p^i}$ such that $x^2 \equiv a \pmod{p^i}$ and $x \equiv b \pmod{p^{i-1}}$. Describe how this x can be computed efficiently.

Answer: Since $b^2 \equiv a \pmod{p^{i-1}}$, we have that $b^2 - a = Kp^{i-1}$ for some integer K. Since $x \equiv b \pmod{p^{i-1}}$, we can write $x = Lp^{i-1} + b$ for some integer L. Now we compute x^2:

$$\begin{aligned} x^2 &= (Lp^{i-1} + b)^2 = L^2 p^{2i-2} + 2bLp^{i-1} + b^2 \\ &= L^2 p^{2i-2} + 2bLp^{i-1} + Kp^{i-1} + a. \end{aligned}$$

Therefore $x^2 \equiv p^{i-1}(2bL + K) + a \pmod{p^i}$, so $x^2 \equiv a \pmod{p^i}$ if and only if $2bL + K \equiv 0 \pmod{p}$. This is true if and only if $L \equiv -K(2b)^{-1} \pmod{p}$.

(b) Illustrate your method in the following situation: starting with the congruence $6^2 \equiv 17 \pmod{19}$, find square roots of 17 modulo 19^2 and modulo 19^3.

Answer: $6^2 - 17 = 1 \times 19$, so $K = 1$. Then

$$L = -1 \times 12^{-1} \bmod 19 = -1 \times 8 \bmod 19 = 11,$$

so

$$x \equiv 11 \times 19 + 6 \equiv 215 \pmod{361}.$$

Next, $215^2 - 17 = 128 \times 361$, so $K = 128$. Then

$$L = -128 \times 8 \bmod 19 = 2$$

(there is no need to recalculate $(2b)^{-1} \bmod p$), so

$$x \equiv 2 \times 361 + 215 \pmod{19^3} \equiv 937 \pmod{19^3}.$$

(c) For all $i \geq 1$, prove that the number of solutions to the congruence $x^2 \equiv a \pmod{p^i}$ is either 0 or 2.

Answer: The proof is by induction on i. For $i = 1$, the congruence

$x^2 \equiv a \pmod{p}$ has no solutions or two solutions in $\mathbb{Z}_p$, depending on the value of the Legendre symbol $\left(\frac{a}{p}\right)$. The result proved in part (a) establishes that the number of solutions modulo p^i is the same as the number of solutions modulo p^{i-1}, for all $i \geq 2$, so the result follows by induction.

6.26 Using various choices for the bound, B, attempt to factor 262063 and 9420457 using the $p-1$ method. How big does B have to be in each case to be successful?

Answer: When $n = 262063$, the factor 521 is computed when $B = 13$, but not when $B = 12$. (Note that $262063 = 521 \times 503$ and $520 = 2^3 \times 5 \times 13$. This illustrates why $B = 13$ is sufficient to find the factor 521.)

When $n = 9420457$, the factor 2351 is computed when $B = 47$, but not when $B = 46$. (Note that $9420457 = 2351 \times 4007$ and $2350 = 2 \times 5^2 \times 47$. This illustrates why $B = 47$ is sufficient to find the factor 2351.)

6.27 Factor 262063, 9420457, and 181937053 using the POLLARD RHO ALGORITHM, if the function f is defined to be $f(x) = x^2 + 1$. How many iterations are needed to factor each of these three integers?

Answer: When $n = 262063$, we get

$$\begin{aligned}&x_{35} = 225384,\\&x_{70} = 94604,\\&\gcd(225384 - 94604, 262063) = 503, \text{and}\\&n = 503 \times 521.\end{aligned}$$

When $n = 9420457$, we get

$$\begin{aligned}&x_{50} = 4559325,\\&x_{100} = 6376648,\\&\gcd(4559325 - 6376648, 262063) = 2351, \text{and}\\&n = 2351 \times 4007.\end{aligned}$$

When $n = 181937053$, we get

$$\begin{aligned}&x_{165} = 2452153,\\&x_{330} = 73737576,\\&\gcd(2452153 - 73737576, 181937053) = 12391, \text{and}\\&n = 12391 \times 14683.\end{aligned}$$

6.28 Suppose we want to factor the integer $n = 256961$ using the RANDOM SQUARES ALGORITHM. Using the factor base

$$\{-1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31\},$$

test the integers $z^2 \bmod n$ for $z = 500, 501, \ldots$, until a congruence of the form

$x^2 \equiv y^2 \pmod{n}$ is obtained and the factorization of n is found.

Answer: The following factorizations over the factor base are obtained:

$$\begin{aligned}
503^2 \bmod n &= (-1) \times 2^4 \times 13 \times 19\\
504^2 \bmod n &= (-1) \times 5 \times 19 \times 31\\
505^2 \bmod n &= (-1) \times 2^4 \times 11^2\\
507^2 \bmod n &= 2^3 \times 11\\
511^2 \bmod n &= 2^6 \times 5 \times 13\\
516^2 \bmod n &= 5 \times 11 \times 13^2\\
519^2 \bmod n &= 2^4 \times 5^2 \times 31
\end{aligned}$$

The first dependence relation that is obtained is:

$$(503 \times 504 \times 511 \times 519)^2 \equiv (2^7 \times 5^2 \times 13 \times 19 \times 31)^2 \pmod{n}.$$

The expressions inside the parentheses simplify to give

$$75319^2 \equiv 91105^2 \pmod{n}.$$

then we compute $\gcd(75319 + 91105, n) = 293$ and $\gcd(75319 - 91105, n) = 877$, so $n = 293 \times 877$.

6.29 In the RANDOM SQUARES ALGORITHM, we need to test a positive integer $w \leq n-1$ to see if it factors completely over the factor base $\mathcal{B} = \{p_1, \dots, p_B\}$ consisting of the B smallest prime numbers. Recall that $p_B = m \approx 2^s$ and $n \approx 2^r$.

(a) Prove that this can be done using at most $B + r$ divisions of an integer having at most r bits by an integer having at most s bits.

Answer: Consider the following algorithm:

```
Algorithm: TRIALDIVIDE ($w, p_1, \dots, p_B$)

$w_0 \leftarrow w$
for $i \leftarrow 1$ to $B$
  do
    $e_i \leftarrow 0$
    while $p_i | w_0$
      do
        $w_0 \leftarrow w_0 / p_i$
        $e_i \leftarrow e_i + 1$
return $(e_1, \dots, e_B, w_0)$
```

At the end of TRIALDIVIDE, we have that

$$w = {p_1}^{e_1} \times \cdots \times {p_B}^{e_B} \times w_0,$$

where w_0 is not divisible by any of $p_1, \ldots, p_B$. The number of divisions performed in the algorithm is

$$B + e_1 + \cdots + e_B.$$

We have that

$$n > w \geq {p_1}^{e_1} \times \cdots \times {p_B}^{e_B} > 2^{e_1 + \cdots + e_B},$$

so

$$e_1 + \cdots + e_B < \log_2 n \approx r.$$

Therefore the number of divisions is (approximately) at most $B + r$.

(b) Assuming that $r < m$, prove that the complexity of this test is $O(rsm)$.

Answer: Each division takes time $O(rs)$ (see page 199). Therefore the total time is $O(rs(B + r))$. However, $B < m$ and we are assuming that $r < m$. Therefore $B + r$ is $O(m)$ and the total time is $O(rsm)$.

6.30 In this exercise, we show that parameter generation for the *RSA Cryptosystem* should take care to ensure that $q - p$ is not too small, where $n = pq$ and $q > p$.

(a) Suppose that $q - p = 2d > 0$, and $n = pq$. Prove that $n + d^2$ is a perfect square.

Answer: $n + d^2 = pq + d^2 = p(p + 2d) + d^2 = (p + d)^2$.

(b) Given an integer n that is the product of two odd primes, and given a small positive integer d such that $n + d^2$ is a perfect square, show how this information can be used to factor n.

Answer: Suppose $n + d^2 = s^2$. Then $n = (s - d)(s + d)$.

(c) Use this technique to factor $n = 2189284635403183$.

Answer: $n + 9^2 = 46789792^2$, and so

$$n = (46789792 - 9)(46789792 + 9) = 46789783 \times 46789801.$$

6.31 Suppose Bob has carelessly revealed his decryption exponent to be $a = 14039$ in an *RSA Cryptosystem* with public key $n = 36581$ and $b = 4679$. Implement the randomized algorithm to factor n given this information. Test your algorithm with the "random" choices $w = 9983$ and $w = 13461$. Show all computations.

Answer: We have that $ab - 1 = 65688480 = 2^5 \times 2052765$, so $r = 2052765$.

When $w = 9983$, we get $v = 35039$ and $v^2 \equiv 1 \pmod{n}$, so the algorithm fails.

When $w = 13461$, we get $v = 11747$, $v^2 \equiv 8477 \pmod{n}$, $v^4 \equiv 14445 \pmod{n}$ and $v^8 \equiv 1 \pmod{n}$. Hence the algorithm succeeds:

$$\gcd(14445 + 1, n) = 233$$

is a factor of n.

6.32 Compute the continued fraction expansion of 144/89.

Answer: The continued fraction expansion is $[1,1,1,1,1,1,1,1,1,2]$.

6.33 If $q_1, \ldots, q_m$ is the sequence of quotients obtained in applying the EUCLIDEAN ALGORITHM with input r_0, r_1, prove that the continued fraction $[q_1, \ldots, q_m] = r_0/r_1$.

Answer: From the Euclidean Algorithm, we have

$$\begin{array}{rcl} r_0 & = & q_1 r_1 + r_2 \\ r_1 & = & q_2 r_2 + r_3 \\ \vdots & \vdots & \vdots \\ r_{m-1} & = & q_m r_m. \end{array}$$

We will prove, by reverse induction on j, that $[q_j, \ldots, q_m] = r_{j-1}/r_j$ for $1 \leq j \leq m$. The base case is $j = m$, where $[q_m] = q_m = r_{m-1}/r_m$. Assume the formula is true for $j = i+1$, and then prove it is true for $j = i$. From the Euclidean Algorithm, we have

$$r_{j-1} = q_j r_j + r_{j+1},$$

so

$$\begin{array}{rcll} r_{j-1}/r_j & = & q_j + r_{j+1}/r_j \\ & = & q_j + 1/(r_j/r_{j+1}) \\ & = & q_j + 1/[q_{j+1}, \ldots, q_m] & \text{(by induction)} \\ & = & [q_j, q_{j+1}, \ldots, q_m]. \end{array}$$

By induction, the result is true for $1 \leq j \leq m$. Setting $j = 1$, we see that $r_0/r_1 = [q_1, ..., q_m]$, as desired.

6.34 Suppose that $n = 317940011$ and $b = 77537081$ in the *RSA Cryptosystem*. Using WIENER'S ALGORITHM, attempt to factor n.

Answer: The continued fraction expansion of b/n is

$$[0,4,9,1,19,1,1,15,3,2,3,71,3,2].$$

The first few convergents are 0, 1/4, 9/37, 10/41, etc. If we let $c = 10$ and $d = 41$, then we obtain the quadratic equation $x^2 - 37980x + 317940011 = 0$. This equation has roots 12457 and 25523, which are the factors of n.

6.35 Consider the modification of the *Rabin Cryptosystem* in which $e_K(x) = x(x+B) \bmod n$, where $B \in \mathbb{Z}_n$ is part of the public key. Supposing that $p = 199$, $q = 211$, $n = pq$, and $B = 1357$, perform the following computations.

(a) Compute the encryption $y = e_K(32767)$.

Answer: $32767(32767 + 1357) \bmod 41989 = 16027$.

(b) Determine the four possible decryptions of this given ciphertext y.

Answer: $B/2 \bmod n = 21673$ and $B^2/4 \bmod n = 29975$. The decryptions are $x = \sqrt{4013} - 21673 \bmod n$. To find one square root of 4013 modulo n, compute $4013^{(p+1)/4} \bmod p = 86$ and $4013^{(q+1)/4} \bmod q = 209$. Then use the Chinese remainder theorem to solve the system $c \equiv 86 \pmod{p}$, $c \equiv 209 \pmod{q}$, yielding $c \equiv 29538 \pmod{n}$. A second square root is obtained by using the Chinese remainder theorem to solve the system $c \equiv 86 \pmod{p}$, $c \equiv -209 \pmod{q}$, yielding $c \equiv 1479 \pmod{n}$. The other two square roots are the negatives (modulo n) of the first two square roots. Therefore we obtain four square roots, namely 29538, 12451, 1479 and 40510. The four decryptions of y are $x = 7865$, 32767, 21795 and 18837.

6.36 The security of the *Rabin Cryptosystem* is established by showing that if $x^2 \equiv y^2 \pmod{n}$, and $x \not\equiv \pm y \pmod{n}$, then $\gcd(x - y, n) = p$ or q, where n is the product of two primes p and q. In this question, we consider a variation where n is the product of three primes.

In what follows, you can assume that $x, y, z \in {\mathbb{Z}_n}^*$.

(a) If n is the product of three primes, and we are given x and y such that that $x^2 \equiv y^2 \pmod{n}$, and $x \not\equiv \pm y \pmod{n}$, show that it is easy to compute at least one prime factor of n using a gcd computation.

Answer: Suppose that $n = pqr$, where p, q, and r are distinct odd primes. Since $x^2 \equiv y^2 \pmod{n}$, we have that $pqr | (x - y)(x + y)$. We cannot have $pqr | (x - y)$ or $pqr | (x + y)$ because $x \not\equiv \pm y \pmod{n}$. Hence $x - y$ is divisible by one of p, q, r and $x + y$ is divisible by the other two; or $x - y$ is divisible by two of p, q, r and $x + y$ is divisible by the other one.

Now, suppose we compute $d_1 = gcd(x - y, n)$ and $d_2 = n/d_1$. It follows that exactly one of d_1 and d_2 is a prime divisor of n. Therefore it suffices to test d_1 for primality, using a fast primality test, to determine which of d_1, d_2 is prime.

(b) Suppose n is the product of three primes, and we are given x, y, and z such that that $x^2 \equiv y^2 \equiv z^2 \pmod{n}$, $x \not\equiv \pm y \pmod{n}$, $x \not\equiv \pm z \pmod{n}$, and $y \not\equiv \pm z \pmod{n}$. Prove that we can compute all three prime factors of n by means of gcd computations.

HINT You can use the Chinese remainder theorem to prove this.

Answer: Suppose that $n = pqr$, where p, q and r are distinct odd primes. From part (a), we know that there is one prime divisor of n, say p, such that one of the following two cases holds:

case 1 $\gcd(x - y, n) = p$

case 2 $\gcd(x + y, n) = p$.

In either case, we can compute p by using the gcd computation $\gcd(x - y, n)$, as described in part (a).

We can also compute a prime divisor p' of n by using the gcd computation $\gcd(x - z, n)$. If $p \neq p'$, then we have two prime divisors of n, and the third one can be computed as $n/(p\,p')$. Therefore we assume that $p = p'$ (this will eventually lead to a contradiction).

Let's first consider case 1. This splits into two subcases:

case 1a $\gcd(x - z, n) = p$

case 1b $\gcd(x + z, n) = p$.

In case 1a, we have that $p|(x - y)$, $qr|(x + y)$, $p|(x - z)$, $qr|(x + z)$. Then it is easy to see that $y \equiv z \pmod{p}$ and $y \equiv z \pmod{qr}$. From the Chinese remainder theorem, we see that $y \equiv z \pmod{n}$, a contradiction.

In case 1b, we have that $p|(x - y)$, $qr|(x + y)$, $p|(x + z)$, $qr|(x - z)$. Then it is easy to see that $y \equiv -z \pmod{p}$ and $y \equiv -z \pmod{qr}$. From the Chinese remainder theorem, we see that $y \equiv -z \pmod{n}$, a contradiction.

Finally, case 2 subdivides into two subcases, exactly as in case 1, and both subcases lead to contradictions.

This completes the proof.

6.37 Prove Equations (6.3) and (6.4) relating the functions **half** and **parity**.

Answer: Denote $y = e_K(x)$, where $0 \leq x < n$. First, suppose that $\mathit{half}(y) = 0$. Then $0 \leq x < n/2$, and hence $0 \leq 2x < n$. Then

$$e_K(2x) \equiv e_K(2)e_K(x) \equiv e_K(2)y \pmod{n}.$$

Therefore $\mathit{parity}(e_K(2)y \bmod n) = 0$, because $2x$ is even.

Conversely, suppose that $\mathit{parity}(e_K(2)y \bmod n) = 0$. This implies that $2x \bmod n$ is even, where $0 \leq x < n$. However,

$$2x \bmod n = \begin{cases} 2x & \text{if } 0 \leq x < n/2 \\ 2x - n & \text{if } n/2 \leq x < n. \end{cases}$$

Because n is odd, we see that $2x \bmod n$ is even if and only if $0 \leq x < n/2$. This implies that $\mathit{half}(y) = 0$.

Now we turn to the other identity. First, suppose that $\mathit{parity}(y) = 0$. Then x is even, and hence $0 \leq x/2 < n/2$, where $x/2$ is an integer. Then

$$e_K(x/2) \equiv e_K(2^{-1})e_K(x) \equiv e_K(2^{-1})y \pmod{n}.$$

Therefore $\mathit{half}(e_K(2^{-1})y \bmod n) = 0$.

Finally, suppose that $\mathit{half}(e_K(2^{-1})y \bmod n) = 0$. This implies that $0 \leq 2^{-1}x \bmod n < n/2$, where $0 \leq x < n$. However,

$$2^{-1}x \bmod n = \begin{cases} x/2 & \text{if } x \text{ is even} \\ (x+n)/2 & \text{if } x \text{ is odd.} \end{cases}$$

We see that $2^{-1}x \bmod n < n/2$ if and only if x is even. This implies that $\mathit{parity}(y) = 0$.

6.38 Prove that Cryptosystem 6.3 is not semantically secure against a chosen ciphertext attack. Given x_1, x_2, a ciphertext (y_1, y_2) that is an encryption of x_i ($i = 1$ or 2), and given a decryption oracle DECRYPT for Cryptosystem 6.3, describe an algorithm to determine whether $i = 1$ or $i = 2$. You are allowed to call the algorithm DECRYPT with any input except for the given ciphertext (y_1, y_2), and it will output the corresponding plaintext.

Answer: Choose a random value $z \neq 0$, define $y_3 = y_2 \oplus z$, and call DECRYPT(y_1, y_2). The oracle outputs a value x, where $y_3 = G(r) \oplus x$. But

$$y_3 = y_2 \oplus z = G(r) \oplus x_i \oplus z,$$

where $i = 1$ or 2. Therefore $x_i = x \oplus z$ where x and z are known, and, hence, it is easy to determine the correct value of i.

Chapter 7

Public-Key Cryptography and Discrete Logarithms

7.1 Implement SHANKS' ALGORITHM for finding discrete logarithms in $\mathbb{Z}_p{}^*$, where p is prime and α is a primitive element modulo p. Use your program to find $\log_{106} 12375$ in $\mathbb{Z}_{24691}{}^*$ and $\log_6 248388$ in $\mathbb{Z}_{458009}{}^*$.

Answer: When $p = 12375$, we have $m = 158$. We find that $j = 141$, $i = 114$ and $\log_{106} 12375 = 22392$.

When $p = 458009$, we have $m = 677$. We find that $j = 625$, $i = 343$ and $\log_6 248388 = 232836$.

7.2 Describe how to modify SHANKS' ALGORITHM to compute the logarithm of β to the base α in a group G if it is specified ahead of time that this logarithm lies in the interval $[s, t]$, where s and t are integers such that $0 \leq s < t < n$, where n is the order of α. Prove that your algorithm is correct, and show that its complexity is $O(\sqrt{t-s})$.

Answer: Define $\gamma = \alpha^{-s}\beta$. Then $\log_\alpha \beta \in [s, t]$ if and only if $\log_\alpha \gamma \in [0, t-s]$. It suffices to compute $\log_\alpha \gamma$ using SHANKS' ALGORITHM with $m = \lceil \sqrt{t-s+1} \rceil$, and then calculate $\log_\alpha \beta = \log_\alpha \gamma + s$.

The proof of correctness is essentially the same as the proof of correctness of SHANKS' ALGORITHM given in Section 7.2. The complexity of the algorithm is $O(\sqrt{t-s+1}) = O(\sqrt{t-s})$.

7.3 The integer $p = 458009$ is prime and $\alpha = 2$ has order 57251 in $\mathbb{Z}_p{}^*$. Use the POLLARD RHO ALGORITHM to compute the discrete logarithm in $\mathbb{Z}_p{}^*$of $\beta = 56851$ to the base α. Take the initial value $x_0 = 1$, and define the partition (S_1, S_2, S_3) as in Example 7.3. Find the smallest integer i such that $x_i = x_{2i}$, and then compute the desired discrete logarithm.

Answer: $x_{444} = 339768$, $a_{444} = 22811$, $b_{444} = 35067$, $x_{888} = 339768$, $a_{888} = 37251$ and $b_{888} = 5360$. Therefore, $\log_\alpha \beta = 40007$.

7.4 Suppose that p is an odd prime and k is a positive integer. The multiplicative group $\mathbb{Z}_{p^k}{}^*$ has order $p^{k-1}(p-1)$, and is known to be cyclic. A generator for this group is called a ***primitive element modulo*** p^k.

(a) Suppose that α is a primitive element modulo p. Prove that at least one of α or $\alpha + p$ is a primitive element modulo p^2.

Answer: Suppose that α has order $p-1$ in $\mathbb{Z}_p{}^*$. Let n denote the order of

α in $Z_{p^2}{}^*$. $\alpha^n \equiv 1 \pmod{p^2}$ implies $\alpha^n \equiv 1 \pmod{p}$, so $n \equiv 0 \pmod{p-1}$. Also, n divides $|\mathbb{Z}_{p^2}{}^*| = p^2 - p$. Therefore $n = p-1$ or $n = p^2 - p$. If $n = p^2 - p$, then we're done, so assume $n = p - 1$. Now consider $\alpha + p$. By the same argument, α has order $p-1$ or $p^2 - p$ in $\mathbb{Z}_{p^2}{}^*$. We show that $\alpha + p$ cannot have order $p-1$, which finishes the proof.
We expand $(\alpha+p)^{p-1}$ using the binomial theorem:

$$(\alpha+p)^{p-1} = \alpha^{p-1} + (p-1)\alpha^{p-2}p + \text{ terms divisible by } p^2.$$

Reducing modulo p^2, we see that

$$\begin{aligned}(\alpha+p)^{p-1} &\equiv \alpha^{p-1} - p\alpha^{p-2} \pmod{p^2}\\ &\equiv 1 - p\alpha^{p-2} \pmod{p^2}.\end{aligned}$$

Therefore $(\alpha+p)^{p-1} \equiv 1 \pmod{p^2}$ if and only if $\alpha^{p-2} \equiv 0 \pmod{p}$. However, $\gcd(\alpha, p) = 1$. Therefore, $\alpha + p$ does not have order $p-1$, and we're done.

(b) Describe how to efficiently verify that 3 is a primitive root modulo 29 and modulo 29^2. Note: It can be shown that if α is a primitive root modulo p and modulo p^2, then it is a primitive root modulo p^k for all positive integers k (you do not have to prove this fact). Therefore, it follows that 3 is a primitive root modulo 29^k for all positive integers k.

Answer: $28 = 2^2 7$. To show that 3 is primitive modulo 29, it suffices to show that $3^{28/2}$ and $3^{28/7}$ are not congruent to 1 modulo 29. Since $3^1 4 \equiv 28 \pmod{29}$ and $3^4 \equiv 23 \pmod{29}$, we conclude that 3 is primitive modulo 29.
As shown in (a), the order of 3 in $\mathbb{Z}_{292}{}^*$ is either 28 or 28×29. To show that 3 is a primitive element, it suffices to show that 3^{28} is not congruent to 1 modulo 29^2. Since $3^{28} \bmod (29^2) = 436$, we're done.

(c) Find an integer α that is a primitive root modulo 29 but not a primitive root modulo 29^2.

Answer: It suffices to find a value α such that $\alpha^{28/2}$ and $\alpha^{28/7}$ are not congruent to 1 modulo 29; but $\alpha^{28} \equiv 1 \pmod{29^2}$. We have $14^{14} \equiv 28 \pmod{19}$, $14^4 \equiv 20 \pmod{29}$ and $14^{28} \equiv 1 \pmod{29^2}$, so $\alpha = 14$ is such an integer.

(d) Use the POHLIG-HELLMAN ALGORITHM to compute the discrete logarithm of 3344 to the base 3 in the multiplicative group $\mathbb{Z}_{24389}{}^*$.

Answer: $|\mathbb{Z}_{24389}{}^*| = 29^2 28 = 29^2 2^2 7^1$. Let a denote the desired discrete logarithm. We need to compute $a \bmod 29^2$, $a \bmod 2^2$ and $a \bmod 7$. We obtain:

$$\begin{aligned} a &\equiv 260 \pmod{29^2}\\ a &\equiv 2 \pmod{2^2}\\ a &\equiv 2 \pmod{7}.\end{aligned}$$

Applying the Chinese remainder theorem, $a = 18762$.

7.5 Implement the POHLIG-HELLMAN ALGORITHM for finding discrete logarithms in $\mathbb{Z}_p$, where p is prime and α is a primitive element. Use your program to find $\log_5 8563$ in $\mathbb{Z}_{28703}$ and $\log_{10} 12611$ in $\mathbb{Z}_{31153}$.

Answer: $28702 = 2^1 113^1 127^1$. We find that

$$\begin{aligned} \log_5 8563 &\equiv 1 \pmod{2}, \\ \log_5 8563 &\equiv 67 \pmod{113}, \quad \text{and} \\ \log_5 8563 &\equiv 99 \pmod{127}. \end{aligned}$$

Using the Chinese remainder theorem, $\log_5 8563 = 3909$.

$31152 = 2^4 3^1 11^1 59^1$. We find that

$$\begin{aligned} \log_{10} 12611 &\equiv 14 \pmod{16}, \\ \log_{10} 12611 &\equiv 2 \pmod{3}, \\ \log_{10} 12611 &\equiv 8 \pmod{11}, \quad \text{and} \\ \log_{10} 12611 &\equiv 51 \pmod{59}. \end{aligned}$$

Using the Chinese remainder theorem, $\log_{10} 12611 = 17102$.

7.6 Let $p = 227$. The element $\alpha = 2$ is primitive in $\mathbb{Z}_p{}^*$.

(a) Compute α^{32}, α^{40}, α^{59}, and α^{156} modulo p, and factor them over the factor base $\{2, 3, 5, 7, 11\}$.

Answer: $2^{32} \equiv 176 = 2^4 \times 11$, $2^{40} \equiv 110 = 2 \times 5 \times 11$, $2^{59} \equiv 60 = 2^2 \times 3 \times 5$ and $2^{156} \equiv 28 = 2^2 \times 7$

(b) Using the fact that $\log 2 = 1$, compute $\log 3$, $\log 5$, $\log 7$, and $\log 11$ from the factorizations obtained above (all logarithms are discrete logarithms in $\mathbb{Z}_p{}^*$ to the base α).

Answer: $\log 2 = 1$, $\log 3 = 46$, $\log 5 = 11$, $\log 7 = 154$ and $\log 11 = 28$.

(c) Now suppose we wish to compute $\log 173$. Multiply 173 by the "random" value $2^{177} \bmod p$. Factor the result over the factor base, and proceed to compute $\log 173$ using the previously computed logarithms of the numbers in the factor base.

Answer: $173 \times 2^{177} \equiv 168 = 2^3 3^1 7^1$. Therefore, $\log 173 = 2\log 2 + \log 3 + \log 7 - 177 \bmod 226 = 26$.

7.7 Suppose that $n = pq$ is an RSA modulus (i.e., p and q are distinct odd primes), and let $\alpha \in \mathbb{Z}_n{}^*$. For a positive integer m and for any $\alpha \in \mathbb{Z}_m{}^*$, define $\mathbf{ord}_m(\alpha)$ to be the order of α in the group $\mathbb{Z}_m{}^*$.

(a) Prove that

$$\mathbf{ord}_n(\alpha) = \mathrm{lcm}(\mathbf{ord}_p(\alpha), \mathbf{ord}_q(\alpha)).$$

Answer: This follows because $\alpha^j \equiv 1 \pmod{n}$ if and only if $\alpha^j \equiv 1 \pmod{p}$ and $\alpha^j \equiv 1 \pmod{q}$.

(b) Suppose that $\gcd(p-1, q-1) = d$. Show that there exists an element $\alpha \in \mathbb{Z}_n{}^*$ such that

$$\mathbf{ord}_n(\alpha) = \frac{\phi(n)}{d}.$$

Answer: Let α_p be a primitive element modulo p and let α_q be a primitive element modulo q. Using the Chinese remainder theorem, there exists $\alpha \in \mathbb{Z}_n{}^*$ such that $\alpha \equiv \alpha_p \pmod p$ and $\alpha \equiv \alpha_q \pmod q$. Then $\mathsf{ord}_p(\alpha) = p - 1$ and $\mathsf{ord}_q(\alpha) = q - 1$. Applying the result proven in part (a), we have that

$$\mathsf{ord}_n(\alpha) = \mathrm{lcm}(p-1, q-1) = \frac{(p-1)(q-1)}{\gcd(p-1, q-1)} = \frac{\phi(n)}{d}.$$

(c) Suppose that $\gcd(p-1, q-1) = 2$, and we have an oracle that solves the **Discrete Logarithm** problem in the subgroup $\langle \alpha \rangle$, where $\alpha \in \mathbb{Z}_n{}^*$ has order $\phi(n)/2$. That is, given any $\beta \in \langle \alpha \rangle$, the oracle will find the discrete logarithm $a = \log_\alpha \beta$, where $0 \leq a \leq \phi(n)/2 - 1$. (The value $\phi(n)/2$ is secret however.) Suppose we compute the value $\beta = \alpha^n \bmod n$ and then we use the oracle to find $a = \log_\alpha \beta$. Assuming that $p > 3$ and $q > 3$, prove that $n - a = \phi(n)$.

Answer: Because $\alpha^n \equiv \alpha^a \pmod n$ and α has order $\phi(n)/2$, we have that $a = n - k\phi(n)/2$ for some integer k. Also, $0 \leq a < \phi(n)/2$, so there is a unique integer k such that $0 \leq n - k\phi(n)/2 < \phi(n)/2$. We will show that $k = 2$ causes this inequality to be satisfied, which will complete the proof.

When $k = 2$, the inequality is equivalent to the following:

$$\phi(n) \leq n < \frac{3\phi(n)}{2}.$$

Clearly $\phi(n) \leq n$, so it suffices to show that $2n < 3\phi(n)$. Assuming WLOG that $p > q$, and using the fact that $q > 3$, this is equivalent to the following:

$$\begin{aligned} 2pq &< 3(p-1)(q-1), \\ pq &> 3(p+q-1), \\ p &> 3 + \frac{6}{q-3}. \end{aligned}$$

Because $q \geq 5$, we have that $3 + \frac{6}{q-3} \leq 6$. However, $p > q \geq 5$ is prime, so $p \geq 7$, and therefore the inequality is satisfied.

(d) Describe how n can easily be factored, given the discrete logarithm $a = \log_\alpha \beta$ from (c).

Answer: Given a, it is simple to compute $\phi(n) = n - a$. Then, given n and $\phi(n)$, it is straightforward to factor n by solving a quadratic equation, as described in Section 6.7.1.

7.8 In this question, we consider a generic algorithm for the **Discrete Logarithm** problem in $(\mathbb{Z}_{19}, +)$.

(a) Suppose that the set C is defined as follows:

$$C = \{(1 - i^2 \bmod 19, i \bmod 19) : i = 0, 1, 2, 4, 7, 12\}.$$

Compute **Good**(C).

Answer: observe that

$$\frac{1 - i^2 - (1 - j^2)}{i - j} = -(i + j),$$

for any $i \neq j$. From this it follows that

$$\mathsf{Good}(C) = \{i + j \bmod 19 : i, j \in \{0, 1, 2, 4, 7, 12\}, i \neq j\}.$$

An easy computation then shows that

$$\mathsf{Good}(C) = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 16\}.$$

(b) Suppose that the output of the group oracle, given the ordered pairs in C, is as follows:

$$\begin{array}{rcl} (0,1) & \mapsto & 10111 \\ (1,0) & \mapsto & 01100 \\ (16,2) & \mapsto & 00110 \\ (4,4) & \mapsto & 01010 \\ (9,7) & \mapsto & 00100 \\ (9,12) & \mapsto & 11001, \end{array}$$

where group elements are encoded as (random) binary 5-tuples. What can you say about the value of "a"?

Answer: Because the encodings are all different, it must be the case that $a \notin \mathsf{Good}(C)$. Therefore $a = 10, 15, 17$ or 18.

7.9 Decrypt the ElGamal ciphertext presented in Table 7.4. The parameters of the system are $p = 31847$, $\alpha = 5$, $a = 7899$ and $\beta = 18074$. Each element of $\mathbb{Z}_n$ represents three alphabetic characters as in Exercise 6.12.

The plaintext was taken from *The English Patient*, by Michael Ondaatje, Alfred A. Knopf, Inc., New York, 1992.

Answer: The first ciphertext element, $(3781, 14409)$, is decrypted to the plaintext element

$$x = 14409((3781)^{7899})^{-1} \bmod 31847 = 12354.$$

$x = 12354$ encodes the triple $18, 7, 4$, which corresponds to the three letters s, h, e.

The complete plaintext is as follows:

TABLE 7.4: ElGamal Ciphertext

$(3781, 14409)$	$(31552, 3930)$	$(27214, 15442)$	$(5809, 30274)$
$(5400, 31486)$	$(19936, 721)$	$(27765, 29284)$	$(29820, 7710)$
$(31590, 26470)$	$(3781, 14409)$	$(15898, 30844)$	$(19048, 12914)$
$(16160, 3129)$	$(301, 17252)$	$(24689, 7776)$	$(28856, 15720)$
$(30555, 24611)$	$(20501, 2922)$	$(13659, 5015)$	$(5740, 31233)$
$(1616, 14170)$	$(4294, 2307)$	$(2320, 29174)$	$(3036, 20132)$
$(14130, 22010)$	$(25910, 19663)$	$(19557, 10145)$	$(18899, 27609)$
$(26004, 25056)$	$(5400, 31486)$	$(9526, 3019)$	$(12962, 15189)$
$(29538, 5408)$	$(3149, 7400)$	$(9396, 3058)$	$(27149, 20535)$
$(1777, 8737)$	$(26117, 14251)$	$(7129, 18195)$	$(25302, 10248)$
$(23258, 3468)$	$(26052, 20545)$	$(21958, 5713)$	$(346, 31194)$
$(8836, 25898)$	$(8794, 17358)$	$(1777, 8737)$	$(25038, 12483)$
$(10422, 5552)$	$(1777, 8737)$	$(3780, 16360)$	$(11685, 133)$
$(25115, 10840)$	$(14130, 22010)$	$(16081, 16414)$	$(28580, 20845)$
$(23418, 22058)$	$(24139, 9580)$	$(173, 17075)$	$(2016, 18131)$
$(19886, 22344)$	$(21600, 25505)$	$(27119, 19921)$	$(23312, 16906)$
$(21563, 7891)$	$(28250, 21321)$	$(28327, 19237)$	$(15313, 28649)$
$(24271, 8480)$	$(26592, 25457)$	$(9660, 7939)$	$(10267, 20623)$
$(30499, 14423)$	$(5839, 24179)$	$(12846, 6598)$	$(9284, 27858)$
$(24875, 17641)$	$(1777, 8737)$	$(18825, 19671)$	$(31306, 11929)$
$(3576, 4630)$	$(26664, 27572)$	$(27011, 29164)$	$(22763, 8992)$
$(3149, 7400)$	$(8951, 29435)$	$(2059, 3977)$	$(16258, 30341)$
$(21541, 19004)$	$(5865, 29526)$	$(10536, 6941)$	$(1777, 8737)$
$(17561, 11884)$	$(2209, 6107)$	$(10422, 5552)$	$(19371, 21005)$
$(26521, 5803)$	$(14884, 14280)$	$(4328, 8635)$	$(28250, 21321)$
$(28327, 19237)$	$(15313, 28649)$		

She stands up in the garden where she has been working and looks into the distance. She has sensed a change in the weather. There is another gust of wind, a buckle of noise in the air, and the tall cypresses sway. She turns and moves uphill towards the house. Climbing over a low wall, feeling the first drops of rain on her bare arms, she crosses the loggia and quickly enters the house.

7.10 Determine which of the following polynomials are irreducible over $\mathbb{Z}_2[x]$: $x^5 + x^4 + 1$, $x^5 + x^3 + 1$, $x^5 + x^4 + x^2 + 1$.

Answer: $x^5 + x^3 + 1$ is irreducible, $x^5 + x^4 + 1 = (x^3 + x + 1)(x^2 + x + 1)$ and $x^5 + x^4 + x^2 + 1 = (x + 1)(x^4 + x + 1)$.

7.11 The field $\mathbb{F}_{2^5}$ can be constructed as $\mathbb{Z}_2[x]/(x^5 + x^2 + 1)$. Perform the following computations in this field.

(a) Compute $(x^4 + x^2) \times (x^3 + x + 1)$.

Answer: In the ring $\mathbb{Z}_2[x]$, we have that

$$(x^4 + x^2) \times (x^3 + x + 1) = x^2(x^5 + x^2 + 1) + x^3,$$

so $(x^4 + x^2) \times (x^3 + x + 1) = x^3$ in the field $\mathbb{Z}_2[x]/(x^5 + x^2 + 1)$.

(b) Using the extended Euclidean algorithm, compute $(x^3 + x^2)^{-1}$.

Answer: $(x^3 + x^2)^{-1} = x^2 + x + 1$ in the field $\mathbb{Z}_2[x]/(x^5 + x^2 + 1)$.

(c) Using the square-and-multiply algorithm, compute x^{25}.

Answer: $x^{25} = x^4 + x^3 + 1$ in the field $\mathbb{Z}_2[x]/(x^5 + x^2 + 1)$.

7.12 We give an example of the *ElGamal Cryptosystem* implemented in $\mathbb{F}_{3^3}$. The polynomial $x^3 + 2x^2 + 1$ is irreducible over $\mathbb{Z}_3[x]$ and hence $\mathbb{Z}_3[x]/(x^3 + 2x^2 + 1)$ is the field $\mathbb{F}_{3^3}$. We can associate the 26 letters of the alphabet with the 26 nonzero field elements, and thus encrypt ordinary text in a convenient way. We will use a lexicographic ordering of the (nonzero) polynomials to set up the correspondence. This correspondence is as follows:

A	$\leftrightarrow$	1	B	$\leftrightarrow$	2	C	$\leftrightarrow$	x
D	$\leftrightarrow$	$x+1$	E	$\leftrightarrow$	$x+2$	F	$\leftrightarrow$	$2x$
G	$\leftrightarrow$	$2x+1$	H	$\leftrightarrow$	$2x+2$	I	$\leftrightarrow$	x^2
J	$\leftrightarrow$	x^2+1	K	$\leftrightarrow$	x^2+2	L	$\leftrightarrow$	x^2+x
M	$\leftrightarrow$	x^2+x+1	N	$\leftrightarrow$	x^2+x+2	O	$\leftrightarrow$	x^2+2x
P	$\leftrightarrow$	x^2+2x+1	Q	$\leftrightarrow$	x^2+2x+2	R	$\leftrightarrow$	$2x^2$
S	$\leftrightarrow$	$2x^2+1$	T	$\leftrightarrow$	$2x^2+2$	U	$\leftrightarrow$	$2x^2+x$
V	$\leftrightarrow$	$2x^2+x+1$	W	$\leftrightarrow$	$2x^2+x+2$	X	$\leftrightarrow$	$2x^2+2x$
Y	$\leftrightarrow$	$2x^2+2x+1$	Z	$\leftrightarrow$	$2x^2+2x+2$			

Suppose Bob uses $\alpha = x$ and $a = 11$ in an *ElGamal Cryptosystem*; then $\beta = x + 2$. Show how Bob will decrypt the following string of ciphertext:

```
(K,H)(P,X)(N,K)(H,R)(T,F)(V,Y)(E,H)(F,A)(T,W)(J,D)(U,J)
```

Answer: The plaintext is *Galois field*.

7.13 Let $\mathcal{E}$ be the elliptic curve $y^2 = x^3 + x + 28$ defined over $\mathbb{Z}_{71}$.

(a) Determine the number of points on $\mathcal{E}$.

Answer: $\#E = 72$.

(b) Show that $\mathcal{E}$ is not a cyclic group.

Answer: This follows from part (c). If E were cyclic, there would be points having order 72, but there are no such points.

Alternatively, the result proven in Exercise 7.14 can be applied, because the congruence $x^3 + x + 28 \equiv 0 \pmod{71}$ has three solutions (namely, $x = 27, 53$ and 62).

(c) What is the maximum order of an element in $\mathcal{E}$? Find an element having this order.

Answer: The maximum order of a point is 36; $(4, 5)$ is one point having order 36.

7.14 Suppose that $p > 3$ is an odd prime, and $a, b \in \mathbb{Z}_p$. Further, suppose that the equation $x^3 + ax + b \equiv 0 \pmod{p}$ has three distinct roots in $\mathbb{Z}_p$. Prove that the corresponding elliptic curve group $(\mathcal{E}, +)$ is not cyclic.

HINT Show that the points of order two generate a subgroup of $(\mathcal{E}, +)$ that is isomorphic to $\mathbb{Z}_2 \times \mathbb{Z}_2$.

Answer: Let a_1, a_2 and a_3 be the three roots, which must be distinct. It is easy to show that $x_1 = (a_1, 0)$, $x_2 = (a_2, 0)$ and $x_3 = (a_3, 0)$ are three distinct points on E having order 2.

Using the fact that $a_1 + a_2 + a_3 = 0$ (which follows because the coefficient of x^2 in the cubic equation $x^3 + ax + b = 0$ is 0), it is straightforward to show that $x_1 + x_2 = x_3$, $x_1 + x_3 = x_2$ and $x_2 + x_3 = x_1$. Hence $\{x_1, x_2, x_3, \mathcal{O}\}$ is isomorphic to $\mathbb{Z}_2 \times \mathbb{Z}_2$. Since G contains a subgroup that is not cyclic, G is not cyclic.

7.15 Consider an elliptic curve $\mathcal{E}$ described by the formula $y^2 \equiv x^3 + ax + b \pmod{p}$, where $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$ and $p > 3$ is prime.

(a) It is clear that a point $P = (x_1, y_1) \in \mathcal{E}$ has order 3 if and only if $2P = -P$. Use this fact to prove that, if $P = (x_1, y_1) \in \mathcal{E}$ has order 3, then

$$3{x_1}^4 + 6a{x_1}^2 + 12x_1 b - a^2 \equiv 0 \pmod{p}. \tag{7.11}$$

Answer: The x-coordinate of $-P$ is x_1. The x-coordinate of $2P$ is

$$\frac{(3{x_1}^2 + a)^2}{(2y_1)^2} - 2x_1.$$

These two x-coordinates must be equal if $P = -2P$. Hence,

$$(3{x_1}^2 + a)^2 = (2y_1)^2(3x_1) = 12x_1{y_1}^2.$$

However,

$${y_1}^2 = {x_1}^3 + ax_1 + b,$$

so

$$(3{x_1}^2 + a)^2 = 12x_1({x_1}^3 + ax_1 + b).$$

This simplifies to give the equation (7.11), which is a necessary condition for P to have order 3.

(b) Conclude from equation (7.11) that there are at most 8 points of order 3 on the elliptic curve $\mathcal{E}$.

Answer: Equation (7.11) has degree equal to four, so it has at most four roots over the field $\mathbb{Z}_p$. For each root x_1 of (7.11), there are at most two values of y_1 such that (x_1, y_1) is a point on E. The total number of points on E having order 3 is therefore at most $2 \times 4 = 8$.

(c) Using equation (7.11), determine all points of order 3 on the elliptic curve $y^2 \equiv x^3 + 34x \pmod{73}$.

Answer: The equation (7.11) becomes

$$\begin{aligned} 3{x_1}^4 + 6 \times 34{x_1}^2 - 34^2 &\equiv 0 \pmod{73}, \\ {x_1}^4 + 68{x_1}^2 - 69 &\equiv 0 \pmod{73}, \\ {x_1}^4 - 5{x_1}^2 + 4 &\equiv 0 \pmod{73}. \end{aligned}$$

This equation factors:

$$({x_1}^2 - 1)({x_1}^2 - 4) \equiv 0 \pmod{73},$$

or

$$x_1 \equiv 1, -1, 2, -2 \pmod{73}.$$

For each of these values of x_1, we need to find the corresponding values of y_1 (if possible).

i. If $x_1 = 1$, then ${y_1}^2 = 35$, and $y_1 = 20$ or 53.
ii. If $x_1 = -1$, then ${y_1}^2 = 38$, and $y_1 = 29$ or 35.
iii. If $x_1 = 2$, then ${y_1}^2 = 2$, and $y_1 = 21$ or 52.
iv. If $x_1 = -2$, then ${y_1}^2 = 71$, and $y_1 = 17$ or 56.

There are eight possible points of order 3, namely $(1, 20)$, $(1, 53)$, $(72, 29)$, $(72, 44)$, $(2, 21)$, $(2, 52)$, $(71, 17)$ and $(71, 56)$. (It can be verified that all eight of these points do in fact have order 3.)

7.16 Suppose that $\mathcal{E}$ is an elliptic curve defined over $\mathbb{Z}_p$, where $p > 3$ is prime. Suppose that $\#\mathcal{E}$ is prime, $P \in \mathcal{E}$, and $P \neq \mathcal{O}$.

(a) Prove that the discrete logarithm $\log_P(-P) = \#\mathcal{E} - 1$.

Answer: Denote $\#E = q$. The order of P divides q, $P \neq \mathcal{O}$ and q is prime, so the order of P must be equal to q. Now we have that $\mathcal{O} = qP = P + (q-1)P$. But we also have $\mathcal{O} = P + (-P)$, so $-P = (q-1)P$ and $\log_P(-P) = q - 1$.

(b) Describe how to compute $\#\mathcal{E}$ in time $O(p^{1/4})$ by using Hasse's bound on $\#\mathcal{E}$, together with a modification of SHANKS' ALGORITHM. Give a pseudocode description of the algorithm.

Answer: Let $P \in E$, $P \neq \mathcal{O}$. Define $s = \lfloor p - \sqrt{p} \rfloor$ and $t = \lfloor p + \sqrt{p} \rfloor - 1$, and use the modification of SHANKS' ALGORITHM described in Exercise 7.2 to find $a = \log_P(-P)$. (We have that $\log_P(-P) = q - 1$, where $q \in [s+1, t+1]$, so $\log_P(-P) \in [s, t]$.)

Note that the interval $[s,t]$ contains $2\lfloor \sqrt{p} \rfloor + 1$ possible values. It will be the case that $q = a + 1$ provided that $q \geq 2\lfloor \sqrt{p} \rfloor + 1$ (this ensures that there is a unique element of the interval $[s,t]$ that is congruent to a modulo q). We have that $q \geq p - \lfloor \sqrt{p} \rfloor$, so everything is all right, provided that $p \geq 3\lfloor \sqrt{p} \rfloor + 1$. This last inequality is true for all primes $p \geq 11$.

For the primes $p = 5$ and 7, it is probably simpler to directly compute the value of q. This does not affect the asymptotic complexity of the algorithm, which is $O(\sqrt{t-s}) = O(p^{1/4})$ by Exercise 7.2.

7.17 Suppose $e : G_1 \times G_2 \to G_3$ is a bilinear pairing. Prove, for all $P \in G_1$ and $Q \in G_2$, that $e(aP, bQ) = e(P,Q)^{ab}$ for any positive integers a and b.

Answer: We will prove this by induction on a. First, fix $a = 1$. Then we can easily prove that $e(P, bQ) = e(P,Q)^b$ for all positive integers b by induction on b. Now, as an induction hypothesis, assume that a is a positive integer such that $e(aP, bQ) = e(P,Q)^{ab}$ for all positive integers b. Then

$$e((a+1)P, bQ) = e(aP, bQ)e(P, bQ) = e(P,Q)^{ab}e(P,Q)^b = e(P,Q)^{(a+1)b},$$

as desired.

7.18 Let $\mathcal{E}$ be the elliptic curve described by the equation $y^2 = x^3 + x + 4$ over $\mathbb{F}_5$. Show that the point $(3,2)$ is a 3-torsion point, and show that the point $(2,3)$ is not a 3-torsion point.

Answer: $(3,2) + (3,2) = (3,3)$ and $(3,2) + (3,3) = \mathcal{O}$, so $(3,2)$ is a 3-torsion point. However, $(2,3) + (2,3) = (0,2)$. Since $(0,2) \neq (2,2) = -(2,3)$, $(2,3)$ is not a 3-torsion point.

7.19 (a) Determine the NAF representation of the integer 87.

Answer: The NAF representation of 87 is $(1, 0, -1, 0, -1, 0, 0, -1)$.

(b) Using the NAF representation of 87, use Algorithm 7.6 to compute $87P$, where $P = (2,6)$ is a point on the elliptic curve $y^2 = x^3 + x + 26$ defined over $\mathbb{Z}_{127}$. Show the partial results during each iteration of the algorithm.

Answer: The algorithm proceeds as follows:

i	c_i	Q
7	1	$[2, 6]$
6	0	$[118, 80]$
5	-1	$[68, 57]$
4	0	$[85, 119]$
3	-1	$[87, 116]$
2	0	$[91, 18]$
1	0	$[102, 39]$
0	-1	$[102, 88]$

Therefore $87P = [102, 88]$.

7.20 Let $\mathcal{L}_i$ denote the set of positive integers that have exactly i coefficients in their NAF representation, such that the leading coefficient is 1. Denote $k_i = |\mathcal{L}_i|$.

(a) By means of a suitable decomposition of $\mathcal{L}_i$, prove that the k_i's satisfy the following recurrence relation:

$$\begin{aligned} k_1 &= 1 \\ k_2 &= 1 \\ k_{i+1} &= 2(k_1 + k_2 + ... + k_{i-1}) + 1 \quad \text{(for } i \geq 2\text{)}. \end{aligned}$$

Answer: It is clear that $k_1 = k_2 = 1$.

For any $x \in \mathcal{L}_{i+1}$, let $a(x)$ denote the number of consecutive zeroes that follow the initial '1'. If $a(x) = i$, then the NAF representation of x is $(1, 0, \ldots, 0)$. If $a(x) \leq i - 1$, then let y denote the entry that follows the $a(x)$ consecutive zeroes in the NAF representation of x. Clearly $y = 1$ or $y = -1$. If $y = 1$, then the last $i - a(x)$ entries in the NAF representation of x form the NAF representation of an integer in $\mathcal{L}_{i-a(x)}$. Suppose that $y = -1$. If we change this '-1' to a '1', then the last $i - a(x)$ entries again form the NAF representation of an integer in $\mathcal{L}_{i-a(x)}$.

(b) Derive a second degree recurrence relation for the k_i's, and obtain an explicit solution of the recurrence relation.

Answer: We have that

$$k_{i+1} = 2(k_1 + k_2 + ... + k_{i-1}) + 1$$

and

$$k_i = 2(k_1 + k_2 + ... + k_{i-2}) + 1,$$

for $i \geq 3$. Subtracting, we see that

$$k_{i+1} - k_i = 2k_{i-1},$$

$i \geq 3$. Also, $k_1 = k_2 = 1$ and $k_3 = 3$.

This recurrence can be solved by standard techniques; the solution is

$$k_i = \frac{2^{i+1}}{3} + \frac{(-1)^i}{3},$$

$i \geq 1$ (this can be proven by induction).

7.21 Find $\log_5 896$ in $\mathbb{Z}_{1103}$ using Algorithm 7.7, given that $L_2(\beta) = 1$ for $\beta = 25$, 219, and 841, and $L_2(\beta) = 0$ for $\beta = 163$, 532, 625, and 656.

Answer: We obtain the following:

$$\begin{array}{ll}
\beta = 896 & x_0 = 1 \\
\beta = 841 & x_1 = 1 \\
\beta = 656 & x_2 = 0 \\
\beta = 532 & x_3 = 0 \\
\beta = 219 & x_4 = 1 \\
\beta = 163 & x_5 = 0 \\
\beta = 625 & x_6 = 0 \\
\beta = 25 & x_7 = 1 \\
\beta = 1 &
\end{array}$$

Therefore $\log_5 896 = 10010011_2 = 147$.

7.22 Throughout this question, suppose that $p \equiv 5 \pmod 8$ is prime and suppose that a is a quadratic residue modulo p.

(a) Prove that $a^{(p-1)/4} \equiv \pm 1 \pmod p$.

Answer: a is a quadratic residue, so $a^{(p-1)/2} \equiv 1 \pmod p$ by Euler's criterion. Now $(a^{(p-1)/4})^2 = a^{(p-1)/2}$, so $a^{(p-1)/4} \equiv \pm 1 \pmod p$.

(b) If $a^{(p-1)/4} \equiv 1 \pmod p$, prove that $a^{(p+3)/8} \bmod p$ is a square root of a modulo p.

Answer:

$$(a^{(p+3)/8})^2 = a^{(p+3)/4} = a^{(p-1)/4}a \equiv a^{(p-1)/4} \pmod p.$$

(c) If $a^{(p-1)/4} \equiv -1 \pmod p$, prove that $2^{-1}(4a)^{(p+3)/8} \bmod p$ is a square root of a modulo p.

HINT Use the fact that $\left(\frac{2}{p}\right) = -1$ when $p \equiv 5 \pmod 8$ is prime.

Answer:

$$\begin{aligned}(2^{-1}(4a)^{(p+3)/8})^2 &\equiv 4^{-1}(4a)^{(p+3)/4} \pmod p\\ &\equiv 4^{(p-1)/4}a^{(p+3)/4} \pmod p\\ &\equiv 2^{(p-1)/2}a^{(p-1)/4}a \pmod p\\ &\equiv \left(\frac{2}{p}\right)a^{(p-1)/4}a \pmod p\\ &\equiv (-1)(-1)a \pmod p\\ &\equiv a \pmod p.\end{aligned}$$

(d) Given a primitive element $\alpha \in \mathbb{Z}_p{}^*$, and given any $\beta \in \mathbb{Z}_p{}^*$, show that $L_2(\beta)$ can be computed efficiently.

HINT Use the fact that it is possible to compute square roots modulo p, as well as the fact that $L_1(\beta) = L_1(p-\beta)$ for all $\beta \in \mathbb{Z}_p{}^*$, when $p \equiv 5 \pmod 8$ is prime.

Answer: Let $a = \log_\alpha \beta$. Then $a = 4a^* + 2a_1 + a_0$, where $a_0 = L_1(\beta)$ and $a_1 = L_2(\beta)$. It is possible to compute a_0 efficiently (see page 262). Let $\beta_0 = \beta/\alpha^{a_0}$; then $\log_\alpha \beta_0 = 4a^* + 2a_1$. Next, compute a square root β_1 of β_0 using the technique described in part (b) or (c). The two square roots of β_0 are $\pm\alpha^{2a^*+a_1}$, or $\alpha^{2a^*+a_1}$ and $\alpha^{(p-1)/2-2a^*-a_1}$. We have that $(p-1)/2$ is even, so $L_1(\alpha^{2a^*+a_1}) = L_1(\alpha^{(p-1)/2-2a^*-a_1})$. Therefore,

$$L_1(\beta_1) = L_1(\alpha^{2a^*+a_1}) = a_1 = L_2(\beta).$$

Since $L_1(\beta_1)$ can be computed efficiently, we have an algorithm to compute $L_2(\beta)$ efficiently.

7.23 The *ElGamal Cryptosystem* can be implemented in any subgroup $\langle\alpha\rangle$ of a finite multiplicative group $(G,\cdot)$, as follows: Let $\beta \in \langle\alpha\rangle$ and define (α,β) to be the public key. The plaintext space is $\mathcal{P} = \langle\alpha\rangle$, and the encryption operation is $e_K(x) = (y_1, y_2) = (\alpha^k, x\cdot\beta^k)$, where k is random.

Here we show that distinguishing ElGamal encryptions of two plaintexts can be Turing reduced to **Decision Diffie-Hellman**, and vice versa.

(a) Assume that ORACLEDDH is an oracle that solves **Decision Diffie-Hellman** in $(G,\cdot)$. Prove that ORACLEDDH can be used as a subroutine in an algorithm that distinguishes ElGamal encryptions of two given plaintexts, say x_1 and x_2. (That is, given $x_1, x_2 \in \mathcal{P}$, and given a ciphertext (y_1, y_2) that is an encryption of x_i for some $i \in \{1,2\}$, the distinguishing algorithm will determine if $i = 1$ or $i = 2$.)

Answer: For $i = 1, 2$, compute $u_i = y_2(x_i)^{-1}$. If

$$\text{ORACLEDDH}(\alpha, \beta, y_1) = u_i,$$

then (y_1, y_2) is an encryption of x_i.

(b) Assume that ORACLE-DISTINGUISH is an oracle that distinguishes ElGamal encryptions of any two given plaintexts x_1 and x_2, for any *ElGamal Cryptosystem* implemented in the group $(G, \cdot)$ as described above. Suppose further that ORACLE-DISTINGUISH will determine if a ciphertext (y_1, y_2) is not a valid encryption of either of x_1 or x_2. Prove that ORACLE-DISTINGUISH can be used as a subroutine in an algorithm that solves **Decision Diffie-Hellman** in $(G, \cdot)$.

Answer: We are given an instance of **Decision Diffie-Hellman**, namely, $\alpha, \beta, \gamma, \delta$. Define $y_1 = \gamma$, $y_2 = \delta$, $x_1 = 1$ and $x_2 \neq 1$ (arbitrarily). Call ORACLEDISTINGUISH$(x_1, x_2, (y_1, y_2))$. If the result is $i = 1$, then answer "yes"; otherwise, answer "no".

Chapter 8

Signature Schemes

8.1 Suppose Alice is using the *ElGamal Signature Scheme* with $p = 31847$, $\alpha = 5$, and $\beta = 25703$. Compute the values of k and a (without solving an instance of the **Discrete Logarithm** problem), given the signature $(23972, 31396)$ for the message $x = 8990$ and the signature $(23972, 20481)$ for the message $x = 31415$.

Answer: First, we compute

$k = (x_1 - x_2)(\delta_1 - \delta_2)^{-1} \bmod (p-1) = -22425 \times 10915^{-1} \bmod 31846 = 1165.$

To determine a, we will solve the congruence

$$\gamma a \equiv x_1 - k\delta_1 \pmod{p-1}$$

for a. This congruence simplifies to

$$23972a \equiv 23704 \pmod{31846}.$$

We use the method described on page 319 to solve it. We have that $\gcd(23972, 31846) = 2$, and $2|23704$, so the congruence is equivalent to

$$11986a \equiv 11852 \pmod{15923}.$$

This congruence has the solution

$$a \equiv 11852 \times 11986^{-1} \pmod{15923} \equiv 7459 \pmod{15923}.$$

Therefore, $a = 7459$ or $a = 7459 + (p-1)/2 = 23382$. By computing $\alpha^{7459} \bmod p = 25703 = \beta$ and $\alpha^{23382} \bmod p = 6144 \neq \beta$, we see that $a = 7459$.

8.2 Suppose we implement the *ElGamal Signature Scheme* with $p = 31847$, $\alpha = 5$, and $\beta = 26379$. Write a computer program that does the following:

(a) Verify the signature $(20679, 11082)$ on the message $x = 20543$.

Answer: $5^{20543} \bmod 31847 = 20688 = 26379^{20679} 20679^{11082} \bmod 31847$.

(b) Determine the private key, a, by solving an instance of the **Discrete Logarithm** problem.

Answer: $a = \log_5 26379 = 7973$.

(c) Then determine the random value k used in signing the message x, without solving an instance of the **Discrete Logarithm** problem.

Answer: To determine k, we will solve the congruence

$$k\delta \equiv x - a\gamma \pmod{p-1}$$

for k. This congruence simplifies to

$$11082k \equiv 13618 \pmod{31846}.$$

We use the method described on page 319 to solve it. $\gcd(11082, 31846) = 2$, and $2|13618$, so the congruence is equivalent to

$$5541k \equiv 6809 \pmod{15923}.$$

This congruence has the solution

$$k \equiv 6809 \times 5541^{-1} \pmod{15923} \equiv 3464 \pmod{15923}.$$

Therefore, $k = 3464$ or $k = 7459 + (p-1)/2 = 19387$. By computing $\alpha^{3464} \bmod p = 11168 \neq \gamma$ and $\alpha^{19387} \bmod p = 20679 = \gamma$, we see that $k = 19387$.

8.3 Suppose that Alice is using the *ElGamal Signature Scheme*. In order to save time in generating the random numbers k that are used to sign messages, Alice chooses an initial random value k_0, and then signs the ith message using the value $k_i = k_0 + 2i \bmod (p-1)$. Therefore,

$$k_i = k_{i-1} + 2 \bmod (p-1)$$

for all $i \geq 1$. (This is not a recommended method of generating k-values!)

(a) Suppose that Bob observes two consecutive signed messages, say $(x_i, \mathbf{sig}(x_i, k_i))$ and $(x_{i+1}, \mathbf{sig}(x_{i+1}, k_{i+1}))$. Describe how Bob can easily compute Alice's secret key, a, given this information, without solving an instance of the **Discrete Logarithm** problem. (Note that the value of i does not have to be known for the attack to succeed.)

Answer: We have the following:

$$\begin{aligned} k_i\delta_1 &\equiv x_1 - a\gamma_1 \pmod{p-1} && (8.1)\\ (k_i+2)\delta_2 &\equiv x_2 - a\gamma_2 \pmod{p-1}. && (8.2) \end{aligned}$$

Multiply (8.1) by δ_2 and multiply (8.2) by δ_1, obtaining the following:

$$\begin{aligned} k_i\delta_1\delta_2 &\equiv x_1\delta_2 - a\gamma_1\delta_2 \pmod{p-1} && (8.3)\\ (k_i+2)\delta_1\delta_2 &\equiv x_2\delta_1 - a\gamma_2\delta_1 \pmod{p-1}. && (8.4) \end{aligned}$$

Then compute (8.4) − (8.3):

$$a(\gamma_2\delta_1 - \gamma_1\delta_2) \equiv x_2\delta_1 - x_1\delta_2 - 2\delta_1\delta_2 \pmod{p-1}. \tag{8.5}$$

(8.5) is a linear congruence in the unknown a. There are

$$\gcd(\gamma_2\delta_1 - \gamma_1\delta_2, p-1)$$

solutions to (8.5) modulo $p-1$, and they can be found using the method described on page 319. However, there is a unique correct value of a modulo p that satisfies the condition $\alpha^a \equiv \beta \pmod{p}$. If (8.5) has more than one solution modulo $p-1$, then it will be necessary to verify which of these solutions is the (unique) correct value of a.

(b) Suppose that the parameters of the scheme are $p = 28703$, $\alpha = 5$, and $\beta = 11339$, and the two messages observed by Bob are

$$\begin{array}{ll} x_i = 12000 & \mathbf{sig}(x_i, k_i) = (26530, 19862) \\ x_{i+1} = 24567 & \mathbf{sig}(x_{i+1}, k_{i+1}) = (3081, 7604). \end{array}$$

Find the value of a using the attack you described in part (a).

Answer: Here, the congruence (8.5) is as follows:

$$14396a \equiv 9964 \pmod{28702}.$$

We have that $\gcd(14396, 28702) = 2$, so this congruence is equivalent to

$$7198a \equiv 4982 \pmod{14351}.$$

This congruence has the solution

$$a = 4982 \times 7198^{-1} \bmod 14351 = 5324.$$

Therefore $a = 5324$ or $a = 5324 + 14351 = 19675$. It can be verified that

$$5^{5324} \bmod 28703 = 17364 \neq \beta$$

and

$$5^{19675} \bmod 28703 = 11339 = \beta.$$

Therefore, $a = 19675$.

8.4 (a) Prove that the second method of forgery on the *ElGamal Signature Scheme*, described in Section 8.3, also yields a signature that satisfies the verification condition.

Answer: We assume that $\alpha^x \equiv \beta^\gamma\gamma^\delta \pmod{p}$. Suppose h, i and j are integers, $0 \le h, i, j \le p-2$, and $\gcd(h\gamma - j\delta, p-1) = 1$. Define

$$\begin{aligned} \lambda &= \gamma^h\alpha^i\beta^j \bmod p \\ \mu &= \delta\lambda(h\gamma - j\delta)^{-1} \bmod (p-1), \quad \text{and} \\ x' &= \lambda(hx + i\delta)(h\gamma - j\delta)^{-1} \bmod (p-1). \end{aligned}$$

We need to show that

$$\beta^\lambda \lambda^\mu \equiv \alpha^{x'} \pmod{p}.$$

We proceed as follows:

$$\begin{aligned}
& & \alpha^{x'} &\equiv \beta^\lambda \lambda^\mu \pmod{p}\\
&\Longleftrightarrow & \alpha^{x'} &\equiv \beta^\lambda (\alpha^i \beta^j \gamma^h)^\mu \pmod{p}\\
&\Longleftrightarrow & \alpha^{x'-i\mu} &\equiv \beta^{\lambda+j\mu}\gamma^{h\mu} \pmod{p}\\
&\Longleftrightarrow & \alpha^{(x'-i\mu)\gamma} &\equiv \beta^{(\lambda+j\mu)\gamma}\gamma^{h\mu\gamma} \pmod{p}\\
&\Longleftrightarrow & \alpha^{(x'-i\mu)\gamma} &\equiv \beta^{(\lambda+j\mu)\gamma}\gamma^{(\lambda+j\mu)\delta} \pmod{p}\\
&\Longleftrightarrow & \alpha^{(x'-i\mu)\gamma} &\equiv (\beta^\gamma\gamma^\delta)^{(\lambda+j\mu)} \pmod{p}\\
&\Longleftrightarrow & \alpha^{(x'-i\mu)\gamma} &\equiv (\alpha^x)^{(\lambda+j\mu)} \pmod{p}\\
&\Longleftrightarrow & (x'-i\mu)\gamma &\equiv x(\lambda+j\mu) \pmod{p-1}\\
&\Longleftrightarrow & x'\gamma - x\lambda &\equiv \mu(xj+i\gamma) \pmod{p-1}\\
&\Longleftrightarrow & (h\gamma-j\delta)(x'\gamma-x\lambda) &\equiv \lambda\delta(xj+i\gamma) \pmod{p-1}\\
&\Longleftrightarrow & x'\gamma(h\gamma-j\delta) &\equiv x\lambda(h\gamma-j\delta)+\lambda\delta(xj+i\gamma) \pmod{p-1}\\
&\Longleftrightarrow & x'\gamma(h\gamma-j\delta) &\equiv \gamma\lambda(i\delta+xh) \pmod{p-1}\\
&\Longleftrightarrow & x'(h\gamma-j\delta) &\equiv \lambda(i\delta+xh)(h\gamma-j\delta)^{-1} \pmod{p-1}.
\end{aligned}$$

(b) Suppose Alice is using the *ElGamal Signature Scheme* as implemented in Example 8.1: $p = 467$, $\alpha = 2$, and $\beta = 132$. Suppose Alice has signed the message $x = 100$ with the signature $(29, 51)$. Compute the forged signature that Oscar can then form by using $h = 102$, $i = 45$, and $j = 293$. Check that the resulting signature satisfies the verification condition.

Answer: $x' = 355$, $\lambda = 363$ and $\mu = 401$. Then

$$\alpha^{x'} \bmod p = 2^{355} \bmod 467 = 255$$

and

$$\beta^\lambda \lambda^\mu \bmod p = 132^{363}363^{401} \bmod 467 = 255,$$

so the signature $(363, 401)$ on the message 355 is verified.

8.5 (a) A signature in the *ElGamal Signature Scheme* or the *DSA* is not allowed to have $\delta = 0$. Show that if a message were signed with a "signature" in which $\delta = 0$, then it would be easy for an adversary to compute the secret key, a.

Answer: If $\delta = 0$ in the *DSA*, then $(\text{SHA3-224}(x) + a\gamma)k^{-1} \equiv 0 \pmod{q}$ and hence $\text{SHA3-224}(x) + a\gamma \equiv 0 \pmod{q}$. Then $a = -\text{SHA3-224}(x)\gamma^{-1} \bmod q$.

(b) A signature in the *DSA* is not allowed to have $\gamma = 0$. Show that if a "signature" in which $\gamma = 0$ is known, then the value of k used in that "signature" can be determined. Given that value of k, show that it is now possible to forge a "signature" (with $\gamma = 0$) for any desired message (i.e., a selective forgery can be carried out).

Answer: If $\gamma = 0$ in the *DSA*, then $\delta = \text{SHA3-224}(x)k^{-1} \bmod q$ and hence $k = \text{SHA3-224}(x)\delta^{-1} \bmod q$.

Now, given an arbitrary message x_0, define

$$\begin{aligned}\gamma_0 &= 0,\\ \delta_0 &= \text{SHA3-224}(x_0)k^{-1} \bmod q,\end{aligned}$$

where k was computed above. Then $e_1 = k$, $e_2 = 0$, and

$$(\alpha^{e_1}\beta^{e_2} \bmod p) \bmod q = (\alpha^k \bmod p) \bmod q = 0 = \gamma_0,$$

so the "signature" is verified.

(c) Evaluate the consequences of allowing a signature in the *ECDSA* to have $r = 0$ or $s = 0$.

Answer: If $s = 0$, then it is possible to compute the secret key, m, in the same way that a was computed in the *DSA* when $\delta = 0$:

$$m = -\text{SHA3-224}(x)r^{-1} \bmod q.$$

If $r = 0$, then the situation is similar to when $\gamma = 0$ in the *DSA*. First, it is possible to compute k: $k = \text{SHA3-224}(x)s^{-1} \bmod q$. Then a signature on an arbitrary message x_0 can be computed using this k by defining

$$\begin{aligned}r &= 0,\\ s &= \text{SHA3-224}(x_0)k^{-1} \bmod q.\end{aligned}$$

Then $i = k$ and $j = 0$, and it is easily seen that the "signature" (r, s) on the message x_0 is verified.

8.6 Here is a variation of the *ElGamal Signature Scheme*. The key is constructed in a similar manner as before: Alice chooses $\alpha \in \mathbb{Z}_p{}^*$ to be a primitive element, $0 \leq a \leq p-2$ where $\gcd(a, p-1) = 1$, and $\beta = \alpha^a \bmod p$. The key $K = (\alpha, a, \beta)$, where α and β are the public key and a is the private key. Let $x \in \mathbb{Z}_p$ be a message to be signed. Alice computes the signature $\mathbf{sig}(x) = (\gamma, \delta)$, where

$$\gamma = \alpha^k \bmod p$$

and

$$\delta = (x - k\gamma)a^{-1} \bmod (p-1).$$

The only difference from the original *ElGamal Signature Scheme* is in the computation of δ. Answer the following questions concerning this modified scheme.

(a) Describe how a signature (γ, δ) on a message x would be verified using Alice's public key.

Answer: We have $a\delta + k\gamma \equiv x \pmod{p-1}$, so $\alpha^{a\delta}\alpha^{k\gamma} \equiv \alpha^x \pmod{p}$. Therefore $\beta^\delta \gamma^\gamma \equiv \alpha^x \pmod{p}$; this is the verification condition.

(b) Describe a computational advantage of the modified scheme over the original scheme.

Answer: The value $a^{-1} \bmod (p-1)$ does not depend on k or x, so it can be precomputed once and for all. In the original version of the *ElGamal Signature Scheme*, a new value $k^{-1} \bmod (p-1)$ must be computed every time a new signature is created.

(c) Briefly compare the security of the original and modified scheme.

Answer: Suppose Oscar tries to forge a signature for a message x. If he chooses a value for γ and tries to solve for δ, he has to solve an instance of the **Discrete Logarithm** problem. If he chooses a value for δ and tries to solve for γ, he has to solve a congruence of the form $\gamma^\gamma \equiv y \pmod{p}$ for γ. This problem does not seem to be one whose difficulty has been studied. In particular, it is a different problem than the one that arises when trying to forge an ElGamal signature by first choosing δ and then trying to solve for γ.

8.7 Suppose Alice uses the *DSA* with $q = 101$, $p = 7879$, $\alpha = 170$, $a = 75$, and $\beta = 4567$, as in Example 8.4. Determine Alice's signature on a message x such that SHA3-224$(x) = 52$, using the random value $k = 49$, and show how the resulting signature is verified.

Answer: We compute $\gamma = 59$ and $\delta = 79$. To verify the signature, $e_1 = 16$, $e_2 = 57$ and

$$(170^{16}4567^{57} \bmod 7879) \bmod 101 = 59.$$

8.8 We showed that using the same value k to sign two messages in the *ElGamal Signature Scheme* allows the scheme to be broken (i.e., an adversary can determine the secret key without solving an instance of the **Discrete Logarithm** problem). Show how similar attacks can be carried out for the *Schnorr Signature Scheme*, the *DSA*, and the *ECDSA*.

Answer: In the *DSA*, Suppose that $k_1 = k_2 = k$. Then $\gamma_1 = \gamma_2 = \gamma$, say, and

$$k(\delta_1 - \delta_2) \equiv \text{SHA3-224}(x_1) - \text{SHA3-224}(x_2) \pmod{q}.$$

this allows k to be determined, provided that $\delta_1 \neq \delta_2$:

$$k = (\text{SHA3-224}(x_1) - \text{SHA3-224}(x_2))(\delta_1 - \delta_2)^{-1} \bmod q.$$

Once k is determined, a can be computed, as follows:

$$a = (k\delta_1 - \text{SHA3-224}(x_1))\gamma^{-1} \bmod q.$$

The situation with *ECDSA* is very similar. First, k can be computed:

$$k = (\text{SHA3-224}(x_1) - \text{SHA3-224}(x_2))(s_1 - s_2)^{-1} \bmod q;$$

and then m can be found:

$$m = (ks_1 - \text{SHA3-224}(x_1))r^{-1} \bmod q.$$

Now we turn to the *Schnorr Signature Scheme*. With high probability, we will have $\gamma_1 \neq \gamma_2$ even when $k_1 = k_2 = k$, because the γ's depend on the messages being signed. It turns out that we can solve for a directly:

$$a = (\delta_1 - \delta_2)(\gamma_1 - \gamma_2)^{-1} \bmod q.$$

8.9 Suppose that two people (say Alice and Bob) using the *Digital Signature Algorithm* happen to use the same k-value to sign two messages. In addition, suppose that the difference between their secret keys is small. In this situation, the scheme can be broken by an adversary.

More precisely, we assume that Alice and Bob employ the same values of p, q, and α in the *DSA*. Alice has $\beta_1 = \alpha^{a_1}$ and Bob has $\beta_2 = \alpha^{a_2}$ where $|a_1 - a_2| \leq c$, for some small constant $c \leq 1000000$. Additionally, Alice has created a signature (γ, δ_1) on a message x_1 and Bob has created a signature (γ, δ_2) on a message x_2. For simplicity, we assume that the scheme is used without a hash function (though this does not affect the attack).

Then it is almost always possible for an adversary to easily compute Alice's and Bob's secret keys (a_1 and a_2, respectively), without solving the corresponding instances of the **Discrete Logarithm** problem.

(a) Describe how an adversary can first compute $c = a_1 - a_2$ by trial and error, and then compute k, a_1, and a_2. Note that it is only necessary to solve one instance of the **Discrete Logarithm** problem, in order to compute c. Further, since c is small in absolute value, it is feasible to do this by trial and error.

HINT After computing c, consider the equations that are used to define γ and δ.

Answer: Since $\beta_1 = \alpha^{a_1}$ and $\beta_2 = \alpha^{a_2}$, we have $\beta_1(\beta_2)^{-1} = \alpha^{a_1 - a_2}$. We know that $-1000000 \leq a_1 - a_2 \leq 1000000$, so we can compute $c = a_1 - a_2$ by trial and error.
Next, we have

$$\begin{aligned} \delta_1 &\equiv (x_1 + a_1\gamma)k^{-1} \pmod q \\ \delta_2 &\equiv (x_2 + a_2\gamma)k^{-1} \pmod q, \end{aligned}$$

so

$$\begin{aligned} k\delta_1 &\equiv (x_1 + a_1\gamma) \pmod q \\ k\delta_2 &\equiv (x_2 + a_2\gamma) \pmod q, \end{aligned}$$

and hence

$$\begin{aligned} k(\delta_1 - \delta_2) &\equiv x_1 - x_2 + (a_1 - a_2)\gamma \pmod{q} \\ k &\equiv (x_1 - x_2 + (a_1 - a_2)\gamma)(\delta_1 - \delta_2)^{-1} \pmod{q}. \end{aligned}$$

Since $a_1 - a_2$ is known, we can easily compute k.
Finally, we have

$$\begin{aligned} (k\delta_1 - x_1)\gamma^{-1} &\equiv a_1 \pmod{q} \\ (k\delta_2 - x_2)\gamma^{-1} &\equiv a_2 \pmod{q}, \end{aligned}$$

so we can easily compute a_1 and a_2.

(b) Suppose that the scheme's parameters are as given below, the two messages are x_1 and x_2, and the signatures are (γ, δ_1) and (γ, δ_2), respectively:

$$\begin{aligned} p &= 19338503263980536075316384051532097468920304555923317073 \\ &\quad 178002594954294412967019 \\ q &= 563670397789087603574646815042556842 1101 \\ \alpha &= 12365662126104529836738929919772082437961500125989127 51 \\ &\quad 33080695670324331879 33534 \\ \beta_1 &= 101590186979186491501456484061936965361908389572617565 7 \\ &\quad 3369802996077953102386282 \\ \beta_2 &= 164343217638803565451481602247364908777541341830200409 4 \\ &\quad 6566045300953546115315558 \\ x_1 &= 331192885868376875495429415067779328 9209 \\ x_2 &= 213554626096795330541883925884865826 5210 \\ \gamma &= 361597028560280214854416249236103562 6294 \\ \delta_1 &= 116608185638931575529296193780223950 5182 \\ \delta_2 &= 317073940448416020133066165229016171 9950 \end{aligned}$$

Verify that Alice's and Bob's signatures are both valid.

Answer: To verify Alice's signature, we compute

$$\begin{aligned} e1 &= 2933305147062697044831568131046863892053 \\ e2 &= 468133888187043042673863698800226122615 \\ (\alpha^{e_1}\beta^{e_2} \bmod p) \bmod q) &= \gamma. \end{aligned}$$

To verify Bob's signature, we compute

$$\begin{aligned} e1 &= 1684911750407716503426711471288517214019 \\ e2 &= 2645780297597144863325958946863398980536 \\ (\alpha^{e_1}\beta^{e_2} \bmod p) \bmod q) &= \gamma. \end{aligned}$$

(c) Illustrate the attack by breaking the instance of *DSA* given above, computing Alice's and Bob's secret keys.

Answer: We obtain the following:

$$\begin{aligned} c &= 819673 \\ k &= 1509406723750396688218286967257406643471 \\ a_1 &= 438385395898820428637513987773739361362 \\ a_2 &= 438385395898820428637513987773738541689. \end{aligned}$$

8.10 Suppose that $x_0 \in \{0,1\}^*$ is a bitstring such that SHA3-224$(x_0) = 0\,0 \cdots 0$. Therefore, when used in *DSA* or *ECDSA*, we have that SHA3-224$(x_0) \equiv 0 \pmod q$.

(a) Show how it is possible to forge a *DSA* signature for the message x_0.

HINT Let $\delta = \gamma$, where γ is chosen appropriately.

Answer: Define $\gamma = \delta = \beta \bmod q$. Then $e_1 = 0$ and $e_1 = 1$, and

$$(\alpha^{e_1} \beta^{e_2} \bmod p) \bmod q = \beta \bmod q = \gamma,$$

so the signature is verified.

(b) Show how it is possible to forge an *ECDSA* signature for the message x_0.

Answer: Let $B = (u, v)$ and define $r = s = u \bmod q$. Then $i = 0$, $j = 1$ and $iA + jB = B = (u, v)$, so the signature is verified.

8.11 (a) We describe a potential attack against the *DSA*. Suppose that x is given, let $z = (\text{SHA3-224}(x))^{-1} \bmod q$, and let $\epsilon = \beta^z \bmod p$. Now suppose it is possible to find $\gamma, \lambda \in \mathbb{Z}_q{}^*$ such that

$$\left((\alpha\, \epsilon^{\gamma})^{\lambda^{-1} \bmod q}\right) \bmod p \bmod q = \gamma.$$

Define $\delta = \lambda\, \text{SHA3-224}(x) \bmod q$. Prove that (γ, δ) is a valid signature for x.

Answer: We have that $e_1 = \lambda^{-1} \bmod q$ and $e_2 = \gamma\lambda^{-1}z \bmod q$. Then (in $\mathbb{Z}_p$) we have that

$$\alpha^{e_1}\beta^{e_2} = \alpha^{\lambda^{-1} \bmod q}\beta^{\gamma\lambda^{-1}z \bmod q} = (\alpha\, \epsilon^{\gamma})^{\lambda^{-1} \bmod q},$$

and it is easily verified that the signature is valid.

(b) Describe a similar (possible) attack against the *ECDSA*.

Answer: Define $z = (\text{SHA3-224}(x))^{-1} \bmod q$, and let $C = zB$. Suppose it is possible to find $r, \lambda \in \mathbb{Z}_q{}^*$ such that

$$\lambda^{-1}(A + rC) = (u, v),$$

where $u \bmod q = r$. Define $s = \lambda\, \text{SHA3-224}(x) \bmod q$; then (r, s) is a valid signature for x.

8.12 In a verification of a signature constructed using the *ElGamal Signature Scheme* (or many of its variants), it is necessary to compute a value of the form $\alpha^c \beta^d$. If c and d are random ℓ-bit exponents, then a straightforward use of the SQUARE-AND-MULTIPLY algorithm would require (on average) $\ell/2$ multiplications and ℓ squarings to compute each of α^c and β^d. The purpose of this exercise is to show that the product $\alpha^c \beta^d$ can be computed much more efficiently.

(a) Suppose that c and d are represented in binary, as in Algorithm 6.5. Suppose also that the product $\alpha\beta$ is precomputed. Describe a modification of Algorithm 6.5, in which at most one multiplication is performed in each iteration of the algorithm.

Answer:

```
Algorithm: MODIFIED SQUARE-AND-MULTIPLY ($\alpha, \beta, c, d$)

  $\gamma \leftarrow \alpha\beta$
  $z \leftarrow 1$
  for $i \leftarrow \ell - 1$ downto 0
    do
      $z \leftarrow z^2$
      if $c_i = 1$
        then
          if $d_i = 1$
            then $z \leftarrow z\gamma$
            else $z \leftarrow z\alpha$
        else
          if $d_i = 1$
            then $z \leftarrow z\beta$
  return ($z$)
```

(b) Suppose that $c = 26$ and $d = 17$. Show how your algorithm would compute $\alpha^c \beta^d$, i.e., what are the values of the exponents i and j at the end of each iteration of your algorithm (where $z = \alpha^i \beta^j$).

Answer: The binary representations of c and d are (respectively) 11010 and 10001. After iteration $i = 4$, $z = \alpha\beta$. After iteration $i = 3$, $z = \alpha^3\beta^2$. After iteration $i = 2$, $z = \alpha^6\beta^4$. After iteration $i = 1$, $z = \alpha^{13}\beta^8$. After iteration $i = 0$, $z = \alpha^{26}\beta^{17}$.

(c) Explain why, on average, this algorithm requires ℓ squarings and $3\ell/4$ multiplications to compute $\alpha^c \beta^d$, if c and d are randomly chosen ℓ-bit integers.

Answer: The number of squarings is exactly ℓ. In any iteration of the algorithm, a multiplication is done if and only if $(c_i, d_i) \neq (0, 0)$. If we estimate that $(c_i, d_i) = (0, 0)$ occurs with probability $1/4$, then, on average, $3\ell/4$ multiplications are performed.

(d) Estimate the average speedup achieved, as compared to using the original SQUARE-AND-MULTIPLY algorithm to compute α^c and β^d separately, assuming that a squaring operation takes roughly the same time as a multiplication operation.

Answer: If we compute α^c and β^d separately and then multiply them together, then, on average, we require $3\ell/2 + 3\ell/2 + 1 = 3\ell + 1$ squarings and/or multiplications.

If we compute $\alpha^c \beta^d$ using the algorithm desribed above, then require $7\ell/4$ squarings and/or multiplications. The ratio of the running times of these two approaches is $(7\ell/4)/(3\ell+1) \approx 7/12$, so the speedup factor is $5/12$ or 42%.

8.13 Prove that a correctly constructed signature in the *ECDSA* will satisfy the verification condition.

Answer: We have that

$$\begin{aligned} iA + jB &= w\text{SHA3-224}(x)A + wrmA \\ &= s^{-1}(\text{SHA3-224}(x) + mr)A \\ &= kA. \end{aligned}$$

Therefore, $iA + jB = (u, v)$, where $u \bmod q = r$.

8.14 Let $\mathcal{E}$ denote the elliptic curve $y^2 \equiv x^3 + x + 26 \bmod 127$. It can be shown that $\#\mathcal{E} = 131$, which is a prime number. Therefore any non-identity element in $\mathcal{E}$ is a generator for $(\mathcal{E}, +)$. Suppose the *ECDSA* is implemented in $\mathcal{E}$, with $A = (2, 6)$ and $m = 54$.

(a) Compute the public key $B = mA$.

Answer: $B = (24, 44)$.

(b) Compute the signature on a message x if SHA3-224$(x) = 10$, when $k = 75$.

Answer: The signature is $(88, 60)$.

(c) Show the computations used to verify the signature constructed in part (b).

Answer: We have $w = 107$, $i = 22$, $j = 115$, $iA + jB = (88, 55) = (u, v)$, and then $r = 88 = u$, so the signature is verified.

8.15 This exercise looks at an RSA-type signature scheme due to Gennero, Halevi, and Rabin. Suppose $n = pq$, where p and q are distinct large safe primes. Let $h : \{0,1\}^* \to \mathbb{Z}_n$ be a public hash function with the property that h only takes on odd values. Let $s \in \mathbb{Z}_n{}^*$ be a random value. The public key consists of the hash function h, n, and s, and the private key is p, q. To sign a message x, perform the following computations:

1. Compute $e = h(s)$. **Note:** This is a typo; it should be $e = h(x)$.

2. Compute $f = e^{-1} \bmod \phi(n)$

3. Compute $y = s^f \bmod n$.

The signature is the value y.

(a) Explain why it is necessary that h only takes on odd values.

Answer: If $e = h(x)$ is even, then $\gcd(e, \phi(n)) > 1$ and $e^{-1} \bmod \phi(n)$ does not exist. The message x cannot be signed in this case.

(b) Derive a formula to verify a signature.

Answer: The verification formula is to check that $y^e \equiv s \bmod n$, where $e = h(x)$. Observe that

$$\begin{aligned} y^e &\equiv (s^f)^e \pmod{n} \\ &\equiv s^{fe} \pmod{n} \\ &\equiv s \pmod{n}, \end{aligned}$$

because $fe \equiv 1 \pmod{\phi(n)}$.

Chapter 9

Post-Quantum Cryptography

9.1 Compute

$$g(x) = (x^4 + 3x^3 + x^2 + 2x + 3)(2x^4 + 5x^2 + 6x + 2)$$

in $\mathbb{Z}[x]/(x^5 - 1)$. Compute $(1,3,1,2,3) \star (2,0,5,6,2)$ and confirm that the entries in the resulting vector correspond to the coefficients of g.

Answer: We have

$$g(x) = 31x^4 + 24x^3 + 35x^2 + 29x + 31$$

and

$$(1,3,1,2,3) \star (2,0,5,6,2) = (31,24,35,29,31).$$

9.2 Let $f(x) = 3x^5 + 3x + 1 \in \mathbb{Z}_{11}[x]$.

(a) Find polynomials $a(x)$ and $b(x)$ in $\mathbb{Z}_{11}[x]$ such that

$$a(x)f(x) + b(x)(x^7 - 1) = 1.$$

Answer: Using the Extended Euclidean Algorithm in $\mathbb{Z}_{11}[x]$, we obtain:

$$(8x^6 + 8x^4 + 4x^3 + 6x^2 + x + 3)f(x) + (9x^4 + 9x^2 + 10x + 2)(x^7 - 1) = 1.$$

(b) Determine the inverse of f in $\mathbb{Z}_{11}[x]/(x^7 - 1)$ mods 11.

Answer: From part (a), we immediately obtain

$$f^{-1}(x) = 8x^6 + 8x^4 + 4x^3 + 6x^2 + x + 3 \bmod 11.$$

Reducing the coefficients of f^{-1} mods 11, we get

$$f^{-1}(x) = -3x^6 + -3x^4 + 4x^3 + -5x^2 + x + 3 \text{ mods } 11.$$

9.3 Let $\{\mathbf{u}, \mathbf{v}\}$ be a basis for a lattice $\mathcal{L}$ in $\mathbb{R}^2$ where $\mathbf{u} = (3,7)$ and $\mathbf{v} = (5,10)$.

(a) Find the norm of $\mathbf{u}$ and the norm of $\mathbf{v}$.

Answer: $\|\mathbf{u}\| = \sqrt{58}$ and $\|\mathbf{v}\| = \sqrt{125}$.

(b) Determine the shortest vectors in $\mathcal{L}$.

Answer: The vector $2(5,10) - 3(3,7) = (1,-1)$ has norm $\sqrt{2}$. We claim that are no vectors in $\mathcal{L}$ having norm 1 or 0.

It is easy to see that there are no vectors of norm 0, because **u** and **v** are linearly independent. There are four possible vectors of norm 1, namely $(1,0)$, $(-1,0)$, $(0,1)$ and $(0,-1)$.

First, we consider $(1,0)$. Suppose that $(1,0) = a(5,10) + b(3,7)$. Then $5a + 3b = 1$ and $10a + 7b = 0$. This system has the unique solution $a = 7/5$, $b = -2$. Since a and b are not both integers, we conclude that $(1,0) \notin \mathcal{L}$.

The other three vectors of norm 1 can also be shown to not be in $\mathcal{L}$. Hence, we conclude that $(1,-1)$ is a shortest vector in $\mathcal{L}$. We also have that $(-1,1) \in \mathcal{L}$, because $-2(5,10) + 3(3,7) = (-1,1)$. The other two vectors of norm $\sqrt{2}$, namely $(1,1)$ and $(-1,-1)$, are not in $\mathcal{L}$.

9.4 Let $\mathcal{C}$ be the $[7,4,3]$ Hamming code with generating matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Find a parity-check matrix for $\mathcal{C}$ having the form $(Y \mid I_3)$, and use syndrome decoding to decode the received vector $(1,0,1,0,0,0,0)$.

Answer: The desired parity-check matrix is

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

To decode the received vector $(1,0,1,0,0,0,0)$, we compute

$$H(1,0,1,0,0,0,0)^T = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}.$$

This is the second column of H. Hence, we conclude that the second bit of the received vector is in error and we correct it to $(1,1,1,0,0,0,0)$.

9.5 Consider Cryptosystem 9.2 with the parameters $n = 3$ and $q = 17$, and public key consisting of the samples

$$((6,4,3),0), ((1,5,8),12) \text{ and } ((7,11,2),4).$$

(a) Encrypt the plaintext bit 1 using each of the seven possible nonempty choices of a random subset S.

Answer: We obtain the following possible encryptions of $x = 1$:

S	y
$\{1\}$	$((6,4,3),8)$
$\{2\}$	$((1,5,8),3)$
$\{3\}$	$((7,11,2),12)$
$\{1,2\}$	$((7,9,11),3)$
$\{1,3\}$	$((13,15,5),12)$
$\{2,3\}$	$((8,16,10),7)$
$\{1,2,3\}$	$((14,3,13),7)$

(b) Given that the private key is $(5,0,1)$, determine whether the decryption algorithm succeeds for each of the seven ciphertexts computed above.

Answer: The decryptions are as follows:

y	$b - \sum a_j s_j \bmod q$	decrypted ciphertext
$((6,4,3),8)$	9	1
$((1,5,8),3)$	7	1
$((7,11,2),12)$	9	1
$((7,9,11),3)$	8	1
$((13,15,5),12)$	10	1
$((8,16,10),7)$	8	1
$((14,3,13),7)$	9	1

Thus all seven encryptions of $x = 1$ decrypt correctly.

9.6 Suppose the elements of $\mathbb{F}_8$ are written in the form $a\theta^2 + b\theta + c$, where $a,b,c \in \mathbb{F}_2$ and where θ satisfies $\theta^3 + \theta + 1 = 0$. Express the equation $X^6 + 1 = 0$ over $\mathbb{F}_8$ as a system of multivariate polynomial equations over $\mathbb{F}_2$.

Answer: Let $X = a\theta^2 + b\theta + c$. Then

$$\begin{aligned} X^2 &= a\theta^4 + b\theta^2 + c \\ &= a(\theta^2 + \theta) + b\theta^2 + c \\ &= (a+b)\theta^2 + a\theta + c \end{aligned}$$

and

$$\begin{aligned} X^4 &= a\theta^8 + b\theta^4 + c \\ &= a\theta + b(\theta^2 + \theta) + c \\ &= b\theta^2 + (a+b)\theta + c. \end{aligned}$$

Now, we compute

$$\begin{aligned}
X^6 &= X^4X^2\\
&= ((a+b)\theta^2 + a\theta + c)(b\theta^2 + (a+b)\theta + c)\\
&= (ab+b)\theta^4 + ab\theta^3 + bc\theta^2 + (a+b)\theta^3 + (a+ab)\theta^2 + (ac+bc)\theta\\
&\quad +bc\theta^2 + (ac+bc)\theta + c\\
&= (ab+b)(\theta^2+\theta) + ab(\theta+1) + bc\theta^2 + (a+b)(\theta+1) + (a+ab)\theta^2\\
&\quad +(ac+bc)\theta + bc\theta^2 + (ac+bc)\theta + c\\
&= (a+b)\theta^2 + a\theta + ab + a + b + c
\end{aligned}$$

and hence

$$X^6 + 1 = (a+b)\theta^2 + a\theta + ab + a + b + c + 1.$$

The desired system of equations is

$$\begin{aligned}
a+b &= 0\\
a &= 0\\
ab+a+b+c+1 &= 0.
\end{aligned}$$

9.7 The ciphertext $(0,1,0)$ was obtained by performing HFE encryption with the parameters and public keys given in Example 9.7. Determine the corresponding plaintext.

Answer: Denote $\mathbf{y} = (0,1,0)$. First, we compute $R^{-1}(\mathbf{y}^T) = (0,1,1)^T$, which corresponds to the field element $\theta+1$. Next, we solve the equation $X^3 = \theta + 1$, obtaining $X = \theta$. Finally, the plaintext is $\mathbf{x} = S^{-1}((0,1,0)^T) = (1,1,1)$.

9.8 Using exhaustive search, determine all solutions to the following system of equations over $\mathbb{F}_2$:

$$\begin{aligned}
x_1x_2 + x_2x_3 + x_3 &= 0\\
x_1x_3 + x_1 + x_2 &= 1\\
x_2x_3 + x_1 &= 1.
\end{aligned}$$

Answer:

x_1	x_2	x_3	$x_1x_2 + x_2x_3 + x_3$	$x_1x_3 + x_1 + x_2$	$x_2x_3 + x_1$	solution?
0	0	0	0	0	0	no
0	0	1	1	0	0	no
0	1	0	0	1	0	no
0	1	1	0	1	1	yes
1	0	0	0	1	1	yes
1	0	1	1	0	1	no
1	1	0	1	0	1	no
1	1	1	1	1	0	no

9.9 Exploit the particular structure of the following system of equations over $\mathbb{F}_2$ in order to find a solution:

$$\begin{aligned} x_1x_2 + x_3x_4 + x_2 + x_5 &= 1 \\ x_2x_3 + x_2x_5 + x_1 &= 1 \\ x_1x_4 + x_2x_5 + x_6 &= 0. \end{aligned}$$

Answer: We can view x_1, x_2 and x_3 as vinegar variables and x_4, x_5 and x_6 as oil variables. Observe that there are no terms that are the product of two oil variables. If we fix values of the vinegar variables, then the resulting system of oil variables is a linear system. So we exhaustively test the $2^3 = 8$ possible combinations of vinegar variables, to see if a solution to the oil variables exists.

x_1	x_2	x_3	linear system	(x_4, x_5, x_6)
0	0	0	$x_5 = 1$, $0 = 1$, $x_6 = 0$	no solution
0	0	1	$x_4 + x_5 = 1$, $0 = 1$, $x_6 = 0$	no solution
0	1	0	$1 + x_5 = 1$, $x_5 = 1$, $x_5 + x_6 = 0$	no solution
0	1	1	$x_4 + 1 + x_5 = 1$, $1 + x_5 = 1$, $x_5 + x_6 = 0$	$(0,0,0)$
1	0	0	$x_5 = 1$, $1 = 1$, $x_4 + x_6 = 0$	$(0,1,0), (1,1,1)$
1	0	1	$x_4 + x_5 = 1$, $1 = 1$, $x_4 + x_6 = 0$	$(0,1,1), (1,0,0)$
1	1	0	$x_5 = 1$, $1 + x_5 = 1$, $x_4 + x_5 + x_6 = 0$	no solution
1	1	1	$x_4 + x_5 = 1$, $x_5 = 1$, $x_4 + x_5 + x_6 = 0$	$(0,1,1)$

9.10 In the *Lamport Signature Scheme*, suppose that two k-tuples, x and x', were signed by Alice using the same key. Let ℓ denote the number of co-ordinates in which x and x' differ, i.e.,

$$\ell = |\{i : x_i \neq x'_i\}|.$$

Show that Oscar can now sign $2^\ell - 2$ new messages.

Answer: For convenience, assume that

$$\{i : x_i \neq {x'}_i\} = \{1, \dots, \ell\}.$$

Then Oscar can create a signature for any message $z = (z_1, \dots, z_k)$ such that $z_i = x_i$ for $\ell + 1 \leq i \leq k$ and $z_i \in \{x_i, x'_i\}$ for $1 \leq i \leq \ell$. There are 2^ℓ such messages, two of which are x and x'. Thus the number of new messages that can be signed by Oscar is $2^\ell - 2$.

Chapter 10

Identification Schemes and Entity Authentication

10.1 Prove that it is impossible to design a secure two-flow mutual identification scheme based on random challenges. (A two-flow scheme consists of one flow from Bob to Alice (say) followed by a flow from Alice to Bob. In a mutual identification scheme, both parties are required to “accept” in order for a session to terminate successfully.)

Answer: Alice must “accept” after the first flow, since this is the only flow in which she receives information from Bob. Oscar can observe a successful session between Alice and Bob, and record the information X transmitted in the first flow. If Oscar later initiates a session with Alice, in which he pretends to be Bob, he can send the same information X to Alice, and Alice will accept. Therefore the scheme is insecure.

10.2 Consider the mutual identification scheme presented in Protocol 10.9. Prove that this scheme is insecure. (In particular, show that Olga can impersonate Bob by means of a certain type of parallel session attack, assuming that Olga has observed a previous session of the scheme between Alice and Bob.)

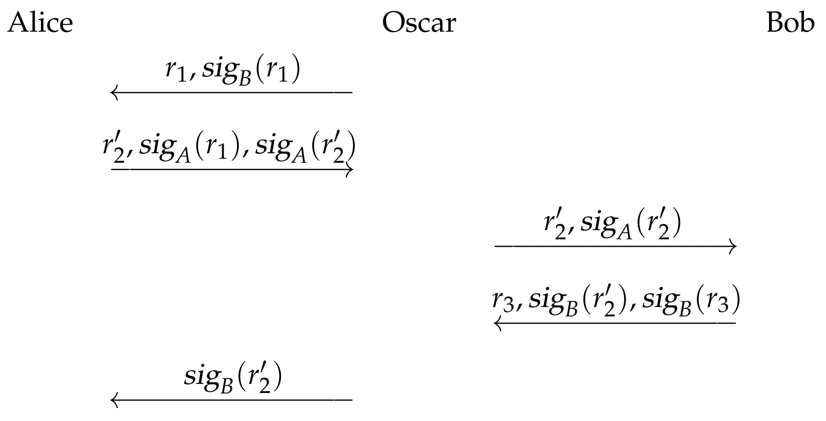
Answer:

First, Oscar observes a session $\mathcal{S}_1$ between Bob (the initiator) and Alice (the responder). In the first flow of $\mathcal{S}_1$, Bob chooses a random number r_1, computes a signature $y_1 = sig_{Bob}(r_1)$, and sends r_1 and y_1 to Alice. Oscar records the values of r_1, y_1, and $\mathbf{Cert}(Bob)$ from $\mathcal{S}_1$.

Later Oscar initiates a session $\mathcal{S}_2$ with Alice, in which he pretends to be Bob. He sends r_1, y_1, and $\mathbf{Cert}(Bob)$ to Alice in the first flow of $\mathcal{S}_2$. Alice responds with some information which includes a random challenge r_2' and a signature $y_3' = sig_{Alice}(r_2')$.

Now Oscar has to obtain Bob's signature on r_2'. Oscar does this by initiating another session, $\mathcal{S}_3$, with Bob, in which he (Oscar) pretends to be Alice. Oscar sends r_2', y_3', and $\mathbf{Cert}(Alice)$ to Bob in the first flow of $\mathcal{S}_3$. Bob's response includes $sig_{Bob}(r_2')$. Oscar can forward this response to Alice as the final flow of $\mathcal{S}_2$.

The attack is as follows, where r_1 and $sig_B(r_1)$ are copied from a previous session between Alice and Bob. (Note that other attacks are also possible.)



Protocol 10.9: INSECURE PUBLIC-KEY MUTUAL AUTHENTICATION

1. Bob chooses a random challenge, r_1. He also computes $y_1 = \mathbf{sig}_{Bob}(r_1)$ and he sends $\mathbf{Cert}(Bob)$, r_1 and y_1 to Alice.
2. Alice verifies Bob's public key, $\mathbf{ver}_{Bob}$, on the certificate $\mathbf{Cert}(Bob)$. Then she checks that $\mathbf{ver}_{Bob}(r_1, y_1) = true$. If not, then Alice "rejects" and quits. Otherwise, Alice chooses a random challenge, r_2. She also computes $y_2 = \mathbf{sig}_{Alice}(r_1)$ and $y_3 = \mathbf{sig}_{Alice}(r_2)$ and she sends $\mathbf{Cert}(Alice)$, r_2, y_2, and y_3 to Bob.
3. Bob verifies Alice's public key, $\mathbf{ver}_{Alice}$, on the certificate $\mathbf{Cert}(Alice)$. Then he checks that $\mathbf{ver}_{Alice}(r_1, y_2) = true$ and $\mathbf{ver}_{Alice}(r_2, y_3) = true$. If so, then Bob "accepts"; otherwise, Bob "rejects." Bob also computes $y_4 = \mathbf{sig}_{Bob}(r_2)$ and he sends y_4 to Alice.
4. Alice checks that $\mathbf{ver}_{Bob}(r_2, y_4) = true$. If so, then Alice "accepts"; otherwise, Alice "rejects."

10.3 Give a complete proof that Protocol 10.10 is secure. (This scheme is essentially identical to one of the schemes standardized in FIPS publication 196.)

Answer: We assume the signature scheme is (q, ϵ)***-secure***: the adversary cannot construct a valid signature for any new message with probability greater than ϵ, given that the adversary has previously seen valid signatures for at most q messages. This assumption holds for Alice's signature scheme, as well as for Bob's signature scheme.

Suppose the adversary has (passively) observed q sessions between Bob and Alice, in which there are a total of q messages signed by Bob and q messages signed by Alice. Then the adversary attempts to have Alice or Bob "accept" in some new session, say $\mathcal{S}_{new}$, in which the adversary is active.

Protocol 10.10: PUBLIC-KEY MUTUAL AUTHENTICATION (VERSION 2)

1. Bob chooses a random challenge, r_1. He sends $\mathbf{Cert}(Bob)$ and r_1 to Alice.
2. Alice chooses a random challenge, r_2. She also computes $y_1 = \mathbf{sig}_{Alice}(ID(Bob) \parallel r_1 \parallel r_2)$ and she sends $\mathbf{Cert}(Alice)$, r_2 and y_1 to Bob.
3. Bob verifies Alice's public key, $\mathbf{ver}_{Alice}$, on the certificate $\mathbf{Cert}(Alice)$. Then he checks that $\mathbf{ver}_{Alice}(ID(Bob) \parallel r_1 \parallel r_2, y_1) = true$. If so, then Bob "accepts"; otherwise, Bob "rejects." Bob also computes $y_2 = \mathbf{sig}_{Bob}(ID(Alice) \parallel r_2 \parallel r_1)$ and he sends y_2 to Alice.
4. Alice verifies Bob's public key, $\mathbf{ver}_{Bob}$, on the certificate $\mathbf{Cert}(Bob)$. Then she checks that $\mathbf{ver}_{Bob}(ID(Alice) \parallel r_2 \parallel r_1, y_2) = true$. If so, then Alice "accepts"; otherwise, Alice "rejects."

Suppose that the random challenges are (positive) k-bit integers. Under these conditions, we can easily give an upper bound on the adversary's probability of deceiving at least one of Alice or Bob. The adversary can reuse a signature from an old session if a challenge is repeated by coincidence, or he can try to guess (i.e., forge) a signature for a new challenge.

First we consider a new session $\mathcal{S}_{new}$ from Bob's point of view. He chooses r_1 and receives r_2 and $sig_{Alice}(ID(Bob) \parallel r_1 \parallel r_2)$. He has to consider the probability that this signature is

(a) newly created by Oscar,

(b) copied from the second flow of some other session $\mathcal{S}_{old}$, or

(c) copied from the third flow of some other session $\mathcal{S}_{old}$.

In case (a), Oscar can guess the new signature with probability at most ϵ. A correct guess will allow Oscar to deceive Bob.

In case (b), Oscar can copy the signature if Bob repeated his challenge r_1 from the session $\mathcal{S}_{old}$. In the session $\mathcal{S}_{old}$, Alice responded with r_2 and $y_1 = sig_{K_{Alice}}(Bob \parallel r_1 \parallel r_2)$, for some r_2. Oscar can use the same r_2 and y_1 in the session $\mathcal{S}_{new}$ to deceive Bob.

In case (c), the session $\mathcal{S}_{old}$ must have begun with the challenge r_2 having been received by Bob in the first flow, and the challenge r_1 having been chosen by Bob in the second flow. Therefore Bob must have repeated his challenge r_1 in the session $\mathcal{S}_{new}$.

In both of cases (b) and (c), Bob must repeat his challenge from a previous session. Therefore the probability of (b) or (c) holding is at most $q/2^k$.

Now we consider a session $\mathcal{S}_{new}$ from Alice's point of view. She receives r_1

and then she chooses r_2 and computes $sig_{Alice}(ID(Bob) \parallel r_1 \parallel r_2)$. Finally, she receives $sig_{Bob}(ID(Alice) \parallel r_2 \parallel r_1)$. She has to consider the probability that this signature is

(a) newly created by Oscar,

(b) copied from the second flow of some other session $\mathcal{S}_{old}$, or

(c) copied from the third flow of some other session $\mathcal{S}_{old}$.

In case (d), Oscar can guess a new signature, $sig_{K_{Bob}}(Alice \parallel r_2)$, with probability at most ϵ. A correct guess will allow Oscar to deceive Alice.

In case (e), the session $\mathcal{S}_{old}$ must have begun with the challenge r_2 having chosen by Alice in the first flow, and the challenge r_1 having been chosen by Bob in the second flow.

In case (f), Alice repeats one of her challenges, say r_2, from the session $\mathcal{S}_{old}$, is at most $q \times 2^{-k}$. In the session $\mathcal{S}_{old}$, Bob responded with $y_2 = sig_{K_{Bob}}(Alice \parallel r_2)$. Oscar can use the same y_2 in the session $\mathcal{S}_{new}$ to deceive Alice.

In both of cases (e) and (f), Alice must repeat her challenge from a previous session. Therefore the probability of (e) or (f) holding is at most $q/2^k$.

Summing up, Oscar's probability of deceiving Alice or Bob is at most $2 \times (q/2^k + \epsilon) = q/2^{k-1} + 2\epsilon$.

10.4 Discuss whether Protocol 10.11 is secure. (Certificates are omitted from its description, but they are assumed to be included in the scheme in the usual way.)

Protocol 10.11: UNKNOWN PROTOCOL

1. Bob chooses a random challenge, r_1, and he sends it to Alice.
2. Alice chooses a random challenge r_2, she computes $y_1 = \mathbf{sig}_{Alice}(r_1)$, and she sends r_2 and y_1 to Bob.
3. Bob checks that $\mathbf{ver}_{Alice}(r_1, y_1) = true$; if so, then Bob "accepts"; otherwise, Bob "rejects." Bob also computes $y_2 = \mathbf{sig}_{Bob}(r_2)$ and he sends y_2 to Alice.
4. Alice checks that $\mathbf{ver}_{Bob}(r_2, y_2) = true$. If so, then Alice "accepts"; otherwise, Alice "rejects."

Answer: The scheme is insecure. An attack is as follows.

$$\begin{tikzcd}
\text{Alice} & \text{Oscar} & \text{Bob} \\
{} & {} \arrow[l, "r_1"] & {} \\
{} \arrow[r, "sig_A(r_1), r_2"] & {} & {} \\
{} & {} \arrow[r, "r_2"] & {} \\
{} & {} & {} \arrow[l, "sig_B(r_2), r_3"] \\
{} & {} \arrow[l, "sig_B(r_2)"] & {}
\end{tikzcd}$$

10.5 Prove that Protocol 10.5 and Protocol 10.10 are both insecure if the identity of Alice (Bob, resp.) is omitted from the signature computed by Bob (Alice, resp.).

Answer: The attack on Protocol 10.5 is as follows. Oscar executes a legitimate session with Bob, and Oscar impersonates Bob in a parallel session with Alice.

$$\begin{tikzcd}
\text{Alice} & \text{Oscar} & \text{Bob} \\
{} & {} \arrow[l, "r_1"] & {} \arrow[l, "r_1"] \\
{} \arrow[r, "sig_A(r_1 \parallel r_2), r_2"] & {} \arrow[r, "sig_O(r_1 \parallel r_2), r_2"] & {} \\
{} & {} \arrow[l, "sig_B(r_2)"] & {} \arrow[l, "sig_B(r_2)"]
\end{tikzcd}$$

The attack on Protocol 10.10 is similar:

$$\begin{tikzcd}
\text{Alice} & \text{Oscar} & \text{Bob} \\
{} & {} \arrow[l, "r_1"] & {} \arrow[l, "r_1"] \\
{} \arrow[r, "sig_A(r_1 \parallel r_2), r_2"] & {} \arrow[r, "sig_O(r_1 \parallel r_2), r_2"] & {} \\
{} & {} \arrow[l, "sig_B(r_2 \parallel r_1)"] & {} \arrow[l, "sig_B(r_2 \parallel r_1)"]
\end{tikzcd}$$

10.6 Consider the following possible identification scheme. Alice possesses a secret key $n = pq$, where p and q are prime and $p \equiv q \equiv 3 \pmod 4$. The value of n will be stored on Alice's certificate. When Alice wants to identify herself to Bob, say, Bob will present Alice with a random quadratic residue modulo n, say x. Then Alice will compute a square root y of x and give it to Bob. Bob then verifies that $y^2 \equiv x \pmod n$. Explain why this scheme is insecure.

Answer: This is basically the same as the chosen-ciphertext attack against the Rabin Cryptosystem. Bob chooses a random $r \in \mathbb{Z}_n$. If $\gcd(r, n) > 1$ then Bob immediately obtains the factorization (this happens with negligible probability, however). Otherwise, Bob computes $x = r^2 \bmod n$, which is a random quadratic residue in $\mathbb{Z}_n$. Alice then gives Bob a value y which is a square root of x modulo n. Since Alice does not know the value of r that Bob initially chose, the probability that $y \not\equiv \pm r \bmod n$ is $1/2$. In this case, the computation of $\gcd(y + r, n)$ yields the factorization $n = pq$. Once Bob has this factorization, he can impersonate Alice.

10.7 Suppose Alice is using the *Schnorr Identification Scheme* where $q = 1201$, $p = 122503$, $t = 10$, and $\alpha = 11538$.

(a) Verify that α has order q in $\mathbb{Z}_p{}^*$.

Answer: Since q is prime, it suffices to verify that $\alpha^q \bmod p = 1$.

(b) Suppose that Alice's secret exponent is $a = 357$. Compute v.

Answer: $v = \alpha^{-357} \bmod p = 14320$.

(c) Suppose that $k = 868$. Compute γ.

Answer: $\gamma = \alpha^{868} \bmod p = 89937$.

(d) Suppose that Bob issues the challenge $r = 501$. Compute Alice's response y.

Answer: $y = k + 501a \bmod q = 776$.

(e) Perform Bob's calculations to verify y.

Answer: Bob verifies that

$$11538^{776} 14320^{501} \equiv 89937 \pmod{122503}.$$

10.8 Suppose that Alice uses the *Schnorr Identification Scheme* with p, q, t, and α as in the previous exercise. Now suppose that $v = 51131$, and Olga has learned that

$$\alpha^3 v^{148} \equiv \alpha^{151} v^{1077} \pmod{p}.$$

Show how Olga can compute Alice's secret exponent a.

Answer: Olga computes $a = (3 - 151)(148 - 1077)^{-1} \bmod 1201 = 441$. To see that this is correct, it suffices to verify that $11538^{-441} \bmod 122503 = 51131$.

10.9 Show that the *Schnorr Identification Scheme* is not secure against an active adversary who changes the messages that are sent from Alice to Bob.

Answer: There are many attacks; here is one. In step 1, Oscar intercepts γ and replaces it by $\gamma' = \alpha\gamma$. Then in step 3, Oscar intercepts y and replaces it by $y' = y + 1$. Since $\gamma = \alpha^y v^r$, it follows that

$$\gamma' = \alpha\gamma = \alpha\alpha^y v^r = \alpha^{y+1} v^r = \alpha^{y'} v^r.$$

10.10 Consider the following identification scheme. Alice has a public key $v = g^a$ and a private key a. Assume that g is a primitive element in $\mathbb{Z}_p{}^*$ where p is prime, and assume that the **Discrete Logarithm** problem in $\mathbb{Z}_p{}^*$ is infeasible. Bob chooses a random b, computes $w = g^b$, and sends w to Alice. Alice computes $K = w^a$ and sends it to Bob. Bob accepts if and only if $K = v^b$. Prove that the above-described scheme zero-knowledge against an honest verifier (i.e., a verifier Bob who chooses the challenges w as described above). That is, show that it is possible to simulate transcripts of Bob's view of the protocol.

Answer: A transcript representing the view of an *honest* verifier can be written as a triple (b, w, k), where b is random, $w = g^b$ and $k = v^b$. Clearly the honest verifier can generate random triples of this form, so the scheme is zero-knowledge for an honest verifier.

10.11 Prove that Protocol 10.10 is not secure if ID strings and random challenges are not required to have a prespecified, fixed length.

HINT Recall that, before the protocol is executed, each user claims an identity to the other user, so each user has an "intended peer." The protocol will proceed only if these two users have a shared secret key. You can assume that the users' IDs are represented in ASCII. Consider the situation where the two users who share a key K are Ali and Alice.

Answer: Oscar can successfully impersonate Alice in a session where he identifies himself to Ali. Ali sends a challenge r to Oscar (who is pretending to be Alice). Oscar computes $r' = ce \parallel r$ (observe that $Ali \parallel r' = Alice \parallel r$). Now Oscar initiates a parallel session with Ali, sending the challenge r' to Ali. Ali computes $y = MAC_K(Ali \parallel r')$ and sends it to Oscar. This completes the parallel session. Now, in the main session, Oscar sends y to Ali. Ali accepts the response y because $MAC_K(Ali \parallel r') = MAC_K(Alice \parallel r)$.

Chapter 11

Key Distribution

11.1 Suppose that $p = 150001$ and $\alpha = 7$ in the *Diffie-Hellman Key Predistribution Scheme*. (It can be verified that α is a generator of $\mathbb{Z}_p{}^*$.) Suppose that the private keys of U, V, and W are $a_U = 101459$, $a_V = 123967$, and $a_W = 99544$.

(a) Compute the public keys of U, V, and W.

Answer: The public keys are $b_U = \alpha^{a_U} \bmod p = 7^{101459} \bmod 150001 = 36138$, $b_V = \alpha^{a_V} \bmod p = 7^{123967} \bmod 150001 = 106355$, and $b_W = \alpha^{a_W} \bmod p = 7^{99544} \bmod 150001 = 93325$.

(b) Show the computations performed by U to obtain $K_{U,V}$ and $K_{U,W}$.

Answer: U computes $K_{U,V} = b_V{}^{a_U} \bmod p = 106355^{101459} \bmod 150001 = 75452$ and $K_{U,W} = b_W{}^{a_U} \bmod p = 93325^{123967} \bmod 150001 = 119360$.

(c) Verify that V computes the same key $K_{U,V}$ as U does.

Answer: V computes $K_{U,V} = b_U{}^{a_V} \bmod p = 36138^{101459} \bmod 150001 = 75452$.

(d) Explain why $\mathbb{Z}_{150001}{}^*$ is a very poor choice of a setting for the *Diffie-Hellman Key Predistribution Scheme* (notwithstanding the fact that p is too small for the scheme to be secure).

HINT Consider the factorization of $p - 1$.

Answer: $150001 - 1 = 3 \times 2^4 \times 5^5$, so the **Discrete Log** problem in $\mathbb{Z}_{150001}{}^*$ can easily be solved by the POHLIG-HELLMAN algorithm.

11.2 Suppose the *Blom KPS* with $k = 2$ is implemented for a set of five users, U, V, W, X, and Y. Suppose that $p = 97$, $r_U = 14$, $r_V = 38$, $r_W = 92$, $r_X = 69$ and $r_Y = 70$. The secret g polynomials are as follows:

$$\begin{array}{ll} g_U(x) = 15 + 15x + 2x^2 & g_V(x) = 95 + 77x + 83x^2 \\ g_W(x) = 88 + 32x + 18x^2 & g_X(x) = 62 + 91x + 59x^2 \\ g_Y(x) = 10 + 82x + 52x^2. & \end{array}$$

(a) Compute the keys for all $\binom{5}{2} = 10$ pairs of users.

Answer: We have

$$\begin{aligned}
K_{U,V} &= 78\\
K_{U,W} &= 87\\
K_{U,X} &= 96\\
K_{U,Y} &= 1\\
K_{V,W} &= 39\\
K_{V,X} &= 58\\
K_{V,Y} &= 32\\
K_{W,X} &= 15\\
K_{W,Y} &= 27\\
K_{X,Y} &= 70.
\end{aligned}$$

(b) Verify that $K_{U,V} = K_{V,U}$.

Answer:

$$K_{U,V} = g_U(r_V) = 15 + 15 \times 38 + 2 \times 38^2 \bmod 97 = 78$$

and

$$K_{V,U} = g_V(r_U) = 95 + 77 \times 14 + 83 \times 14^2 \bmod 97 = 78.$$

11.3 Suppose that the *Blom KPS* is implemented with security parameter k. Suppose that a coalition of k users, say $W_1, \dots, W_k$, pool their secret information. Additionally, assume that a key $K_{U,V}$ is exposed, where U and V are two other users.

(a) Describe how the coalition can determine the polynomial $g_U(x)$ by polynomial interpolation, using known values of $g_U(x)$ at $k+1$ points.

Answer: The coalition knows the k values

$$\begin{aligned}
g_U(r_{W_1}) &= g_{W_1}(r_U)\\
g_U(r_{W_2}) &= g_{W_2}(r_U)\\
\vdots \quad & \quad \vdots\\
g_U(r_{W_k}) &= g_{W_k}(r_U).
\end{aligned}$$

Additionally, the coalition knows

$$g_U(r_V) = K_{U,V}.$$

Therefore they have $k+1$ points on the polynomial $g_U(x)$, which has degree at most k, and the coalition can compute $g_U(x)$ by the interpolation formula.

(b) Having computed $g_U(x)$, describe how the coalition can compute the bivariate polynomial $f(x,y)$ by bivariate polynomial interpolation.

Answer: The coalition knows the $k+1$ polynomials

$$\begin{aligned} g_{W_1}(x) &= f(x, r_{W_1}) \\ g_{W_2}(x) &= f(x, r_{W_2}) \\ \vdots \quad & \quad \vdots \\ g_{W_k}(x) &= f(x, r_{W_k}), \quad \text{and} \\ g_U(x) &= f(x, r_U). \end{aligned}$$

Therefore the coalition can compute $f(x,y)$ by the bivariate interpolation formula.

(c) Illustrate the preceding two steps, by determining the polynomial $f(x,y)$ in the sample implementation of the *Blom KPS* where $k = 2$, $p = 34877$, and $r_i = i$ ($1 \leq i \leq 4$), supposing that

$$\begin{aligned} g_1(x) &= 13952 + 21199x + 19701x^2, \\ g_2(x) &= 25505 + 24549x + 15346x^2, \quad \text{and} \\ K_{3,4} &= 9211. \end{aligned}$$

Answer: First, we compute $g_3(x)$. We have $g_3(r_1) = g_1(r_3) = 10719$, $g_3(r_2) = g_2(r_3) = 28004$ and $g_3(r_4) = K_{3,4} = 9211$. Then, interpolating the points $(1, 10719)$, $(2, 28004)$ and $(4, 9211)$, we obtain

$$g_3(x) = 31796x^2 + 26528x + 22149.$$

Then we can interpolate the three polynomials $g_1(x)$, $g_2(x)$ and $g_3(x)$ using the bivariate Lagrange interpolation formula to compute

$$\begin{aligned} f(x,y) = \; & (27841x^2 + 16753x + 9984)y^2 + (16753x^2 + 22845x + 16478)y \\ & +9984x^2 + 16478x + 22367. \end{aligned}$$

11.4 Consider the *Lee-Stinson Linear KPS* with parameters p and k.

(a) Suppose $K_{i,j}$ and $K_{i',j'}$ are two keys in the scheme, where $i \neq i'$. Prove that there is a unique node in the scheme that contains both of these keys.

Answer: $U_{a,b}$ possesses $K_{i,j}$ and $K_{i',j'}$ if and only if $j = ai + b \bmod p$ and $j' = ai' + b \bmod p$. We have two linear equations in the two unknowns a and b. This system has the unique solution $a = (j - j')(i - i')^{-1} \bmod p$ and $b = j - i(j - j')(i - i')^{-1} \bmod p$. Note that $(i - i')^{-1} \bmod p$ exists because $i \neq i'$ and p is prime.

(b) Suppose that $U_{a,b}$ and $U_{a',b'}$ are two nodes that do not have a common key. Prove that there are exactly $(k-1)^2$ nodes that have a common key with $U_{a,b}$ and a (different) common key with $U_{a',b'}$.

Note: There is a typo in this part of the question: the number of nodes having the stated properties is $k^2 - k$.

Answer: Consider $U_{a,b}$ and $U_{a',b'}$, where $(a, b) \neq (a', b')$. These two nodes have at most one common key, by part (a). They have a common key $K_{i,j}$ if $ai + b \equiv a'i + b' \pmod p$ for some i. This is equivalent to $(a - a')i \equiv b' - b \pmod p$. This equation has a solution for i if and only if $a \neq a'$. When $a = a'$ and $b \neq b'$, the equation has no solution and the two given nodes do not have a common key.

So, suppose $a = a'$ and $b \neq b'$. Consider a key $K_{i,j}$ held by $U_{a,b}$ and a key $K_{i',j'}$ held by $U_{a',b'}$. There are k^2 such pairs of keys. If $i = i'$ there is no node that holds both of these keys, since the system of equations analyzed in part (a) will not have a solution. For each of the remaining $k^2 - k$ pairs of keys, we proved in part (a) that there is a unique node that holds both keys.

11.5 We describe a secret-key based three-party session key distribution scheme in Protocol 11.8. In this scheme, K_{Alice} is a secret key shared by Alice and the *TA*, and K_{Bob} is a secret key shared by Bob and the *TA*.

(a) State all consistency checks that should be performed by Alice, Bob, and the *TA* during a session of the protocol.

Answer: The *TA* receives two identifying strings, say $ID(Alice)$ and $ID(Bob)$, in the clear. Then the *TA* should verify that $d_{K_{Alice}}(y_A)$ has the form

$$(ID(Alice) \parallel ID(Bob) \parallel r_A);$$

and $d_{K_{Bob}}(y_B)$ has the form

$$(ID(Alice) \parallel ID(Bob) \parallel r_B);$$

where r_A and r_B are arbitrary.

Alice should verify that $d_{K_{Alice}}(z_A)$ has the form $(r_A \parallel K)$, where r_A was the challenge that she chose in step 1.

Bob should verify that $d_{K_{Bob}}(z_A)$ has the form $(r_B \parallel K)$, where r_B was the challenge that he chose in step 2.

(b) The protocol is vulnerable to an attack if the *TA* does not perform the necessary consistency checks you described in part (a). Suppose that Oscar replaces $ID(Bob)$ by $ID(Oscar)$, and he also replaces y_B by

$$y_O = e_{K_{Oscar}}(ID(Alice) \parallel ID(Bob) \parallel r'_B)$$

in step 2, where r'_B is random. Describe the possible consequences of this attack if the *TA* does not carry out its consistency checks properly.

Answer: The *TA* receives identity strings $ID(Alice)$, $ID(Oscar)$, so it computes

$$d_{K_{Alice}}(y_A) = (ID(Alice) \parallel ID(Bob) \parallel r_A)$$

Protocol 11.8: SESSION KEY DISTRIBUTION SCHEME

1. Alice chooses a random number, r_A. Alice sends $ID(Alice)$, $ID(Bob)$, and
$$y_A = e_{K_{Alice}}(ID(Alice) \parallel ID(Bob) \parallel r_A)$$
to Bob.

2. Bob chooses a random number, r_B. Bob sends $ID(Alice)$, $ID(Bob)$, y_A and
$$y_B = e_{K_{Bob}}(ID(Alice) \parallel ID(Bob) \parallel r_B)$$
to the *TA*.

3. The *TA* decrypts y_A using the key K_{Alice} and it decrypts y_B using the key K_{Bob}, thus obtaining r_A and r_B. It chooses a random session key, K, and computes
$$z_A = e_{K_{Alice}}(r_A \parallel K)$$
and
$$z_B = e_{K_{Bob}}(r_B \parallel K).$$
z_A is sent to Alice and z_B is sent to Bob.

4. Alice decrypts z_A using the key K_{Alice}, obtaining K; and Bob decrypts z_B using the key K_{Bob}, obtaining K.

and
$$d_{K_{Oscar}}(y_O) = (ID(Alice) \parallel ID(Bob) \parallel r'_B).$$
The *TA* should reject because $ID(Bob)$ occurs in the decryption of the strings y_A and y_O (rather than $ID(Oscar)$). If the *TA* fails to perform this check, it may go ahead and compute
$$z_A = e_{K_{Alice}}(r_A \parallel K)$$
and
$$z_O = e_{K_{Oscar}}(r'_B \parallel K).$$
z_A is sent to Alice and z_O is sent to Oscar. Alice and Oscar both obtain the key K, but Alice thinks she shares a session key with Bob. If Alice encrypts information using the key K, Oscar will be able to decrypt it and Bob will not be able to do so.

(c) In this protocol, encryption is being done to ensure both confidentiality and data integrity. Indicate which pieces of data require encryption for the purposes of confidentiality, and which ones only need to be authenticated. Rewrite the protocol, using MACs for authentication in the appropriate places. In general, the first transmission of challenges and IDs does not need to be encrypted or authenticated. (Authentication does not achieve anything useful, because the adversary can reuse values from a previous session, even if they are authenticated.) Responses to challenges (including IDs) need to be authenticated (not encrypted). Session keys should be encrypted, and the encrypted session keys should be authenticated, as shown in the following modified protocol.

Protocol: SESSION KEY DISTRIBUTION SCHEME (MODIFIED)

1. Alice chooses a random number, r_A. Alice sends $ID(Alice)$, $ID(Bob)$ and r_A to Bob.

2. Bob chooses a random number, r_B. Bob sends $ID(Alice)$, $ID(Bob)$, r_A and r_B to the *TA*.

3. The *TA* chooses a random session key, *K*, and computes

$$\begin{aligned} z_A &= e_{K_{Alice}}(K), \\ w_A &= MAC_{Alice}(ID(Alice) \parallel ID(Bob) \parallel z_A \parallel r_A), \\ z_B &= e_{K_{Bob}}(K), \quad \text{and} \\ w_B &= MAC_{Bob}(ID(Alice) \parallel ID(Bob) \parallel z_B \parallel r_B) \end{aligned}$$

   (z_A, w_A) is sent to Alice and (z_B, w_B) is sent to Bob.

4. Alice decrypts z_A using the key K_{Alice}, obtaining K; and Bob decrypts z_B using the key K_{Bob}, obtaining K. Alice and Bob also verify the MACs, w_A and w_B, respectively.

11.6 We describe a public-key protocol, in which Alice chooses a random session key and transmits it to Bob in encrypted form, in Protocol 11.9 (this is another example of key transport). In this scheme, K_{Bob} is Bob's public encryption key. Alice and Bob also have private signing keys and public verification keys for a signature scheme.

(a) Determine if the above protocol is a secure mutual identification scheme. If it is, then analyze an active adversary's probability of successfully deceiving Alice or Bob, given suitable assumptions on the security of the signature scheme. If it is not, then demonstrate an attack on the scheme.

Answer: The scheme is a secure mutual identification scheme. The analysis of deception probabilities of an active adversary is similar to anal-

Protocol 11.9: PUBLIC-KEY KEY TRANSPORT SCHEME

1. Bob chooses a random challenge, r_1. He sends r_1 and $\mathbf{Cert}(Bob)$ to Alice.
2. Alice verifies Bob's public encryption key, K_{Bob}, on the certificate $\mathbf{Cert}(Bob)$. Then Alice chooses a random session key, K, and computes
$$z = e_{K_{Bob}}(K).$$
She also computes
$$y_1 = \mathbf{sig}_{Alice}(r_1 \parallel z \parallel ID(Bob))$$
and sends $\mathbf{Cert}(Alice)$, z and y_1 to Bob.
3. Bob verifies Alice's public verification key, $\mathbf{ver}_{Alice}$, on the certificate $\mathbf{Cert}(Alice)$. Then he verifies that
$$\mathbf{ver}_{Alice}(r_1 \parallel z \parallel ID(Bob), y_1) = true.$$
If this is not the case, then Bob "rejects." Otherwise, Bob decrypts z obtaining the session key K, and "accepts." Finally, Bob computes
$$y_2 = \mathbf{sig}_{Bob}(z \parallel ID(Alice))$$
and sends y_2 to Alice.
4. Alice verifies Bob's public verification key, $\mathbf{ver}_{Bob}$, on the certificate $\mathbf{Cert}(Bob)$. Then she checks that
$$\mathbf{ver}_{Bob}(z \parallel ID(Alice), y_2) = true.$$
If so, then Alice "accepts"; otherwise, Alice "rejects."

yses considered in previously studied protocols. We assume the signature scheme is (q, ϵ)***-secure***: the adversary cannot construct a valid signature for any new message with probability greater than ϵ, given that the adversary has previously seen valid signatures for at most q messages. This assumption holds for Alice's signature scheme, as well as for Bob's signature scheme.

Suppose the adversary has (passively) observed q sessions between Bob and Alice, in which there are a total of q messages signed by Bob and q messages signed by Alice. Then the adversary attempts to have Alice or Bob "accept" in some new session, say $\mathcal{S}_{new}$, in which the adversary is active.

Suppose that Bob's challenge is a random k-bit integer, and that z (the encryption of a random key K) is also a random k-bit integer. Observe that z functions as Alice's challenge for purposes of identification. Under these assumptions, we can easily give an upper bound on the adversary's probability of deceiving at least one of Alice or Bob. The adversary can reuse a signature from an old session if a challenge is repeated by coincidence, or he can try to guess (i.e., forge) a signature for a new challenge.

i. The probability that Bob repeats one of his challenges, say r_1, from a previous session, say $\mathcal{S}$, is at most $q \times 2^{-k}$. In the session $\mathcal{S}$, Alice responded with z and $y_1 = \mathbf{sig}_{K_{Alice}}(r_1 \parallel z \parallel ID(Bob))$, for some z. Oscar can use the same z and y_1 in the session $\mathcal{S}_{new}$ to deceive Bob.
ii. The probability that Alice repeats one of her challenges, say z, from a previous session, say $\mathcal{S}$, is at most $q \times 2^{-k}$. In the session $\mathcal{S}$, Bob responded with $y_2 = \mathbf{sig}_{K_{Bob}}(z \parallel ID(Alice))$. Oscar can use the same y_2 in the session $\mathcal{S}_{new}$ to deceive Alice.
iii. Oscar can guess a new signature, $\mathbf{sig}_{K_{Alice}}(r \parallel z \parallel ID(Bob))$, with probability at most ϵ. A correct guess will allow Oscar to deceive Bob.
iv. Oscar can guess a new signature, $\mathbf{sig}_{K_{Bob}}(z \parallel ID(Alice))$, with probability at most ϵ. A correct guess will allow Oscar to deceive Alice.

Summing up, Oscar's probability of deceiving Alice or Bob is at most $2 \times (q/2^k + \epsilon) = q/2^{k-1} + 2\epsilon$.

(b) What type of key authentication or confirmation is provided by this protocol (from Alice to Bob, and from Bob to Alice)? Justify your answer briefly.

Answer: There is a known session key attack against the scheme. First, Oscar records a session $\mathcal{S}_1$ between Alice and Bob. In this session, Alice chooses a key and sends an encrypted version of it, denoted z to Bob.

Next, Oscar carries out a session $\mathcal{S}_2$ with Bob. Oscar identifies himself correctly, so $\mathcal{S}_2$ is a legitimate session. In this session, Oscar sends to Bob the same value z that Alice used in the session $\mathcal{S}_1$.

After session $\mathcal{S}_2$ is completed successfully (Oscar and Bob both "accept"), Oscar does not know the value of K. However, because we are considering the known session key attack model, Oscar is allowed to ask for the key for the session $\mathcal{S}_2$ to be revealed. So Oscar is given the decrypted key $K = d_{K_{Bob}}(z)$. This key, K, is also the key for the session $\mathcal{S}_1$, so Oscar has carried out a successful attack.

Because Oscar can learn the value of a key in a session between Alice and Bob in a certain attack model, it follows that no level of key authentication or confirmation is achieved in the protocol. Even though the scheme is a secure identification scheme (which prevents active attacks) the scheme does not fulfill the requirements of a secure key distribution scheme.

There are various ways to fix the flaw in the scheme. Perhaps the easiest is to use K as a "pre-key", and then derive the actual key by concatenating some identifying information and hashing the result. For example, the real session key for a session $\mathcal{S}$ can be computed (by Alice and Bob) to be $K_{\mathcal{S}} = h(ID(Alice) \parallel ID(Bob) \parallel K)$. Then, knowing the value of the session key $K_{\mathcal{S}_2} = h(ID(Oscar) \parallel ID(Bob) \parallel K)$ does help Oscar in computing the session key $K_{\mathcal{S}_1} = h(ID(Alice) \parallel ID(Bob) \parallel K)$. This is because Oscar no longer learns the value of K in the known session key attack.

11.7 Suppose we want to simultaneously revoke r users, say $U_{i_1}, \dots, U_{i_r}$, in the *Logical Key Hierarchy*. Assuming that the tree depth is equal to d and the tree nodes are labeled as described in Section 11.4, we can assume that $2^d \leq U_{i_1} < \cdots < U_{i_r} \leq 2^{d+1} - 1$.

(a) Informally describe an algorithm that can be used to determine which keys in the tree need to be updated.

Answer: For a node that is not a leaf node, let $\mathsf{left}(\cdot)$ and $\mathsf{right}(\cdot)$ denote the left and right children (respectively) of the given node.
Define a set $\mathcal{D}$ recursively, as follows:

$U_{i_j} \in \mathcal{D}, 1 \leq j \leq r$
for all non-leaf nodes $X, \mathsf{left}(X), \mathsf{right}(X) \in \mathcal{D} \Rightarrow X \in \mathcal{D}$.

The set $\mathcal{D}$ contains all nodes in the tree for which keys are no longer required. It can easily computed in a bottom-up fashion, as follows:

```
(1) $\mathcal{D} \leftarrow \{U_{i_j} : 1 \leq j \leq r\}$
(2) for $X \leftarrow 2^d - 1$ downto 1
      do if $\{2X, 2X+1\} \subseteq \mathcal{D}$
      then $\mathcal{D} \leftarrow \mathcal{D} \cup \{X\}$
```

The keys to be updated are those in the set

$$\mathcal{Q} = \left(\bigcup_{j=1}^{r} \mathcal{P}(U_{i_j}) \right) \setminus \mathcal{D},$$

where, for any leaf node U, $\mathcal{P}(U)$ denotes the set of nodes in the unique path from the node U to the root node R.

(b) Describe the broadcast that is used to update the keys. Which keys are used to encrypt the new, updated keys?

Answer: Define $\mathcal{Q}$ and $\mathcal{D}$ as above. For all nodes $X \in \mathcal{Q}$ and for $Y \in \{\mathsf{left}(X), \mathsf{right}(X)\}$, the following item is broadcast by the *TA*:

$$\begin{array}{ll} e_{k'(Y)}(k'(X)) & \text{if } Y \in \mathcal{Q} \\ e_{k(Y)}(k'(X)) & \text{if } Y \notin \mathcal{D} \cup \mathcal{Q} \\ \emptyset & \text{if } Y \in \mathcal{D} \end{array}$$

(c) Illustrate your algorithm by describing the updated keys and the broadcast if users 18, 23, and 29 are to be revoked in a tree with depth $d = 4$ (this tree is depicted in Figure 9.1. How much smaller is the broadcast in this case, as compared to the three broadcasts that would be required to revoke these three users one at a time in the basic *Logical Key Hierarchy*?

Answer: Here, $\mathcal{Q} = \{1, 2, 3, 4, 5, 7, 9, 11, 14\}$ and $\mathcal{D} = \{18, 23, 29\}$.
The following items are broadcast:

$$\begin{array}{ll}
e_{k'(2)}(k'(1)) & e_{k'(3)}(k'(1)) \\
e_{k'(4)}(k'(2)) & e_{k'(5)}(k'(2)) \\
e_{k(6)}(k'(3)) & e_{k'(7)}(k'(3)) \\
e_{k(8)}(k'(4)) & e_{k'(9)}(k'(4)) \\
e_{k(10)}(k'(5)) & e_{k'(11)}(k'(5)) \\
e_{k'(14)}(k'(7)) & e_{k(15)}(k'(7)) \\
e_{k(19)}(k'(9)) & \\
e_{k(22)}(k'(11)) & \\
e_{k(28)}(k'(14)) &
\end{array}$$

The size of the broadcast is reduced from $3 \times 7 = 21$ to 15 in this example.

11.8 Write a computer program to compute the key for the *Shamir* (t, w)-*Threshold Scheme* implemented in $\mathbb{Z}_p$. That is, given t public x-coordinates, $x_1, x_2, \ldots, x_t$, and t y-coordinates $y_1, \ldots, y_t$, compute the resulting key using the Lagrange interpolation formula.

(a) **Note:** A typo in this question has been corrected: the shares are named $y_1, \ldots, y_{10}$.
Test your program if $p = 31847$, $t = 5$, and $w = 10$, with the following shares:

$$\begin{array}{lcrlcr}
x_1 & = & 413 & y_1 & = & 25439 \\
x_2 & = & 432 & y_2 & = & 14847 \\
x_3 & = & 451 & y_3 & = & 24780 \\
x_4 & = & 470 & y_4 & = & 5910 \\
x_5 & = & 489 & y_5 & = & 12734 \\
x_6 & = & 508 & y_6 & = & 12492 \\
x_7 & = & 527 & y_7 & = & 12555 \\
x_8 & = & 546 & y_8 & = & 28578 \\
x_9 & = & 565 & y_9 & = & 20806 \\
x_{10} & = & 584 & y_{10} & = & 21462
\end{array}$$

Verify that the same key is computed by using several different subsets of five shares.

Answer: The key is 31318.

(b) Having determined the key, compute the share that would be given to a participant with x-coordinate equal to 10000. (Note that this can be done without computing the whole secret polynomial $a(x)$.)

Answer: Basically, it suffices to evaluate the Lagrange interpolation formula, setting $x = 10000$. Any five shares can be used. If desired, we can use $(0, 31318)$ as one of these shares.

The desired share is computed to be 24834.

11.9 (a) Suppose that the following are the nine shares in a $(5, 9)$-*Shamir Threshold Scheme* implemented in $\mathbb{Z}_{94875355691}$:

i	x_i	y_i
1	11	537048626
2	22	89894377870
3	33	65321160237
4	44	18374404957
5	55	24564576435
6	66	87371334299
7	77	60461341922
8	88	10096524973
9	99	81367619987

Exactly one of these shares is defective (i.e., incorrect). Your task is to determine which share is defective, and then figure out its correct value, as well as the value of the secret.

The "primitive operations" in your algorithm are polynomial interpolations and polynomial evaluations. Try to minimize the number of polynomial interpolations you perform.

HINT The question can be answered using at most three polynomial interpolations.

Answer: We assume that exactly one of the ten shares is incorrect. Suppose we interpolate five shares to construct a polynomial f of degree four. There are two possibilities:

case 1 The five shares are all correct. In this case, four of the remaining five shares will lie on the polynomial f.

case 2 The five shares contain one incorrect share. Then none of the remaining shares will lie on the polynomial f, since f can take on the same value as the correct polynomial for at most four points.

Thus, for a given set of five shares, we can distinguish between case 1 and case 2 using one interpolation and two polynomial evaluations.

Suppose we start with shares $1, \ldots, 5$. If we are unlucky, we are in case 2, which would require choosing a second set of five shares, WLOG, shares $5, \ldots, 9$. If we are still in case two, then this means that share 5 is incorrect. So we try again, a third time, say with shares $1, 2, 3, 4, 6$. This is guaranteed to work.

The following is obtained when we carry out these computations.

Interpolating the first five shares, we get

$$\begin{aligned} f_1(x) &= 71897634192x^4 + 3443928561x^3 + 80026893211x^2 \\ &\quad +49864276750x + 69147041214. \end{aligned}$$

None of the last four shares are on this polynomial.

Interpolating the last five shares, we get

$$\begin{aligned} f_2(x) &= 59044943323x^4 + 88984944477x^3 + 12459873939x^2 \\ &\quad +234698644x + 1024975809. \end{aligned}$$

Interpolating shares $1, 2, 3, 4, 6$, we get

$$\begin{aligned} f_3(x) &= 71897634192x^4 + 22112608111x^3 + 35706149573x^2 \\ &\quad +25129729395x + 45623204649. \end{aligned}$$

Shares 7,8 and 9 are on this polynomial, but share 5 is not. Therefore share 5 is the incorrect share.

The correct value of the fifth share can be obtained by computing $f_3(33) = 51317866721$.

(b) Suppose that a (t, w)-*Shamir Threshold Scheme* has exactly one defective share, and suppose that $w - t \geq 2$. Describe how it is possible to determine which share is defective using at most $\lceil \frac{w}{w-t} \rceil$ polynomial interpolations. Why is this problem impossible to solve if $w - t = 1$?

Answer: Denote $\mu = \lceil \frac{w}{w-t} \rceil$. Let $T_1, \dots, T_\mu$ be subsets of $\{1, \dots, w\}$ such that

$$|T_j| = w - t$$

for $1 \leq j \leq \mu$, and

$$\bigcup_{j=1}^{\mu} T_j = \{1, \dots, w\}.$$

This is not hard to do, for example by taking

$$T_1 = \{1, \dots, w - t\}, T_2 = \{w - t + 1, \dots, 2(w - t)\}, \dots.$$

For each T_j, interpolate a polynomial $f_j(x)$ using the t shares in the set

$$S_j = \{s_i : i \notin T_j\}.$$

If S_j contains no bad shares, then $f_j(x)$ is the correct polynomial and $w - 1$ shares will lie on this polynomial. On the other hand, if S_j contains the bad share, then there are at most $t - 1$ good shares and one bad share (i.e., a total of t shares) that lie on the polynomial $f_j(x)$ (recall that two different polynomials of degree $t-1$ have at most $t-1$ points in common). Provided that $w-1 > t$, we can distinguish between these two cases. That is, an interpolated polynomial $f_j(x)$ is the correct one if and only if exactly one share does not lie on $f_j(x)$. Therefore we can determine the defective share after at most μ polynomial interpolations.

If $w = t+1$, then any interpolated polynomial will pass through exactly $w-1$ of the w shares. Here $t = w-1$ and the two cases considered above are indistinguishable. Therefore there is no way to tell which interpolated polynomial is correct.

(c) Suppose that a (t, w)-*Shamir Threshold Scheme* has exactly τ defective shares, and suppose that $w \geq (\tau+1)t$. Describe how it is possible to determine which shares are defective using at most $\tau+1$ polynomial interpolations.

Answer: For $i \leftarrow 1$ to $\tau+1$, perform the following steps:

step (1) Interpolate shares in the set $S_i = \{(i-1)t+1, \dots, it\}$ to yield a polynomial $f_i(x)$.

step (2) Determine the set $B_i = \{j : f_i(x_j) \neq y_j\}$.

step (3) If $|B_i| \leq \tau$, then K is the constant term of f_i.

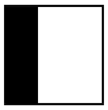
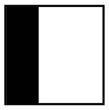
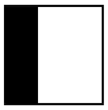
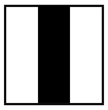
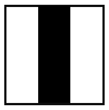
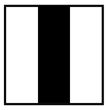
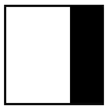
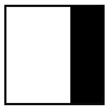
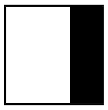
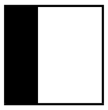
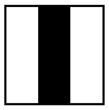
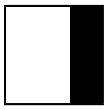
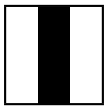
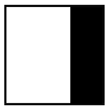
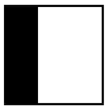
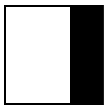
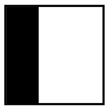
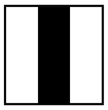
We now give a brief proof sketch that the algorithm described above will yield the correct value of the secret. Let $f(x)$ denote the polynomial used to generate the shares.

First, since $w \geq (\tau+1)t$, there exists at least one i such that S_i contains no bad shares. For any such i, we have that $f_i = f$, and at most τ of the w shares will not lie on f_i (i.e, $|B_i| \leq \tau$). In step (3), we will therefore obtain the correct value K as the secret.

On the other hand, suppose S_i contains at least one bad share. Note that S_i contains at most τ bad shares. Here, f_i agrees with at most $t-1$ good shares and at most τ bad shares, so $|B_i| \geq w - t - \tau + 1$. We claim that $w - t - \tau + 1 > \tau$. This inequality is equivalent to $w \geq t + 2\tau$, which is true because $w \geq (\tau+1)t$ and $t \geq 2$. This proves the claim. It then follows that step (3) will fail for this value of i.

11.10 Devise a $(2,3)$-visual threshold scheme with pixel expansion equal to 9. Each pixel in each of three shares is replaced by a 3×3 grid of subpixels. The gray level of a reconstructed black pixel should be equal to $2/3$ and the gray level of a reconstructed white pixel should be equal to $1/3$. Finally, no 3×3 grid of subpixels (from a single share) should give any information as to whether the reconstructed pixel will be white or black.

Answer: Here is the scheme:

pixel		y_1	y_2	y_3
□	$p = 1/3$			
	$p = 1/3$			
	$p = 1/3$			
■	$p = 1/3$			
	$p = 1/3$			
	$p = 1/3$			

It is straightforward to verify the following:

- Any given share has one of three possible forms, each with probability $1/3$, independent of whether the pixel is white or black
- the superposition of two white shares is $1/3$ black
- the superposition of two black shares is $2/3$ black.

Chapter 12

Key Agreement Schemes

12.1 Suppose that U and V take part in a session of the *Diffie-Hellman KAS* with $p = 27001$ and $\alpha = 101$. Suppose that U chooses $a_U = 21768$ and V chooses $a_V = 9898$. Show the computations performed by both U and V, and determine the key that they will compute.

Answer: U computes $b_U = \alpha^{a_U} = 7580$ and V computes $b_V = \alpha^{a_V} = 22181$. Then V computes $K = {b_U}^{a_V} = 10141$ and U computes ${b_V}^{a_U} = 10141$.

12.2 Consider the modification of the *STS KAS* that is presented as Protocol 12.8. In this modification of the protocol, the signatures omit the intended receiver. Show how this renders the protocol insecure, by describing an intruder-in-the-middle attack. Discuss the consequences of this attack, in terms of key authentication properties and how they are violated. (This attack is known as an ***unknown key-share attack***.)

Answer: : The attack is depicted below. In this attack, the intended peer of U is O. At the end of the session, U computes a key K that can be computed by V, but not by V. Since someone other than U's intended peer can compute K, the scheme does not provide implicit key authentication.

Remark: A similar attack can be carried out against V.

$$
\begin{array}{ccccc}
U & & O & & V \\
& \xrightarrow{\quad b_U, \mathbf{Cert}(U) \quad} & & \xrightarrow{\quad b_U, \mathbf{Cert}(U) \quad} & \\
& \xleftarrow{b_V, \mathbf{Cert}(O), sig_O(b_V \parallel b_U)} & & \xleftarrow{b_V, \mathbf{Cert}(V), sig_V(b_V \parallel b_U)} & \\
& \xrightarrow{\quad sig_U(b_U \parallel b_V) \quad} & & \xrightarrow{\quad sig_U(b_U \parallel b_V) \quad} &
\end{array}
$$

12.3 Discuss whether the property of perfect forward secrecy (which was defined in Section 11.1) is achieved in the *STS KAS*, assuming that the secret signing keys of one or more users are revealed.

Answer: The signing keys do not yield any information about the encrypted values that are transmitted during a session of the scheme. Therefore the *STS KAS* has perfect forward secrecy.

12.4 Suppose that U and V carry out the *MTI/A0 KAS* with $p = 30113$ and $\alpha =$

Protocol 12.8: MODIFIED STATION-TO-STATION KAS

The public domain parameters consist of a group $(G, \cdot)$ and an element $\alpha \in G$ having order n.

1. U chooses a random number a_U, $0 \leq a_U \leq n-1$. Then she computes

$$b_U = \alpha^{a_U}$$

and she sends **Cert**(U) and b_U to V.

2. V chooses a random number a_V, $0 \leq a_V \leq n-1$. Then he computes

$$\begin{aligned} b_V &= \alpha^{a_V} \\ K &= (b_U)^{a_V}, \quad \text{and} \\ y_V &= \mathbf{sig}_V(b_V \parallel b_U). \end{aligned}$$

Then V sends **Cert**(V), b_V and y_V to U.

3. U verifies y_V using $\mathbf{ver}_V$. If the signature y_V is not valid, then she "rejects" and quits. Otherwise, she "accepts," she computes

$$\begin{aligned} K &= (b_V)^{a_U}, \quad \text{and} \\ y_U &= \mathbf{sig}_U(b_U \parallel b_V), \end{aligned}$$

and she sends y_U to V.

4. V verifies y_U using $\mathbf{ver}_U$. If the signature y_U is not valid, then he "rejects"; otherwise, he "accepts."

52. Suppose that U has $a_U = 8642$ and he chooses $r_U = 28654$; and V has $a_V = 24673$ and she chooses $r_V = 12385$. Show the computations performed by both U and V, and determine the key that they will compute.

Answer: We have $b_U = 19392$ and $b_V = 7592$. U computes $s_U = 18143$ and sends it to V; V computes $s_V = 12$ and sends it to U. Now U computes $K = {s_V}^{a_U} {b_V}^{r_U} \bmod p = 7841$ and V computes $K = {s_U}^{a_V} {b_U}^{r_V} \bmod p = 7841$.

12.5 Discuss whether the property of perfect forward secrecy is achieved in *MTI/A0* for a session key that was established between U and V in the following two cases:

(a) one LL-key a_U is revealed.

Answer: Here we have perfect forward secrecy if the **Decision Diffie-Hellman** problem is infeasible in the group in question. An adversary can store the values s_U and s_V and they now know a_U. The session key is $K = (s_U)^{a_V}(s_V)^{a_U}$. The adversary can compute $(s_V)^{a_U}$, so computing K is equivalent to computing $(s_U)^{a_V} = \alpha^{r_U a_V}$. The adversary has $s_U = \alpha^{r_U}$ and $b_V = \alpha^{a_V}$, where a_V and r_U are not known. Computing any information about $\alpha^{r_U a_V}$ from this information cannot be done if the **DDH** problem is infeasible.

(b) both LL-keys a_U and a_V are revealed.

Answer: We do not have perfect forward secrecy in this situation, because a previous session key $K = (s_U)^{a_V}(s_V)^{a_U}$ can be computed by an adversary who has stored s_U and s_V.

12.6 If a passive adversary tries to compute the key K constructed by U and V by using the *MTI/A0 KAS*, then he is faced with an instance of what we might term the **MTI** problem, which we present as Problem 12.1.

Prove that any algorithm that can be used to solve the **MTI** problem can be used to solve the **Computational Diffie-Hellman** problem, and vice versa. (i.e., give Turing reductions between these two problems).

Answer: Suppose first that we have an oracle to solve **CDH**, say CDH-SOLVER. Given an instance of the **MTI** problem, $I = (p, \alpha, \beta, \gamma, \delta, \epsilon)$, compute $A = \text{CDHSOLVER}(p, \alpha, \beta, \gamma)$ and $B = \text{CDHSOLVER}(p, \alpha, \delta, \epsilon)$. The desired answer is $AB \bmod p$.

Conversely, suppose we have an oracle to solve **MTI**, say MTISOLVER. Let $I = (p, \alpha, \beta, \gamma)$ be an instance of **MTI**. To solve the instance I of **CDH**, compute $\text{MTISOLVER}(p, \alpha, \beta, \gamma, 1, 1)$.

Problem 12.1: MTI

Instance: $I = (p, \alpha, \beta, \gamma, \delta, \epsilon)$, where p is prime, $\alpha \in \mathbb{Z}_p{}^*$ is a primitive element, and $\beta, \gamma, \delta, \epsilon \in \mathbb{Z}_p{}^*$.
Question: Compute $\beta^{\log_\alpha \gamma} \delta^{\log_\alpha \epsilon} \bmod p$.

12.7 Analyze the deniability properties of *X3DH*. Specifically, show how an adversary with access to V's private keys can forge a transcript that appears to correspond to a session between U and V. Carefully describe how the associated transcript is created and how the associated key is computed.

Answer: A transcript would consist of the following information: $a_V, b_V, b_U, r_V, s_V, s_U$ and K. Note that b_U and b_V are fixed, since they are contained in the public certificates of U and V. The value a_V is V's private key. The values r_V and s_U are chosen randomly by the adversary and the adversary computes $s_V = \alpha^{r_V}$. Finally, the adversary computes

$$K = \textbf{KDF}({s_U}^{r_V} \parallel {s_U}^{a_V} \parallel {b_U}^{r_V}).$$

This is a "real key" that could have been computed in a session between U and V using the information in the transcript.

12.8 Show that *X3DH* provides perfect forward secrecy. That is, assume that an adversary records all the information transmitted in a particular session between U and V and later obtains the long-term private keys belonging to U and V. Despite learning all this information, the adversary should be unable to compute the session key for this specific session.

Answer: . The adversary obtains the following information: a_V, b_V, a_U, b_U, s_V and s_U. The session key is

$$\mathbf{KDF}(\alpha^{r_U r_V} \parallel \alpha^{r_U a_V} \parallel \alpha^{r_V a_U}).$$

To compute K, it is necessary to know the three values $\alpha^{r_U r_V}$, $\alpha^{r_U a_V}$ and $\alpha^{r_V a_U}$. The adversary can compute the second and third of these three values, but they cannot compute $\alpha^{r_U r_V}$ if the **Compuational Diffie-Hellman** problem is infeasible.

12.9 The purpose of this question is to perform the required computations in a session of the *Burmester-Desmedt Conference KAS*. Suppose we take $p = 128047$, $\alpha = 8$, and $n = 21341$. (It can be verified that p is prime and the order of α in $\mathbb{Z}_p{}^*$ is equal to n.) Suppose there are $m = 4$ participants, and they choose secret values $a_0 = 4499$, $a_1 = 9854$, $a_2 = 19887$, and $a_3 = 10002$.

(a) Compute the values b_0, b_1, b_2, and b_3.

Answer: $b_0 = 91104$, $b_1 = 62356$, $b_2 = 49681$ and $b_3 = 50726$.

(b) Compute the values X_0, X_1, X_2 and X_3.

Answer: $X_0 = 56204$, $X_1 = 67188$, $X_2 = 90645$ and $X_3 = 15115$.

(c) Show the computations performed by U_0, U_1, U_2, and U_3 to construct the conference key Z.

Answer: U_0 computes $Z = {b_3}^{4a_0} {X_0}^3 {X_1}^2 {X_2}^1 \bmod p$, U_1 computes $Z = {b_0}^{4a_1} {X_1}^3 {X_2}^2 {X_3}^1 \bmod p$, U_2 computes $Z = {b_1}^{4a_2} {X_2}^3 {X_3}^2 {X_0}^1 \bmod p$ and U_3 computes $Z = {b_2}^{4a_3} {X_3}^3 {X_0}^2 {X_1}^1 \bmod p$. The result is $Z = 12395$ in each case.

12.10 Show all the computations performed in a session of the *Steiner-Tsudik-Waidner Conference KAS* involving four participants. Use the same values of p, α, n, a_0, a_1, a_2, and a_3 as in the previous exercise.

Answer: First the lists $\mathcal{L}_0$, $\mathcal{L}_1$ and $\mathcal{L}_2$ are computed, in order, by U_0, U_1, U_2, resp.:

$$\begin{aligned}\mathcal{L}_0 &= (91104)\\ \mathcal{L}_1 &= (91104, 64376)\\ \mathcal{L}_2 &= (91104, 64376, 24433).\end{aligned}$$

Then U_3 computes the key $Z = 24433^{10002} \bmod p = 94723$ and the list

$$\mathcal{M}_3 = (50726, 34904, 120239).$$

Then U_2 computes the key $Z = 120239^{19887} \bmod p = 94723$ and the list

$$\mathcal{M}_2 = (73264, 63789).$$

Then U_1 computes the key $Z = 63789^{9854} \bmod p = 94723$ and the list

$$\mathcal{M}_1 = (115484).$$

Finally, U_0 computes the key $Z = 115484^{4499} \bmod p = 94723$.

Chapter 13

Miscellaneous Topics

13.1 In the *Cocks Identity-based Cryptosystem*, verify that

$$\left(\frac{1 + K_U^{priv}\,(t_1)^{-1}}{n}\right) = \pm 1.$$

Answer: Note: This question is incorrect: the Jacobi symbol could in fact turn out to be 0 (this also seems to be an error in the original paper describing this cryptosystem). The Jacobi symbol would equal 0 if

$$p \mid (1 + K_U^{priv}\,(t_1)^{-1}) \bmod n,$$

which is equivalent to

$$p \mid (t_1 + K_U^{priv}),$$

because $\gcd(t_1, n) = 1$.

For example, suppose we use the parameters from question 2. Here $n = 128099 \times 128047$. Denote $p = 128099$ and suppose we take $t_1 = p - K_U^{priv} = 31634$. It is easy to verify that $(\frac{t_1}{n}) = 1$ and $1 + K_U^{priv}/t_1 \mod n = 4396485779$, which is divisible by p.

Of course, finding t_1 such that

$$\left(\frac{1 + K_U^{priv}\,(t_1)^{-1}}{n}\right) = 0$$

is equivalent to factoring n, so it is very unlikely to occur in practice if n is large enough that factoring it is infeasible.

13.2 Suppose the *Cocks Identity-based Cryptosystem* is implemented with master public key $n = 16402692653$, and suppose that a user U has public key $K_U^{pub} = 9305496225$.

(a) Let $t_1 = 3975333024$ and $t_2 = 4892498575$. Verify that $(\frac{t_1}{n}) = (\frac{t_2}{n}) = -1$.

Answer: This is just an evaluation of two Jacobi symbols.

(b) Encrypt the plaintext $x = -1$ using the "random" values t_1 and t_2, obtaining the ciphertext (y_1, y_2).

Answer: We obtain the values $y_1 = 2650909904$ and $y_2 = 4071118542$.

(c) Given that $K_U^{priv} = 96465$, verify that the decryption of (y_1, y_2) is equal to x.

Answer: We have $(K_U^{priv})^2 \equiv K_U^{pub} \pmod{n}$, so $s = y_1$. Then we compute x to be the Jacobi symbol $(\frac{y_1+2K_U^{priv}}{n}) = -1$.

13.3 Suppose you are given an instance of the **BDH** problem, specifically, consisting of the following:

- additive groups $(G_1, +)$ and $(G_2, +)$ of prime order q and a multiplicative group $(G_3, \cdot)$ of order q,
- a pairing $e_q : G_1 \times G_2 \to G_3$,
- an element $P \in G_1$,
- an element $Q \in G_2$ having order q, and
- elements $aQ, bQ \in G_2$ for some $a, b \in {\mathbb{Z}_q}^*$.

Show that, if you can solve the **CDH** problem in G_3, then you can solve the given instance of the **BDH** problem.

Answer: Compute $\alpha = e_q(P, Q)$, $\beta = e_q(P, aQ)$ and $\gamma = e_q(P, bQ)$. Then solve the **CDH** in G_3, given the instance α, β, γ, obtaining δ. This value δ is the desired solution to the given instance of the **BDH** problem.

13.4 The purpose of this question is to perform some computations using the *Paillier Cryptosystem*. Suppose $p = 1041817$ and $q = 716809$.
Note: the computations do not work correctly because the given value of p is not prime. We instead show the computations when $p = 1041823$ (which is prime).

(a) Suppose $x_1 = 726095811532$, $r_1 = 270134931749$, $x_2 = 450864083576$, and $r_2 = 378141346340$. Compute $y_1 = e_K(x_1, r_1)$ and $y_2 = e_K(x_2, r_2)$.

Answer: We have
$$n = 746788102807$$
and
$$\phi(n) = 746786344176.$$
Then we compute
$$y_1 = 442697833349572462807160$$
and
$$y_2 = 247027496947548022966686.$$

(b) Let $y_3 = y_1 y_2 \bmod n^2$. Compute $x_3 = d_K(y_3)$ using the decryption algorithm for the *Paillier Cryptosystem*.

Answer: We have
$$y_3 = 155666267426885069548049$$

and

$$x_3 = 430171792301.$$

(c) Verify that $x_3 \equiv x_1 + x_2 \pmod{n}$. We have

$$x_1 + x_2 - x_3 = n \equiv 0 \pmod{n}.$$

13.5 Suppose that $n = pq$, where p and q are the values from the previous exercise. Determine if 22980544317200183678448 is an nth residue modulo n^2.

Answer: Note: As in the previous question, since p is not prime, the question must be restated.

We show how to determine if $y = 384680746356111062443683$ is an nth residue modulo n^2, where use use the modified value of n from the previous question, i.e., $n = 746788102807$ and $\phi(n) = 746786344176$. Here, we can compute

$$y^{\phi(n)} \bmod n^2 = 1.$$

Thus $d_K(y) = 0$ and y is an nth residue modulo n^2.

13.6 Prove the "if" part of Theorem 13.5; i.e., that an (ℓ, n, q) code $\mathcal{C}$ is a 2-IPP code if $A(\mathcal{C})$ is simultaneously a PHF$(\ell; n, q, 3)$ and an SHF$(\ell; n, q, \{2, 2\})$.

Answer: Suppose that

$$\mathbf{f} = (f_1, \dots, f_n) \in \mathbf{desc}_2\mathcal{C})$$

and

$$\bigcap_{\mathcal{C}_0 \in \mathbf{susp}_2(\mathbf{f})} \mathcal{C}_0 = \emptyset.$$

Suppose first that there are two disjoint pairs of codewords $\{\mathbf{c}_1, \mathbf{c}_2\}$ and $\{\mathbf{c}_3, \mathbf{c}_4\}$ in the set $\mathbf{susp}_2(\mathbf{f})$, (where "disjoint" means that $\{\mathbf{c}_1, \mathbf{c}_2\} \cap \{\mathbf{c}_3, \mathbf{c}_4\} = \emptyset$). Consider the corresponding rows $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4$ in the array $A = A(\mathcal{C})$. For every column j of A, we have that

$$f_j \in \{A(\mathbf{c}_1, j), A(\mathbf{c}_2, j)\} \cap \{A(\mathbf{c}_1, j), A(\mathbf{c}_2, j)\}.$$

Hence, it follows that there is no column j such that

$$\{A(\mathbf{c}_1, j), A(\mathbf{c}_2, j)\} \cap \{A(\mathbf{c}_3, j), A(\mathbf{c}_4, j)\} = \emptyset.$$

Therefore A is not an SHF$(\ell; n, q, \{2, 2\})$.

Now if there are not two disjoint pairs of codewords in the set $\mathbf{susp}_2(\mathbf{f})$, it follows that any two pairs of codewords in this set must intersect in one codeword. From this, it is easy to see that $\mathbf{susp}_2(\mathbf{f})$ consists of three pairs of codewords that form a "triangle", i.e.,

$$\mathbf{susp}_2(\mathbf{f}) = \{\{\mathbf{c}_1, \mathbf{c}_2\}, \{\mathbf{c}_1, \mathbf{c}_3\}, \{\mathbf{c}_2, \mathbf{c}_3\}\}$$

for three codewords $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3$. Now consider the three rows $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3$ in the array $A = A(\mathcal{C})$. For every column j, we have that

$$f_j \in \{A(\mathbf{c}_1, j), A(\mathbf{c}_2, j)\} \cap \{A(\mathbf{c}_1, j), A(\mathbf{c}_3, j)\} \cap \{A(\mathbf{c}_2, j), A(\mathbf{c}_3, j)\}.$$

From this, we that the three values $A(\mathbf{c}_1, j), A(\mathbf{c}_2, j), A(\mathbf{c}_3, j)$ cannot all be different. Hence, A is not a PHF$(\ell; n, q, 3)$.

13.7 (a) Consider the $(3, 3r^2, r^2 + 2r)$ 2-IPP code $\mathcal{C}$ that was described in Section 13.3.3. Give a complete description of a $O(1)$ time algorithm TRACE, which takes as input a triple $\mathbf{f} = (f_1, f_2, f_3)$ and attempts to determine an identifiable parent of $\mathbf{f}$. If $\mathbf{f} \in \mathcal{C}$, then TRACE$(\mathbf{f}) = \mathbf{f}$; if $\mathbf{f} \in \mathbf{desc}_2(\mathcal{C}) \backslash \mathcal{C}$, then TRACE$(\mathbf{f})$ should find an identifiable parent of $\mathbf{f}$; and TRACE$(\mathbf{f})$ should return the output "fail," if $\mathbf{f} \notin \mathbf{desc}_2(\mathcal{C})$.

Answer: Let (x_1, x_2, x_3) be the input to the tracing algorithm. If $(x_1, x_2, x_3) \in \mathsf{desc}_2(\mathcal{C})$, then there are six cases that can occur:

(1) If $x_1 \in L$, compute $m_2 \in M$ such that $m_2 \equiv x_1 \pmod{n}$. Then compute $m_1 = (x_1 + r^2 - m_2)/r$. The codeword (x_1, m_1, m_2) is a parent of (x_1, x_2, x_3).

(2) If $x_2 \in L$, compute $m \in M$ such that $m \equiv x_2 \pmod{n}$. Then compute $s = (x_2 - m)/r$. The codeword (m, x_2, s) is a parent of (x_1, x_2, x_3).

(3) If $x_3 \in L$, compute $s_2 \in S$ such that $s_2 \equiv x_3 \pmod{n}$. Then compute $s_1 = (x_3 - s_2 - r)/r$. The codeword (s_1, s_2, x_3) is a parent of (x_1, x_2, x_3).

(4) If $x_1, x_2 \in S$, let $s_1 = x_1$ and $s_2 = x_2$. The codeword $(s_1, s_2, rs_1 + s_2 + r) = (x_1, x_2, rx_1 + x_2 + r)$ is a parent of (x_1, x_2, x_3).

(5) If $x_2, x_3 \in M$, let $m_1 = x_2$ and $m_2 = x_3$. The codeword $(rm_1 + m_2 - r^2, m_1, m_2) = (rx_2 + x_3 - r^2, x_2, x_3)$ is a parent of (x_1, x_2, x_3).

(6) If $x_1 \in M$ and $x_3 \in S$, let $m = x_1$ and $s = x_3$. The codeword $(m, sr + m, s) = (x_1, rx_3 + x_1, x_3)$ is a parent of (x_1, x_2, x_3).

If none of these six cases arise, then $(x_1, x_2, x_3) \notin \mathsf{desc}_2(\mathcal{C})$ and the tracing algorithm fails.

(b) Illustrate the execution of your algorithm in the case $r = 10$ ($q = 120$) for the following triples: $(13, 11, 17)$; $(44, 9, 14)$; $(18, 108, 9)$.

Answer: When $(x_1, x_2, x_3) = (13, 11, 17)$, we have $x_2, x_3 \in M$, so we are in case (5). $10 \times 11 + 17 - 10^2 = 27$, and hence $(27, 11, 17)$ is a parent of $(13, 11, 17)$.

When $(x_1, x_2, x_3) = (44, 9, 14)$, we have $x_1 \in L$, so we are in case (1). We have $m_2 = 14$ and $m_1 = (44 + 100 - 14)/10 = 13$, and hence $(44, 13, 14)$ is a parent of $(44, 9, 14)$.

When $(x_1, x_2, x_3) = (18, 108, 9)$, we have $x_2 \in L$, so we are in case (2). We have $m = 18$ and $s = (108 - 18)/10 = 9$, and hence $(18, 108, 9)$ is a parent of $(18, 108, 9)$. (In this example, (x_1, x_2, x_3) is already a codeword.)

13.8 We describe a $(4, r^3, r^2)$ 2-IPP code due to Hollman, van Lint, Linnartz, and Tolhuizen. The alphabet is $Q = \mathbb{Z}_r \times \mathbb{Z}_r$. The code $\mathcal{C} \subseteq Q^4$ consists of the following set of r^3 4-tuples:

$$\{((a,b),(a,c),(b,c),(a+b \bmod r, c)) : a,b,c \in \mathbb{Z}_r\}.$$

(a) Give a complete description of a $O(1)$ time algorithm TRACE, which takes as input a 4-tuple $\mathbf{f} = ((\alpha_1,\alpha_2),(\beta_1,\beta_2),(\gamma_1,\gamma_2),(\delta_1,\delta_2))$ and attempts to determine an identifiable parent of $\mathbf{f}$. The output of TRACE should be as follows:

- if $\mathbf{f} \in \mathcal{C}$, then TRACE$(\mathbf{f}) = \mathbf{f}$;
- if $\mathbf{f} \in \mathbf{desc}_2(\mathcal{C}) \backslash \mathcal{C}$, then TRACE$(\mathbf{f})$ should find one identifiable parent of $\mathbf{f}$; and
- TRACE$(\mathbf{f})$ should return the output "fail," if $\mathbf{f} \notin \mathbf{desc}_2(\mathcal{C})$.

In order for the algorithm to be an $O(1)$ time algorithm, there should be no linear searches, for example. You can assume that an arithmetic operation can be done in $O(1)$ time, however.

HINT In designing the algorithm, you will need to consider several cases. Many of the cases (and resulting subcases) are quite similar, however. You could initially divide the problem into the following four cases:

- $\alpha_1 \neq \beta_1$
- $\alpha_2 \neq \gamma_1$
- $\beta_2 \neq \gamma_2$
- $\alpha_1 = \beta_1$, $\alpha_2 = \gamma_1$, and $\beta_2 = \gamma_2$.

Answer: The algorithm is as follows:

```
Algorithm: TRACE (f)

if $\alpha_1 \neq \beta_1$   (case 1)
  then
    if $\gamma_1 \neq \alpha_2$
      then
        if $\beta_2 \neq \gamma_2$
          then return ("fail")
          else
            if $(\delta_1 = \alpha_1 + \alpha_2)$ or $((\delta_1, \delta_2) = (\beta_1 + \gamma_1, \gamma_2))$
              then
            return $(((\beta_1, \gamma_1), (\beta_1, \beta_2), (\gamma_1, \beta_2), (\beta_1 + \gamma_1, \beta_2)))$
              else
            return ("fail")
      else
        if $\beta_2 \neq \gamma_2$
          then
            if $(\delta_2 = \beta_2)$ or $((\delta_1, \delta_2) = (\alpha_1 + \alpha_2, \gamma_2))$
              then
            return $(((\alpha_1, \alpha_2), (\alpha_1, \gamma_2), (\alpha_2, \gamma_2), (\alpha_1 + \alpha_2, \gamma_2)))$
              else
            return ("fail")
          else
            if $(\delta_1, \delta_2) = (\alpha_1 + \alpha_2, \gamma_2)$
              then
            return $(((\alpha_1, \alpha_2), (\alpha_1, \gamma_2), (\alpha_2, \gamma_2), (\alpha_1 + \alpha_2, \gamma_2)))$
              else if $(\delta_1, \delta_2) = (\beta_1 + \gamma_1, \beta_2)$
              then
            return $(((\beta_1, \gamma_1), (\beta_1, \beta_2), (\gamma_1, \beta_2), (\beta_1 + \gamma_1, \beta_2)))$
              else
            return ("fail")
  else if $\alpha_2 \neq \gamma_1$   (case 2)
  then
    if $\beta_2 \neq \gamma_2$
      then
        if $(\delta_2 = \gamma_2)$ or $((\delta_1, \delta_2) = (\alpha_1 + \alpha_2, \beta_2))$
          then
        return $(((\alpha_1, \alpha_2), (\alpha_1, \beta_2), (\alpha_2, \beta_2), (\alpha_1 + \alpha_2, \beta_2)))$
          else
        return ("fail")
      else
        if $(\delta_1, \delta_2) = (\alpha_1 + \alpha_2, \beta_2)$
          then
        return $(((\alpha_1, \alpha_2), (\alpha_1, \beta_2), (\alpha_2, \beta_2), (\alpha_1 + \alpha_2, \beta_2)))$
          else if $(\delta_1, \delta_2) = (\alpha_1 + \gamma_1, \beta_2)$
          then
        return $(((\alpha_1, \gamma_1), (\alpha_1, \beta_2), (\gamma_1, \beta_2), (\alpha_1 + \gamma_1, \beta_2)))$
          else
        return ("fail")
  else if $\beta_2 \neq \gamma_2$   (case 3)
  then
    if $(\delta_1, \delta_2) = (\alpha_1 + \alpha_2, \beta_2)$
      then
    return $(((\alpha_1, \alpha_2), (\alpha_1, \beta_2), (\alpha_2, \beta_2), (\alpha_1 + \alpha_2, \beta_2)))$
      else if $(\delta_1, \delta_2) = (\alpha_1 + \alpha_2, \gamma_2)$
      then
    return $(((\alpha_1, \alpha_2), (\alpha_1, \gamma_2), (\alpha_2, \gamma_2), (\alpha_1 + \alpha_2, \gamma_2)))$
      else
    return ("fail")
  else   (case 4)   return $(((\alpha_1, \alpha_2), (\alpha_1, \beta_2), (\alpha_2, \beta_2), (\alpha_1 + \alpha_2, \beta_2)))$
```

Here is a partial explanation of the algorithm. Denote

$$\mathbf{f} = (\alpha, \beta, \gamma, \delta) = ((\alpha_1, \alpha_2), (\beta_1, \beta_2), (\gamma_1, \gamma_2), (\delta_1, \delta_2)),$$

and let $\mathbf{c}_1$ and $\mathbf{c}_2$ denote two (hypothetical) parents of $\mathbf{f}$.

Suppose we are in case 1: $\alpha_1 \neq \beta_1$. Then α comes from $\mathbf{c}_1$, say, and β comes from $\mathbf{c}_2$.

Subcase 1(a): $\gamma_1 \neq \alpha_2$. Then γ comes from $\mathbf{c}_2$, so we have:

$$\begin{aligned} \mathbf{c}_1 &= (\alpha, -, -, -) \\ \mathbf{c}_2 &= (-, \beta, \gamma, -) \end{aligned}$$

Subcase 1(a)-1: $\beta_2 \neq \gamma_2$. Then β and γ cannot come from the same codeword. This contradiction shows that $\mathbf{f}$ is not in the 2-descendant code.

Subcase 1(a)-2: $\beta_2 = \gamma_2$. Then we have two (consistent) coordinates of $\mathbf{c}_2$. This determines all four coordinates of $\mathbf{c}_2$,

$$\mathbf{c}_2 = ((\beta_1, \gamma_1), (\beta_1, \beta_2), (\gamma_1, \beta_2), (\beta_1 + \gamma_1, \beta_2)).$$

If $\mathbf{f}$ is in the 2-descendant code, then we know at this point that $\mathbf{c}_2$ is a parent of $\mathbf{f}$. However, we have to examine the fourth coordinate of $\mathbf{f}$, namely δ, before making this determination. Namely, we need to check that either $\delta = (\beta_1 + \gamma_1, \beta_2)$ (in which case δ comes from $\mathbf{c}_2$) or $\delta_1 = \alpha_1 + \alpha_2$ (in which case δ is consisent with α, i.e., there exists a codeword of the form $(\alpha, -, -, \delta)$. If neither of these two conditions holds, then $\mathbf{f}$ is not in the 2-descendant code.

Subcase 1(b): $\gamma_1 = \alpha_2$

Subcase 1(b)-1: $\beta_2 \neq \gamma_2$. Then γ comes from $\mathbf{c}_1$, so we have:

$$\begin{aligned} \mathbf{c}_1 &= (\alpha, -, \gamma, -) \\ \mathbf{c}_2 &= (-, \beta, -, -) \end{aligned}$$

We have two (consistent) coordinates of $\mathbf{c}_1$. This determines all four coordinates of $\mathbf{c}_1$,

$$\mathbf{c}_1 = ((\alpha_1, \alpha_2), (\alpha_1, \gamma_2), (\alpha_2, \gamma_2), (\alpha_1 + \alpha_2, \gamma_2)).$$

If $\mathbf{f}$ is in the 2-descendant code, then we know at this point that $\mathbf{c}_1$ is a parent of $\mathbf{f}$. However, we have to examine the fourth coordinate of $\mathbf{f}$, namely δ, before making this determination. Namely, we need to check that either $\delta = (\alpha_1 + \alpha_2, \gamma_2)$ (in which case δ comes from $\mathbf{c}_1$) or $\delta_2 = \beta_2$ (in which case δ is consisent with β, i.e., there exists a codeword of the form $(-, \beta, -, \delta)$. If neither of these two conditions holds, then $\mathbf{f}$ is not in the 2-descendant code.

Subcase 1(b)-2: $\beta_2 = \gamma_2$. Then γ is consistent with both α and β, so we have:

$$\begin{aligned} \mathbf{c}_1 &= (\alpha, -, \gamma, -) \\ \mathbf{c}_2 &= (-, \beta, \gamma, -) \end{aligned}$$

At this point, we have two consistent coordinates in each of two possible codewords. We can compute the fourth coordinate in each of $\mathbf{c}_1$ and $\mathbf{c}_2$, and check if we obtain δ in one of these two cases. If so, then the relevant codeword is a parent of $\mathbf{f}$.

There remain three more cases to consider, which can be analyzed in a similar fashion.

(b) Illustrate the execution of your algorithm in detail in the case $r = 100$ for each of the following 4-tuples $\mathbf{f}$:

$$\begin{gathered}((37,71),(37,96),(71,96),(12,96))\\((25,16),(83,54),(16,54),(41,54))\\((19,11),(19,12),(11,15),(30,12))\\((32,40),(32,50),(50,40),(82,30))\end{gathered}$$

Answer: If $\mathbf{f} = ((37,71),(37,96),(71,96),(12,96))$, then we have $\alpha_1 = \beta_1$, $\alpha_2 = \gamma_1$ and $\beta_2 = \gamma_2$. Then $37 + 71 = 8$ and we return

$$((37,71),(37,96),(71,96),(8,96)).$$

If $\mathbf{f} = ((25,16),(83,54),(16,54),(41,54))$, then we have $\alpha_1 \neq \beta_1$, $\alpha_2 = \gamma_1$ and $\beta_2 = \gamma_2$. Then we verify that $(\delta_1,\delta_2) = (41,54) = (\alpha_1+\alpha_2,\gamma_2)$. Then we return

$$((25,16),(25,54),(16,54),(41,54)).$$

If $\mathbf{f} = ((19,11),(19,12),(11,15),(30,12))$, then we have $\alpha_1 = \beta_1$, $\alpha_2 = \gamma_1$ and $\beta_2 \neq \gamma_2$. Then we verify that $(\delta_1,\delta_2) = (30,12) = (\alpha_1+\alpha_2,\beta_2)$. Then we return

$$((19,11),(19,12),(11,12),(30,12)).$$

If $\mathbf{f} = ((32,40),(32,50),(50,40),(82,30))$, then we have $\alpha_1 = \beta_1$, $\alpha_2 \neq \gamma_1$ and $\beta_2 \neq \gamma_2$. Then we find that $\delta_2 \neq \gamma_2$ and $\delta_1 \neq \alpha_1 + \alpha_2$. Hence, we return "fail".

13.9 Consider the 3-TA code constructed by applying Theorem 13.8 with $\ell = 19$ and $q = 101$. This code is a $(19, 101^3, 101)$-code.

(a) Write a computer program to construct the 101^3 codewords in this code.

(b) Given the vector

$$\mathbf{f} = (14,66,46,56,13,31,50,30,77,32,0,93,48,37,16,66,24,42,9)$$

in the 3-descendant code, compute a parent of $\mathbf{f}$ using nearest neighbor decoding.

Answer: The nearest neighbor to $\mathbf{f}$ is the following codeword obtained from the polynomial $14 + 20x + 99x^2$:

$$(14,32,46,56,62,64,62,56,46,32,14,93,67,37,3,66,24,79,29).$$

This codeword has distance 11 from $\mathbf{f}$.

It can be shown that $\mathbf{f}$ is a descendant of the following three codewords:

$$(14, 32, 46, 56, 62, 64, 62, 56, 46, 32, 14, 93, 67, 37, 3, 66, 24, 79, 29)$$
$$(96, 52, 74, 61, 13, 31, 14, 63, 77, 56, 0, 10, 86, 26, 32, 3, 40, 42, 9)$$
$$(12, 66, 74, 36, 53, 24, 50, 30, 65, 54, 98, 96, 48, 55, 16, 32, 2, 27, 6)$$

13.10 There are many online programs to compute *SHA-1* message digests. Typically, the input will be given in ascii form and the output will be a sequence of 40 hexadecimal characters $(0, \ldots, 9, A, B, C, D, F)$. By computing the *SHA-1* message digests of the strings $0, 1, 2, 3 \ldots,$ determine the smallest positive integer x whose *SHA-1* message digest starts with the hexadecimal digit 0. Then determine the smallest positive integer x whose corresponding *SHA-1* message digest starts with hexadecimal digits 00.

Answer: *SHA-1*$(127) = 008451A05E1E7AA32C75119DF950D405265E0904$.