Exercises for
COMP 70009 Cryptography Engineering

Exercises 54 and 55 are questions from past Exams of this Course.

Exercise 1 *Let $x \oplus y$ denote the exclusive or defined over bit-vectors of length $n > 0$. Show that $\oplus$ is*

1. *commutative: for all x and y we have $x \oplus y$ equals $y \oplus x$*
2. *associative: for all x, y, z we have that $(x \oplus y) \oplus z$ equals $x \oplus (y \oplus z)$*
3. *idempotent: for all x we have that $x \oplus x = 0$ where 0 is the bit-vector of n zeros*
4. *0 is a unit: for all x we have that $0 \oplus x$ equals $x \oplus 0$ which equals x*
5. *for all x and y we have that $(x \oplus y) \oplus y = x$*
6. *all equations $a \oplus x = b$ have a unique solution x*

In particular, $\oplus$ makes bit-vectors of length n into a commutative group with identity element 0 and where each element is its own inverse.

Solution 1 *First, we note that it suffices to show these relationships for bit-vectors of length 1 as all operations are applied bit-wise and independently. That is to say, we really just have to show this for single bits.*

1. *This follows from the definition of $\oplus$ since $x = y$ is a symmetric relation.*
2. *This boils down to a case analysis. For bits there are eight cases. For example, one case is $(1 \oplus 0) \oplus 1 = 1 \oplus 1 = 0 = 1 \oplus 1 = 1 \oplus (0 \oplus 1)$. The other cases can be shown in the same vain.*
3. *This follows directly from the definition of $\oplus$.*

4. *If $x = 0$, then $0 \oplus x = 0$, which equals x; if $x = 1$, then $0 \oplus x = 1$, which equals x as well.*

5. *We have $(x \oplus y) \oplus y = x \oplus (y \oplus y) = x \oplus 0 = x$ by previous items.*

6. *First of all $x = a \oplus b$ is a solution since $a \oplus (a \oplus b)$ equals b be the previous item, noting that $\oplus$ is commutative as well. Second, let x' be another solution. From $a \oplus x' = b$ we get $a \oplus (a \oplus x') = a \oplus b$ and so $x' = a \oplus b$ follows in a similar vein. Thus the solution is unique.*

Exercise 2 *Let $\mathbb{P} = \{a, b, c, d\}$, $\mathbb{C} = \{1, 2, 3, 4\}$, and $\mathbb{K} = \{k_1, k_2, k_3\}$. Let $p(P = a) = 0.2$, $p(P = b) = 0.3$, $p(P = c) = 0.1$, and $p(P = d) = 0.4$. Let $p(K = k_1) = 0.3 = p(K = k_2)$ and $p(K = k_3) = 0.4$. Finally, the encryption $e_k(c)$ for this is given in the table*

	a	b	c	d
k_1	4	2	3	1
k_2	3	2	4	1
k_3	1	2	4	3

1. *Compute $p(C = c)$ for each c in $\mathbb{C}$.*

2. *Compute $p(C = c \mid P = m)$ for each pair (c, m) in $\mathbb{C} \times \mathbb{P}$.*

3. *Compute $p(P = a \mid C = 2)$ and $p(P = c \mid C = 3)$.*

4. *Explain why the above cryptosystem is not perfectly secure.*

Solution 2 1. *We show this in detail for $C = 1$. The reasoning for other choices of C follows the principle. According to equation (7) in the textbook, we have that $p(C = 1)$ equals*

$$p(K = k_1) \cdot p(P = d) + p(K = k_2) \cdot p(P = d) + p(K = k_3) \cdot p(P = a) \quad (1)$$

Note that each plaintext represents $d_k(c)$. For example, for k_1 we have $P = d$ since $d_{k_1}(1) = a$. We can now replace each formal expression in (1) with their probability as so this computes to $0.3 \cdot 0.4 + 0.3 \cdot 0.4 + 0.4 \cdot 0.2 = 0.32$.

2. *Again, we only show this for one such pair as the calculations for other pairs are similar and follow the same recipe. For* $p(C = 1 \mid m = a)$, *this equals* $\sum_{k \mid a = d_k(1)} p(K = k) = p(K = k_3)$ *since only* k_3 *decrypts* 1 *to* a. *Therefore,* $p(C = 1 \mid m = a)$ *equals* 0.4.

3. *We only compute* $p(P = a \mid C = 2)$, *the computations for the other case are very similar. By definition,*

$$p(P = a \mid C = 2) = \frac{p(P = a) \cdot p(C = 2 \mid P = a)}{p(C = 2)} \tag{2}$$

   Now $p(C = 2 \mid P = a)$ *is defined as* $\sum_{k \mid c \in \mathbb{C}(k)} p(K = k) \cdot p(P = d_k(c))$. *But this sum is empty here, and so equals* 0, *since there is no key in* $\mathbb{K}$ *that decrypts* 2 *into* a. *Therefore, we conclude that the expression in (2) computes to* 0 *as well.*

4. *For this crypto system to have perfect secrecy, we would need that* $P(P = m \mid C = c)$ *equals* $p(P = m)$ *for all plaintexts* m *and all ciphertexts* c. *Above, we computed* $p(P = a \mid C = 2)$ *to be* 0. *But this does not equal* $P(p = a)$, *which is* 0.2. *Therefore, the crypto system is not perfectly secure.*

Exercise 3 *Recall the definition of perfect secrecy from our textbook: "for all plaintexts* m *and all ciphertexts* c, *we have that* $P(P = m \mid C = c)$ *equals* $p(P = m)$."

Prove that this definition is equivalent to the one based on "for all plaintexts m *and all ciphertexts* c, *we have that* $P(C = c \mid P = m)$ *equals* $p(C = c)$."

Solution 3 *We show that we may assume, without loss of generality, that for all ciphertexts* c *we have* $p(C = c) > 0$. *To see this, we first note that we may assume that* $p(K = k)$ *and* $p(P = m)$ *are positive for all keys* k *and plaintexts* m. *Otherwise, we would be saying that a particular key or plaintext would never be used, which begs the question of why this would be in the set* $\mathbb{K}$, *respectively,* $\mathbb{P}$ *in the first place. Given that, suppose that there is some* c *with* $p(C = c) = 0$. *This then means by equation (7) of the textbook* *that $\sum_{k|c\in\mathbb{C}(k)} p(K=k)\cdot p(P=d_k(c)) =$. Since the probabilities for keys and plaintexts are all positive, the sum can only be equal to 0 if it is an empty sum. But that would mean that there is no key k with c in $\mathbb{C}(k)$. And this would mean that c cannot be the ciphertext of any plain text. Therefore, we could remove c from the set $\mathbb{C}$ without impacting the crypto system in any way.*

Since we have established now that $p(C=c)$ is positive for all c in $\mathbb{C}$, we use this to prove the statement:

1. *First, assume perfect secrecy. Let m be a plaintext and c a ciphertext. We need to show that $p(C=c \mid P=m) = p(C=c)$. By perfect secrecy, we have $p(P=m \mid C=c) = p(P=m)$. We use this and the fact that $p(P=m) > 0$ to compute*

$$\begin{aligned} p(C=c \mid P=m) &= \frac{p(C=c)\cdot p(P=m \mid C=c)}{p(P=m)} && \textit{(Bayes' Theorem)} \\ &= \frac{p(C=c)\cdot p(P=m)}{p(P=m)} && \textit{(Perfect Secrecy)} \\ &= p(C=c) && \textit{(Basic algebra)} \end{aligned}$$

2. *Conversely, assume that $P(C=c \mid P=m) = p(C=c)$ for all ciphertexts c and plaintexts m. We need to show that $p(P=m \mid C=c)$ equals $p(P=m)$. We now use that $p(C=c)$ is positive and compute*

$$\begin{aligned} p(P=m \mid C=c) &= \frac{p(P=m)\cdot p(C=c \mid P=m)}{p(C=c)} && \textit{(Bayes' Theorem)} \\ &= \frac{p(P=m)\cdot p(C=c)}{p(C=c)} && \textit{(Perfect Secrecy)} \\ &= p(P=m) && \textit{(Basic algebra)} \end{aligned}$$

Exercise 4 *For a perfectly secure cryptosystem, prove that we always have $\#\mathbb{K} \geq \#\mathbb{C} \geq \#\mathbb{P}$, where $\#A$ denotes the number of elements in a finite set A.*

Solution 4 *See the proof of Lemma 9.3 in the textbook, on page 166. Note that this uses that $p(C=c)$ is positive for all c, and we already explained why we can make that assumption.*

Exercise 5 *Prove the second part of Theorem 9.4 in our textbook, Shannon's Theorem: assume that*

1. *every key is used with equal probability $1/\#\mathbb{K}$, and*
2. *for each m in $\mathbb{P}$ and c in $\mathbb{C}$, there is a unique k in $\mathbb{K}$ such that $e_k(m) = c$.*

Use these assumptions to prove that the cryptosystem has perfect secrecy.

Solution 5 *Please see the second part of the proof of Theorem 9.4 in the textbook, on pages 167-168, for details on this.*

Exercise 6 *Let us reconsider the one-time pad in the setting where Alice encrypts binary strings of length $n > 1$ to be sent to Bob. Assume that Eve can intercept such encrypted messages but that Eve cannot control which messages Alice will encrypt. However, you may assume that Alice will encrypt more than one message with the same key k.*

Explain what Eve can learn about two ciphertexts encrypted with the same key.

Solution 6 *Suppose that Eve intercepts two ciphertexts c_1 and c_2 that Alice encrypted with the same key k, using the one-time pad:*

$$c_1 = m_1 \oplus k \qquad\qquad c_2 = m_2 \oplus k$$

Then Eve knows c_1 and c_2 and can therefore compute $c_1 \oplus c_2$ and so knows that value of that bit string as well. We claim that this bit string equals $m_1 \oplus m_2$ since

$$c_1 \oplus c_2 = (m_1 \oplus k) \oplus (m_2 \oplus k) = m_1 \oplus k \oplus m_2 \oplus k = (m_1 \oplus m_2) \oplus (k \oplus k) = m_1 \oplus m_2$$

So Eve now knows for which bits the two plaintexts m_1 and m_2 coincide: they are the positions at which $c_1 \oplus c_2$ equals 0. This discovers a relationship between two plain-texts.

Even worse, suppose that Eve would know some portion of m_1, for example the values of the first 32 bits. Then the above equation means that Eve can compute the first 32 bit values of m_2 as well.

Background information: *A crypto system is called non-malleable if, given a ciphertext c corresponding to an unknown plaintext m, an attacker cannot find a ciphertext c' for a plaintext m' where m' is "related" to m. This definition is informal, as we don't say what "related" means here. But it is clear from the solution above, that the one-time pad is malleable: Eve can discover a relationship between two plain-texts if Alice reuses a key.*

Exercise 7 *Show equation (10) in our textbook on page 172:*

$$(8) \qquad H(K \mid C) = H(K) + H(P) - H(C)$$

and discuss the significance of this equation in relation to cryptanalysis.

Solution 7 *We will not cover entropies in this module. But it you want to post a solution for this exercise on Piazza, we will be happy to discuss that there.*

Exercise 8 *Recall the design of a symmetric key encryption schemes that relied on the use of a Feistel function for round-based encryption, with DES being a good example thereof.*

Assume that encryption and decryption operate in three rounds, and show the correctness of this scheme: the decryption of ciphertext recovers correctly the plaintext.

Solution 8 *We refer to Figure 13.4 of the textbook on page 247, where we now assume that there are only* 3 *and not* 16 *rounds and so r in the figure equals* 3.

Let m be the single data block that is encrypted with key K and round keys K_1, K_2, and K_3. By definition of this device, we have

$$L_0R_0 = IP(m) \tag{3}$$

The ciphertext is then

$$c = IP^{-1}(R_3L_3)$$

Decryption next uses the encryption device in that figure, but with two changes: the input is now c and the key schedule is reversed, so K_3 is used in round 1, K_2 in round 2, and K_1 in round 3. We write L'_i and R'_i to denote these block halfs computed in this decryption, so that we can distinguish them from the L_i and R_i computed during encryption. We then have

$$L'_0R'_0 = IP(c) = IP(IP^{-1}(R_3L_3)) = R_3L_3$$

Let us see the effect of the first decryption round on $L'_0R'_0$:

$$\begin{aligned} L'_1 &= R'_0 = L_3 = R_2 \\ R'_1 &= L'_0 \oplus F(K_3, R'_0) \\ &= R_3 \oplus F(K_3, L_3) = (L_2 \oplus F(K_3, R_2)) \oplus F(K_3, R_2) = L_2 \end{aligned}$$

So one round of decryption gives us the new data block R_2L_2, meaning that we have "pealed off" one layer of the decryption. Note that we used the definitions of L_3 and R_3 and that K_3 is the key for the first decryption round (the leftmost occurrence of that key in the equation above).

It is also noteworthy that the expressions that involve the Feistel function F cancelled out merely because of the algebraic properties of $\oplus$ and not because of any properties of function F (other than having the same arguments to allow reasoning such as $a \oplus f \oplus f = a$).

In the same manner, we can now reason that $L'_2 = R_1$ and $R'_2 = L_1$. Similarly, we get that $L'_3 = R_0$ and $R'_3 = L_0$. The device in Figure 4 then outputs the swapped pair $R'_3L'_3$ which equals L_0R_0, and then applies IP^{-1} to it. So the output of the decryption is, using equation (3) above:

$$IP^{-1}(L_0R_0) = IP^{-1}(IP(m)) = m$$

and so this recovers the plaintext correctly, as claimed.

Exercise 9 *Specify the decryption operation for the OFB mode of operation of a block cipher and show that it correctly decrypts ciphertext back to plaintext.*

Solution 9 *We inspect the definition of encryption in the OFB node and observe that the generation of the key stream $E_1 E_2 \ldots E_q$ only depends on the initialization vector IV and the use of the block cipher $e_k(\cdot)$ with key k. In particular, for decryption we generate the key stream $E_1 E_2 \ldots E_q$ in the same manner as done for encryption! In particular, this will generate the same key stream.*

To see this is correct, let c_i be a j-bit "block" of cipher text. We know that c_i equals $m_i \oplus E_i$ where m_i is the ith, j-bit plain text "block". So if we apply $\oplus E_i$ to this we recover $(m_i \oplus E_i) \oplus E_i = m_i$ showing that this is correct.

To emphasize, the above says that the decryption operation is exactly the same as the encryption operation. We just need the ciphertext as input. So whilst this looks like the one-time pad, it is not the one-time pad because the key stream $E_1 E_2 \ldots E_q$ is not entirely random: for example E_2 depends on X_1 and so also on E_1 by definition.

Exercise 10 *Specify the decryption operation for the CFB mode of operation of a block cipher and show that it correctly decrypts ciphertext back to plaintext.*

Solution 10 *Recall that for this mode we have*

$$c_i = m_i \oplus E_i$$

So decryption is the same operation as encryption, as long as we manage to generate the same keystream. Note that for decryption, we only have access to the initialization vector IV and the block cipher $e_k(\cdot)$. The question therefore is only: how can we generate the same keystream as for encryption, given IV, $e_k(\cdot)$, and the ciphertext $c_1 c_2 \ldots c_q$.

For E_1, this is easy as this is defined to be $Z_1 = e_k(Z_0) = e_k(IV)$. For E_{i+1}, this is $Z_{i+1} = e_k(Y_i)$, where Y_i refers to the definition in the encryption state of the mode. But for that, Y_i equals $m_i \oplus E_i$ and so equals c_i. But c_i is given to us, and so we can compute Z_{i+1} as $e_k(c_i)$.

That this is correct is clear because we generate the same key stream as for decryption, so the correctness argument is the same as for the one-time pad.

Exercise 11 *In this exercise, assume the use of a block cipher $e_k(\cdot)$ on a plaintext m with k data blocks $m_1 m_2 \dots m_k$, and the resulting ciphertext of k blocks $c_1 c_2 \dots c_k$.*

1. *Let one bit in block c_i of ciphertext c be flipped prior to decryption. Which portions of plaintext will be affected, and how, when decrypting c in ECB mode?*

2. *Repeat the previous exercise for CBC mode.*

3. *Suppose that a single bit is inserted in some block c_i of ciphertext and the last bit of ciphertext is deleted. Which portions of the plaintext will be affected, and how, when decrypting the modified ciphertext in ECB mode?*

4. *Repeat the previous exercise for CBC mode.*

Solution 11 *Let c'_i be result of flipping single bit in ciphertext block c_i.*

1. *This change will only affect the recovery of m_i, as decryption of c_j is independent from that of c_i for $j \neq i$. But decryption c'_i should change about 50% of m_i due to the avalanche effect of the block cipher.*

2. *Avalanche effect will mean $d_k(c'_i)$ changes a lot, and so recovery of m_i via $d_k(c'_i) \oplus c_{i-1}$ severly affected. Recovery $d_k(c_{i+1}) \oplus c'_i$ of m_{i+1} affected only in bit changed in c'_i. All other blocks unaffected.*

3. *Because of the shift of ciphertext from c_i onwards, this will affect the recovery of all plaintext blocks m_j with $j \geq i$.*

4. *Same argument as for the previous item.*

Exercise 12 *A stream cipher is a device that generates a keystream k that is as long as the message m, and then encrypts and decrypts in the same manner as done with the one-time pad. However, the keystream is not completely random but generated from a seed and a state-based feedback loop (where the details vary from design to design and are rooted in mathematics). Assume that these designs are such that the keystream will eventually be periodic.*

Informally discuss the advantages and disadvantages of stream ciphers in comparison to block ciphers, based on the above and your understanding of block ciphers.

Solution 12 *We did not cover stream ciphers, so it is a bit of stretch to answer this. But if you read up on stream ciphers, there are some good things about them:*

1. *we can often prove mathematical properties about them (e.g. the period of the key stream)*
2. *we can build more complex stream ciphers out of simpler ones and reason about that composition mathematically*
3. *the key generation may have good fault-tolerance regarding transmission errors*
4. *decryption and encryption are as for the one-time pad, simple and easy to implement*

But block ciphers also have advantages. They can operation on bulk on an entire block of data, and then can emulate stream ciphers if used in an appropriate mode such as OFB.

Exercise 13 *Discuss the possible meanings of* computationally infeasible *along various dimensions such as mathematical or complexity-theoretic concepts, abilities and resources of an attacker, resiliency to future technological developments, etc.*

For sake of concreteness, you may consider one or all of the three properties that cryptographically secure hash functions should possess.

Solution 13 *The notion of computational feasibility may have at least two types of meaning:*

A practical one: *does an individual, an organization, or an other entity (such as a territorial state) have the computational resources to solve a certain problem within a period of time that would be a threat to us? The resources may not be available for different reasons, two of them being lack of funds and the other one being access to the right technology (e.g. through embargos or because the technology is still in the future!)*

A theoretical one: *can the problem be solved theoretically within certain complexity bounds, for example within probabilistic polynomial time? Failure* *to do so will also suggest that it may be hard to solve this practically; but there is no guarantee for this. For example, scalable quantum computers may be able to solve problems that are in some complexity class that this theoretical model of infeasibility used as a basis.*

Concretely for a hash functions with 160 *bit outputs, the birthday paradox suggests that to break collision resistance no more than about* 2^{80} *computations are needed. This is a considerable hurdle from a practical point of view. But if the hash function is an iterated one, someone may come up with ways of reducing collisions of the hash function to collisions of its compression function, for which the search space may be smaller and within reach of powerful adversaries.*

To take RSA encryption as another example (we cover this later), its security rests on the hardness of integer factorization. And the latter would be solvable with scalable quantum computers. So that is a threat to RSA and also for its use for digital signatures. Imagine a world with scalable quantum computers in which contracts signed with RSA would now be contestable in a court of law.

Exercise 14 *1. Assume that we have an oracle* $\mathcal{O}$ *that, given* y *in the image of a hash function* h*, produces for us some legitimate input* x *of* h *such that* $h(x) = y$*. You may assume that calls to the oracle are efficient. Show that you can use this oracle to efficiently break second preimage resistance of* h*.*

2. Show that the ability to break second preimage resistance can be used to also break collision resistance of a hash function.

Solution 14 *1. This is part of Lemma 10.2 in the textbook on page 154. To break second preimage resistance, we are given some input* x*. We need to find some* $x' \neq x$ *with* $h(x) = h(x')$*. We use the oracle as follows:*

```
$y = h(x);$
$x' = \mathcal{O}(y);$
while $(x' == x)\{\ x' = \mathcal{O}(y); \}$
```

return x';

Since the oracle gives us a "random" x' with the property that $h(x') = y$, it will gives us new such x' with high probability each time we make this call. Since there are intuitively many such x', we can argue that the above procedure terminates with probability 1, and so clearly break second preimage resistance:

- *the returned x' satisfies $h(x') = y$ as the oracle was called with argument y*
- *the returned x' is different from x as otherwise the while loop would not have been exited.*

2. *Now we assume that we have an oracle $\mathcal{O}'$ which, on input x, will find some x' with $x \neq x'$ and $h(x) = h(x')$. We can use this to break collision resistance: choose x at random as input and feed this into $\mathcal{O}'$ to get some $x' \neq x$ with $h(x) = h(x')$. This then breaks the collision resistance of h.*

Exercise 15 1. *Show that the h in Algorithm 14.1 on page 276 of our textbook is* not *collision resistant. HINT: let $s = 8$, $n = 4$, and consider $m_1 = 010$ and as m_2 a "padded" version of m_1.*

2. *Assume that Algorithm 14.1 on page 276 is modified so that, before it divides the message into blocks it adds another block that encodes the length of the original, unpadded message in bits.*
   (a) *Show that this means that messages can have at most 2^l bits.*
   (b) *Assume that function f is collision resistant. Use this to show that the collisions found for h in the previous item are no longer collisions.*

Solution 15 1. *For $s = 8$ and $n = 4$, the type of the compression function is $f\colon \{0,1\}^8 \to \{0,1\}^4$. Messages are padded to multiples of $l = s-n = 8-4 = 4$ bits. There is a constant 4-bit string H. The first input message is $m_1 = 010$. Since this is just one block and not a multiple* *of four, the actual padded input is then $m_1 0$, which adds a zero bit to bring it to block size 4. The second input message is $m_2 = 0100$ which is of block size and so no padding is applied to this message.*

For both messages, Algorithm 14.1 will perform only one iteration as there is only one data block to process. Therefore, we will have $h(m_i) = f(H \mid\mid p(m_i))$ where $p(m_i)$ denotes the padded version of m_i. We can now compute

$$\begin{aligned} h(m_1) &= f(H \mid\mid p(m_1)) \\ &= f(H \mid\mid m_1 0) \\ &= f(H \mid\mid m_2) \\ &= f(H \mid\mid p(m_2)) \\ &= h(m_2) \end{aligned}$$

Since m_1 is different from m_2, we have found a collision of h.

2. (a) *This is easy to see: each data block has l bits, and so this also applies to the last padded block which encodes the length of the unpadded message in binary. Therefore, the unpadded message cannot have more than 2^l bits ($2^l - 1$ to be more accurate), as otherwise the length could not be encoded in a single block.*

   (b) *For $m_1 = 010$, the padding now results into $p(m_1) = m_1 00011$ which adds a zero bit as before but then adds another block 0011 that encodes in binary the length 3 of the unpadded message m_1. Similarly, we now get $p(m_2) = m_1 00100$. Note that the first blocks of these padded versions agree as in the previous item. But the second blocks differ. For m_2 this is 0100, the binary encoding of 4 which is the length of m_2.*

   Therefore, the collision from the previous item will only occur after the first iteration in Algorithm 14.1. But in the second iteration, both computations will evaluate $f(H \mid\mid 0011)$, respectively $f(H \mid\mid 0100)$ and so this will not result in a collision for h unless this is a collision of f itself.

   This illustrates typical reasoning about hash functions: one works hard at making f collision resistant, and then one tries to show that collisions of h would result in collisions of f.

Exercise 16 *We saw that one can build a MAC function out of a hash function h. Show that such a design*

$$MAC_K(m) = h(m \mid\mid K)$$

is not secure, so appending a key to the message before hashing must not be used for computing MACs.

Hint: assume that hash function h is iterated, e.g. as in the Merkle-Damgard construction, and that h is not given in a length-strengthened version. Further assume that you can find two messages $m_1 \neq m_2$ of equal length that give a collision of h (e.g. by relying on a birthday attack). Also assume that the key K has the same length as a data block for h. Now show that the MAC of m_1 equals the MAC of m_2.

Solution 16 *We will also assume that m_1 and m_2 are of length that is a multiple of the block size l for Algorithm 14.1. For example, our birthday attack could only look at messages of such length.*

Since we have that $h(m_1) = h(m_2)$ and since m_1 and m_2 have the same length, a multiple of block size l, and since K is a single data block, we can simply reason as follows:

$$\begin{aligned} MAC_K(m_1) &= h(m_1 \mid\mid K) \\ &= f(h(m_1) \mid\mid K) \\ &= f(h(m_2) \mid\mid K) \\ &= h(m_2 \mid\mid K) \\ &= MAC_K(m_2) \end{aligned}$$

So in this computation we expressed the first $t-1$ iterations in the algorithm as the result $h(m_i)$ and then there was only one iteration left since K is one block. So the remaining computation is $f(h(m_i) \mid\mid K)$ and this is the same for both messages as they are a collision for h.

Exercise 17 *Discuss the benefits and pitfalls of using "padding method zero" and "padding method three" from page 275 in the textbook. You may also want to look up the term "reversible padding scheme" on the web to inform this discussion.*

Solution 17 *This is essentially a discussion of reversible padding. For padding method zero, for example, we saw that both messages* 010 *and* 0100 *have the same padded version* 0100 *if the block size is* 4*. And this caused problems with the hash function. One cause of this collision problem is that the padding scheme is not reversible: for padded version* 0100 *we cannot know whether it came from* 010, 01, *or etc.*

The padding method three is better as it is reversible (for messages for which it is defined, meaning those whose length is less than 2^l *bits). The last block tells us how long the unpadded message is. We can then remove that last block and determine which of the rightmost* 0 *bits are padding bits; we can do this as we know how long the unpadded message is.*

For example, we had 01000100*. We remove the last block* 0100*, noting that the unpadded message has length* 4*. But the remaining message is* 0100 *and has that length. So we remove nothing else.*

Applying the same procedure to 01000011 *means that we remove* 0011 *and note that the unpadded message has length* 3*. But then it means that we move the rightmost* 0 *in* 0100 *to get* 010*.*

Block ciphers need padding schemes that are reversible. But the discussion in previous exercises makes clear that even reversible padding schemes cannot shield against other design weaknesses, e.g. those that allow for length-extension attacks of hash functions.

Exercise 18 *This exercise is about CBC-MAC. Let* $m = m_1m_2\dots m_q$ *be a message of* q *many data blocks, where* m_q *is padded as in padding method zero from page 285 in the textbook. Let*

$$M = \mathit{CBC}\text{-}\mathit{MAC}_K(m)$$

Show that

$$M = \mathit{CBC}\text{-}\mathit{MAC}_K(m \mid\mid M \oplus m_1 \mid\mid m_2m_3\dots m_q)$$

Solution 18 *We know that* M *equals* O_q *as this processed the first* q *blocks*

$m_1 m_2 \ldots m_q$. By definition of CBC-MAC, we then continue to compute

$$\begin{aligned}
I_{q+1} &= (M \oplus m_1) \oplus O_q && O_q \text{ is previous } O \\
&= (M \oplus m_1) \oplus M && \text{as reasoned above} \\
&= m_1 && \text{property of } \oplus \\
&= I_1 && \text{by definition of CBC-MAC}
\end{aligned}$$

So the CBC-MAC computation renders I_1 for input $m \mid\mid (M \oplus m_1)$ and it still has to process $m_2 m_3 \ldots m_q$. But this is the same computational state as the one for computing M after m_1 has just been processed. Therefore, we conclude that O_{2q} equals O_q, which we know is M.

Exercise 19 *This exercise is also about CBC-MAC. It shows that even the padding from padding method three can result in a length-extension attack.*

Assume: the attacker has MAC M_1 of message m_1, and has MAC M_2 of message m_2, where both m_1 and m_2 are just a single data block.

Notation: we write $\mathbb{P}(n)$ for a single data block that encodes in binary the natural number n; this of course limits the size of n by the block size. Let the MAC M of m_1 then be defined as $e_K(e_K(m_1) \oplus \mathbb{P}(b))$ where b is the block size of the block cipher $e_K(\cdot)$, and similarly for messages with more than one block.

1. *Let m_3 be a new data block. Then CBC-MAC$_K(m_1 \mid\mid \mathbb{P}(b) \mid\mid m_3)$ equals*

$$e_K(e_K(e_K(e_K(m_1) \oplus \mathbb{P}(b)) \oplus m_3) \oplus \mathbb{P}(3b))$$

2. *Let the three-block message m be $m_2 \mid\mid \mathbb{P}(b) \mid\mid m_3 \oplus M_1 \oplus M_2$. Then we have*

$$\text{CBC-MAC}_K(m) = \text{CBC-MAC}_K(m_1 \mid\mid \mathbb{P}(b) \mid\mid m_3)$$

Solution 19 *The solution to this exercise can be seen in our textbook on page 286.*

Exercise 20 1. *Describe how you generate a key pair for RSA encryption. In doing so, choose prime numbers p and q that are larger than* 10 *but smaller than* 100, *which is of course unrealistically small.*

2. *State which parts of this RSA setup should be secret, and explain why that is so.*

3. *State the set of plaintexts and the set of ciphertexts for the RSA key pair you computed in the first item above.*

4. *Then choose a plaintext, show details of its encryption, and explain why and how the corresponding ciphertext decrypts to the original plaintext.*

Solution 20 1. *We chose two primes $p = 13$ and $q = 97$. Then $N = 1261$ and $\phi(N) = (p-1) \cdot (q-1) = 12 \cdot 96 = 1152$. Suppose we wanted to choose $e = 3$. This is a popular choice that would not be wise for the basic RSA without randomization. But we cannot choose this here at all since* 3 *has a common factor with $\phi(N)$, in fact it is a factor of $\phi(N)$ and so there is no d with $3 \cdot d = 1 \mod \phi(N)$.*

Let us choose $e = 5$ instead. This has no common factor with $\phi(N)$. Therefore we can run Extended Euclid algorithm and it computes for us that $461 \cdot 5 - 2 \cdot 1152 = 1$. Therefore, we can choose 461 *as multiplicative inverse d of e.*

The public key is $(5, 1261)$ and the private key is $(461, 1261)$.

2. *Secret should be $\phi(N)$, p, q, and d. Public should be N and e.*

3. *The sets $\mathbb{C}$ and $\mathbb{P}$ are equal for RSA and are here $\{0, 1, \ldots, N-1\} = \{0, 1, \ldots, 1260\}$.*

4. *Let $m = 7$. Then $c = m^e \mod N = 7^5 \mod 1261 = 414$.*

To decrypt c, we compute $c^d \mod 1261 = 414^{461} \mod 1261 = 7$. The reason this recovers the plaintext is rooted in Lagrange's Theorem and in the fact that $\mathbb{Z}_N^*$ has $\phi(N)$ many elements.

Exercise 21 *Recall the definition of the Euler totient function $\phi(n)$ for a natural number $m \geq 2$: it is defined as the size of the set $\{a \mid 1 \leq a < m \mid \gcd(a, m) = 1\}$:*

$$\phi(m) = \#\{a \mid 1 \leq a < m \mid \gcd(a, m) = 1\}$$

For example, we have $\phi(9) = 6$ as only the numbers $1, 2, 4, 5, 7, 8$ from $\{1, 2, \dots, 8\}$ are relative prime to 9.

1. *Compute $\phi(60)$.*
2. *Let p and q be primes. Prove that $\phi(p \cdot q)$ equals $(p-1) \cdot (q-1)$. (Note that we used this identity in the definition of RSA.)*
3. *Show that an attacker Eve can compute the private RSA key d if she knows the value of $\phi(N)$. This means that the security of RSA also depends on whether or not computing $\phi(N)$ is a hard problem.*

Solution 21

1. *Euler's product formula states that $\phi(n) = n \cdot \prod(1 - \frac{1}{p_i})$ where i ranges over all primes p_i that are a factor of n. In particular, it is easy to compute $\phi(n)$ is one knows all prime factors of n.*

   We use that formula here for sake of illustration. Since 60 equals $2^2 \cdot 3 \cdot 5$, we get $\phi(60) = 60 \cdot (1 - \frac{1}{2}) \cdot (1 - \frac{1}{3}) \cdot (1 - \frac{1}{5}) = 60 \cdot \frac{1}{2} \cdot \frac{2}{3} \cdot \frac{4}{5} = 16$.

2. *This identity follows readily from Euler's product formula mentioned in the previous item.*

   Let's give a direct proof here, though. We assume that p is not equal to q, which is also the assumption in RSA. Then p and q have no common factor. Using the Chinese Remainder Theorem, we conclude that $\phi(p \cdot q)$ must equal $\phi(p) \cdot \phi(q)$. But $\phi(r)$ equals $r - 1$ for any prime $r > 1$ since all non-zero elements of $\{0, 1, \dots, r-1\}$ have a no common factor with r. This proves the claim.

3. *Suppose that Eve knows $\phi(N)$. Since e is public, she can then run the Extended Euclid algorithm to compute d. She can find a solution since a proper RSA setup ensures that e and $\phi(N)$ have no common factor. If d happens to be negative (a possibility when running Extended Euclid), she will simple take it modulo $\phi(N)$ again to get a natural number.*

   Note that any d that differs from the secret d by a multiple of $\phi(N)$ will work correctly for decryption, courtesy of Lagrange's Theorem.

Exercise 22 *Let $N \geq 2$ be a natural number and let $\mathbb{Z}_N^*$ be the set of natural numbers a with $1 \leq a < N$ such that* $\gcd(a, N) = 1$.

1. *Show that for all a in $\mathbb{Z}_N^*$ there is some b in $\mathbb{Z}_N^*$ such that $a \cdot b = 1 \mod N$. (Hint: use the facts around the Extended Euclid algorithm for example.)*

2. *Assume that N equals $p \cdot q$ for primes p and q. Show that $\mathbb{Z}_N^*$ has $\phi(N)$ many elements.*

3. *Use the fact shown in the previous item to prove that RSA decryption successfully recovers the plain text. (Hint: we basically did this in the notes already but we did not explain why*

$$(m^{(p-1)(q-1)})^s = 1 \mod N$$

Solution 22 1. *Extended Euclid has as input two numbers (here a and N) and computes two integers x and y such that*

$$a \cdot x + N \cdot y = \gcd(a, N)$$

Since $\gcd(a, N)$ is assumed to be 1, we therefore get values x and y with $a \cdot x + N \cdot y = 1$. And so $a \cdot x = 1 \mod N$. Therefore, we can choose as b the value of x returned by Extended Euclid.

2. *This follows from the previous item and the definition of $\phi(N)$.*

3. *To see that $(m^{(p-1)(q-1)})^s = 1 \mod N$, it suffices to show that $m^{(p-1)(q-1)} = 1 \mod N$. Since $\mathbb{Z}_N^*$ has $(p-1) \cdot (q-1)$ many elements, this follows from basic group theory: raising any group element to the power of the group size computes the identity element of that group, which in this case is 1.*

Exercise 23 *Let (e, N) and (d, N) be an RSA key pair. A message m is called* unconcealed *iff*

$$m^e = m \mod N$$

1. *Show that there are unconcealed messages m that are unconcealed independent of the choice of N.*

2. *Assume that there are $(\gcd(e-1,p-1)+1)\cdot(\gcd(e-1,q-1)+1)$ many unconcealed plain-text messages m. Discuss whether this is a concern for the security of RSA and, if so, how implementations can mitigate or prevent such security concerns.*

Solution 23

1. *Well, we always have that 0 and 1 are unconcealed, regardless of the value of N. This problem is fixed if we randomize the input m prior to encryption. Is $N-1$ also unconcealed?*

2. *Whoever generates the RSA system will know e, p, and q. So they will know the value of $(\gcd(e-1,p-1)+1)\cdot(\gcd(e-1,q-1)+1)$. If that value is "large", whatever this would or should mean, a new e should be generates for which this value is not "large". If no such e exists, p or q should be changed and then one would try again.*

   This is a genuine issue. For example, when $p=19$, $q=37$, and $e=181$ we have $N=703$. Then $(\gcd(e-1,p-1)+1)\cdot(\gcd(e-1,q-1)+1)$ equals $(18+1)\cdot(36+1)=N$. So the encryption of all possible plaintext messages will be unconcealed, a disaster!

Exercise 24 *Prove that the RSA problem is no harder than the FACTORING problem. Concretely, consider an RSA system with a modulus N and a key pair for that modulus and then show the following:*

> *"Knowing the factors p and q of the modulus N, allows an attacker to efficiently compute the private RSA key, and to decrypt all ciphertext of that RSA system."*

Solution 24 *This is pretty obvious. Once we have the factors p and q, we exploit that N and e are public knowledge and so we know those values as well. But from p and q we can compute $\phi(N)$ as $(p-1)\cdot(q-1)$. And knowing e and $\phi(N)$ we can then run Extended Euclid to compute a secret key d. With the knowledge of d, we can now solve the RSA problem as we can decrypt all ciphertexts correctly.*

Exercise 25 *Consider an RSA key pair* (N, e) *and* (N, d) *where* d *is the secret key and* $N = p \cdot q$ *for primes* p *and* q*. Below, we write* $c = m^e \mod N$ *to identify the ciphertext of* m*.*

Let $parity(c)$ *denote the least significant bit of* m*. Let* $half(c)$ *be* 0 *if* $0 \leq m \leq N/2$*, and set* $half(c) = 1$ *otherwise.*

Below you should complete staged exercises that will prove that any algorithm that computes $parity(c)$ *or* $half(c)$ *from the ciphertext* c *of* m *can be used as an oracle to build an algorithm that computes* m *from its ciphertext:*

1. *Show that* $half(c) = parity(c \cdot 2^e \mod N)$*.*

2. *Show that* $parity(c) = half(c \cdot (2^{-1})^e \mod N)$*.*

3. *Conclude that the computations of* $parity(c)$ *and* $half(c)$ *are polynomially equivalent: one can convert an algorithm for computing one of these into an algorithm for computing the other one – with polynomial overhead in that conversion.*

4. *Let* $\mathcal{O}$ *be an oracle such that* $\mathcal{O}(c)$ *outputs* $half(c)$*. Design an algorithm that uses this oracle, has ciphertext* c *as input, and computes the corresponding plain-text* m *efficiently (assuming that calls to the oracle take constant time).*

5. *Conclude that the security of RSA also depends on the hardness of computing the least significant bit of a plaintext, given its ciphertext.*

Solution 25 1. *This is easy to see from basic algebra, noting that*

$$(x^e \mod N) \cdot (y^e \mod N) \mod N = (x \cdot y)^e \mod N \qquad (4)$$

2. *This is also easy to see from basic algebra using (4), keeping in mind that* 2^{-1} *is the multiplicative inverse of* 2 *modulo* N*.*

3. *This follows directly from the last two items, as the conversions implicit in these equations can be computed efficiently.*

4. *One such algorithm is given here:*

```
$k = \lfloor \log_2 N \rfloor$;
for $(i = 0 \dots k)$ {
        $y_i = half(c)$;
        $c = (c \cdot 2^e) \mod N$;
    }
    $low = 0$;
    $high = N$;
    for ($i = 0 \ldots k$) {
        $mid = (low + high)/2$;
        if ($y_i == 1$) { $low = mid$; }else { $high = mid$; }
    }
    return $\lfloor high \rfloor$;
```

We leave it to the reader to reason that this correctly returns the plaintext but we give an example of its working: let $N = 1457$, $e = 779$, *and* $c = 722$. *Note that* $2^e \mod N$ *equals* 946. *For these data, we get the following computations of* y_i*:*

i	0	1	2	3	4	5	6	7	8	9	10
y_i	1	0	1	0	1	1	1	1	1	0	0

5. *The last item makes clear that any ability to compute the least significant bit of a plaintext from its ciphertext can be turned into an algorithm that recovers the complete plaintext. Therefore, the security of RSA depends on that the least significant bit of the plaintext cannot be computed from publicly available information such as the ciphertext.*

Exercise 26 *More efficient RSA decryption and RSA signing: Let* (N, e) *and* (N, d) *be an RSA key pair with a private key* d *and* $N = p \cdot q$ *for primes* p *and* q. *The operation* $c^d \mod N$ *is of course used for either decryption or for signing a message, depending on whether the key pair is used as a cryptosystem or as a digital signature system.*

Here is a technique that makes this operation more efficient and is therefore often implemented in practice. First, we precompute the values

$$d_p = d \mod p - 1 \qquad d_q = d \mod q - 1 \qquad q_{inv} = q^{-1} \mod p$$

Then, to compute $m = c^d \mod N$ we compute

$$\begin{array}{ll} m_1 = c^{d_p} \mod p & m_2 = c^{d_q} \mod q \\ h = q_{inv} \cdot (m_1 - m_2) \mod p & m = m_2 + h \cdot q \end{array}$$

1. *Show that the definitions above indeed compute the plaintext m that satisfies $m = c^d \mod N$.*

2. *Explain why this new computation of m may be more efficient that computing $c^d \mod N$, keeping in mind that p and q would have at least 1024 bits.*

Solution 26 1. *Note that $m_1 = c^{d_p} \mod p = c^d \mod p$ as $p-1$ is the order of $\mathbb{Z}_p^*$. Similarly, we get that $m_2 = c^{d_q} \mod q$. Assuming that $q > p$, the setting above is a special case of Garner's Algorithm (e.g. see Algorithm 14.71 in Handbook of Applied Cryptography by Menezes, van Oorschot, and Vanstone.) for computing m.*

2. *This involves two modular exponentiations whereas the normal RSA decryption involves only one modular exponentiation. But the former two are computed for smaller exponents and smaller moduli, making this more efficient for these sizes of p and q.*

   Incidentally, the theoretical speed-up is about a factor of 4. So this is a method you often find on smaller devices. One problem with that method, though, is that p and q have to be stored as they are needed for these computations. The normal implementation of RSA can safely destroy p and q after key generation.

Exercise 27 *Illustrate the workings of the Diffie-Hellman key exchange protocol where p equals 701 and g equals 220:*

1. *If Alice chooses $a = 254$ at random, what concrete value will she send to Bob?*

2. *If Bob chooses $b = 478$ at random, what concrete value will he send to Alice?*

3. *Given the choices of a and b above, what key will Alice and Bob agree upon at the end of the protocol?*

Background information: *The g above is such that it generates the entire group $\mathbb{Z}^*_{701}$. Such elements are called primitive roots. Whenever p is prime, the group $\mathbb{Z}^*_p$ has primitive roots, and these can be computed efficiently. Please see*

`http://www.bluetulip.org/programs/primitive.html`

for an online calculator of primitive roots for small prime values. For example, it informs us that there are 240 different primitive roots for $p = 701$, and it lists them all for us.

It turns out that the use of primitive roots leaks at least one bit of information. This is a reason, apart from efficiency, why one would like to pick a g that generates a large but proper subgroup of $\mathbb{Z}^*_p$. One technique is to use primes p of form $2q+1$ for some prime q. It is then not hard to find g whose subgroup has order q.

Solution 27

1. *Alive will send $g^a \mod p = 220^{254} \mod 701 = 668$ to Bob.*
2. *Bob will send $g^b \mod p = 220^{478} \mod 701 = 272$ to Alice.*
3. *Alice will compute as key $272^{254} \mod 701 = 29$. Similarly, Bob will compute as key $668^{478} \mod 701 = 29$, and so they agree on key 29.*

Exercise 28 *Assume that we use an RSA signature scheme based on a hash function h and key pairs (e, N) and (d, N) as follows: The signature of a message m is $s = h(m)^d \mod N$ and verifiers receive the pair (m, s). Such a pair is verified in three steps:*

- *Compute $h' = s^e \mod N$, an 'encryption'*
- *Compute $h(m)$ from m*
- *Accept the signature if and only if $h(m)$ equals h'.*

Analyze what capabilities an attacker might have when the following properties of the used hash function h are not *true:*

1. *preimage resistance*
2. *collision resistance*
3. *second preimage resistance*

Solution 28

1. *preimage resistance: this prevents an attacker from cooking up a message with a given signature; failure of this would allow the attacker to compute $h' = r^e \mod N$ for some random r, and compute a preimage $m = h^{-1}(h')$ of h' under h. Now the attacker has a signature (m, r) for message m. This is called an* existential forgery *since the attacker cannot control the content of the message m.*
2. *collision resistance: failure of this makes the signer into a potential attacker: he chooses two m and m' with $h(m) = h(m')$. He signs m to get signature (m, s). We can later repudiate this signature, saying that he really signed m'.*
3. *second preimage resistance: failure of this allows the attacker to obtain your signature (m, s) for message m, find another m' with $h(m') = h(m)$ and now has signature (m', s) on message m'.*

Exercise 29 *Let $\mathcal{P} = \{A, B, C, D\}$ be a set of parties and let $m(\Gamma)$ be the set of all subsets of $\mathcal{P}$ is size 2. Assume that the secret s is a binary string of length l.*

1. *Compute shares s_P for each party P in $\mathcal{P}$, using the Ito-Nishizeki-Seito secret sharing scheme.*
2. *Informally explain why this scheme is information-theoretically secure for this example. Specify any assumptions needed for this argument.*

Solution 29

1. *The solution for this can be seen in the Example of our textbook on pages 405 (bottom) and page 406.*

2. *The only sets that must not be able to recover the secret are those that have less than 2 elements. Clearly, we cannot recover the secret from the empty set of parties, there is no information in that. Let us consider a set of one party. By symmetry, it suffices to consider the set $\{A\}$. The share s_A equals $s_1 \mid\mid s_3 \mid\mid s_5$ where s_1 represents A in $\{A, B\}$, s_3 represents A in $\{A, C\}$, and s_5 represents A in $\{A, D\}$. The attacker therefore knows s_1, s_3, s_5, and the equations*

   $$\begin{aligned} s &= s_1 \oplus s_2 \\ s &= s_3 \oplus s_4 \\ s &= s_5 \oplus s_6 \end{aligned}$$

   We assume that these equations are generated in a truly random fashion. More precisely, we assume that one of the s_1 or s_2 is randomly generated, say s_1 and so s_2 is determined as $s_1 \oplus s$. Similarly, we proceed with the other equations. The attacker knows nothing about the secret (another assumption). The properties of $\oplus$ and the fact that the known information is either truly random or determined by the exclusive or of the secret with truly random information mean that the attacker cannot learn anything about the secret.

   This is very much like the situation of properly implementing a one-time pad, in this case three of them.

Exercise 30 *Let $\mathcal{P}$ be $\{A, C, B, D\}$ and let*

$$m(\Gamma) = \{\{A, C, D\}, \{A, B\}, \{B, C\}, \{B, D\}\}$$

Let s be a secret represented as a binary string of length l. Develop the Replicated Secret Sharing Scheme for this access structure:

1. *Compute all maximally non-qualifying sets $\mathcal{O}_i$.*
2. *Compute the complements B_i of all sets in the previous item.*
3. *Explain how the secret is split up into shares s_i, in particular specify which strings are generated randomly.*
4. *Compute the shares for each party.*

5. *Explain why no information can be computed from a combination of the shares in the non-qualifying set* $\{A, D\}$.

Solution 30 1. *We get*

$$\begin{array}{rcl} \mathcal{O}_1 & = & \{A, C\} \\ \mathcal{O}_2 & = & \{A, D\} \\ \mathcal{O}_3 & = & \{B\} \\ \mathcal{O}_4 & = & \{C, D\} \end{array}$$

All these sets are non-qualifying, and all non-qualifying sets are contained in one of these sets.

2. *The complements are therefore*

$$\begin{array}{rcl} B_1 & = & \{B, D\} \\ B_2 & = & \{B, C\} \\ B_3 & = & \{A, C, D\} \\ B_4 & = & \{A, B\} \end{array}$$

3. *We generate* s_1*,* s_2*, and* s_3 *at random and set* s_4 *to be equal to* $(s_1 \oplus s_2 \oplus s_3) \oplus s$*. This will ensure that* s *equals* $s_1 \oplus s_2 \oplus s_3 \oplus s_4$*. And it will ensure that the scheme is secure.*

4. *For the share* s_P *of party* P*, we need to concatenate all* s_i *for which* P *is in set* B_i*. Therefore, we get*

$$\begin{array}{rcl} s_A & = & s_3 \,||\, s_4 \\ s_B & = & s_1 \,||\, s_2 \,||\, s_4 \\ s_C & = & s_2 \,||\, s_3 \\ s_D & = & s_1 \,|\, s_3 \end{array}$$

(Note that the set of complements equals $m(\Gamma)$ *here, this is an artefact of the example and is not generally the case for this scheme.)*

5. *Let the attacker know the shares in set* $\{A, D\}$*. So the attacker knows* $s_3 \mid\mid s_4$ *and* $s_1 \mid s_3$ *as well as the equation* $s = s_1 \oplus s_2 \oplus s_3 \oplus s_4$*. We may rewrite that equation as*

$$s = (s_1 \oplus s_3 \oplus s_4) \oplus s_2$$

We can now see that this is perfectly secure assuming that s_2 *was generated truly randomly, as solving for* s *is then like solving a one-time pad.*

Exercise 31 *Consider the following set-up for the Reed-Solomon code:* q *equals* 101*,* n *equals* 7*,* t *equals* 2*, and* X *equals* $\{1, 2, 3, 4, 5, 6, 7\}$*.*

1. *Let* f *in* $\mathcal{P}$ *be given by* $f(X) = 26 + 67X + 93X^2$*. Compute the Reed-Solomon code* c *for this* f *and the above setting.*

2. *Let* $Y = \{1, 4, 7\}$*. Let* $s_i = f(i)$ *for all* i *in* X*. Explain in detail how you can recover from* $\{s_1, s_4, s_7\}$ *the secret* $f(0) = 26$*.*

Solution 31

1. *We need to evaluate all elements of* X *at* f *to get* $c = (f(1), f(2), \ldots, f(7))$*. For example,* $f(1) = 26 + 67 \cdot 1 + 93 \cdot 1^2 = 85 \mod 101$*. In similar fashion we compute*

$$c = (85, 27, 54, 65, 60, 39, 2)$$

2. *We choose the approach in which we compute the recombination vector* $r_Y = (r_{1,Y}, r_{4,Y}, r_{7,Y})$*. We do this as follows:*

   - $r_{1,Y} = ((-4) \cdot (1-4)^{-1}) \cdot ((-7) \cdot (1-7)^{-1}) = 4 \cdot 7 \cdot (-3)^{-1} \cdot (-6)^{-1} = 28 \cdot ((-3) \cdot (-6))^{-1} = 28 \cdot 18^{-1} = 28 \cdot 73 = 24 \mod 101$*. Note that we used that* $\mathbb{Z}^*_{101}$ *is a commutative group, so* $a^{-1} \cdot b^{-1}$ *equals* $(a \cdot b)^{-1}$ *and so we only have to compute one multiplicative inverse, not two.*

   - *Similarly, we compute* $r_{4,Y} = ((-1) \cdot (4-1)^{-1}) \cdot ((-7) \cdot (4-7)^{-1}) = 1 \cdot 7 \cdot 3^{-1} \cdot (-3)^{-1} = 7 \cdot (3 \cdot (-3))^{-1} = 7 \cdot (-9)^{-1} = 7 \cdot 56 = 89 \mod 101$.

- *And* $r_{7,Y} = ((-1) \cdot (7-1)^{-1}) \cdot ((-4) \cdot (7-4)^{-1}) = 1 \cdot 4 \cdot 6^{-1} \cdot 3^{-1} = 4 \cdot (6 \cdot 3)^{-1} = 4 \cdot 18^{-1} = 4 \cdot 73 = 90 \mod 101$.

We know that

$$f(0) = r_{1,Y} \cdot s_1 + r_{4,Y} \cdot s_4 + r_{7,Y} \cdot s_7 \mod 101$$

and so this evaluates to

$$24 \cdot 85 + 89 \cdot 65 + 90 \cdot 2 = 20 + 28 + 79 = 26 \mod 101$$

This is correct since $f(0)$ *equals* 26.

Exercise 32 *Consider an arithmetic variant of the Ito-Nishizeki-Saito Secret Sharing Scheme: in the same manner as for that scheme, we can write a monotone access structure* Γ *over a set of parties* $\mathcal{P}$ *as a Boolean formula* $\phi_{m(\Gamma)}$ *that is in DNF form, contains no negation, and has parties from* $\mathcal{P}$ *as atomic propositions.*

Let m *be a large integer such that the secret* s *is a number in* $\{0, 1, \ldots, m\}$. *For each disjunct* C_i *of* $\phi_{m(\Gamma)}$, *we do the following. Let* $A_i^1, \ldots, A_i^{k_i}$ *be the parties that occur in* C_i *in that order. We randomly generate numbers* a_i^j *in* $\{0, 1, \ldots, m\}$ *for all* $1 \leq j < k_i$ *and let* a_i^j *be the share of* A_i^j *for* C_i. *For* $A_i^{k_i}$ *we set the share for* C_i *to be* $s - \sum_{l=1}^{k_i-1} a_i^l$. *The share of a party* A *in* $\mathcal{P}$ *is the set of all shares that it has for clauses* C_i *in which* A *occurs.*

In the exercises below, you can understand how this scheme works by means of a simple example. Let $\mathcal{P} = \{A, B, C, D\}$ *and*

$$m(\Gamma) = \{\{A, B, D\}, \{A, C, D\}, \{B, C\}\}$$

Let the secret s *be an integer with* $0 \leq s \leq m$ *for a large positive integer* m. *Then clause* C_1 *is* $A \wedge B \wedge D$, *clause* C_2 *is* $A \wedge C \wedge D$, *and clause* C_3 *is* $B \wedge C$. *The parties that occur in clause* C_1 *are* A, B, *and* D. *Etc.*

1. *Write down the shares for each party.*
2. *Show that all* $\mathcal{O}$ *of* Γ *can recover the secret.*
3. *Explain why no* $\mathcal{O}$ *with* $\mathcal{O} \notin \Gamma$ *can recover the secret.*

Solution 32 1. *The shares for parties are as follows:*

- *Party A receives shares a_1^1 and a_2^1.*
- *Party B receives shares a_1^2 and a_3^1.*
- *Party C receives shares a_2^2 and $s - a_3^1 \mod m$.*
- *Party D receives shares $s - a_1^1 - a_1^2 \mod m$ and $s - a_2^1 - a_2^2 \mod m$.*

2. *We need to look at all of the three elements of $m(\Gamma)$, and this suffices by monotonicity:*

   - *For set $\{A, B, D\}$ they can recover s as $a_1^1 + a_1^2 + (s - a_1^1 - a_1^2 \mod m) \mod m$.*
   - *For set $\{A, C, D\}$ they can recover s as $a_2^1 + a_2^2 + (s - a_2^1 - a_2^2 \mod m) \mod m$.*
   - *For set $\{B, C\}$ then can recover s as $a_3^1 + (s - a_3^1 \mod m) \mod m$.*

3. *Let $\mathcal{O} \notin \Gamma$ be given. We know that $\mathcal{O}$ is contained in a maximally non-qualifying set $\mathcal{O}_i$. So it suffices, by monotonicity, to show that no maximally non-qualifying set can recover s. These sets are*

$$\{A, B\}, \{A, C\}, \{A, D\}, \{B, D\}, \{C, D\}$$

We only illustrate this here for one of these sets, $\{A, B\}$. Together, A and B own the shares a_1^1, a_2^1, a_1^2, and a_3^1. Since all elements a_i^j are generated at random, we can see that they cannot recover s.

The example above generalizes to what is called the Monotone Circuit Construction. Please refer to Chapter 11 in Stinson's book Cryptography: Theory and Practice *for further details on this.*

Exercise 33 *Let p be an odd prime (i.e. $p \neq 2$), x a natural number with $1 \leq x \leq p - 1$. Then x is a quadratic residue modulo p iff the equation $y^2 = x \mod p$ has a solution y in $\mathbb{Z}_p$.*

Fermat's Little Theorem states that $b^{p-1} = 1 \mod p$ for prime p and all b in $\mathbb{Z}_p^*$. Use that theorem to prove that one can decide, in polynomial time, whether x is a quadratic residue modulo p:

1. *Show that if x is a quadratic residue modulo p, then $x^{(p-1)/2} = 1 \mod p$.*

2. *Conversely, show that if $x^{(p-1)/2)} = 1 \mod p$, then x is a quadratic residue modulo p.*

3. *Describe a polynomial-time decision procedure for quadratic residues modulo p.*

4. *Now suppose that, instead of a prime p we have $N = p \cdot q$ for large primes p and q but where p and q are not known. Do you think that we can efficiently decide whether x is a quadratic residue modulo N?*

Solution 33 1. *We know that x is a quadratic residue modulo p. Therefore, there is some y in $\mathbb{Z}_p$ such that $y^2 = x \mod p$. Since $1 \leq x \leq p-1$, it follows that y is neither 0 nor p. Therefore, y is in $\mathbb{Z}_p^*$. So we can compute*

$$\begin{aligned} x^{(p-1)/2} &= (y^2)^{(p-1)/2)} \mod p \\ &= y^{2\cdot(p-1)/2} \mod p \\ &= y^{p-1} \mod p \\ &= 1 \mod p \qquad \textit{(y in } \mathbb{Z}_p^* \textit{ and Fermat's Little Theorem)} \end{aligned}$$

2. *Conversely, let $x^{(p-1)/2} = 1 \mod p$. Since p is prime, we know that $\mathbb{Z}_p^*$ has a generator g (a primitive root of p). Since x is in $\mathbb{Z}_p^*$, there is therefore some natural number i with $x = g^i \mod p$. We can then compute*

$$\begin{aligned} 1 &= x^{(p-1)/2} \\ &= (g^i)^{(p-1)/2)} \mod p \\ &= g^{i\cdot(p-1)/2} \mod p \end{aligned}$$

Since g generates all of $\mathbb{Z}_p^*$, it has order $p-1$. But since $g^{i\cdot(p-1)/2} \mod p = 1$, this implies that $i \cdot (p-1)/2$ is a multiple of $p-1$, and so $p-1$ divides $i \cdot (p-1)/2$. Since both are integers, this means that i is even. Therefore, $x = (g^{i/2})^2 \mod p$ shows that x is a quadratic residue modulo p.

3. *This directly follows from the previous two items: given p and x as above, the algorithm compute $x^{(p-1)/2} \mod p$, which it can do efficiently in p (and so in x as well). If that computation results in 1 it replies with "Yes", otherwise it replies with "No". This correctly decides quadratic residues modulo p because of the facts shown in the previous two items.*

4. *It can be shown that, under the assumptions of this item, the decision problem for quadratic residues modulo N is polynomial-time equivalent to the problem of factoring N into p and q. The latter is believed to be a hard problem and so we think that the former is a hard problem as well.*

Exercise 34 **RSA Factoring and Computing Square roots.** *Let $N = p \cdot q$ for two large primes p and q, and let (N, e) and (N, d) be the public and private key, respectively. Consider the equation $x^2 = 1 \mod N$.*

1. *Use the Chinese Remainder Theorem (CRT) to show that there are four solutions to this equation.*

2. *Explain that two of these four solutions are always the same values, these are called trivial solutions.*

3. *Suppose that x is a non-trivial solution for $x^2 = 1 \mod N$. Show that you can now factor N, knowing the value of x.*

4. *Give an example of small values p and q that illustrates the three points above.*

5. *It can be shown that any oracle for computing a secret key d from a public key (N, e) can be used to compute (with a so called Las Vegas algorithm) a non-trivial square root of 1 modulo N. Discuss what ramifications this fact has for the usage of RSA.*

Solution 34 1. *By the CRT, we know that each solution of $x^2 \mod N$ is determined by a solution of both $x^2 = 1 \mod p$ and $x^2 = 1 \mod q$. The latter equations have as solutions the values 1 and -1. Since we can combine each of these solutions to get a solution for $x^2 = 1 \mod N$, we get four solutions for this latter equation.*

2. *Clearly, $x = 1$ and $x = -1$ are solutions to $x^2 \mod N$, and these are the trivial solutions.*

3. *Let x be such that $x^2 = 1 \mod N$ and $x \notin \{-1, 1\}$. In particular, $x^2 - 1 = 0 \mod N$. Since $x^2 - 1$ equals $(x - 1) \cdot (x + 1)$, we infer that $(x - 1) \cdot (x + 1) = 0 \mod N$. Since x is not in $\{-1, 1\}$, it follows that neither $x - 1$ nor $x + 1$ equal 0 and so they are in $\{1, 2, \ldots, N - 1\}$. But then N is neither a factor of $(x + 1)$ nor of $(x - 1)$. From this and the CRT it follows that $\gcd(x + 1, N)$ is in $\{p, q\}$ or $\gcd(x - 1, N)$ is in $\{p, q\}$: from $(x - 1) \cdot (x + 1) = 0 \mod p$ we infer that p divides $x - 1$ or $x + 1$. Dually, from $(x - 1) \cdot (x + 1) = 0 \mod q$ we infer that q divides $x - 1$ or $x + 1$. (It is possible to show that both $\gcd$ expressions are in $\{p, q\}$.) Since N only has p and q as factors, we infer that $\gcd(x+1, N)$ is in $\{p, q\}$ or $\gcd(x - 1, N)$ is in $\{p, q\}$ as claimed; and if they are not in this set, they have to equal 1. So we can run extended Euclid on these two inputs and will know that at least one output will be different from 1 and will equal a factor of N.*

4. *Let $N = 403$, $p = 13$, and $q = 31$. The four square roots of 1 are 1, 92, 311, and 402 (which is $-1 \mod 403$). For example, 92 is the solution using the CRT on the two equations $x = 1 \mod 13$ and $x = -1 \mod 31$. We have that $\gcd(93, 403) = 31$ and $\gcd(312, 403) = 13$.*

5. *It means that any method that can compute the secret key d from the private key (N, e) can be used to factor the RSA modulus N. For example, if someone has access to an API service that computes private keys given a public key, he or she could use this to factor the N. This also suggests that recomputing public/private keys for the same modulus is a dangerous idea that should be limited if not disallowed.*

Exercise 35 *Let $N = p \cdot q$ for large primes p and q. Let m be a quadratic residue modulo N, i.e. the equation $r^2 = m \mod N$ has a solution in r. Assume that Peggy knows neither the factorization of N nor a square root of m modulo N. Let $X = \mathbb{Z}_N^*$ and $Y = QR(N)$, the latter being the set of elements of $\mathbb{Z}_N$ that are quadratic residues modulo N. We define a bit commitment scheme (a scheme that commits to a single bit b) as follows:*

$$f\colon \{0, 1\} \times X \to Y, \qquad f(b, x) = m^b \cdot x^2 \mod N$$

1. *Show that f is well defined.*

2. *Show that the bit commitment scheme is information-theoretically concealing, and at best computationally binding.*

3. *Show that, if Peggy could compute squares roots of quadratic residues modulo N, then she could completely break the binding of this bit commitment scheme.*

Solution 35 1. *We only have to show that $f(b,x)$ is a quadratic residue modulo N, and so a member of Y as claimed. If $b = 0$, then $f(b,x) = x^2 \mod N$ which is clearly a quadratic residue modulo N. If $b = 1$, then $f(b,x) = m^1 \cdot x^2 = m \cdot x^2 = r^2 \cdot x^2 = (r \cdot x)^2 \mod N$ where r is a square root of m modulo N (which exists by assumption on m). Therefore, $f(b,x)$ is a quadratic residue modulo N as claimed.*

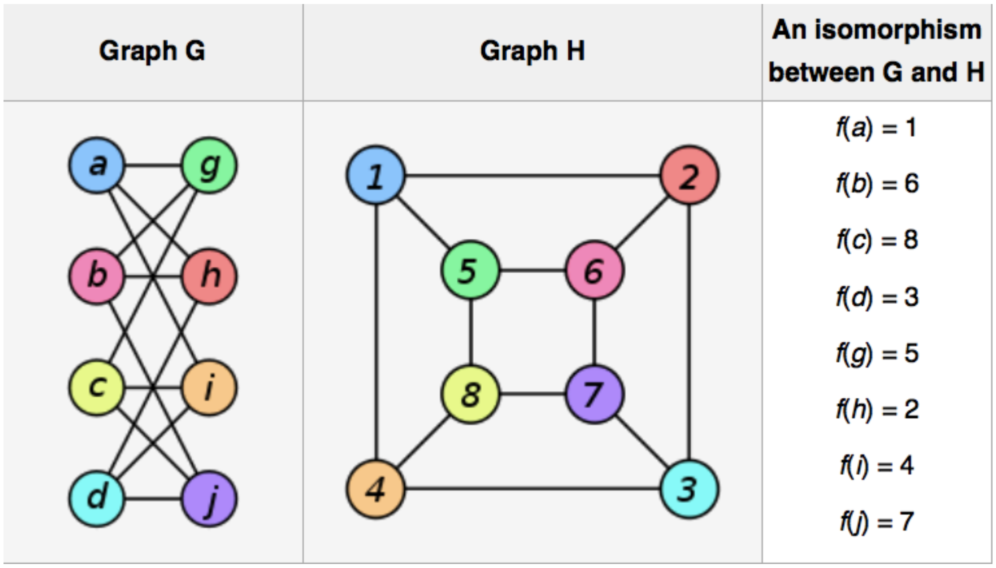
2. *We know that all commitments are quadratic residues modulo N. Moreover, all quadratic residues y modulo N can be the commitment of bit 0 or of bit 1. To see this, for all such y there is some r_y with $r_y^2 = y \mod N$. Using that $m = r^2 \mod N$ as above, we get*

$$\begin{aligned} f(0, r_y) &= m^0 \cdot r_y^2 = r_y^2 = y \mod N & (5)\\ f(1, r_y \cdot r^{-1} \mod N) &= m^1 \cdot (r_y \cdot r^{-1})^2 & (6)\\ &= m \cdot r_y^2 \cdot r^{-2} = m \cdot y \cdot m^{-1} = y \mod N \end{aligned}$$

Therefore, y is of form $f(0,\cdot)$ and of form $f(1,\cdot)$ which makes this scheme information-theoretically concealing.

But we also know that an information-theoretically concealing commitment scheme can at best be computationally binding – it cannot be information-theoretically binding as well.

3. *Since Peggy can compute square roots modulo N, she knows r with $r^2 = m \mod N$. Let y in Y be given. Since this is also a quadratic residue modulo N, Peggy can compute r_y with $r_y^2 = y \mod N$. Therefore, Peggy can compute both expressions $f(0, r_y)$ and $f(1, r_y)$ as in (5) and so could claim for any bit b that her commitment y is really for that bit b. Clearly, this breaks the binding property of the scheme.*



Graph G	Graph H	An isomorphism between G and H
		f(a) = 1 f(b) = 6 f(c) = 8 f(d) = 3 f(g) = 5 f(h) = 2 f(i) = 4 f(j) = 7

Figure 1: Two isomorphic graphs and an isomorphism between them

Exercise 36 *Consider the two graphs and an isomorphism f between them in Figure 1, taken from* `en.wikipedia.org/wiki/Graph_isomorphism}`. *In the notation of zero knowledge proofs, let G_1 be the G from the figure, G_2 the H from the figure, and $\phi\colon G_1 \to G_2$ be the f from the figure.*

1. *Describe a round of a zero knowledge proof in which Peggy wants to prove to Victor that she knows an isomorphism between G_1 and G_2. That round should use the isomorphism from the figure and the challenge of Victor should be $b = 1$.*

2. *Show that Victor would accept the proof of the round in the above item.*

3. *Let the challenge of Victor now be $b = 2$. How does the round now differ, and why does Victor still accept the proof?*

Solution 36 *1. First Peggy generates a random (secret) permutation ψ on $V_2 = \{1, 2, \ldots, 7, 8\}$. For example, suppose that ψ is $(2, 5, 4, 3, 7)$. She then computes $H = \psi(G_2)$ for this, as for example in Figure 2, and sends H to Victor.*

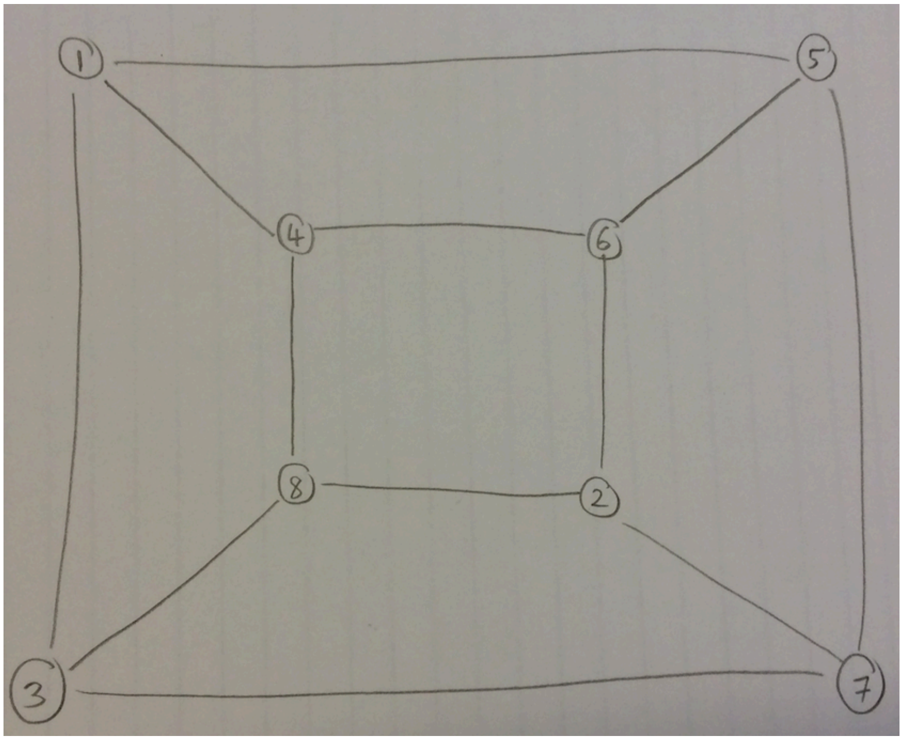


Figure 2: Graph H computes as $\psi(G_2)$ for permutation $\psi = (2, 5, 4, 3, 7)$.

Victor now replies with a random b in $\{1, 2\}$, which we are asked to assume equals 2. So Victor's challenge is 1.

Peggy now computes $\chi = \psi \circ \phi$ where $\phi \colon G_1 \to G_2$ is the isomorphism she knows. Note that $\chi \colon G_1 \to H$ as desired. We have that

$$\begin{array}{lll} \chi(a) = 1 & \chi(b) = 6 & \chi(c) = 8 \\ \chi(d) = \psi(3) = 7 & \chi(g) = \psi(5) = 4 & \chi(h) = \psi(2) = 5 \\ \chi(i) = \psi(4) = 3 & \chi(j) = \psi(7) = 2 & \end{array}$$

This concludes the round of this zero knowledge proof.

2. *Victor wants to verify whether $\chi \colon G_1 \to H$ is an isomorphism of graphs. It is clearly a bijection between the respective sets of nodes. He now has to check that the edge relations are isomorphic. We illustrate this for node h in G_1. He knows that $\chi(h) = 5$. In G_1, node h has an edge to a, b, and d and to no other node. So Victor verifies that, in H, node 5 has edges to $\chi(a) = 1$, $\chi(b) = 6$, and $\chi(d) = 7$ and to no other node. This is clearly the case. Victor would check all remaining nodes in this fashion (with obvious optimizations if desired).*

3. *Let's say that Peggy would choose ψ to be $(2, 5, 4, 3, 7)$ as above (assuming that it is the same run just with a different challenge of Victor). Since Victor sends 2 as a challenge, Peggy replies with $\chi = \psi$ and Victor has to verify that $\chi \colon G_2 \to H$ is a graph isomorphism. But this is the case by construction of H (assuming that Peggy did not make a mistake when computing the image of G_2 under ψ).*

Exercise 37 *Show that the decision problem of "graph isomorphism" is in NP. That is to say, one can check efficiently whether a claimed isomorphism between two graphs is indeed an isomorphism.*

Solution 37 *Let $\phi \colon G_1 \to G_2$ be a claimed isomorphism of graphs $G_i = (V_i, E_i)$. Then we can check the correctness of this claim as follows:*

- *We verify that $\phi \colon V_1 \to V_2$ is a bijection; failure means that the graphs are not isomorphic and the claim is then false.*

- *We verify for each node v in V_1 that the sets $\{\phi(w) \mid (v, w) \in E_1\}$ and $\{w' \mid (\phi(v), w') \in E_2\}$ are equal; this verifies that nodes v and $\phi(v)$ have the same edges "modulo" ϕ. Failure of this at any node means that ϕ is not an isomorphism. Success at all nodes means that ϕ is a graph isomorphism.*

This procedure is clearly efficient in the sizes of G_1, G_2, and ϕ.

Exercise 38 *Suppose that Peggy plays honestly in the protocol for zero knowledge proofs of graph isomorphism (in particular, she knows a graph isomorphism $\phi \colon G_1 \to G_2$), and that she generates the same H on two different runs of this protocols.*

Show that Victor can then learn the graph isomorphism between G_1 and G_2 that Peggy knows (and that forms the basis of her protocol moves).

Solution 38 *In the first run we have*

1. *$P \to V \colon H$ where $\psi \colon G_2 \to H$ chosen by Peggy*
2. *$V \to P \colon b = 1$*
3. *$P \to V \colon \chi = \psi \circ \phi$*

In the second run we have

1. $P \to V \colon H$ where $\psi \colon G_2 \to H$ is the same ψ chosen by Peggy from the first run.
2. $V \to P \colon b = 2$ so Victor now choses the other bit
3. $P \to V \colon \chi = \psi$

From the second run, Victor learns $\psi \colon G_2 \to H$ and so can easily compute $\psi^{-1} \colon H \to G_2$ from it. From the first run, Victor knows $\psi \circ \phi \colon G_1 \to H$ and so can compose this with ψ^{-1} to obtain

$$\psi^{-1} \circ (\psi \circ \phi) = (\psi^{-1} \circ \psi) \circ \phi = \phi$$

as claimed.

Exercise 39 *A graph $G = (V, E)$ is k-colorable for $k > 1$ if there is a function $\phi\colon V \to \{1, 2, \dots k\}$ such that for all (v, w) in E we have $\psi(v) \neq \psi(w)$.*

1. *Consider the graph G in Figure 1. Show that G is 2-colorable.*
2. *Find a graph that is 3-colorable but not 2-colorable.*

Solution 39

1. *Let $\psi\colon V \to \{1, 2\}$ be given by $\psi(x) = 1$ if x is in $\{a, b, c, d\}$, and $\phi(x) = 2$ if x is in $\{g, h, i, j\}$. It is clear that this is a 2-coloring: graph G is bipartite.*
2. *Let $H = (\{a, b, c\}, \{(a, b), (b, c), (c, a)\}$. Then H has a cycle abc in E of odd length 3, and so H cannot be 2-colorable. But $\psi(a) = 1$, $\psi(b) = 2$, and $\psi(c) = 3$ shows that H is 3-colorable.*

Exercise 40 *In the proof of Theorem 21.1, it speaks of the "obvious" simulation for the zero-knowledge proof of 3-colorability. Describe such a simulation for a round of that zero-knowledge proof.*

Solution 40 *Note that Peggy permutes her 3-colorability witness $\psi\colon V \to \{1, 2, 3\}$ by composing it with a permutation π of $\{1, 2, 3\}$. Then she generates the tuple of commitments*

$$(C(\pi(\psi(v_i)); r_i))_{v_i \in V}$$

for randomly generated and secret r_i. Victor's challenge consists of an edge (v_i, v_j) in E, upon which Peggy needs to decommit and send Victor the pairs $(\pi(\psi(v_i)), r_i)$ and $(\pi(\psi(v_j), r_j)$. Victor then verifies whether $\pi(\psi(x_i))$ equals $\pi(\psi(x_j))$.

We now have to simulate this. We may think of a commitment as a tuple whose elements are pairs (c, r) where c is a color (an element in $\{1, 2, 3\}$) and r is random. So Victor can simulate a transcript of the real protocol as follows:

1. *He issues a challenge (v_i, v_j) from relation E.*

2. *He generates independently at random an element c_i of $\{1, 2, 3\}$ and a random number r_i for all v_i in V. He computes as commitment the tuple*
$$r = ((c_i, r_i))_{v_i \in V}$$

3. *He poses as response the pairs (c_i, r_i) and (c_j, r_j). The transcript is therefore*

   1. $P \to V$: $r = ((c_i, r_i))_{v_i \in V}$
   2. $V \to P$: (v_i, v_j) *in* E
   3. $P \to V$: (c_i, r_i) *and* (c_j, v_j).

   This transcript is not distinguishable from a real transcript of this zero knowledge protocol.

   (Note that simulated transcripts may or may not be accepted, this is orthogonal to whether something is a legitimate transcript, in the sense that it could *have come from a protocol run (in which Peggy may or may not know the secret.)*

Exercise 41 *Prove that the Chaum-Pederson Sigma Protocol satisfied soundness, assuming an honest verifier.*

Solution 41 *Since Victor is assumed to be honest, it suffices to show the special soundness property. So let there be two runs of this protocol that (i) have the same commitment, (ii) have different challenges, and (iii) are both accepted by Victor. We need to build from this an efficient extractor of the knowledge, which in this case is the two discrete logarithms of $y_1 = g^{x_1}$ and $y_2 = g^{x_2}$ and the fact that x_1 equals x_2.*

We can write the two protocol runs as tuples $((r_1, r_2), e, s)$ and $((r_1, r_2), e', s')$, which clearly share the same commitment (r_1, r_2), have different challenges (meaning that $e \neq e'$), and that both are accepted by Victor. The latter means that we have

$$\begin{aligned} r_1 &= g^s \cdot y_1^e = g^{s'} \cdot y_1^{e'} \\ r_2 &= h^s \cdot y_2^e = h^{s'} \cdot y_2^{e'} \end{aligned}$$

Since h and g have the same order as the group G, this implies that

$$\begin{array}{rcl} y_1^{e-e'} & = & g^{s'-s} \\ y_2^{e-e'} & = & h^{s'-s} \end{array}$$

From this we infer that

$$\begin{array}{rcl} (e-e') \cdot dlog_g(y_1) & = & s'-s \\ (e-e') \cdot dlog_h(y_2) & = & s'-s \end{array} \tag{7}$$

where $dlog_z(y)$ is the discrete logarithm of y with respect to z, i.e. the a in $\mathbb{Z}_q$ with $y = z^x$. But from the equations in (7), we see that both $dlog_g(y_1)$ and $dlog_h(y_2)$ are equal, and have value $(s'-s) \cdot (e-e')^{-1} \mod q$. Victor can compute this value efficiently.

Exercise 42 *Consider the standard notation for the Sigma Protocol of the Pedersen Commitments, where Peggy wants to prove that $y = g_1^{x_1} \cdot g_2^{x_2}$ where g_1 and g_2 are publicly known element of order q in a group G of prime order q, and x_1 and x_2 are known to Peggy:*

$$\begin{array}{rcl} R(x,(k_1,k_2)) & := & (r_1,r_2) \\ S(e,(x_1,x_2),(k_1,k_2)) & := & (s_1,s_2) \\ V((r_1,r_2),e,(s_1,s_2)) & := & true \Leftrightarrow (g_1^{s_1} \cdot g_2^{s_2} = y^e \cdot r_1 \cdot r_2) \\ S'(e,(s_1,s_2)) & := & (r_1, g_1^{s_1} \cdot g_2^{s_2} \cdot y^{-e} \cdot r_1^{-1} \mod q) \end{array}$$

where $r_i = g_i^{k_i}$ for random k_i in $\mathbb{Z}_q$, $s_i = k_i + e \cdot x_i \mod q$, and r_1 is a random element of G.

1. *Prove completeness of this protocol.*

2. *Assume an honest verifier. Further assume that Peggy does not cheat in two runs of this protocol in which her commitments are the same, the challenges are different, but Victor accepts both runs. Show that Victor can now extract all secret information.*

3. *Why does the argument from the previous item not show the special soundness property?*

4. *Prove zero knowledge of this protocol.*

Solution 42

1. *We need to show that Peggy, knowing x_1 and x_2 and knowing that $y = g_1^{x_1} \cdot g_2^{x_2}$ will make Victor accept if she plays as stated in the protocol. This amounts to showing that $g_1^{s_1} \cdot g_2^{s_2} = y^e \cdot r_1 \cdot r_2$. We compute*

$$\begin{aligned} g_1^{s_1} \cdot g_2^{s_2} &= g_1^{k_1+e\cdot x_1} \cdot g_2^{k_2+e\cdot x_2} \\ &= g_1^{k_1} \cdot (g_1^{x_1})^e \cdot g_2^{k_2} \cdot (g_2^{x_2})^e \\ &= r_1 \cdot (g_1^{x_1} \cdot g_2^{x_2})^e \cdot r_2 \\ &= y^e \cdot r_1 \cdot r_2 \end{aligned}$$

and so Victor will accept this run.

2. *Victor is an honest verifier. Let there be two runs of the protocol that (i) have the same commitment (k_1, k_2), (ii) different challenges $e \neq e'$, that (iii) Victor both verifies successfully, and (iv) where Peggy plays as stated in the protocol (she does not deviate from it). Let these runs be represented by $((k_1, k_2), e, (s_1, s_2))$ and $((k_1, k_2), e', (s'_1, s'_2))$, respectively. Since Peggy won't deviate from the protocol, we know that*

$$\begin{aligned} s_i &= k_i + e \cdot x_i \mod q \\ s'_i &= k_i + e' \cdot x_i \mod q \end{aligned}$$

where we use that $k_i = k'_i$ as her commitments are the same in both runs. From these equations, we get

$$s_i - s'_i = (e - e') \cdot x_i \mod q$$

Since $e \neq e'$, Verifier can then compute x_i from public information as follows:

$$x_i = (s_i - s'_i) \cdot (e - e')^{-1} \mod q$$

3. *This is so because for the special soundness property we cannot assume that Peggy abides by the protocol when considering two accepted runs that have the same commitment but different challenges.*

4. *Consider a random choice of r_1. This looks like a random choice k_1 with $r_1 = g_1^{k_1}$. To pass the verification for given e, s_1, and s_2, there is a unique element of G that, chosen as r_2 will pass this:*

$$r_2 = g_1^{s_1} \cdot g_2^{s_2} \cdot y^{-e} \cdot r_1^{-1}$$

1. Peggy chooses a random v in $\mathbb{Z}_N^*$, computes $y = v^2 \mod N$, and sends y to Victor as her commitment.

2. Victor chooses a random bit b from $\{0, 1\}$ and sends this to Peggy as his challenge.

3. Peggy computes $z = u^b \cdot v \mod N$ where u is a square root of x modulo N, and sends z to Victor as response.

Figure 3: A round in this zero knowledge proof that Peggy knows a quadratic residue modulo N.

> *This means that nobody can distinguish a simulation from a real transcript of this protocol.*

Exercise 43 *Let $N = p \cdot q$ for large primes p and q. Assume that Peggy does not know the factorization of N. But Peggy knows that x is a quadratic residue modulo N, so she knows u with $u^2 = x \mod N$. She wishes to construct a zero knowledge proof to convince Victor that x is a quadratic residue modulo N, without revealing anything about x to Victor.*

The protocol proceeds in $\lfloor \log_2 N \rfloor$ rounds. In each round the steps from Figure 3 are executed.

Victor then verifies that $z^2 = x^b \cdot y \mod N$ and, if so, accepts that round. As always, Victor accepts the proof if he accepts all $\lfloor \log_2 N \rfloor$ rounds.

1. *Show that this protocol is complete.*

2. *Show that this protocol is sound. Specifically, show that Peggy's probability of deceiving Victor in all $\lfloor \log_2 N \rfloor$ rounds is $1/N$.*

3. *Show that the protocol has the zero-knowledge property.*

Solution 43 *1. We assume that Peggy knows the root u or m modulo N and that she plays as stated in the protocol. For the random challenge b, we have to show that $z^2 = x^b \cdot y \mod N$. We do a case analysis:*

- *If $b = 0$, then $y = v^2 \mod N$ and $z = u^0 \cdot v = v \mod N$. So $x^0 \cdot y = y = v^2 = z^2 \mod N$ means that Victor will accept that round.*

- *If $b = 1$, then $y = v^2 \mod N$ and $z = u^1 \cdot v = u \cdot v \mod N$. So $x^1 \cdot y = x \cdot y = u^2 \cdot v^2 = (u \cdot v)^2 = z^2 \mod N$ means that Victor will also accept that round in this case.*

2. *Let x not be a quadratic residue. For every y, we then have that y is a quadratic residue modulo N iff $x \cdot y$ is not a quadratic residue modulo N. So let's consider what Peggy can do now:*

   - *Let us say she picks y to be a quadratic residue. Then she may at best convince Victor if $b = 0$. But then Victor would surely not accept if $b = 1$ as $x^1 \cdot y$ cannot be a quadratic residue.*
   - *Let her pick y to be not a quadratic residue. Then she may at best convince Victor when $b = 1$ but Victor will surely not accept when $b = 0$ as then $x^0 \cdot y$ cannot be a quadratic residue.*

   Therefore, Peggy can convince Victor with probability at most $1/2$ in a round. The probability of her convincing Victor in all $\lfloor \log_2 N \rfloor$ rounds will be $2^{-\log_2 N} = 1/N$. Note that this says that the probability of Peggy to deceive Victor is exponentially small in the size of the problem instance (which is measures in terms of bits).

3. *To show zero knowledge of this protocol, we proceed as follows: Victor can generate a random b and a random z, and then he can define $y = z^2(x^b)^{-1} \mod N$. The triples (y, b, z) produced in this manner have the same probability distribution as those from actual runs of the interactive protocol, assuming that Victor chooses b and z at random. Therefore, we get zero knowledge of the interactive protocol as claimed.*

Exercise 44 *Consider the protocol we discussed that a voter (as Peggy) uses to prove to Victor that she submitted a vote x in $\{-1, 1\}$ but where Victor cannot learn the value of x.*

1. *Show that, if Peggy plays honestly in the protocol, then Victor will accept the protocol run.*

2. *Discuss in what sense this protocol proves that Peggy knows $dlog_g(B_a(x) \cdot h)$ or knows $dlog_g(B_a(x) \cdot h^{-1})$, and why this would demonstrate to Victor that she has committed to some x in $\{-1, 1\}$.*

3. *If Peggy committed to some x not in $\{-1, 1\}$, she will find it hard to respond to Victor's challenge so that Victor will accept the protocol run.*

4. *The protocol reveals no information to any party as to the exact value of Peggy's commitment, except that it came from $\{-1, 1\}$.*

Solution 44 *1. Here we know that Peggy committed to x from $\{-1, 1\}$. We do a case analysis:*

- *Let $x = 1$. Since she plays honestly, we get*

$$\begin{aligned} \alpha_1 &= g^r \cdot (B_a(x) \cdot h)^{-d} && (8) \\ \alpha_2 &= g^w && (9) \\ (e_1, e_2, r_1, r_2) &= (d, d', r, r') && (10) \end{aligned}$$

  where $d' = e - d \mod q$ and $r' = w + a \cdot d' \mod q$. Victor will now accept that protocol run if he can verify the following three equations:

$$\begin{aligned} e &= e_1 + e_1 \mod q && (11) \\ g^{r_1} &= \alpha_1 \cdot (B_a(x) \cdot h)^{e_1} \\ g^{r_2} &= \alpha_2 \cdot (B_a(x) \cdot h^{-1})^{e_2} \end{aligned}$$

  Feeding the definitions of (8) into (11), we see that

$$\begin{aligned} e_1 + e_2 &= d + d' = d + (e - d) = e \mod q \\ g^r \cdot (B_a(x) \cdot h)^{-d} \cdot (B_a(x) \cdot h)^d &= g^r = g^{r_1} \\ g^w \cdot (B_a(x) \cdot h^{-1})^{d'} &= g^{r_2} \end{aligned}$$

  and so Victor will accept that run.

- *Let $x = -1$. Since she plays honestly, we get*

$$\begin{aligned} \alpha_2 &= g^r \cdot (B_a(x) \cdot h^{-1})^{-d} && (12) \\ \alpha_1 &= g^w \\ (e_1, e_2, r_1, r_2) &= (d', d, r', r) \end{aligned}$$

  where $d' = e - d \mod q$ and $r' = w + a \cdot d' \mod q$. Victor will now accept that protocol run if he can verify the three equations *in (11) by feeding the definitions of (12) into (11). As before, we have that*

$$e_1 + e_2 = d' + d = (e - d) + d = e \mod q$$

As for $g^{r_1} = \alpha_1 \cdot (B_a(x) \cdot h)^{d_1}$, note that $r_1 = r'$ and $d_1 = d'$. Therefore we have

$$\begin{array}{rcll} \alpha_1 \cdot (B_a(x) \cdot h)^{d_1} & = & g^w \cdot ((h^x \cdot g^a) \cdot h)^{d'} & \\ & = & g^{w+a\cdot d'} \cdot h^{(x+1)\cdot d'} & \\ & = & g^{w+a\cdot d'} \cdot h^{(-1+1)\cdot d'} & \\ & = & g^{w+a\cdot d'} \cdot h^0 & \\ & = & g^{w+a\cdot d'} & \\ & = & g^{r'} & \textit{as } r' = w + a \cdot d' \mod q \end{array}$$

Finally, for $g^{r_2} = \alpha_2 \cdot (B_a(x) \cdot h^{-1})^{e_2}$ we note that $r_2 = r$, $e_2 = d$, and so we get

$$\begin{array}{rcl} \alpha_2 \cdot (B_a(x) \cdot h^{-1})^{e_2} & = & (g^r \cdot (B_a(x) \cdot h^{-1})^{-d}) \cdot (B_a(x) \cdot h^{-1})^d \\ & = & g^r \end{array}$$

shows this. Therefore, Victor will accept that run as well.

2. *We have $B_a(x) \cdot h = h^x \cdot g^a \cdot h$, which equals g^a if x equals -1. Similarly, we have $B_a(x) \cdot h^{-1} = h^x \cdot g^a \cdot h^{-1}$, which equals g^a is x equals 1. So Victor realizes that, in checking the equations for g^{r_1} and g^{r_2}, he checks a "simulation" for one possible value of x and checks whether x equals -1 (respectively, 1) for the value x that Peggy commits to.*

3. *If x is not in $\{-1, 1\}$, then neither $B_a(x) \cdot h$ nor $B_a(x) \cdot h^{-1}$ are likely to be equal to g^a. So then it seems that Peggy would have to be able to compute the discrete logarithm of h with respect to g in order to compute values that Victor may accept. But we assumed that doing this would be hard for any one.*

4. *The protocol is like an "or" of two Sigma protocols: if $x = -1$, it proves that Peggy knows $dlog_g(B_a(x) \cdot h)$. If x equals 1, it proves that Peggy knows $dlog_g(B_a(x) \cdot h^{-1})$. Since Victor cannot distinguish which of the two cases occur in a run, it follows that he won't know anything about the value of x, bar it being in $\{-1, 1\}$ (if the run is accepted by him).*

Exercise 45 *A Boolean circuit may also contain NOT gates. The garbled tables we defined for gates with two input wires need to be adjusted to work for NOT gates, which have only one input wire.*

1. *Define how to garble the table for a NOT gate with input wire w_i and output wire w_j.*
2. *Illustrate your garbled table for the case when party A uses $\rho_i = 0$ and $\rho_j = 1$.*
3. *What potential problems do you see in using NOT gates in secure multi party computations? Can you suggests how to address some of these problems?*

Solution 45

1. *The input wire will be annotated with $k_i^0 \mid\mid 0 \oplus \rho_i$ and $k_i^1 \mid\mid 1 \oplus \rho_i$. The output wire will be annotated with $k_j^0 \mid\mid 0 \oplus \rho_j$ and $k_j^1 \mid\mid 1 \oplus \rho_j$ as for binary gates. However, when we define the encryptions we only have one argument a instead of a pair a, b. Similarly, we only have one key k_i^a for encryption and so we infer that both k_i^0 and k_i^1 need to be twice as long as the key halves for binary gates. In particular, A needs to be aware of this when generating keys for the entire Boolean circuit.*

   We can now define

$$c_a^j = E_{k_i^{a \oplus \rho_i}}(k_j^{o_a} \mid\mid o_a \oplus \rho_j)$$

   where a is in $\{0, 1\}$ and $o_a = NOT(a \oplus \rho_i)$.

   Note that $k_j^{o_a}$ may be either a key half or an entire key, depending on whether w_j is an input wire for a binary gate or a NOT gate, respectively.

2. *We compute*

$$\begin{aligned}
k_i^0 \mid\mid 0 \oplus \rho_i &= k_i^0 \mid\mid 0 \oplus 0 = k_i^0 \mid\mid 0 \\
k_i^1 \mid\mid 1 \oplus \rho_i &= k_i^0 \mid\mid 1 \oplus 0 = k_i^0 \mid\mid 1 \\
k_j^0 \mid\mid 0 \oplus \rho_j &= k_j^0 \mid\mid 0 \oplus 1 = k_j^0 \mid\mid 1 \\
k_j^1 \mid\mid 1 \oplus \rho_j &= k_j^0 \mid\mid 1 \oplus 1 = k_j^0 \mid\mid 0
\end{aligned}$$

for the ciphertexts c_a^j we first note that

$$\begin{aligned} o_0 &= NOT(0 \oplus \rho_i) = NOT(0 \oplus 0) = NOT(0) = 1 \\ o_1 &= NOT(1 \oplus \rho_i) = NOT(1 \oplus 0) = NOT(1) = 0 \end{aligned}$$

So we get

$$\begin{aligned} c_0^j &= E_{k_i}^{0 \oplus \rho_i}(k_j^{o_0} \mid\mid o_0 \oplus \rho_j) = E_{k_i}^{0}(k_j^{1} \mid\mid 1 \oplus 1) = E_{k_i}^{0}(k_j^{1} \mid\mid 0) \\ c_1^j &= E_{k_i}^{1 \oplus \rho_i}(k_j^{o_1} \mid\mid o_1 \oplus \rho_j) = E_{k_i}^{1}(k_j^{0} \mid\mid 0 \oplus 1) = E_{k_i}^{1}(k_j^{0} \mid\mid 1) \end{aligned}$$

3. *Here are potential issues and how to deal with them:*

   (a) *The first thing to note is that we can no longer use key halves, as there is only one argument. Therefore, the keys for all input wires of NOT gates in a Boolean circuit have to be keys as long as needed for the symmetric cryptosystem. All other keys (for input wires to binary gates such as AND gates) will need to be key halves of the cryptosystem.*

   It is pretty straightforward to deal with this, though, since wires are not shared across different gates.

   (b) *A NOT gate is invertible: if we know its output, we know its input! The binary gates we discussed are not invertible. For example, consider an AND gate. Although an AND gate is invertible on output* 1*, as then both inputs have to be equal to* 1*, an output of* 0 *means that are three possible input scenarios. Say that w_1 and w_2 are the input wires and that w_1 is private to A. Then if w_2 is* 1*, party B will know that w_1 is* 0*. However, if w_2 is* 0*, then party B knows nothing about the value of w_1. In summary, AND gates can have information flow from outputs to inputs, but this flow does not always completely determine inputs.*

   We may address this as follows: note that we already have AND, OR, NOR, and XOR gates at our disposal. So we just need to be able to express a NOT gates through the latter. Then we won't need NOT gates per se. Observe that we may define $NOT(x)$ as $NOR(0, x)$ where the NOR gate has 0 *as constant first input. This constant input* 0 *may be problematic, but in an arbitrary Boolean circuit, an attacker may not know which input wires that do into NOR gates model NOT gates, and which ones don't.*

Another interesting solution is to use a NAND gate and to provide the input for NOT in both input wires for the NAND gate. Please convince yourself that this has the right semantics of NOT. Moreover, each input wire will have an independent ρ_i and so this will shield against knowing that the two input wires have the same actual values.

Exercise 46 *The protocol for secure two-party computation with garbled Boolean circuits uses oblivious transfer protocols so that party B learns key halves for each of her input wires. We also saw that this protocol always puts party B into the possession of two key halves (and no more) for each garbled table in the circuit.*

1. *Discuss why it may be a problem if party B does not know which of the four cipher-texts these two key halves correspond to.*

2. *Explain why the protocol allows party B to uniquely identify the correct ciphertext amongst the four candidates for the next decryption in the secure two-party computation.*

Solution 46

1. *We could of course try the key consisting of the two halves on all four cipher-texts. The problem with that is that this will decrypt something that is a random key half, followed by a random encrypted bit. So party B would have no way of knowing that she decrypted the right one of the four cipertexts.*

2. *We have $c^m_{z_i,z_j} = E_{k_\cdot^{e(z_i)},k_\cdot^{e(z_j)}}(\cdots)$ where $e(z_k) = z_k \oplus \rho_k$. So if we apply e again, we get*

   $$c^m_{e(z_i),e(z_j)} = E_{k_\cdot^{e(e(z_i))},k_\cdot^{e(e(z_j))}}(\cdots) = E_{k_\cdot^{z_i},k_\cdot^{z_j}}(\cdots)$$

   since $\oplus$ satisfies $a \oplus x \oplus x = a$. Therefore, party B gets the right pair x, y of cipher-text by pairing up the values she finds after both $||$ for these key halves, as those values are defined to be $e(e(z_i)$ and $e(e(z_j))$.

   For example, in Figure 3 of our textbook, party B learns $k_1^0 \mid\mid 0$ and $k_3^1 \mid\mid 1$ and so looks up $c^3_{1,1}$ as the two values after $||$ are 1 and 1 in that order.

Exercise 47 *Consider the Boolean function*

$$f(\{w_1, w_4\}, \{w_2, w_3\}) = (w_1 \wedge w_2) \oplus \neg(w_3 \vee w_4) \tag{13}$$

where $\{w_1, w_4\}$ is the private input of party A and $\{w_2, w_3\}$ is the private input of party B.

1. *Draw a Boolean circuit that implements function f above and only uses gates for AND, NOR, and XOR. Make sure that all wires have names.*
2. *Suppose that party A generates random keys $\rho_1 = \rho_2 = \rho_5 = 0$ and $\rho_3 = \rho_4 = \rho_6 = \rho_7 = 1$ and that the input values v_i for input wires w_i are $v_1 = v_4 = 0$ and $v_2 = v_3 = 1$.*

   (a) *Write down the corresponding garbled circuit in the style seen in Figure 22.2 on page 443 of our textbook.*

   (b) *Show the progress of the secure Two-Party Computation, in the style seen in Figure 22.3 on page 444 of our textbook.*

Solution 47 *1. The circuit can be seen in Figure 4. Bear in mind that you may have named the wires differently, or in different order.*

2. The information flow for the garbled Boolean circuit for the above input values is shown in Figure 5. This does not show the ciphertexts, these

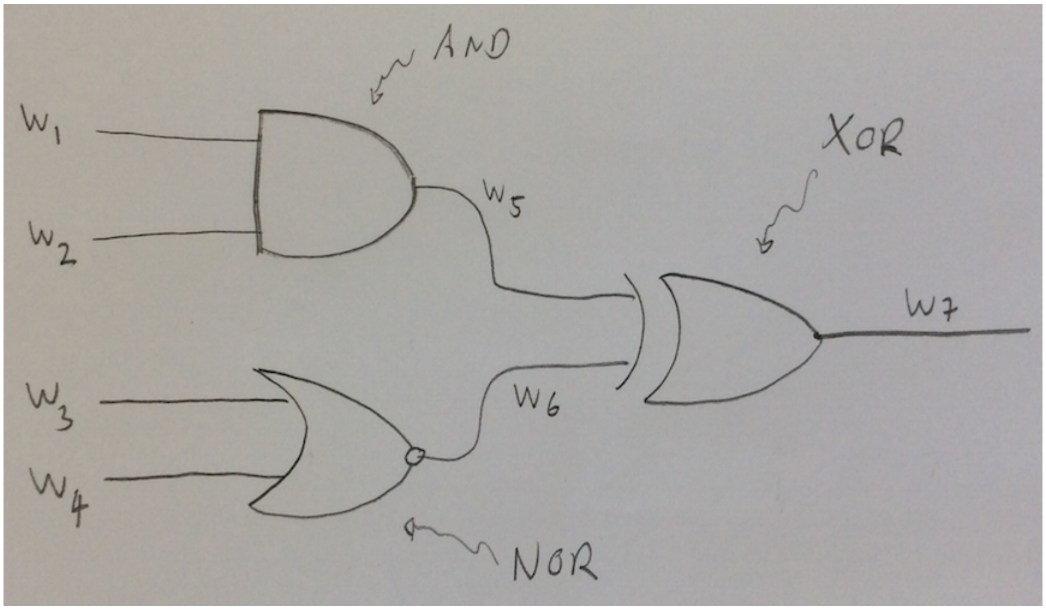


Figure 4: Boolean circuit for the function in (13).

are listed separately here:

$$\begin{aligned}
c^5_{0,0} &= E_{k_1^0,k_2^0}(k_5^0 \mid\mid 0)\\
c^5_{0,1} &= E_{k_1^0,k_2^1}(k_5^0 \mid\mid 0)\\
c^5_{1,0} &= E_{k_1^1,k_2^0}(k_5^0 \mid\mid 0)\\
c^5_{1,1} &= E_{k_1^1,k_2^1}(k_5^1 \mid\mid 1)
\end{aligned}$$

$$\begin{aligned}
c^6_{0,0} &= E_{k_3^1,k_4^1}(k_6^0 \mid\mid 1)\\
c^6_{0,1} &= E_{k_3^1,k_4^0}(k_6^0 \mid\mid 1)\\
c^6_{1,0} &= E_{k_3^0,k_4^1}(k_6^0 \mid\mid 1)\\
c^6_{1,1} &= E_{k_3^0,k_4^0}(k_6^1 \mid\mid 0)
\end{aligned}$$

$$\begin{aligned}
c^7_{0,0} &= E_{k_5^0,k_6^1}(k_7^1 \mid\mid 0)\\
c^7_{0,1} &= E_{k_5^0,k_6^0}(k_7^0 \mid\mid 1)\\
c^7_{1,0} &= E_{k_5^1,k_6^1}(k_7^0 \mid\mid 1)\\
c^7_{1,1} &= E_{k_5^1,k_6^0}(k_7^1 \mid\mid 0)
\end{aligned}$$

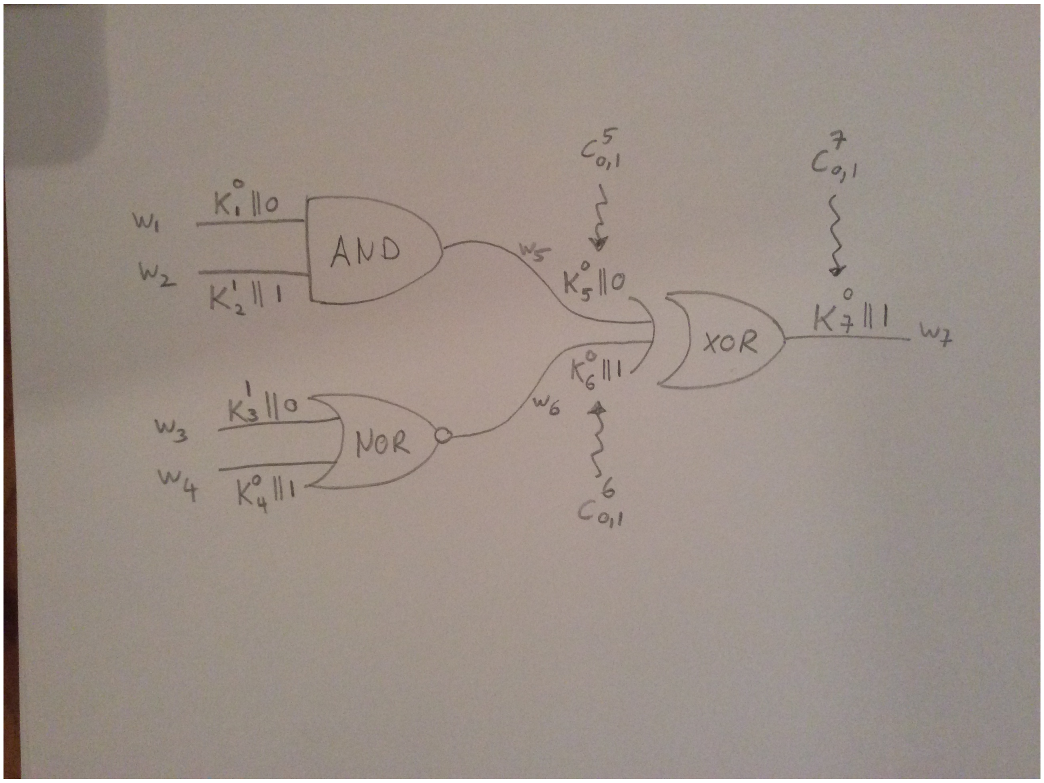


Figure 5: Garbled Boolean circuit showing the flow of information for the computation of the output for the function in (13) and the private inputs specified above

Recall the definition of these ciphertexts:

$$c^{w_{i_2}}_{a,b} = E_{K^{a\oplus\rho_{i_0}}_{w_{i_0}}, K^{b\oplus\rho_{i_1}}_{w_{i_1}}}(K^{o_{a,b}}_{w_{i_2}} \mid\mid o_{a,b} \oplus \rho_{i_2})$$

We will show the details for how to compute each of these 12 ciphertext forms above:

(a) *Ciphertext forms for* c^5*:*

   i. *Let* $a = 0$ *and* $b = 0$*. Then* $a \oplus \rho_1 = 0 \oplus 0 = 0$*,* $b \oplus \rho_2 = 0 \oplus 0 = 0$*,* $o_{a,b} = 0\, AND\, 0 = 0$*, and* $o_{a,b} \oplus \rho_5 = 0 \oplus 0 = 0$*.*

    ii. Let $a = 0$ and $b = 1$. Then $a \oplus \rho_1 = 0 \oplus 0 = 0$, $b \oplus \rho_2 = 1 \oplus 0 = 1$, $o_{a,b} = 0\, AND\, 1 = 0$, and $o_{a,b} \oplus \rho_5 = 0 \oplus 0 = 0$.
    iii. Let $a = 1$ and $b = 0$. Then $a \oplus \rho_1 = 1 \oplus 0 = 1$, $b \oplus \rho_2 = 0 \oplus 0 = 0$, $o_{a,b} = 1\, AND\, 0 = 0$, and $o_{a,b} \oplus \rho_5 = 0 \oplus 0 = 0$.
    iv. Let $a = 1$ and $b = 1$. Then $a \oplus \rho_1 = 1 \oplus 0 = 1$, $b \oplus \rho_2 = 1 \oplus 0 = 1$, $o_{a,b} = 1\, AND\, 1 = 1$, and $o_{a,b} \oplus \rho_5 = 1 \oplus 0 = 1$.

  (b) Ciphertext forms for c^6:

    i. Let $a = 0$ and $b = 0$. Then $a \oplus \rho_3 = 0 \oplus 1 = 1$, $b \oplus \rho_4 = 0 \oplus 1 = 1$, $o_{a,b} = 1\, NOR\, 1 = 0$, and $o_{a,b} \oplus \rho_6 = 0 \oplus 1 = 1$.
    ii. Let $a = 0$ and $b = 1$. Then $a \oplus \rho_3 = 0 \oplus 1 = 1$, $b \oplus \rho_4 = 1 \oplus 1 = 0$, $o_{a,b} = 1\, NOR\, 0 = 0$, and $o_{a,b} \oplus \rho_6 = 0 \oplus 1 = 1$.
    iii. Let $a = 1$ and $b = 0$. Then $a \oplus \rho_3 = 1 \oplus 1 = 0$, $b \oplus \rho_4 = 0 \oplus 1 = 1$, $o_{a,b} = 0\, NOR\, 1 = 0$, and $o_{a,b} \oplus \rho_6 = 0 \oplus 1 = 1$.
    iv. Let $a = 1$ and $b = 1$. Then $a \oplus \rho_3 = 1 \oplus 1 = 0$, $b \oplus \rho_4 = 1 \oplus 1 = 0$, $o_{a,b} = 0\, NOR\, 0 = 1$, and $o_{a,b} \oplus \rho_6 = 1 \oplus 1 = 0$.

  (c) Ciphertext forms for c^7:

    i. Let $a = 0$ and $b = 0$. Then $a \oplus \rho_5 = 0 \oplus 0 = 0$, $b \oplus \rho_6 = 0 \oplus 1 = 1$, $o_{a,b} = 0\, XOR\, 1 = 1$, and $o_{a,b} \oplus \rho_7 = 1 \oplus 1 = 0$.
    ii. Let $a = 0$ and $b = 1$. Then $a \oplus \rho_5 = 0 \oplus 0 = 0$, $b \oplus \rho_6 = 1 \oplus 1 = 0$, $o_{a,b} = 0\, XOR\, 0 = 0$, and $o_{a,b} \oplus \rho_7 = 0 \oplus 1 = 1$.
    iii. Let $a = 1$ and $b = 0$. Then $a \oplus \rho_5 = 1 \oplus 0 = 1$, $b \oplus \rho_6 = 0 \oplus 1 = 1$, $o_{a,b} = 1\, XOR\, 1 = 0$, and $o_{a,b} \oplus \rho_7 = 0 \oplus 1 = 1$.
    iv. Let $a = 1$ and $b = 1$. Then $a \oplus \rho_5 = 1 \oplus 0 = 1$, $b \oplus \rho_6 = 1 \oplus 1 = 0$, $o_{a,b} = 1\, XOR\, 0 = 1$, and $o_{a,b} \oplus \rho_7 = 1 \oplus 1 = 0$.

Please note that party B, at this stage only knows the values of these 12 ciphertexts.

3. For the computation, we have $v_1 = 0$ and $v_4 = 0$. Their respective external values are $v_1 \oplus \rho_1 = 0 \oplus 0 = 0$ and $v_4 \oplus \rho_4 = 0 \oplus 1 = 1$. Party A gives to party B data of form $k_i^{v_i} \mid\mid v_i \oplus \rho_i$ for party A's input wire w_i. Therefore, party A gives to party B the information

$$k_1^0 \mid\mid 0 \text{ and } k_4^0 \mid\mid 1$$

Party B cannot learn the values v_1 and v_4 as they are encrypted with ρ_1 and ρ_4 respectively; and because the key halves k_1^0 and k_4^0 are just random strings to B.

Next, both parties run an oblivious transfer protocol for each of the input wires of party B:

(a) *Consider the input wire w_2. We have $v_2 = 1$. So this is an OT protocol in which A has two messages $k_2^0 \mid\mid 0$ and $k_2^1 \mid\mid 1$ and party B has bit $b = 1$. So at the end of the OT protocol, party B will have learned $k_2^1 \mid\mid 1$.*

(b) *For the input wire w_3, we have $v_3 = 1$. So this is an OT protocol in which A has two messages $k_3^0 \mid\mid 1$ and $k_3^1 \mid\mid 0$ and party B has bit $b = 1$. So at the end of the OT protocol, party B will have learned $k_3^1 \mid\mid 0$.*

So now party B knows $k_1^0 \mid\mid 0$ and $k_2^1 \mid\mid 1$ and so can decrypt $c_{0,1}^5$ to obtain $k_5^0 \mid\mid 0$. Similarly, party B knows $k_3^1 \mid\mid 0$ and $k_4^0 \mid\mid 1$ and so can decrypt $c_{0,1}^6$ to obtain $k_6^0 \mid\mid 1$.

Next, party B can use k_5^0 and k_6^0 to decrypt $c_{0,1}^7$ to $k_7^0 \mid\mid 1$. So the encrypted output of f is the 1 in $k_7^0 \mid\mid 1$. Since party B knows ρ_7 (it was sent by A as such), she can compute $1 \oplus \rho_7 = 1 \oplus 1 = 0$ as the actual output of f on the given input values $v_1, \ldots, v_4$ above. To see that this is correct, we compute

$$\begin{aligned} f(v_1, v_2, v_3, v_4) &= (0 \wedge 1) \oplus \neg(1 \vee 0) \\ &= 0 \oplus \neg 1 \\ &= 0 \oplus 0 \\ &= 0 \end{aligned}$$

Exercise 48 *Consider a finite field $\mathbb{F}_q$ where $q = p^k$ for some prime $p \geq 2$ is prime. The characteristic of $\mathbb{F}_p$ is the minimal n such that $\sum_{i=1}^{n} 1 = 0$. One can show that the characteristic of $\mathbb{F}_q$ is p.*

Now suppose that $p > 2$. Show that all Boolean Circuits can be faithfully encoded as polynomials over $\mathbb{F}_q$:

1. *First show that all Boolean operators can be represented using only constants 0, 1, the exclusive-or $\oplus$, and the AND operator denoted here as $\odot$.*

2. *Show how one can encode $\oplus$ faithfully as a polynomial in two variables over $\mathbb{F}_q$.*

3. *Show how one can encode $\odot$ faithfully as a polynomial in two variables over $\mathbb{F}_q$.*

Solution 48 *1. It is known that all Boolean functions can be expressed with the NAND operator alone:*

$$\begin{array}{rcl} \neg x & = & x\ NAND\ x \\ x \wedge y & = & \neg(x\ NAND\ y) \\ x \vee y & = & (x\ NAND\ y)\ NAND\ (x\ NAND\ y) \end{array}$$

This shows the claim as it is known that $\neg$, $\wedge$, and $\vee$ can generate all Boolean formulas. Therefore, it suffices to show how we can express NAND from $\oplus$, $\odot$, and constants. Since $\odot$ is AND, it remains to show how to express NOT. But we have that $\neg x$ is equivalent to $1 \oplus x$ which uses the constant 1 and $\oplus$.

2. *The field has constants 0 and 1, which we identify with the corresponding Boolean values 0 and 1. We encode $\oplus$ by*

$$x \oplus y = -2 \cdot x \cdot y + x + y$$

Let x and y be from $\{0, 1\}$. If $x = y = 0$, then $x \oplus y$ evaluates to 0 as well. Similarly, if exactly one of x and y is 1 and the other one is 0, then the term $-2 \cdot x \cdot y$ evaluates to 0 and $x + 1$ evaluates to 1, so the result 1 is correct. Finally, if $x = y = 1$, then $x \oplus y$ evaluates to $-2 \cdot 1 \cdot 1 + 1 + 1$. Since the characteristic of the field is larger than 2, the term $1 + 1$ is not 0 and so $x \oplus y$ evaluates to 0 as desired.

3. *We define $x \odot y$ as $x \cdot y$, the multiplication in the field. This works since 1 is a unit for multiplication $\cdot$ and since $0 \cdot x = x \cdot 0$ in a field.*

Exercise 49 *Consider the elliptic curve*

$$E\ :\ Y^2 = X^3 + X + 3$$

and the finite field $K = \mathbb{F}_7$.

1. *Show that* $(4, 1)$ *is in* $E(\mathbb{F}_7)$.

2. *Show that* $(4, 6)$ *is in* $E(\mathbb{F}_7)$.

3. *Compute* $(4, 1) + (4, 6)$.

4. *Compute* $(4, 1) + (6, 6)$.

5. *Compute* $4 \star (4, 1)$.

Solution 49 1. *We use* $\equiv$ *for equations modulo 7 in these solutions. Then* $Y^2 = 1^2 \equiv 1$ *and* $X^3+X+3 \equiv 4^3+4+3 \equiv 4^3 \equiv 4\cdot 16 \equiv 4\cdot 2 \equiv 1$ *and so* $(4, 1)$ *is a point in* $E(\mathbb{F}_7)$.

2. *We have* $Y^2 = 6^2 \equiv 1$ *and* $X^3+X+3 \equiv 4^3+4+3 \equiv 4^3 \equiv 4\cdot 16 \equiv 4\cdot 2 \equiv 1$ *and so* $(4, 6)$ *is a point in* $E(\mathbb{F}_7)$.

3. *To compute* $(4, 1) + (4, 6)$, *note that* $P_2 = (4, 6) = -P_1 = (4, -1)$ *since* $-1 \equiv 6$ *modulo 7. By definition of the chord-tangent process, we have that* $(4, 1) + (4, 6) = \mathcal{O}$.

4. *To compute* $(4, 1) + (6, 6)$, *we have that* $x_1 = 4 \not\equiv x_2 = 6$. *Therefore, we are in the case when* $x_1 \neq x_2$ *in Smart's textbook. We first compute* λ *and* μ *for that case:*

$$\begin{aligned}
\lambda &= \frac{y_2 - y_1}{x_2 - x_1} \\
&\equiv \frac{6-1}{6-4} \\
&\equiv \frac{5}{2} \\
&\equiv 5 \cdot 4 \\
&\equiv 20 \\
&\equiv 6
\end{aligned}$$

$$\begin{aligned}
\mu &= \frac{y_1 \cdot x_2 - y_2 \cdot x_1}{x_2 - x_1} \\
&\equiv \frac{1 \cdot 6 - 6 \cdot 4}{6 - 4} \\
&\equiv \frac{6 - 24}{2} \\
&\equiv -18 \cdot 4 \\
&\equiv 3 \cdot 4 \\
&\equiv 12 \\
&\equiv 5
\end{aligned}$$

But then $(4,1) + (6,6) = (x_3, y_3)$ *where*

$$\begin{aligned}
x_3 &= \lambda^2 - x_1 - x_2 \\
&\equiv 6^2 - 4 - 6 \\
&\equiv 1 - 10 \\
&\equiv 5
\end{aligned}$$

$$\begin{aligned}
y_3 &= -\lambda \cdot x_3 - \mu \\
&\equiv -6 \cdot x_3 - 5 \\
&\equiv -6 \cdot 5 - 5 \\
&\equiv -35 \\
&\equiv 0
\end{aligned}$$

Therefore, $(4,1) + (6,6) = (5,0)$.

5. *To compute* $4 \star (4,1)$, *it is of advantage to compute first* $2 \star (4,1)$, *and then* $2 \star (2 \star (4,1))$. *This general technique is the subject of the next exercise. Let us compute* $(4,1) + (4,1)$ *therefore first.*

   Let $(4,1) + (4,1) = (x_3, y_3)$. *Since both arguments of* $+$ *are the same, we are in a case in which* $x_1 = x_2$ *and* $P_2 \neq -P_1$ *in the notation of our slides and Smart's textbook. This means that we have*

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ &\equiv \lambda^2 - 4 - 4 \\ &\equiv 0^2 - 8 \\ &\equiv 6 \end{aligned}$$

since $\lambda \equiv 0$. *The latter is seen by*

$$\begin{aligned} \lambda &\equiv \frac{3 \cdot 4^2 + 1}{2 \cdot 1} \\ &\equiv (3 \cdot 16 + 1) \cdot 2^{-1} \\ &\equiv (3 \cdot 2 + 1) \cdot 2^{-1} \\ &\equiv 0 \cdot 2^{-1} \\ &\equiv 0 \end{aligned}$$

Moving on to computing y_3 *we get*

$$\begin{aligned} y_3 &= -\lambda \cdot x_3 - \mu \\ &\equiv -0 \cdot x_3 - \mu \\ &\equiv -\mu \\ &\equiv -\frac{-4^3 + 1 \cdot 4 + 2 \cdot 3}{2 \cdot 1} \\ &\equiv -\frac{-8 + 4 + 6}{2} \\ &\equiv -\frac{2}{2} \\ &\equiv -1 \\ &\equiv 6 \end{aligned}$$

Therefore, we have $(4,1) + (4,1) = (6,6)$. *To compute* $4 \star (4,1)$, *we may therfore compute instead* $(6,6) + (6,6)$. *Since both arguments are the same, we are also in the same definitional case of* $+$. *Now* (x_3, y_3) *refers to the result of* $(6,6) + (6,6)$, *i.e. it refers to* $4 \star (4,1)$.

We first compute λ and μ:

$$\begin{aligned}
\lambda &= \frac{3x_1^2+1}{2y_1} \\
&\equiv \frac{3\cdot 6^2+1}{2\cdot 6} \\
&\equiv \frac{3+1}{12} \\
&\equiv \frac{4}{5} \\
&\equiv 4\cdot 5^{-1} \\
&\equiv 4\cdot 3 \\
&\equiv 5
\end{aligned}$$

$$\begin{aligned}
\mu &= \frac{-x_1^3+1\cdot x_1+2\cdot 3}{2\cdot 6} \\
&\equiv \frac{-6^3+1\cdot 6+6}{12} \\
&\equiv \frac{-6+6+6}{5} \\
&\equiv \frac{6}{5} \\
&\equiv 6\cdot 5^{-1} \\
&\equiv 6\cdot 3 \\
&\equiv 4
\end{aligned}$$

Now we compute

$$\begin{aligned}
x_3 &\equiv \lambda^2-x_1-x_2 \\
&\equiv 5^2-6-6 \\
&\equiv 4-12 \\
&\equiv -8 \\
&\equiv 6
\end{aligned}$$

$$\begin{aligned} y_3 &\equiv -\lambda \cdot x_3 - \mu \\ &\equiv -5 \cdot 6 - 4 \\ &\equiv -30 - 4 \\ &\equiv -34 \\ &\equiv 1 \end{aligned}$$

Therefore, $4 \star (4,1)$ *equals* $(x_3, y_3) = (6,1)$.

Exercise 50 *Let* K *be a finite field and* E *an elliptic curve in the short Weierstrass form*

$$E \,:\, Y^2 = X^3 + aX + b$$

for constants a *and* b*. Let* $+$ *denote the point addition defined through the chord-tangent process. Assume that there is an implementation of this addition operation* $+$ *and of* $\lambda P \in E(K)\colon 2 \star P$*, that is of* $\lambda P \in E(K)\colon P + P$*. Let* k *be an* l*-bit natural number* > 2 *with binary representation*

$$k = \sum_{j=0}^{l-1} k_j \cdot 2^l \qquad\qquad (k_j \in \{0,1\})$$

Write an algorithm that takes such a vector $(k_j)_{j=0}^{l-1}$ *and a point* P *in* $E(K)$ *as input, outputs* $k \star P$*, and has running time logarithmic in* k*.*

Solution 50 *This uses a technique that goes by various names, one of them being* iterative squaring*:*

```
$Q = \mathcal{O}$;
for ($j = l - 1$ to $0$ by $-1$) {
    $Q = 2 \star Q$;
    if ($k_j == 1$) { $Q = Q + P$; }
}
return $Q$;
```

A real concern in an implementation is the if-statement: it performs an additional point addition whenever the guard is true. Such iterations *may therefore take longer (recall that point additions may involve expensive operations such). But anyone who can distinguish which iterations execute the if-branch and which ones do not, can recover all k_j and therefore k as well. This would be fatal when invoked with a private key k.*

Exercise 51 *Consider an elliptic curve*

$$E \,:\, Y^2Z = X^3 + XZ^2 + 3Z^3$$

in the projective plane, and a short Weierstrass form eliptive curve

$$E' \,:\, Y^2 = X^3 + X + 3$$

1. *Let K be the finite field $\mathbb{F}_7$.*

   (a) Show that $(1,5,2)$ is in $E(\mathbb{F}_7)$.

   (b) Show that $(1/2, 5/2)$ is in $E'(\mathbb{F}_7)$.

2. *Show that for all fields K and for all $(X,Y,Z) \in \mathbb{P}^2(K)$ with $Z \neq 0$ we have that $(X,Y,Z) \in E(K)$ implies that $(X/Z, Y/Z) \in E'(K)$. That is to say, we can transform points on $E(K)$ to points on $E'(K)$.*

Solution 51 *1. We compute this as follows:*

(a) We have $Z = 2 \neq 0$ and so $(1,5,2)$ is in $\mathbb{P}^2(\mathbb{F}_7)$. We have $Y^2Z \equiv 5^2 \cdot 2 \equiv 4 \cdot 2 \equiv 1$. And $X^3 + XZ^2 + 3Z^3 \equiv 1^3 + 1 \cdot 2^2 + 3 \cdot 2^3 \equiv 1 + 4 + 3 \cdot 8 \equiv 5 + 3 \cdot 1 \equiv 1$. So this is indeed a point in $E(\mathbb{F}_7)$.

(b) We have $2^{-1} = 4$ modulo 7 and so $(1/2, 5/2) \equiv (1 \cdot 4, 5 \cdot 4) \equiv (4, 20) \equiv (4, 6)$. Now $Y^2 \equiv 6^2 \equiv 1$ and $X^3 + X + 3 \equiv 4^3 + 4 + 3 \equiv 4^3 \equiv 4 \cdot 16 \equiv 4 \cdot 2 \equiv 1$. And so $(1/2, 5/2)$ is in $E'(\mathbb{F}_7)$.

2. *First of all, the argument appeals to equations that follow from the axioms of fields, and so the proof is valid for all chosen fields K.*

   We have

$$\begin{aligned}(Y/Z)^2 &\equiv (Y\cdot Z^{-1})^2 \equiv Y^2\cdot Z^{-2} \equiv (Y^2\cdot Z)\cdot Z^{-3}\\ &\equiv (X^3+XZ^2+3Z^3)\cdot Z^{-3} \qquad \textit{(since } (X,Y,Z)\in E(K)\textit{)}\\ &\equiv (X/Z)^3+(X/Z)+3\end{aligned}$$

which shows that $(X/Z, Y/Z)$ is in $E'(K)$. Note that this is well defined since $Z \neq 0$ and so Z^{-1} exists in the field K.

Exercise 52 *Recall the Replicated Secret Sharing Scheme.*

1. *Find an example of a monotone access structure Γ over a set of parties $\mathcal{P}$ such that*

   $$\mathcal{B} = \{\mathcal{P}\setminus\mathcal{O} \mid \mathcal{O} \text{ maximally non-qualifying}\}$$

   is disjoint from $m(\Gamma)$, the set of minimal elements of Γ with respect to subset inclusion.

2. *Find an example for $\mathcal{B}\cap m(\Gamma) = \emptyset$ in which $\mathcal{B}$ equals the set of maximally non-qualifying sets.*

3. *Find an example for $\mathcal{B}\cap m(\Gamma) = \emptyset$ in which $\mathcal{B}$ has less elements than $m(\Gamma)$.*

Solution 52 *We give an example that answers all threee parts of this exercise in one:*

Consider $\mathcal{P} = \{A, B, C, D\}$ and $\Gamma = \{\{A, B\}, \{A, C\}, \{A, D\}\}$. This Γ indirectly specifies two necessary and sufficient conditions for a set of parties to be able to recover the secret:

1. *Party A has to be in that set.*

2. *Some party other than A has to be in that set.*

From this alternative characterisation of Γ it is pretty easy to read off the maximally non-qualifying sets: $\mathcal{O}_1 = \{A\}$ and $\mathcal{O}_2 = \{B, C, D\}$. Both are clearly non-qualifying.

By item 2. above, $\mathcal{O}_1$ is maximally non-qualifying. By item 1. above, $\mathcal{O}_2$ is maximally non-qualifying. But the union of $\mathcal{O}_1$ and $\mathcal{O}_2$ is $\mathcal{P}$ and so there are no other maximally non-qualifying sets. Therefore, $B_1 = \mathcal{P} \setminus \{A\} = \{B, C, D\}$ and $B_2 = \mathcal{P} \setminus \{B, C, D\} = \{A\}$ and so

$$\mathcal{B} = \{\{A\}, \{B, C, D\}\}$$

In particular, $\mathcal{B}$ equals the set of all maximally non-qualifying sets.

(Let us look at the shares as well, although this was not part of the question: We generate shares s_1 and s_2 for a secret s such that $s = s_1 \oplus s_2$. The share s_A for party A is s_1 as A is only in set B_1 not in B_2. The shares s_B, s_C, and s_D are all equal to s_2 since these parties are only in B_2 not in B_1.)

Clearly, B is different from $m(\Gamma)$, which here equals Γ. In fact, B and $m(\Gamma)$ are disjoint and the former has 2 elements whereas the latter has three elements.

Exercise 53 *Let $p = 2q + 1$ be a safe prime, and so q is prime as well. Let $g \in \mathbb{Z}_p^*$ be an element such that $g \neq 1 \mod p$ but where g is a quadratic residue modulo p.*

Show that the subgroup generated by g in the multiplicative group $\mathbb{Z}_p^*$ has order (i.e. size) q.

Solution 53 *By Lagrange's Theorem, we know that for each subgroup of $\mathbb{Z}_p^*$ its order must be a divisor of the order of $\mathbb{Z}_p^*$, which is $p - 1$. In our setting, $p - 1 = (2q + 1) - 1 = 2q$. The only divisors of $2q$ are 1, 2, q, and $2q$ since 2 and q are prime.*

We know that g is a quadratic residue modulo p. This implies that all elements in the subgroup G, generated by g, are quadratic residues modulo p:

- *$1 = 1^2 \mod p$ and so the unit 1 of the subgroup G is a quadratic residue modulo p*
- *let h be in G and a quadratic residue modulo p; then $h = r^2 \mod p$ for some r. But then $h^{-1} = (r^2)^{-1} = (r^{-1})^2 \mod p$ shows that h^{-1} is also a quadratic residue modulo p*

- *let h_1 and h_2 be in G and quadratic residues modulo p; then there are r_1 and r_2 with $h_i = r_i^2 \mod p$ for $i = 1, 2$. But then $h_1 \cdot h_2 = (r_1 \cdot r_2)^2 \mod p$ shows that $h_1 \cdot h_2$ is a quadratic residue as well*

Since G is generated by g and the group operations, the above shows that all *elements of G are quadratic residues. But this rules out that the size of G is $2q$: not all elements in Z_p^* are quadratic residues. In fact, we have*

$$r^2 = (p - r)^2 \mod p$$

and so the list of all quadratic residues is symmetric around $p/2$. In fact, for $p > 2$ we can say that there are exactly $(p+1)/2$ quadratic residues modulo p, which in our setting is $q + 1$. Now 0 is one of these quadratic residues modulo p; however, we exclude 0 since we operate in $\mathbb{Z}_p^*$ here, and so we have q quadratic residues in Z_p^*. This concludes our proof, provided *we can show that G has neither size 1 nor size 2.*

But G clearly does not have size 1 since g is different from 1 and G contains the unit 1 as a subgroup.

To see that G does not have size 2, we note that in a cyclic group such as $\mathbb{Z}_p^*$ there is a unique subgroup of a given order if there is a subgroup of that order at all. By Lagrange's Theorem, we know that there is a subgroup of $\mathbb{Z}_p^*$ of order 2. So what is that unique subgroup? It turns out to be $\{1, p-1\}$. Note that for this claim it suffices to show that $p - 1$ is its own multiplicative inverse in $\mathbb{Z}_p^*$, i.e. that

$$(p - 1) \cdot (p - 1) \equiv 1 \mod p$$

But $(p-1) \cdot (p-1)$ is $p^2 - 2 \cdot p + 1 \equiv 1 \mod p$ and so this is true. Since $g \neq 1$, we therefore will have shown that the order of G is larger than 2 if we can rule out that $g \equiv p - 1 \mod p$. We use **Proof by Contradiction:** *Assume that $g \equiv f^2 \mod p$ is the case. Then $g \equiv p - 1 \mod p$ would imply that*

$$g^2 \equiv f^4 = (p - 1) \cdot (p - 1) = 1 \mod p$$

But then $f^4 \equiv 1 \mod p$ implies that the order of the group generated by f divides 4. The divisors of 4 are 1, 2, and 4. We already ruled out 1 since $g = f^2 \neq 1$. And it cannot be 4 since 4 is not a divisor of $p - 1 = 2 \cdot q$ and so there is no subgroup of $\mathbb{Z}_p^*$ of order 4. Therefore, the order of (the subgroup *generated by) f has to equal 2. But then f^2, which equals g, has to equal 1, which is a contradiction.*

Exercise 54 (Q1 of CO409 – now COMP 70009 – Exam in 2016) Shannon's notion of perfect secrecy:

Let $\mathbb{P} = \{a, b, c, d, e\}$, $\mathbb{C} = \{1, 2, 3, 4, 5\}$, and $\mathbb{K} = \{k_1, k_2, k_3, k_4, k_5\}$. The probability distribution $p(P = p)$ for plain-texts p is given by

	a	b	c	d	e
$p(P = p)$	0.31	0.15	0.21	0.06	0.27

Let $p(K = k_i) = 0.2$ for all $1 \leq i \leq 5$. Finally, the encryption $e_k(m)$ for this cryptosystem is given in the table

	a	b	c	d	e
k_1	4	2	1	3	5
k_2	5	1	4	2	3
k_3	1	3	2	5	4
k_4	2	4	5	1	3
k_5	2	5	3	4	1

1. *Compute $p(C = 3)$ and $p(C = 5)$.*

2. *Compute $p(C = 3 \mid P = a)$ and $p(P = c \mid C = 3)$.*

3. *Explain why the above cryptosystem is not perfectly secure in the sense of Shannon.*

4. *The table for $e_k(m)$ above has 25 entries from $\mathbb{C}$. Without changing $\mathbb{P}$, $\mathbb{C}$, and $\mathbb{K}$, make a* minimal *number of changes to these 25 entries so that the resulting cryptosystem* is *perfectly secure in the sense of Shannon.*

5. *Suppose you need to engineer a perfectly secure cryptosystem for* GlobalCorp*, for communications between its headquarter in Singapore and its British office.*

   Briefly discuss what cryptosystem you would choose, and sketch what security and implementation issues such a choice would entail.

Solution 54 1. *We have* $p(C = 3) = \sum_{k \in \mathbb{K}} p(K = k) \cdot p(P = d_k(3))$*. Since all* $p(K = k)$ *equal* 0.2 *here, this computes to* $0.2 \cdot p(P = d) + 0.2 \cdot p(P = e) + 0.2 \cdot p(P = b) + 0.2 \cdot p(P = e) + 0.2 \cdot p(P = c) = 0.2 \cdot (0.06 + 0.27 + 0.15 + 0.27 + 0.21) = 0.2 \cdot 0.96 = 0.192$*.*

Similarly, we get $p(C = 5)$ *as* $\sum_{k \in \mathbb{K}} p(K = k) \cdot p(P = d_k(5)) = 0.2 \cdot p(P = e) + 0.2 \cdot p(P = a) + 0.2 \cdot p(P = d) + 0.2 \cdot p(P = c) + 0.2 \cdot p(P = b) = 0.2 \cdot (0.27 + 0.31 + 0.06 + 0.21 + 0.15) = 0.2 \cdot 1 = 0.2$*.*

2. *We have* $p(C = 3 \mid P = a) = \sum_{k \mid a = d_k(3)} p(K = k) = 0$ *as no key encrypts* a *to* 3*.*

   We compute $p(P = c \mid C = 3)$ *using Bayes' Theorem. We first have to compute* $p(C = 3 \mid P = c) = \sum_{k \mid c = d_k(3)} p(K = k) = p(K = k_5) = 0.2$*. Then we use* $p(C = 3) = 0.192$ *from above to compute*

$$p(P = c \mid C = 3) = \frac{p(P = c) \cdot p(C = 3 \mid P = c)}{p(C = 3)} = \frac{0.21 \cdot 0.2}{0.192} = 0.21875$$

3. *The sizes of* $\mathbb{P}$*,* $\mathbb{C}$*, and* $\mathbb{K}$ *are the same and finite. By Shannon's Theorem of Perfect Secrecy, the cryptosystem is therefore perfectly secure iff the probability distribution on the key space is uniform (which it is)* and *for all* m *in* $\mathbb{P}$ *and* c *in* $\mathbb{C}$*, there is a unique* k *in* $\mathbb{K}$ *with* $e_k(m) = c$*. But the latter is not the case, for example,* $2 = e_{k_4}(a) = e_{k_5}(a)$*.*

   Alternative answer which does not appeal to Shannon's Theorem: *We have* $p(P = c \mid C = 3) = 0.322916667$ *which does not equal* $p(P = c) = 0.21$ *and so this is not perfectly secure by definition.*

4. *Following on from the answer to the previous subpart, we merely have to ensure that each pair* $(m, c) \in \mathbb{P} \times \mathbb{C}$ *has a unique* k *with* $e_k(m) = c$*. In terms of the above table for* $e_k(m)$*, this means that we need that each column is a permutation of* $\mathbb{C}$*. In order to realise this, we need to maintain that each row of that table is a permutation as well, otherwise,* $e_k(\cdot)$ *for row* k *is no longer invertible as an encryption function and so no longer correct.*

   The columns for b*,* c*, and* d *are already permutations. The column for* a *has two* 2 *and the column for* e *has two* 3*. Fortunately, in row* k_4 *we* *have one of these 2 and one of these 3 so we can simply swap these two elements to arrive at*

	a	b	c	d	e
k_1	4	2	1	3	5
k_2	5	1	4	2	3
k_3	1	3	2	5	4
k_4	**3**	4	5	1	**2**
k_5	2	5	3	4	1

This swap maintains that each row is still a permutation of $\mathbb{C}$. *The resulting cryptosystem is therefore perfectly secure by Shannon's Theorem. This required two changes. We cannot have just one change as this would break that rows are permutations of* $\mathbb{C}$. *And we need at least one change as the original system is not perfectly secure. Thus* 2 *changes are minimal.*

5. *The cryptosystem has to be in essence a one-time pad, where the only variabilities are coding specifics such as the choice of alphabet and length of message. Suppose we choose the same set of plaintexts and ciphertexts, binary words of length n for some fixed $n > 1$. Then the keys are all binary strings of the same length where each bit value is chosen uniformly and independently at random. Encryption and decryption are then the same operation, the bitwise exclusive-or of the message with the key. In particular, the key is identical for encryption and decryption.*

   Issues for this system are numerous ones, let us just mention some:

   - *keys have to be as long as the message: not very practical*
   - *need to guarantee that keys are truly random and uniformly and independently in bits: hard to do*
   - *have to ensure that same key is* never *reused: cipher is malleable*
   - *need to get all of these keys to communicating parties securely: perfectly securely to be precise if we want to maintain perfect secrecy*
   - *requires authentication mechanism*

Exercise 55 (Past Exam Question) *Let p be a large prime of ≥ 1024 bits, and also a* safe prime, *i.e.* $p = 2 \cdot q + 1$ *where q is a prime as well. Let G be the abelian group with elements in $\{1, 2, \ldots, p-1\}$, group operation $x \cdot y \mod p$, and unit element 1. Let $\mathbb{P} = \mathbb{C} = \{1, 2, \ldots, p-1\}$.*

Below are the first three steps of a protocol in which Alice wants to send a message $m \in \mathbb{P}$ to Bob in encrypted form over an insecure communication channel:

- *(S1) Alice generates a random natural number x_A co-prime to $p-1$ and sends Bob a, defined as $m^{x_A} \mod p$.*
- *(S2) Bob generates a random natural number x_B co-prime to $p-1$ and sends Alice b, defined as $a^{x_B} \mod p$.*
- *(S3) Alice computes $x_A^{-1} \mod p-1$ and sends Bob a', defined as $b^{x_A^{-1}} \mod p$.*

1. *Briefly explain why Alice and Bob are able to perform the computations of these three steps (S1)-(S3), and efficiently so.*
2. *Specify what Bob does in the last step (S4) to recover the message m.*
3. *Assume that Alice and Bob follow steps (S1)-(S4) and messages are not tampered with in transit. Explain why your step (S4) correctly recovers m.*
4. Is this protocol subject to the man-in-the-middle attack? *If not, explain why not. If it is, describe such an attack and any assumptions that it requires to succeed.*
5. *Considering that p is a safe prime, explain why $\mathbb{P}' = \{2, 3, \ldots, p-2\}$ is a better set of plaintexts for the above protocol.* **Hint:** Lagrange's Theorem says that the size of any sub-group H of group G must be a divisor of the size of G.

1. *There are three operations that need to be supported: random sampling of a natural number, exponentiation in the group G, and multiplicative inverses modulo $p-1$. Random sampling can be done efficiently, although there is a tradeoff in the entropy of the randomly generated number and the effort taken to produce it. Exponentiation can be made*

   more efficient with techniques such as iterative squaring (polynomial in the logarithm of the exponent). Multiplicative inverses modulo $p-1$ can be computed efficiently using Extended Euclid Algorithm (although one may consider this to be expensive in practical computations).

2. *(S4) Bob computes $x_b^{-1} \mod p-1$ and computes m as $(a')^{x_B^{-1}}$.*

3. *This follows then since, all equations are modulo p, $(a')^{x_B^{-1}} = (m^{x_B})^{x_B^{-1}} = m^{x_B \cdot x_B^{-1}} = m$. Note that this is valid since the group G has order $p-1$ and so exponents that are equal modulo $p-1$ yield the same results.*

4. *Without proper authentication, this is subject to such attacks:*

   - *Suppose that Eve can persuade Alice that Eve is Bob. Then Eve can run the protocol with Alice as initiator and Eve as responder. This will allow Eve to learn the plaintext m that Alice had meant to sent to Bob securely. This is possible since participation in the protocol does not require the knowledge of a preexisting or shared secret.*
   - *Suppose that Eve can persuade Bob that Eve is Alice. Then Eve can run the protocol with Eve as initiator and Bob as responder. This will allow Eve to make Bob believe that he received some plaintext m from Alice.*
   - *We could combine both attacks so that Alice would think she sent m to Bob securely, and Bob thinks he has received some m' (which Eve now controls) from Alice securely.*
   - *The plaintexts 1 and $p-1$ are problematic: 1 generates the small subgroup $\{1\}$; and $p-1$ generates the small subgroup $\{1, p-1\}$ since $(p-1)^2 = 1 \mod p$. So these are insecure values: the exponents are co-prime to $p-1$ and so an attacker can learn the base when seeing 1 or $p-1$ as a message.*

     For plaintexts m in $\{2, 3, \ldots, p-2\}$ we claim that $H = \{m^x \mid x \geq 0\}$ is a sub-group of size q in G, and so this is secure enough. The reasoning for this involves Lagrange's Theorem and the fact that in this cyclic group there is at most one *one subgroup of order d for each divisor d of the group order – which allows us to exclude that the H has certain orders.*