Full name: _____

Student ID: _____

- This exam is made of 16 pages and **10 questions**.
- You work individually on your answers. With the submission of your answers, you declare that this is your **individual work** and you did not ask any other person for assistance. You neither discussed with fellow students nor asked in an internet forum or chat for help.
- Explain and **motivate** all your answers.
- You have **180 minutes** to do the exam (+45 if you are entitled for extra time).
- Make sure your **handwriting is clear** and legible. Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, continue on the back of the page.
- This is a **closed-book** exam.

Exercise 1 – Perfect Secrecy [10 points]

Consider the following cipher with three messages, $\mathcal{M} := \{m_1, m_2, m_3\}$, four keys $\mathcal{K} := \{k_1, k_2, k_3, k_4\}$, and four ciphertexts $\mathcal{C} := \{c_1, c_2, c_3, c_4\}$, defined by the following encryption map:

	k_1	k_2	k_3	k_4
m_1	c_2	c_1	c_3	c_4
m_2	c_4	c_3	c_1	c_2
m_3	c_1	c_2	c_4	c_3

(a) **[5 points]** Assuming plaintext and key distributions to be uniform, does this cipher satisfy perfect secrecy? Motivate your answer, and show at least one example of calculation of the conditional probabilities $Pr(M = m|C = c)$ in detail.

(b) **[5 points]** Suppose that we *swap* the plaintext space with the key space above, i.e. we now have $\mathcal{M} = \{m_1, m_2, m_3, m_4\}$ and $\mathcal{K} = \{k_1, k_2, k_3\}$. The ciphertext space remains unchanged. The encryption table is instead re-defined as the *transpose* of the encryption table above. Write the new encryption table in full. Does the marginal distribution $Pr(C = c)$ of the ciphertexts change? Is this new cipher perfectly secret? Motivate your answers. As above, you can assume that the new plaintexts and keys are uniformly distributed.

Solution

(a) Each ciphertext c_i occurs three times in the table, thus we have $\Pr[C = c_i] = \frac{3}{12} = \frac{1}{4}$ for each $i \in \{1, 2, 3, 4\}$. If we condition the probability of C after observing a specific plaintext message, then we can see that $\Pr[C = c_i|M = m_j] = \frac{1}{4}$ for all $i \in \{1, 2, 3, 4\}$ and $j \in \{1, 2, 3\}$. The reason is that each ciphertext appears exactly once in each row of the table. Since plaintexts are uniformly distributed, Bayes's rule gives us the following result for all $i \in \{1, 2, 3, 4\}$ and $j \in \{1, 2, 3\}$ as follows:

$$\Pr[M = m_i|C = c_i] = \frac{\Pr[C = c_i|M = m_i] \cdot \Pr[M = m_i]}{\Pr[C = c_i]} = \frac{\frac{1}{4} \cdot \frac{1}{3}}{\frac{1}{4}} = \frac{1}{3} = \Pr[M = m_i] \ .$$

Thus, the cipher satisfies perfect secrecy.

(b) The new encryption table is constructed transposing the table above:

	k_1	k_2	k_3
m_1	c_2	c_4	c_1
m_2	c_1	c_3	c_2
m_3	c_3	c_1	c_4
m_4	c_4	c_2	c_3

The marginal distribution of the ciphertext space does not change, since we have simply transposed the matrix. Hence, we still have $\Pr[C = c_i] = \frac{1}{4}$ for all $i \in \{1, 2, 3, 4\}$. On the other hand, if we compute the conditional probabilities $\Pr[C = c_i|M = m_j]$, we now see that something changes. For instance, taking $j = 1$ and $i = 3$ we have that $\Pr[C = c_3|M = m_1] = 0$, because c_3 does not occur on the row of m_1. Thus, using Bayes's rule to compute the reverse conditional probability $\Pr[M = m_1|C = c_3]$, we get

$$\Pr[M = m_1|C = c_3] = \frac{\Pr[C = c_3|M = m_1] \cdot \Pr[M = m_1]}{\Pr[C = c_3]} = \frac{0 \cdot \frac{1}{4}}{\frac{1}{4}} = \frac{1}{4} = 0 \neq \Pr[M = m_1] \ .$$

Thus, the cipher is no longer perfectly secret.

Exercise 2 – PRG and PRF [10 points]

(a) **[3 points]** Let G be a PRG with expansion factor ℓ such that $\ell(n) > n$ for all $n \in \mathbb{N}$, and consider this definition for G':

$$G'(s) = G(s^*) \ ,$$

where for all $s \in \{0,1\}^*$ such that $|s| = n$ and $s = s_1 s_2 \cdots s_n$, we have $s^* = s_1^* s_2^* \cdots s_n^*$ where

$$s_i^* = \begin{cases} 0 & \text{, if } i \equiv 0 \mod 2 \\ s_i & \text{, otherwise} \end{cases}$$

Is G' a PRG? Motivate your answer. It is not necessary to provide a full proof by reduction or a worked out counterexample, but you should at least provide the intuition behind your answer (for example: if you think that G' is not a PRG, then you should describe a strategy for a distinguisher).

(b) **[3 points]** Let $\mathbb{F} = \{f_s\}$ be a pseudorandom function where $f_s : \{0,1\}^{|s|} \to \{0,1\}^{|s|}$, and consider the new function f'_s defined for all $x \in \{0,1\}^*$ as:

$$f'_s(x) = f_s(0^{|s|}) f_s(x)$$

is f'_s a PRF as well? Motivate your answer, again by describing the intuition behind it. In particular, if you don't think this is a PRF, you should show a successfull strategy for the distinguisher.

(c) **[4 points]** Explain how a pseudorandom generator can be used to obtain an EAV-secure private-key encryption scheme. Why can't this scheme achieve CPA-security?

Solution

(a) Although the expansion factor condition is satisfied (since $\ell' \equiv \ell$ and thus $\ell'(n) > n$ for all $n \in \mathbb{N}$), G' is not necessarily a PRG, depending on the definition of G. The solution is similar to exercise 2.2 in Practical 2. Intuitively, the reason is that G' is calling G over the same seed, but setting the bits in the even positions to zero. Hence, G' is effectively working on half of the seed space. More specifically, suppose that we define the original PRG G as follows: $G(s) = \tilde{G}(s_2 s_4 \dots s_n)$, where $\tilde{G}$ is a PRG with expansion factor $\tilde{\ell}(n) > 2n$. In other words, G is now filtering only the bits in the even positions, and calling $\tilde{G}$ on them[1]. Now, if we call G' on a seed s of length $|s| = n$, what happens is the following:

$$G'(s) = G(s^*) = G(s_1 0 s_3 0 \dots s_{n-1} 0) = \tilde{G}(0 \dots 0) \ .$$

Hence, G' is always returning the string $\tilde{G}(0 \dots 0)$ in this specific definition of G, which suggests the following strategy for a distinguisher D':

1. Receive the string $x \in \{0,1\}^{\ell(n)}$.
2. Compute $y = \tilde{G}(0 \dots 0)$.
3. If $x = y$, return 1; otherwise, return 0.

It can be seen that this strategy yields a PPT distinguisher, since $\tilde{G}$ runs in polynomial time. Details for the probability calculations are omitted since they were not requested in the exam, but you should see (if not, try it out yourself) that with this strategy the distinguisher has a non-negligible advantage of correctly identifying strings generated by G'.

[1] For simplicity, here we assume that the length of the original seed is always an even number. It can be proved that G is still a PRG, however for now you can just assume that G is a PRG, without proving it formally.

(b) f'_s is also not a PRF. Intuitively, f'_s concatenates the output of f_s evaluated on the all-zero input vector with the output of f_s computed on the input x, so the first half of the output of f'_s is always the same. This observation suggests the following strategy for a PPT distinguisher D', assuming oracle access to a function $\varphi : \{0,1\}^{|s|} \to \{0,1\}^{2|s|}$, and to the original PRF f_s:

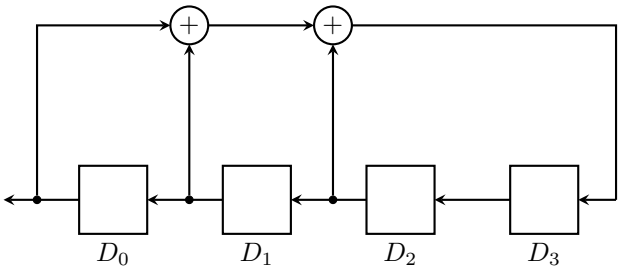
1. Query the oracle for f_s on the all-zero input vector, i.e. set $w = f_s(0^{|s|}$.
2. Query the oracle for ϕ on any input vector $x \in \{0,1\}^{|s|}$, and split the output $y = \varphi(x)$ in two halves, i.e. $y = y_L y_R$.
3. Repeat step 2 for a polynomial number of times $q(n)$, where $n = |s|$ is the security parameter. If $w = y_L$ every time, return 1; otherwise, return 0.

From the strategy above, we can see that the distinguisher always succeeds if φ corresponds to f'_s. This fact can be exploited to show that D' has a non-negligible advantage in distinguishing f'_s from a random function (details of the probability calculations omitted).

(c) For the construction of the pseudo-OTP scheme, refer to Section 3.3.3 in the textbook. Either a textual description or a diagram is fine as an answer to this question. The reason why the pseudo-OTP cannot achieve CPA-security is that the scheme is deterministic, and randomized encryption is required to resist chosen-plaintext attacks.

Exercise 3 – LFSR, Boolean Functions and Stream Ciphers [10 points]

Consider the following LFSR:



(a) **[4 points]** Draw the cycle of the LFSR transition diagram obtained by starting from the initial state $(0, 0, 0, 1)$. What is the length of this cycle?

(b) **[3 points]** Consider the following Boolean function $f : \mathbb{F}_2^3 \to \mathbb{F}_2$ described by its truth table:

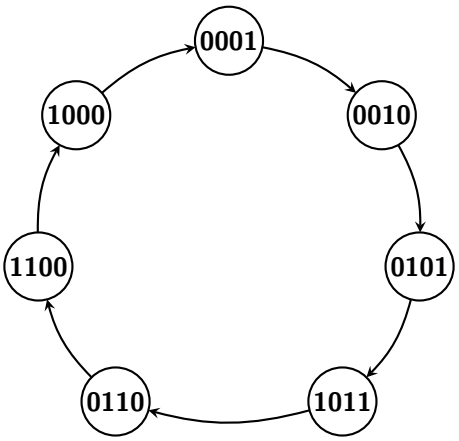
x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Compute the Walsh coefficient for the vector $a = (1, 0, 1)$. Based just on this coefficient, what can we say about the nonlinearity of f (i.e. assume that this is the maximum absolute value in the Walsh spectrum of f)?

(c) **[3 points]** Briefly describe the combiner model for stream cipher. Could the LFSR in part (a) be used as part of this model? If not, why?

Solution

(a)



Length of the cycle is 7.

(b) The Walsh coefficient of $a = (1, 0, 1)$ is computed as follows:

$$\begin{aligned}
W_f(1,0,1) &= \sum_{x \in \mathbb{F}_2^3} (-1)^{f(x) \oplus a \cdot x} = \\
&= (-1)^{f(0,0,0) \oplus (1,0,1) \cdot (0,0,0)} + (-1)^{f(0,0,1) \oplus (1,0,1) \cdot (0,0,1)} + \\
&+ (-1)^{f(0,1,0) \oplus (1,0,1) \cdot (0,1,0)} + (-1)^{f(0,1,1) \oplus (1,0,1) \cdot (0,1,1)} + \\
&+ (-1)^{f(1,0,0) \oplus (1,0,1) \cdot (1,0,0)} + (-1)^{f(1,0,1) \oplus (1,0,1) \cdot (1,0,1)} + \\
&+ (-1)^{f(1,1,0) \oplus (1,0,1) \cdot (1,1,0)} + (-1)^{f(1,1,1) \oplus (1,0,1) \cdot (1,1,1)} = \\
&= (-1)^{0 \oplus 0} + (-1)^{1 \oplus 1} + (-1)^{0 \oplus 0} + (-1)^{1 \oplus 1} + \\
&+ (-1)^{1 \oplus 1} + (-1)^{0 \oplus 0} + (-1)^{1 \oplus 1} + (-1)^{0 \oplus 0} = \\
&= 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = 8 \ .
\end{aligned}$$

Hence, considering $W_f(1, 0, 1) = 8$ as the maximum absolute value in the Walsh spectrum of f, the nonlinearity of the function is:

$$nl(f) = 2^{3-1} - \frac{1}{2} \cdot W_f(1, 0, 1) = 4 - \frac{1}{2} \cdot 8 = 0 \ ,$$

or equivalently, the function is linear considering only this coefficient. However, we do not have to compute other coefficients to be sure that $W_f(1, 0, 1)$ is the largest absolute value in the spectrum: Since $W_f(1, 0, 1)^2 = 8^2 = 64 = 2^{2 \cdot 3}$, by Parseval relation we know that all other Walsh coefficients must be zero. Hence, we can conclude that function f is indeed linear. Having a closer look at the truth table, one can see that f amounts to the XOR of of the first and the third variable. The fact that the largest absolute value in the spectrum occurs for the coefficient $(1, 0, 1)$ is no coincidence: as we know, the Walsh transform measures the correlation between function f and the linear functions $a \cdot x$. Since f is already defined as the XOR of the first and the third variable, the correlation is maximal when $a = (1, 0, 1)$.

(c) The combiner model requires n Linear Feedback Shift Registers (LFSRs) and a Boolean function of n variables $f : \mathbb{F}_2^n \to \mathbb{F}_2$. The LFSR are clocked together, and the output of each LFSR is fed as an input variable to the Boolean function. Specifically, at each step the bit generated in output by the i-th LFSR for $i \in \{1, \cdots, n\}$ represents the i-th input variable x_i of f. Then, the next output bit of the keystream is given by the output value $f(x_1, \cdots, x_n)$. The security of this model depends on the properties of f (for example, f needs to be balanced and nonlinear). In principle, the LFSR defined in (a) could be used as one of the three LFSRs that feed the combiner function f defined in (b), although it is not a maximum length LFSR (we know from (a) that one one cycle has length 7 instead of 15). In practice, one wants to use only maximum length LFSRs in the combiner model, to get an overall period that is the least common multiple of the periods of the composing LFSRs.

Exercise 4 – Block Ciphers [10 points]

(a) [**5 points**] Briefly describe the SPN paradigm for designing a block cipher.

(b) [**2 points**] Name a block cipher that is designed following the SPN paradigm. What is the block length of this cipher?

(c) [**3 points**] What do block ciphers correspond to from a theoretical point of view? In particular, explain which pseudorandom object block ciphers are a good candidate of.

Solution

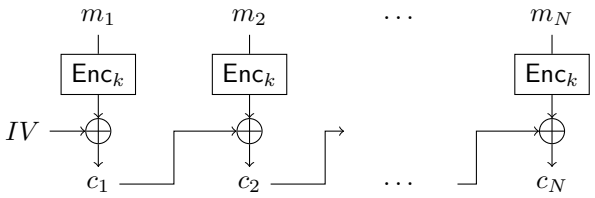
(a) SPN stands for Substitution-Permutation Network. It is a design method for block ciphers where each plaintext block is processed through several rounds of a so-called round function. The round function, in particular, is structured by first applying a confusion layer, then a permutation layer, and finally a (sub-)key mixing phase. The confusion layer is usually composed of small substitution boxes (S-boxes), which are bijective vectorial Boolean functions of the form $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$ that take a small n-bit chunk of the plaintext block and replace it with another one. The permutation layer, on the other hand, acts on the whole block state, by applying an invertible transformation that permutes the positions of the bits. Finally, the key mixing phase XORs the output of the permutation layer with a sub-key, computed from the initial key with a key expansion algorithm.

(b) An example of block cipher design that follows the SPN paradigm is the Advanced Encryption Standard (AES), also known as Rijndael. The block size of this cipher is always 128 bits, independently of the size of the key (which can be 128, 192 or 256 bits long).

(c) From a theoretical point of view, a block cipher is an efficient keyed permutation $F : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}$. In particular, a block cipher is a model for a strong pseudorandom permutation (SPRP), i.e., a pseudorandom permutation that stays pseudorandom even if the adversary has oracle access to the inverse F_k^{-1}.

Exercise 5 – Block Ciphers and Modes of Operations

[10 points] Given a block cipher with block size n, we define the following mode of operation:



Where Enc_k denotes the block cipher encryption algorithm with key k, and IV is a (public) initialization vector in $\{0,1\}^n$.

(a) **[2 points]** Provide a formal description of the encryption algorithm, i.e. write the ciphertexts as a function of the plaintexts, key, and initialization vector.

(b) **[2 points]** Provide a formal description of the corresponding decryption algorithm, i.e. write the plaintexts as a function of the ciphertexts, key, and initialization vector. For this point, let Dec_k be the block cipher decryption algorithm with key k.

(c) **[6 points]** Discuss the security of this encryption scheme in detail.

Solution

(a) Given the plaintext blocks $m_1, \ldots, m_N \in \{0,1\}^n$, a key k, and an initialization vector $IV \in \{0,1\}^n$, the encryption algorithm produces $c_1, \ldots, c_N \in \{0,1\}^n$ defined as:

$$\begin{aligned} c_1 &= \mathsf{Enc}_k(m_1) \oplus IV \\ c_i &= \mathsf{Enc}_k(m_i) \oplus c_{i-1} \qquad \text{for } i > 1 \end{aligned}$$

(b) Given the ciphertext blocks $c_1, \ldots, c_N \in \{0,1\}^n$, a key k, and an initialization vector $IV \in \{0,1\}^n$, the decryption algorithm produces $m_1, \ldots, m_N \in \{0,1\}^n$ defined as:

$$\begin{aligned} m_1 &= \mathsf{Dec}_k(c_1 \oplus IV) \\ m_i &= \mathsf{Dec}_k(c_i \oplus c_{i-1}) \qquad \text{for } i > 1 \end{aligned}$$

(c) This mode of operation might resemble the CBC mode, thus looking secure. However, here the xoring is applied after the block cipher, making the diffusion property reversible. In fact, given ciphertexts $c_1, \ldots, c_N$ and the initialization vector IV, we can compute the encryption of each plaintext block through the block cipher, as follows:

$$\begin{aligned} \mathsf{Enc}_k(m_1) &= c_1 \oplus IV \\ \mathsf{Enc}_k(m_i) &= c_i \oplus c_{i-1} \qquad \text{for } i > 1 \end{aligned}$$

Thus, making this mode of operation equivalent to the ECB mode in terms of security. Hence, one of the security issues of this mode is that it leaks possible patterns in the plaintext.

Exercise 6 – MACs [10 points]

Unconditionally Secure MACs

Alice and Bob use an unconditionally secure MAC where $\mathcal{M} = \{0, 1, \cdots, 8\}$, $\mathcal{A} = \mathbb{Z}_{11}$ and $\mathcal{K} = \mathbb{Z}_{11} \times \mathbb{Z}_{17}$. The tag generation algorithm is defined for all keys $k = (i, j) \in \mathcal{K}$ and messages $m \in \mathcal{M}$ as:

$$\mathsf{Mac}_k(m) = (i + mj) \mod 11 \ .$$

(a) **[2 points]** Suppose that Alice and Bob chose the key $k = (3, 6)$, and Alice wants to send $m = 5$ to Bob. What is the corresponding tag?

(b) **[3 points]** Suppose that Alice and Bob use the same key to authenticate two messages. Specifically, Oscar observes the two consecutive message/tag pairs $(4, 1)$ and $(6, 9)$. What is the key k? Explain how Oscar can retrieve it, assuming that the tag generation algorithm is public.

Computationally Secure MACs

(c) **[3 points]** Let $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be a length-preserving pseudorandom function. Show that the following MAC based on F is not secure: Gen outputs a uniform n-bit string k, and to authenticate a message $m \in \{0,1\}^n$ we compute the tag as $t := \mathsf{Mac}_k(m) = F_k(m)||F_k(m_R)$, where m_R is the reverse of m. The verification algorithm is defined in the canonical way.

(d) **[2 points** What are the main differences between the CBC mode of operation for encryption and the CBC-MAC construction?

Solution.

Unconditionally Secure MACs.

(a) We apply the formula for $\mathsf{Mac}_k(m)$ on $m = 5$ and $k = (3, 6)$:

$$\mathsf{Mac}_{(3,6)}(5) = (3 + 6 \cdot 5) \mod 11 = 33 \mod 11 = 0 \ .$$

Hence, the tag is $a = 0$.

(b) Since the key is reused, Oscar needs to solve the following system of modular equations:

$$\begin{cases} 1 &= (i + 4j) \mod 11 \\ 9 &= (i + 6j) \mod 11 \end{cases}$$

Subtracting the first equation from the second one, we obtain:

$$8 = (2j) \mod 11 \ ,$$

from which we deduce that $j = 4$. Substituting this value back in the first equation gives:

$$1 = (i + 16) \mod 11$$

Since $16 \mod 11 = 5$, we have that $i = 7$. Thus, the key is $k = (7, 4)$.

Computationally Secure MACs.

(c) A possible forgery attack is the following: the adversary queries the oracle with a message $m \in \{0,1\}^n$, obtaining the tag $t := t_0||t_1 = F_k(m)||F_k(m_R)$. Then, the adversary takes the message $m' := m_R$ and forges the tag $t' := t_1||t_0$ (i.e., the original tag t with the left and right halves swapped). Verification of (m', t') always succeeds, because

$$F_k(m')||F_k(m'_R) = F_k(m_R)||F_k((m_R)_R)) = F_k(m_R)||F_k(m) = t_1||t_0 \ .$$

(d) A first notable difference between CBC-mode and CBC-MAC are that CBC-mode outputs each intermediate ciphertext block, while CBC-MAC only outputs the last block as the tag of the message. The second difference is that CBC-mode uses a random Initialization Vector (IV) to bootstrap the encryption of the first block, while CBC-MAC does not (or equivalently, it uses an IV fixed to the all-zero string). In both cases, the use (or lack thereof) an IV is crucial to prove the security of the corresponding construction.

Exercise 7 – Hash Functions [10 points]

(a) **[3 points]** Suppose that we have a hash function $H : \{0,1\}^* \to \{0,1\}^{16}$. How many input messages do we need to hash before finding a collision with a probability of at least 50%?

(b) **[3 points]** Name an example of a standard cryptographic hash function, including its output length. Would it be feasible to perform a birthday attack with a success probability of at least 50%? Motivate your answer.

(c) **[4 points]** Provide a formal definition for the second preimage resistance property, by formulating a security game (similar to the one defined for collision resistance).

Solution

(a): Since $\mathcal{H}$ is a cryptographic hash function, we can assume its output to be uniformly distributed. Thus, the probability that a random input is mapped to y is $1/N$, where $N = 2^{16} = 65,536$ is the number of possible digests. Let us define $A(n)$ as the event "at least one collision within n hashes". Then

$$Pr[A] = 1 - Pr[\text{not } A]$$

where not A is "no collisions within n hashes". In this case, a collision means that no digest should repeat twice, and this happens with probability:

$$\frac{N}{N} \cdot \frac{N-1}{N} \cdot \frac{N-2}{N} \cdots \frac{N-n+1}{N}$$

and thus we have that

$$Pr[A] = 1 - Pr[\text{not } A] = 1 - \frac{N!}{(N-n)! \cdot N^n} \ .$$

We want to find the minimum value n for which $Pr[A] \geq 0.50$, that is to find the minimum value n such that

$$\frac{65536!}{(65536-n)! \cdot 65536^n} \leq \frac{1}{2} \ .$$

Through different methods we can find the value of n to be 302.

(b): Standard cryptographic hash function: SHA-256

(c): Second pre-image resistance is defined informally as: "Given a specific input and its hash, it is computationally infeasible to find a different input that produces the same hash. "

A hash function scheme $\mathcal{H} = (\mathsf{Gen}, H)$ is second preimage resistant if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ there exists a negligible function negl such that

$$\Pr[\mathsf{Hash\text{-}2ndPre}_{\mathcal{A},\mathcal{H}}(n) = 1] \leq \mathsf{negl}(n),$$

where the experiment $\mathsf{Hash\text{-}2ndPre}_{\mathcal{A},\mathcal{H}}(n)$ is defined as follows:

1. $s \leftarrow \mathsf{Gen}(1^n)$
2. $x \leftarrow \mathcal{A}_0(s)$
3. $y := H^s(x)$
4. $x' \leftarrow \mathcal{A}_1(s, x, y)$
5. **Return** 1 if $x' \neq x$ and $H^s(x') = y$, and 0 otherwise.

Exercise 8 – RSA Cryptosystem

[10 points] Alice wants to send a message to Bob using the RSA cryptosystem.

(a) **[4 points]** Bob creates an RSA key pair with primes $p = 5$ and $q = 7$, and public exponent $e = 5$. Compute the public and private key pair for Bob.

(b) **[4 points]** Alice encrypts the message $m = 5$. Compute the corresponding ciphertext.

(c) **[2 points]** Briefly explain how to make the basic RSA encryption scheme CCA-secure. In particular, do we need additional assumptions beside the hardness of factoring integers?

Solution

(a) Bob computes the modulus as $n = pq = 35$. Then he computes $\phi(n) = (p-1)(q-1) = 24$ and the private exponent as $d = e^{-1} \pmod{\phi(n)} = 5^{-1} \pmod{24}$. To compute this inverse, Bob uses the extended Euclidean algorithm: since we know that $\gcd(e, \phi(n)) = 1$, that means there exist d, a such that $ed + a\phi(n) = 1$, thus $ed - 1 = a\phi(n)$, so $\phi(n) \mid ed - 1$, that means $ed = 1 \pmod{\phi(n)}$, and by definition d is the inverse of e modulo $\phi(n)$. We find d (and a) by performing the extended Euclidean algorithm:

$$\begin{aligned} \mathbf{24} &= 4 \cdot \mathbf{5} + \mathbf{4} && \text{[eq1]} \\ \mathbf{5} &= 1 \cdot \mathbf{4} + \mathbf{1} && \text{[eq2]} \end{aligned}$$

and by going backward:

$$\begin{aligned} \mathbf{1} &= \mathbf{5} - 1 \cdot \mathbf{4} && \text{[isolating } \mathbf{1} \text{ in eq2]} \\ &= \mathbf{5} - 1 \cdot (\mathbf{24} - 4 \cdot \mathbf{5}) && \text{[replacing } \mathbf{4} \text{ by eq1]} \\ &= 5 \cdot \mathbf{5} - 1 \cdot \mathbf{24} && \text{[rearranging terms]} \end{aligned}$$

So $d = 5$. So the public key and private key coincide and are equal to $(n, e) = (n, d) = (35, 5)$ (not very safe).

(b) To compute $c = m^e \pmod{n} = 5^5 \pmod{35}$, Alice uses the square and multiply algorithm. She writes 5 in binary as 101, she initializes $c = 1$ then by starting from the MOST significant bit, at each position, she squares c and, if the bit is 1, she also multiplies it by m, so:

b=1 so $c = c^2 m \pmod{n} = 1^2 \cdot 5 \pmod{35} = 5$

b=0 so $c = c^2 \pmod{n} = 5^2 \pmod{35} = 25$

b=1 so $c = c^2 m \pmod{n} = 25^2 \cdot 5 \pmod{35} = 10$

So the corresponding ciphertext is $c = 10$.

(c) : Use padding based on hash function, again with CCA security. We need to add a non-deterministic element to the encryption scheme by using a secure padding scheme, like OAEP, which combines the plaintext with random data before encryption. A plain, deterministic RSA encryption scheme is not CCA-secure because it produces the same ciphertext everytime when given the same plaintext, allowing an attacker to easily manipulate messages.

Exercise 9 – Diffie-Hellman Key Agreement [10 points]

[10 points] Alice and Bob want to agree on a common secret key using the Diffie-Hellman Key Agreement protocol.

(a) **[5 points]** They agree on the multiplicative group $G = \mathbb{Z}_{19}^*$. Show that Alice and Bob can use $g = 2$ as a generator for G.

(b) **[3 points]** Manually execute the DH protocol between Alice and Bob, with secret exponents being respectively $a = 6$ and $b = 2$.

(c) **[2 points]** Beside its small size, is $p = 19$ a good choice for the security of the scheme? Motivate your answer.

Solution:

(a): We must show that $g = 2$ is a generator of G. g is a generator if $k = |\mathbb{Z}_{19}^*|$ when $2^k \equiv 1 \pmod{19}$. In other words, the multiplicative order of g is equal to $\mathbb{Z}_{19}^*|$. Since $|\mathbb{Z}_{19}^*| = 18$, the order of 2 modulo 19 must be 18. We check that $2^d \not\equiv 1 \pmod{19}$ for all proper divisors d of 18, and $2^{18} \equiv 1 \pmod{19}$.

$$
\begin{aligned}
2^1 &\equiv 2 \pmod{19},\\
2^2 &\equiv 4 \pmod{19},\\
2^3 &\equiv 8 \pmod{19},\\
2^6 &= (2^3)^2 \equiv 8^2 = 64 \equiv 7 \pmod{19},\\
2^9 &= 2^6 \cdot 2^3 \equiv 7 \cdot 8 = 56 \equiv 18 \pmod{19},\\
2^{18} &\equiv (2^9)^2 \equiv 18^2 = 324 \equiv 1 \pmod{19}.
\end{aligned}
$$

Since $2^d \not\equiv 1$ for $d = 1, 2, 3, 6, 9$, and $2^{18} \equiv 1$, the order of 2 is 18. Hence 2 is a generator of $\mathbb{Z}_{19}^*$.

(b): Execute the Diffie–Hellman protocol with $a = 6$ and $b = 2$.
Public parameters: $p = 19$, $g = 2$.

$$
\begin{aligned}
Alice &= g^a \bmod 19 = 2^6 \bmod 19 \equiv 64 \equiv 7,\\
Bob &= g^b \bmod 19 = 2^2 \bmod 19 \equiv 4.
\end{aligned}
$$

Alice sends $A = 7$, Bob sends $B = 4$.

The shared secret key is computed for Alice as: $K_A = B^a \bmod 19 = 4^6 \bmod 19$. To do this it is good to know to the following values:

$$4^2 \equiv 16, \quad 4^4 \equiv 16^2 = 256 \equiv 9 \pmod{19},$$

From this we compute $\Rightarrow K_A = 4^6 = 4^4 \cdot 4^2 \equiv 9 \cdot 16 = 144 \equiv 11 \pmod{19}$.

For Bob we compute the key to be: $K_B = A^b \bmod 19 = 7^2 \bmod 19 = 49 \equiv 11 \pmod{19}$.

Hence, both obtain the shared secret $K = 11$.

(c): Besides its small size, $p = 19$ is not a good choice for security for a few reasons, including but not limited to:

1. The group order $p-1 = 18 = 2 \cdot 3^2$ has only small prime factors, making the discrete logarithm problem (DLP) relatively easy.

2. Secure Diffie-Hellman requires a large prime modulus (typically at least 2048 bits) and often a safe prime $p = 2q + 1$ with q prime to avoid small-subgroup attacks.

3. For such a small p, DLP can be solved instantly by brute force or generic attacks.

Therefore, $p = 19$ is not secure for practical use.

Exercise 10 – Digital Signatures [10 points]

Alice wants to sign a message for Bob using the Digital Signature Algorithm (DSA).

(a) **[3 points]** Let the prime parameters be $p = 11$, $q = 5$, and let the generator be $g = 2$. Alice chooses a private signing key $x = 3$. Compute the corresponding public key y.

(b) **[4 points]** Alice wants to sign the message $m = 6$, with nonce $k = 1$. Use $H(m) = m$, and write down the signature pair (r, s).

(c) **[3 points]** In the standard DSA algorithm, what function H is used instead of $H(m) = m$? In this case, which additional assumption do we need beside the hardness of the discrete logarithm problem for the security of the scheme?

Solutions:

$$p = 11, \quad q = 5, \quad g = 2, \quad x = 3 \text{ (private key)}, \quad m = 6, \quad k = 1, \quad H(m) = m.$$

(a): The public key is $y = g^x \bmod p$. So:

$$y = 2^3 \bmod 11 = 8.$$

So $y = 8$.

(b): In DSA:

$$r = (g^k \bmod p) \bmod q, \qquad s = k^{-1}\big(H(m) + x \cdot r\big) \bmod q.$$

Compute $g^k \bmod p = 2^1 \bmod 11 = 2$, hence

$$r = 2 \bmod 5 = 2.$$

Since $k = 1$, $k^{-1} \bmod q = 1$. With $H(m) = m = 6$ and $x = 3$,

$$s = 1 \cdot (6 + 3 \cdot 2) \bmod 5 = (6 + 6) \bmod 5 = 12 \bmod 5 = 2.$$

Thus the signature is $(r, s) = (2, 2)$.

(c): In practice DSA uses a cryptographic hash function (e.g. SHA-1, SHA-256, etc.). The hash output is interpreted as an integer and (if necessary) truncated to an element of $\mathbb{Z}_q$.

For a provable security reduction one typically needs a stronger assumption about the hash than merely collision-resistance in isolation: security proofs for (standard) DSA are usually given in the random oracle model (i.e. the hash is modeled as a random oracle), or at minimum it is assumed that the hash being used behaves like a cryptographic hash (collision-resistant and pseudo-random / unpredictable). Additionally, secure DSA requires that the nonces, km are uniformly random and never reused. The leakage or reuse of k completely breaks security.