Full name: _____

Student ID: _____

- This exam is made of 13 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 Security

[10 points] We have a cipher with three messages $\mathcal{M} := \{m_1, m_2, m_3\}$, five keys $\mathcal{K} := \{k_1, k_2, k_3, k_4, k_5\}$, and four possible 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	k_5
m_1	c_1	c_3	c_2	c_4	c_4
m_2	c_4	c_4	c_4	c_3	c_1
m_3	c_2	c_1	c_3	c_1	c_4

(a) **[2 points]** Based on the cardinalities of the message, ciphertext and key spaces, can the above cipher be perfectly secret, in principle? Explain why.

(b) **[4 points]** Assuming plaintext and key distributions to be uniform, compute $\Pr[C = c]$ for each ciphertext $c \in \mathcal{C}$.

(c) **[4 points]** Under the same distribution assumptions, determine whether this cipher is perfectly secure, and prove your statement.

Solution

(a) A necessary condition for perfect secrecy is that $|\mathcal{K}| \geq \mathcal{C} \geq \mathcal{M}$, i.e. we have at least as many keys as ciphertexts, and in turn we have at least as many ciphertexts as plaintexts. This is the case for the cipher above since we have $|\mathcal{K}| = 5$, $|\mathcal{C}| = 4$ and $|\mathcal{M}| = 3$. So, in principle this cipher can satisfy perfect secrecy.

(b) As the distributions of keys and messages are uniform (and independent), we can simply count the number of times a given ciphertext appears in the table:

$$\Pr[C = c_1] = 4/15$$
$$\Pr[C = c_2] = 2/15$$
$$\Pr[C = c_3] = 3/15$$
$$\Pr[C = c_4] = 6/15$$

(c) No, this cipher is not perfectly secure. We can prove it by showing that

$$\Pr[M = m_2 | C = c_4] = \frac{\Pr[M = m_2, C = c_4]}{\Pr[C = c_4]} = \frac{3/15}{6/15} = \frac{1}{2} \neq \frac{1}{3} = \Pr[M = m_2] \ .$$

An alternative way is to use Bayes's theorem to compute $\Pr[M = m_2 | C = c_4]$ as:

$$\Pr[M = m_2 | C = c_4] = \frac{\Pr[C = c_4 | M = m_2] \cdot \Pr[M = m_2]}{\Pr[C = c_4]} = \frac{\frac{3}{5} \cdot \frac{1}{3}}{\frac{6}{15}} = \frac{1}{2} \neq \frac{1}{3} = \Pr[M = m_2] \ .$$

Exercise 2 – PRG and PRF [10 points]

(a) **[4 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 $n \in \mathbb{N}$ and $i \in \{1, \cdots, \ell(n)\}$ we have

$$G(s)_i^{\#} = \begin{cases} 1 & \text{, if } i \equiv 1 \mod 2 \\ G(s)_i & \text{, otherwise} \end{cases}$$

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:

$$f'_s(x) = f_s(x_R)$$

where for all $x \in \{0,1\}^*$, x_R is the string x taken in reverse order. Is f'_s a PRF as well? Motivate your answer, again by describing the intuition behind it.

(c) **[3 points]** Provide the formal definition of a strong pseudorandom permutation. What cryptographic primitive is a good candidate of such a permutation?

Solution.

(a) Although the expansion factor condition is met here, G' is not a PRG, under any definition of G. To see intuitively why, consider the following distinguisher D': upon having in input a string $x \in \{0,1\}^{\ell(n)}$, D' checks whether every odd position of x is set to 1. If so, it returns 1, otherwise it returns 0. In this way, D' always returns 1 when given as an input a string generated by G'. On the other hand, a true random string $r \in \{0,1\}^{2\ell(n)}$ contains 1 in all odd positions only with probability $2^{-\ell(n)/2}$. Therefore, the distinguisher has a non-negligible advantage of $1 - 2^{\ell(n)/2}$ to correctly distinguish the output of G'.

(b) f'_s is not a PRF. Consider the following strategy for a distinguisher D': given oracle access to φ, D' submits $x \leftarrow \{0,1\}^{|s|}$ to the oracle of φ, getting in output $y \in \{0,1\}^{|s|}$. Then, D' submits x_R (i.e., x in reverse order) to the oracle of f_s, getting in output y', and checks if $y' = y$. D' repeats these steps for a polynomial number of queries $q(n)$, where $n = |s|$, always choosing a fresh random input x at each iteration. If it sees $y' = y$ in each step, it will output 1 (that is, D' believes that φ is an oracle of f'_s), otherwise it will return 0. Now, if $\varphi \equiv f'_s$, then D' will always correctly guess that it is interacting with f'_s. On the other hand, the probability that a true random function $f^* : \{0,1\}^{|s|} \to \{0,1\}^{|s|}$ satisfies the check of D' for a polynomial number of queries is $q(n)2^{-n}$, since the probability that a single output of f^* coincides with the output of f_s when the input is reversed is 2^{-n}. Therefore, D' correctly identifies f'_s with a non-negligible probability of $1 - q(n)2^{-n}$.

(c) See Definition 3.27 on the textbook. Basically, a strong pseudorandom permutation (S-PRP) is a PRP F such that there is no PPT adversary able to distinguish F from a random permutation, even if F is given access to an oracle of the inverse permutation F^{-1}. Block ciphers are believed to be good concrete models for S-PRP.

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

Consider the Boolean function $f : \{0,1\}^3 \to \{0,1\}$ defined for all $(x_1, x_2, x_3) \in \{0,1\}^3$ as:

$$f(x_1, x_2, x_3) = \begin{cases} x_1 & , \text{ if } x_2 = 1 \text{ and } x_3 = 1 \\ x_1 \oplus 1 & , \text{ otherwise} \end{cases}$$

1. **[4 points]** Write the truth table of the function.
2. **[2 points]** Is the function balanced?
3. **[4 points]** Briefly describe the combiner generator model for stream ciphers. Could function f be used in such a model?

Solution

(a) The truth table of f is as follows:

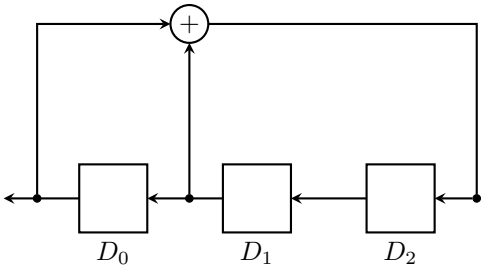
x_1	x_2	x_3	$f(x)$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(b) Yes, f is balanced since the output column of its truth table has an equal number of zeros and ones (that is, $|f^{-1}(0)| = |f^{-1}(1)| = 4 = 2^{3-1}$).

(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 LFSRs 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, function f defined in (a) could be used as a combiner for the output of three LFSR, since it is balanced and thus does not have an obvious statistical bias to predict the next bit. However, other properties (such as the nonlinearity of f) should also be checked.

Exercise 4 – LFSR [10 points]

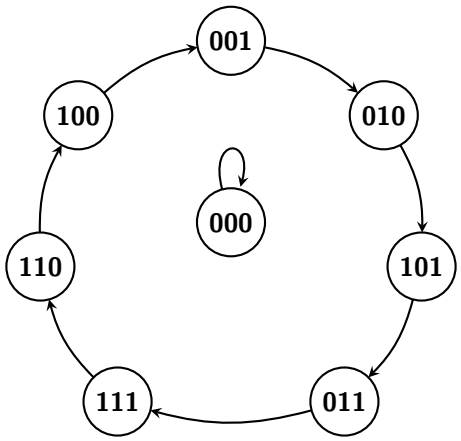
Consider the following LFSR:



1. **[6 points]** Is this a maximum-length LFSR? Draw the full transition graph where the nodes represent the states of the registers.
2. **[4 points]** If we start from the initial state $(0, 1, 1)$, what are the first 8 bits generated in output for the keystream? When do they start repeating?

Solution

(1) Yes, the LFSR has maximum length $2^3 - 1 = 7$. To show this, we construct the transition diagram of all $2^3 = 8$ LFSR states:



(2) Since the leftmost register is holding the bit that is given in output at each clock cycle, we can simply look at the transition diagram and check what is the leftmost bit of the states in the path of length 8 which starts from $(0, 1, 1)$. Hence, the first 8 bits given in output are $(0, 1, 1, 1, 0, 0, 1, 0)$.

It will repeat at the 8th bit.

Exercise 5 – Block Ciphers and Mode of Operations [10 points]

(a) [**10 points**] A company is developing their own symmetric cipher. Their website is claiming that the underlying block cipher is based on the substitution-permutation network paradigm. For enhanced security, the block cipher has both a block length and key length of 1024 bits, and it runs for 128 rounds. The confusion layer of this SPN cipher, on the other hand, is defined by S-boxes of the form $F : \{0,1\}^8 \to \{0,1\}^4$, to ensure irreversibility of the ciphertexts. Further, they extend this cipher with CBC mode to encrypt plaintexts of arbitrary length. For efficiency, only the first plaintext is encrypted using a random IV. Then, they use the last ciphertext block as a new IV for the next plaintext to encrypt.

Analyze the security and efficiency issues of the cipher described above.

Solution

We list the security and efficiency issues of the above cipher description, along with the corresponding points if they are mentioned in your answer:

- *Block length and key length of 1024 bits (2 points)*: this is way larger than necessary. Going beyond 256-bit keys does not make sense in a symmetric cipher from a security standpoint, especially to thwart brute force attacks (since exhaustively searching a space of 2^{256} keys would require more energy/matter than what is presumably contained in the observable universe). So, it is not a reasonable choice from an efficiency point of view.
- *128 rounds (1 points)*: this is also overly cautious on the security side. Typical block ciphers have between 10-30 rounds. Doing more than 100 rounds is also a waste of computational resources.
- *SPN design and S-boxes (3 points)*: the website claims that the design of the cipher is based on the SPN paradigm, but then it also mentions S-boxes where the output space (4 bits) is smaller than the input space (8 bits). This is contradictory, since SPN ciphers require S-boxes with the same number of input and output bits to ensure invertibility (and thus decryption).
- *CBC mode (3 points)*: The idea of "recycling" the final ciphertext block as a new IV for the next message is tempting from an efficiency point of view, so one doesn't have to come up with a fresh random IV every time. However, this breaks the CPA-security property of CBC mode. To see why, think of how an adversary could win the CPA indistinguishability game by interacting with an encryption oracle of the cipher above. The BEAST attack exploited this issue in TLS 1.0.

Finally, 1 point is awarded if your answer is sufficiently clear/well written.

Exercise 6 – Message Authentication Codes [10 points]

Unconditionally Secure MACs

[6 points] Alice and Bob use a secure MAC $(\mathcal{M}, \mathcal{A}, \mathcal{K}, \mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ to authenticate their messages, where $\mathcal{M} = \{0, 1, \cdots, 11\}$, $\mathcal{A} = \mathbb{Z}_{13}$ and $\mathcal{K} = \mathbb{Z}_{13} \times \mathbb{Z}_{13}$. The key generation algorithm Gen simply samples with uniform probability a key from $\mathcal{K}$, while the tag generation algorithm is defined for all keys $k = (i, j) \in \mathcal{K}$ and messages $m \in \mathcal{M}$ as:

$$\mathsf{Mac}_k(m) = (i + mj) \mod 13 \ .$$

Since 13 is prime, we know that the corresponding authentication matrix is an orthogonal array, so the MAC is unconditionally secure.

(a) **[2 points]** Oscar observes the pair $(10, 6)$ on the channel, where the first component is the message while the second is the corresponding authenticator. What is the probability for Oscar to guess the correct key?

(b) **[4 points]** Alice and Bob commit a mistake and use the same key $k = (i, j)$ to authenticate another message after $m = 10$. In particular, after seeing $(10, 6)$, Oscar sees the pair $(7, 4)$. What is the key shared by Alice and Bob? Explain in detail how Oscar can retrieve it, assuming that the tag generation algorithm is public.

Solution

(a) Since the MAC is unconditionally secure, we know that the total number of keys for a given pair of message and authenticator equals the number of authenticators. Hence, the probability for Oscar to guess the correct key is

$$\frac{1}{|\mathcal{A}|} = \frac{1}{|\mathbb{Z}_{13}|} = \frac{1}{13} \ .$$

(b) The key is recovered by solving the following system of equations:

$$\begin{cases} 6 &= (i + 10j) \mod 13 \\ 4 &= (i + 7j) \mod 13 \end{cases}$$

In particular, the first equation is the one related to message/tag pair $(10, 6)$, while the second one is related to $(7, 4)$. To find i and j, let us subtract the second equation from the first one, obtaining:

$$2 = 3j \mod 13 \ .$$

By bruteforcing the values of j from 0 onwards, we can find that $j = 5$ gives the correct result, since $15 \mod 13 = 2$. Hence, we have $j = 5$. Substituting this value back in the first equation gives us

$$6 = (i + 50) \mod 13 \ .$$

Here, remark that $13 \cdot 4 = 52$. Thus, in order to obtain 6 when reducing modulo 13, we need to add $i = 8$ to 50. Therefore, the key is $k = (8, 5)$.

Computationally Secure MACs

[4 points] Turn the following informal description of a MAC scheme into a formal description, then prove its insecurity: to authenticate a message, we XOR it with its reverse (i.e. the same message but with the bits in reverse order), and apply a pseudorandom function to the result of the XOR.

Solution

The MAC scheme can be formally defined as follows: given a pseudorandom function $F_k : \{0,1\}^n \to \{0,1\}^n$ and a message $m \in \{0,1\}^n$ where $m = m_0 m_1 \dots m_n$ and $m_R = m_n \dots m_1 m_0$ is the reverse of m, the authentication tag a is computed as $\mathsf{Mac}_k(m) = F_k(m \oplus m_R)$. To show its insecurity, given a pair (m, a), the tag a is also valid for $m' = m_R$, since $m_{R_R} = m$. Formally, one has that $a = F_k(m \oplus m_R) = F_k(m_R \oplus m) = F_k(m' \oplus m'_R)$.

Exercise 7 – Hash Functions

[10 points] Let $\mathcal{H} = (\mathsf{Gen}, H)$ be a hash function.

(a) **[3 points]** Provide a formal definition for the collision resistance property, by formulating a security game (similar to the ones defined below for similar properties).

Remember now the following two definitions.

Definition 1 (Preimage Resistance). A hash function $\mathcal{H} = (\mathsf{Gen}, H)$ is preimage resistant if for all PPTA $\mathcal{A}$ there is a negligible function negl such that

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

where $\mathsf{Hash\text{-}preimage}_{\mathcal{A},\mathcal{H}}(n)$ is the following game:

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

Definition 2 (Second Preimage Resistance). A hash function $\mathcal{H} = (\mathsf{Gen}, H)$ is second preimage resistant if for all PPTA $\mathcal{A}$ there is a negligible function negl such that

$$\Pr[\mathsf{Hash\text{-}preimage2}_{\mathcal{A},\mathcal{H}}(n) = 1] \leq \mathsf{negl}(n)$$

where $\mathsf{Hash\text{-}preimage2}_{\mathcal{A},\mathcal{H}}(n)$ is the following game:

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

(b) **[7 points]** Does one of the two properties defined above imply the other? If yes, which one implies which? Formally prove your statement.

Solution

(a) A hash function $\mathcal{H} = (\mathsf{Gen}, H)$ is collision resistant if for all PPTA $\mathcal{A}$ there is a negligible function negl such that

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

where $\mathsf{Hash\text{-}coll}_{\mathcal{A},\mathcal{H}}(n)$ is the following game:

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

(b) Yes, any hash function that is second preimage resistant is also preimage resistant. We can prove this by reduction, showing that an adversary who is able to win the Hash-preimage game can also win the Hash-preimage2 game. Let $\mathcal{A}$ be a PPTA that can find a preimage with non-negligible probability, we use it to build an adversary $\mathcal{A}'$ for the Hash-preimage2 game in the following way. $\mathcal{A}'$ takes as input s and x, it computes $y := H^s(x)$ and then it invokes $\mathcal{A}$ on (s, y). It is trivial to see that the PPTA $\mathcal{A}'$ defined in this way wins the Hash-preimage2 game with non-negligible probability.

Exercise 8 – Primality Testing and Group Generators [10 points]

(a) **[6 points]** Using the Fermat's primality test, show that the number 29 is composite with a probability less than 20%.

(b) **[4 points]** Consider the multiplicative group $\mathbb{Z}_{29}^*$. What are the possible orders of its subgroups?

Solution

(a) The probability that p is composite after finding k witnesses for its primality is $\frac{1}{2^k}$. Thus, if we want our probability to be less than 20%, we have to find $k = 3$ witnesses, since $\frac{1}{2^3} = \frac{1}{8} < 0.20$.

Let us take $a = 2$. To verify that this is a witness for the primality of 29, we compute $a^{p-1} \bmod p$ with $p = 29$. The binary expansion of $p - 1 = 28$ is

$$28 = (11100)_2.$$

Setting $c = 1$ and scanning the bits from most significant to least significant, we obtain:

- $(b = 1)$ $c \leftarrow c^2 \cdot a \bmod p = 1^2 \cdot 2 \bmod 29 = 2$
- $(b = 1)$ $c \leftarrow c^2 \cdot a \bmod p = 2^2 \cdot 2 \bmod 29 = 8$
- $(b = 1)$ $c \leftarrow c^2 \cdot a \bmod p = 8^2 \cdot 2 \bmod 29 = 128 \bmod 29 = 12$
- $(b = 0)$ $c \leftarrow c^2 \bmod p = 12^2 \bmod 29 = 144 \bmod 29 = 28$
- $(b = 0)$ $c \leftarrow c^2 \bmod p = 28^2 \bmod 29 = 784 \bmod 29 = 1$

Hence, $2^{28} \equiv 1 \pmod{29}$, so $a = 2$ passes Fermat's test.

We now provide two additional witnesses.

Witness 2: $a = 3$**.** Compute $3^{28} \bmod 29$ with the same exponent bits $(11100)_2$:

- $(b = 1)$ $c = 1^2 \cdot 3 \bmod 29 = 3$
- $(b = 1)$ $c = 3^2 \cdot 3 \bmod 29 = 27$
- $(b = 1)$ $c = 27^2 \cdot 3 \bmod 29 = 729 \cdot 3 \bmod 29 = 4 \cdot 3 \bmod 29 = 12$
- $(b = 0)$ $c = 12^2 \bmod 29 = 144 \bmod 29 = 28$
- $(b = 0)$ $c = 28^2 \bmod 29 = 784 \bmod 29 = 1$

Thus, $3^{28} \equiv 1 \pmod{29}$.

Witness 3: $a = 10$**.** Compute $10^{28} \bmod 29$:

- $(b = 1)$ $c = 1^2 \cdot 10 \bmod 29 = 10$
- $(b = 1)$ $c = 10^2 \cdot 10 \bmod 29 = 100 \cdot 10 \bmod 29 = 13 \cdot 10 \bmod 29 = 14$
- $(b = 1)$ $c = 14^2 \cdot 10 \bmod 29 = 196 \cdot 10 \bmod 29 = 22 \cdot 10 \bmod 29 = 17$
- $(b = 0)$ $c = 17^2 \bmod 29 = 289 \bmod 29 = 28$
- $(b = 0)$ $c = 28^2 \bmod 29 = 784 \bmod 29 = 1$

Hence, $10^{28} \equiv 1 \pmod{29}$.

Since we found $k = 3$ witnesses, the probability that 29 is composite is at most $\frac{1}{2^3} = \frac{1}{8} = 12.5\% < 20\%$.

(b) By Lagrange theorem, we know that nay subgroup must have order dividing the order of the group, which is $\phi(29) = 29 - 1 = 28$, since 29 is prime. So the possible orders are the divisors or 28, which are: 1, 2, 4, 7, 14, 28.

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

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

(a) **[5 points]** They are using the multiplicative group $G = \mathbb{Z}_7^*$ and the generator $g = 2$. Manually execute the DH protocol between Alice and Bob, with secret exponents being respectively $a = 6$ and $b = 2$.

(b) **[5 points]** Explain the correctness of the DH Key Agreement protocol (i.e., show why, in general, Alice and Bob generate the same key).

Solution

(1) Alice computes $A = g^6 \pmod 7 = 1$, while Bob computes $B = g^b \pmod 7 = 4$. Alice sends A to Bob, while Bob sends B to Alice. Now, Alice computes the shared key as $k = B^a \pmod 7 = 1$, while Bob computes it as $k = A^b \pmod 7 = 1$.

(2) Alice and Bob generate the same key since $B^a = (g^b)^a = g^{ab} = (g^a)^b = A^b$.

Exercise 10 – RSA Cryptosystem [10 points]

Alice wants to send a message to Bob using the RSA cryptosystem.

(a) Bob creates an RSA key pair with primes $p = 7$ and $q = 11$, and public exponent $e = 7$. Compute the public and private key pair for Bob.

(b) Alice encrypts the message $m = 6$. Compute the corresponding ciphertext.

Solution

(a) Bob computes the modulus as $n = pq = 77$. Then he computes $\phi(n) = (p-1)(q-1) = 60$ and the private exponent as $d = e^{-1} \pmod{\phi(n)} = 7^{-1} \pmod{60}$.

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:

$$\begin{aligned} \mathbf{60} &= 8 \cdot \mathbf{7} + \mathbf{4} && \text{[eq1]} \\ \mathbf{7} &= 1 \cdot \mathbf{4} + \mathbf{3} && \text{[eq2]} \\ \mathbf{4} &= 1 \cdot \mathbf{3} + \mathbf{1} && \text{[eq3]} \end{aligned}$$

and by going backward:

$$\begin{aligned} \mathbf{1} &= \mathbf{4} - 1 \cdot \mathbf{3} && \text{[isolating } \mathbf{1} \text{ in eq3]} \\ &= \mathbf{4} - (\mathbf{7} - 1 \cdot \mathbf{4}) && \text{[replacing } \mathbf{3} \text{ by eq2]} \\ &= 2 \cdot \mathbf{4} - \mathbf{7} && \text{[rearranging terms]} \\ &= 2 \cdot (\mathbf{60} - 8 \cdot \mathbf{7}) - \mathbf{7} && \text{[replacing } \mathbf{4} \text{ by eq1]} \\ &= 2 \cdot \mathbf{60} - 17 \cdot \mathbf{7} && \text{[rearranging terms]} \end{aligned}$$

So $-17 \cdot 7 + 2 \cdot 60 = 1$. This means that $d = -17 = 43 \pmod{60}$. In the end we find that the public key is $(n, e) = (77, 7)$ while the private key is $(n, d) = (77, 43)$

(b) To compute $c = m^e \pmod{n} = 6^7 \pmod{77}$, Alice uses the square and multiply algorithm. She writes 7 in binary as 111, 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 $c = c^2 m \pmod{n} = 1^2 \cdot 6 \pmod{77} = 6$

b=1 so $c = c^2 m \pmod{n} = 6^2 \cdot 6 \pmod{77} = 62$

b=1 so $c = c^2 m \pmod{n} = 62^2 \cdot 6 \pmod{7} = 41$

So the corresponding ciphertext is $c = 41$.