TECHNISCHE UNIVERSITEIT EINDHOVEN
Faculty of Mathematics and Computer Science
Introduction to Cryptology, Monday 27 January 2020

Name :

TU/e student number :

Exercise	1	2	3	4	5	6	7	total
points								

Notes: Please hand in this sheet at the end of the exam. You may keep the sheet with the exercises.
This exam consists of 7 exercises. You have from 13:30 – 16:30 to solve them. You can reach 100 points.
Make sure to justify your answers in detail and to give clear arguments. Document all steps, in particular of algorithms; it is not sufficient to state the correct result without the explanation. If the problem statement asks for usage of a particular algorithm other solutions will not be accepted even if they give the correct result.
All answers must be submitted on TU/e letterhead; should you require more sheets ask the proctor. State your name on every sheet.
Do not write in red or with a pencil.
You are not allowed to use any books, notes, or other material.
You are allowed to use a simple, non-programmable calculator without networking abilities. Usage of laptops and cell phones is forbidden.

1. This exercise is about LFSRs. Do the following subexercises for the sequence

$$s_{i+5} = s_{i+4} + s_{i+2} + s_i.$$

   (a) Draw the LFSR corresponding this sequence. | 2 points

   (b) State the characteristic polynomial f and compute its factorization. You do not need to do a Rabin irreducibility test but you do need to argue why a factor is irreducible. | 10 points

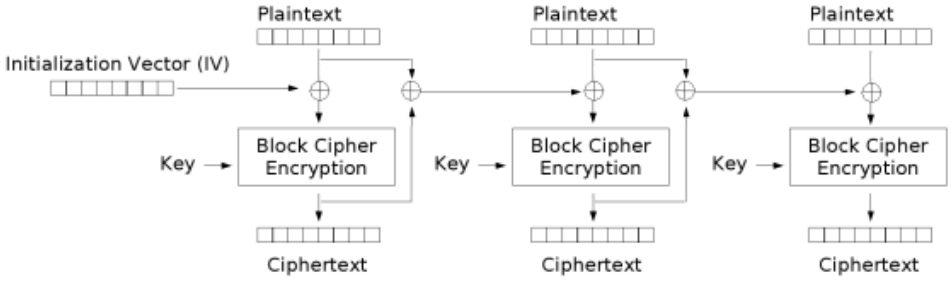
   (c) For each of the factors of f compute the order. | 8 points

   (d) What is the longest period generated by this LFSR? Make sure to justify your answer. | 3 points

   (e) State the lengths of all subsequences so that each state of n bits appears exactly once. Make sure to justify your answer. | 8 points

2. This exercise is about modes.

   Here is a schematic description of the Propagating CBC (PCBC) mode.



Propagating Cipher Block Chaining (PCBC) mode encryption

   [Picture by White Timberwolf, public domain]

   This encryption uses a block cipher of block size b. Let $\mathrm{Enc}_k(m)$ denote encryption of a single block m using this block cipher with key k and let $\mathrm{Dec}_k(c)$ denote decryption of a single block c using the block cipher with key k. Let IV be the initialization vector of length b, let $m_i, i = 0, 1, 2, \ldots$ be the b-bit strings holding the message and $c_i, i = 0, 1, 2, \ldots$ be the b-bit strings holding the ciphertexts.

(a) Describe how encryption and decryption of long messages work, i.e., write c_0 and a general c_i in terms of IV, m_0, m_i, and (if necessary) other m_j and c_j; and write m_0 and a general m_i in terms of IV, c_0, c_i, and (if necessary) other m_j and c_j. 5 points

(b) Assume that ciphertext c_j gets modified in transit. Show which messages get decrypted incorrectly and explain why others get decrypted correctly. 5 points

3. This problem is about RSA encryption.

   Let $p = 337$ and $q = 433$. Compute the public key using $e = 5$ and the corresponding private key.
   Reminder: The private exponent d is a positive number. 6 points

4. This problem is about the DH key exchange. The public parameters are that the group is $(\mathbb{F}_{907}^*, \cdot)$ and that it is generated by $g = 2$.

   (a) Compute the public key belonging to the secret key $b = 17$. 4 points

   (b) Alice's public key is $h_a = 418$. Using $b = 17$ from the previous part, compute the shared DH key with Alice. 8 points

5. The integer $p = 13$ is prime. You are the eavesdropper and know that Alice and Bob use the Diffie-Hellman key-exchange in $\mathbb{F}_{13}^*$ with generator $g = 2$. Alice's public key is $h_a = g^a = 7$. Use the Baby-Step Giant-Step method to compute Alice's private key a.
   Verify your result, i.e. compute g^a. 10 points

6. NSA cryptographer Rick Proto noticed independently of Diffie and Hellman that exponentiation in finite fields is easy while computing discrete logarithms can be hard. The scheme he came up with requires several interactions between the sender and receiver.

   The system parameter is a large prime p.

   To send message $m < p$ to user Bob, Alice chooses a random integer $A < p$ with $\gcd(A, p-1) = 1$ and computes $a \equiv A^{-1} \bmod (p-1)$. She then computes $c_1 \equiv m^A \bmod p$ and sends c_1 to Bob.

   Bob chooses a random integer $B < p$ with $\gcd(B, p-1) = 1$ and computes $b \equiv B^{-1} \bmod (p-1)$. Bob then computes $c_2 \equiv c_1^B \bmod p$ and sends c_2 to Alice.

   Alice then computes $c_3 \equiv c_2^a \bmod p$ and sends c_3 to Bob.

   Bob decrypts the message by computing $m' \equiv c_3^b \bmod p$.

   (a) Let $p = 23$. Execute one run of the protocol in which Alice wants to send $m = 2$ and picks $A = 5$ and computes $a = 5^{-1} \equiv 9 \bmod 22$, and Bob picks $B = 3$ and computes $b = 3^{-1} \equiv 15 \bmod 22$. Compute c_1, c_2, c_3, and m'. **8 points**

   (b) Explain why the system works, i.e., explain why $m' = m$. **5 points**

   (c) Attacker Eve observes the conversations between Alice and Bob and obtains c_1, c_2, and c_3; she also knows the system parameter p. Assume that Eve can compute discrete logarithms in $\mathbb{F}_p^*$. Show how she can obtain m from the observed ciphertexts.
   Note that you need to show where to find a DLP in this system and how to use the solution. **4 points**

7. This exercise is about a different variant of the Diffie-Hellman key exchange used until 2013 in the Telegram chat system. We describe the version working in $\mathbb{F}_p^*$ for some fixed, publicly known prime p. Let $g \in \mathbb{F}_p^*$ be a generator of a large group of prime order ℓ.

   Telegram users generate their public and private keys as in the regular Diffie-Hellman key exchange. For forward secrecy, keys are used only once.

   If Alice wants to communicate with Bob, she computes a fresh key pair and sends her public key $h_A = g^a$ to the server with the request that it should be passed to Bob.

   The server generates a nonce $n \in \mathbb{F}_p^*$ and passes h_A and n to Bob.

   Bob computes a fresh key pair, sends $h_B = g^b$ to the server and computes the shared key as $k_B = \text{hash}_1(h_A^b + n)$.

   The server passes h_B and n to Alice.

   Alice computes the shared key as $k_A = \text{hash}_1(h_B^a + n)$.

   To protect against man-in-the-middle attacks, Alice and Bob compute a second, somewhat shorter hash value (computed using hash function hash_2) and compare $\text{hash}_2(h_A^b + n) = \text{hash}_2(h_B^a + n)$, e.g. by calling each other.

   (a) Explain why the system works, i.e., explain why $k_A = k_B$ and why the comparison of the hash_2 values works. 6 points

   (b) The server has an interest in spying on its users. In the proper execution of the protocol the server cannot compute the shared secret without solving for a or b.

   Show how to mount a man-in-the-middle attack, i.e., show how the server can modify the protocol in a way that 1) $k_A = k_B$, 2) the server knows the shared key k_A, and 3) the modification is not caught by Alice and Bob comparing the hash_2 values.

   Hint: Note that the hash functions are secure, i.e. their inputs must be equal for the check to work and the keys to be equal. However the server can send different nonces n_B and n_A to Bob and Alice. 8 points