TECHNISCHE UNIVERSITEIT EINDHOVEN
Faculty of Mathematics and Computer Science
Introduction to Cryptology, Monday 8 April 2024

Name :

TU/e student number :

Exercise	1	2	3	4	5	6	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 6 exercises. You have from 18:00 – 21:00 to solve them. You can reach 100 points.
Make sure to justify your answers in detail and to give clear arguments. Document all steps and intermediate results, in particular of algorithms; it is not sufficient to state the correct result without the explanation and the steps that lead to it. 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

$$s_{i+6} = s_{i+5} + s_{i+1} + s_i.$$

(a) Draw the LFSR corresponding this sequence. 3 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.
Reminder: Factors may appear with multiplicity larger than one. 10 points

(c) Write the factorization of f from (b) in the form $f = \prod f_i^{e_i}$ with integers $e_i > 0$ and f_i different irreducible polynomials, i.e., group equal factors.
For each of the $f_i^{e_i}$ compute the order.
Reminder: We have shown in class that $f_i^{e_i}$ has order different from that of f_i (for $e_i > 1$) and you need to compute the order directly. We do not have a theorem for this. 7 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 6 bits appears exactly once.
Make sure to justify your answer and to check that all 2^6 states are covered. 13 points

2. FIDES is a cipher for authenticated encryption which permits to authenticate additional data which is not encrypted but only authenticated. FIDES is designed to be light weight and deliver 92-bits of security. FIDES requires the message and additional data each to be split into blocks of length 96 bits and internally uses a non-linear mixing function F which takes as input 192 bits and produces 192 bits of output. Assume for simplicity that both the additional data and the message have lengths that are multiples of 96. Let the additional data A be one block of 96 bits and let the message M be split into 96-bit blocks $M_0, \ldots, M_{\ell-1}$. Let k denote the 96-bit key shared by Alice and Bob and IV be a 96-bit initialization vector, picked fresh for each message M. Then $\mathsf{FIDESenc}(k, IV, A, M) = (C, T)$, where ciphertext C consists of ℓ blocks $C_0, \ldots, C_{\ell-1}$ of 96 bits each and a 96-bit authentication tag T so that $\mathsf{FIDESdec}(k, IV, A, C, T) = M$ and that the decryption function outputs "failure" if the tag T is not correct.

   Here is a schematic description of FIDES.

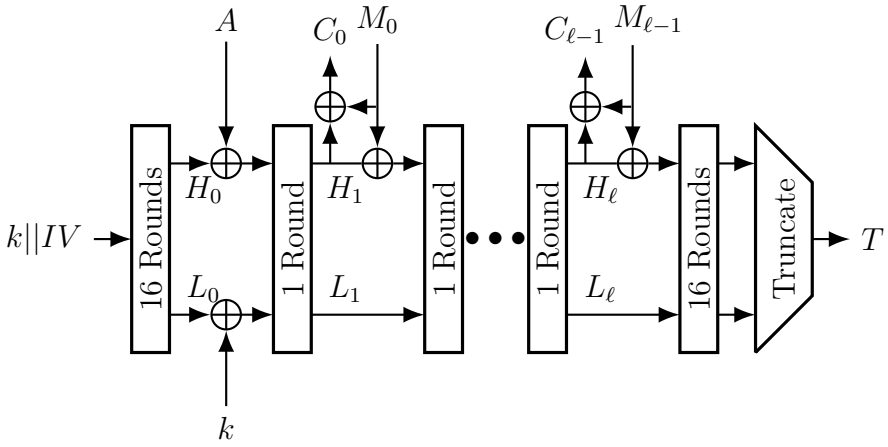

Image credit: adapted from Jérémy Jean.

In this schematic description, each of the horizontal arrows carries 96 bits and the number of rounds indicates how many times the non-linear mixing function F is run. The thick dots indicate that the intermediate blocks are handled the same way as M_0 and $M_{\ell-1}$.

Hinti for all subexercises: To structure your answer, it helps to use the high and low parts H_i and L_i indicated in the diagram. Make sure to describe how to compute them when you use them.

   (a) Describe how FIDES encryption of long messages works by writing

C_0 and a general C_i in terms of k, IV, A, M_0, M_i, and other M_j and C_j as necessary. 4 points

(b) Describe how decryption of long messages and verification of the authentication tag work by writing M_0 and a general M_i in terms of k, IV, A, C_0, C_i, and (if necessary) other M_j and C_j and describing how the receiver can check the authentication tag T. 5 points

(c) Assume that ciphertext C_j gets modified in transit. Show which message blocks get decrypted incorrectly and explain why others get decrypted correctly. Show how the authentication tag T catches this error. 5 points

(d) Assume that the additional data A gets modified in transit. Show which message blocks get decrypted incorrectly and explain why others get decrypted correctly. Show how the authentication tag T catches this error. 4 points

3. This problem is about RSA encryption. Let $p = 307$ and $q = 439$. Compute the public key using $e = 65537$ and the corresponding private key.
Reminder: The private exponent d is a positive number. 8 points

4. This problem is about the DH key exchange. The public parameters are the group G and generator g, where $G = (\mathbb{F}_{1031}^*, \cdot)$ and $g = 37$. Alice's public key is $h_A = 231$. Bob's private key is $b = 22$,
Compute the DH key that Bob shares with Alice. 8 points

5. The integer $p = 29$ is prime. You are the eavesdropper and know that Alice and Bob use the Diffie-Hellman key-exchange in $\mathbb{F}_{29}^*$ with generator $g = 3$. Alice's public key is $h_A = g^a = 16$. Use the Baby-Step Giant-Step method to compute Alice's private key a.
Verify your result, i.e. compute g^a. 11 points

6. Multi-prime RSA uses a modulus n that is a product of more than two primes, so $n = p_1 \cdot p_2 \cdots p_\ell$ for $p_i \neq p_j$ and $\ell > 2$. Like in regular RSA the public exponent is a positive integer e that is coprime to $\varphi(n) = (p_1 - 1) \cdot (p_2 - 1) \cdots (p_\ell - 1)$ and the private exponent is a positive integer $d < \varphi(n)$ with $d \equiv e^{-1} \bmod \varphi(n)$. The public key is (n, e) and the private key is (n, d). Encryption and decryption work like in regular RSA as $c \equiv m^e \bmod n$ and $m' \equiv c^d \bmod n$ for plaintext m and ciphertext c.

   (a) Show that the system correctly recovers the message, i.e., that $m = m'$. 3 points

   (b) The primes p_i still need to be very large to achieve security, but RSA-4096 can be built with 4 primes of 1024 bits each. An interesting feature of multi-prime RSA is that the receiver can speed up the computations by using the Chinese Remainder Theorem (CRT). Instead of computing $m' \equiv c^d \bmod n$ the receiver can compute $m'_i \equiv c_i^{d_i} \bmod p_i$ for $1 \leq i \leq \ell$ and then compute m' from the m'_i by using the CRT, where $c_i \equiv c \bmod p_i$, so that the base of exponentiation is smaller, speeding up all multiplications, and where $d_i \equiv d \bmod p_i - 1$, so that the exponents are also smaller. In the example of 4 primes, each d_i has a quarter of the bits of d and each c_i has a quarter of the bits of c.

   The private key of multi-prime RSA with CRT then is $(p_1, d_1, p_2, d_2, \ldots, p_\ell, d_\ell)$. The product n need not be included but is recomputed in the CRT computation. Users may choose to extend their keys with n and other precomputed values depending on the p_i that are otherwise computed in the CRT computation such as the values $p_i^{-1} \bmod N_i$, where $N_i = n/p_i$.

   Explain why multi-prime RSA with CRT works and how much faster decryption is with CRT compared to regular RSA.

   Note: Your explanation should include reasoning why d_i is taken modulo $p_i - 1$ and not modulo p_i.
   Hint: You may assume schoolbook multiplication where the product of two b-bit numbers takes time b^2. 8 points

   (c) Bob and Alice talk a lot and even with the CRT speedup Bob is getting tired of all the work of decrypting. Furthermore, Alice's messages are often very short, all much smaller than Bob's p_1. Bob is weary that the messages are too short but has also heard concerning things about padding schemes, so he comes up with his own variation.

While he does not fully trust Alice, he trusts her enough to reveal p_1 because $N_1 = n/p_1$ is still a large unfactored number and she would not learn $\varphi(n)$ from knowing just p_1. To send message $0 \neq m < p_1$, Alice should pick a random $0 < \bar{m} < N_1$, compute M with

$$\begin{aligned} M &\equiv m \bmod p_1 \\ M &\equiv \bar{m} \bmod N_1 \end{aligned}$$

and then send the encryption of M, i.e. $c \equiv M^e \bmod n$. To decrypt, Bob just needs to compute $m \; equivc_1^d \bmod p_1$. That way, Alice can send short messages repeatedly and because of the randomness in $\bar{m}$ the ciphertexts do not reveal that she is sending the same messages.

Eve has gotten suspicious of the many messages between Alice and Bob and knows Alice as somebody who does not have a lot to send, but trial encrypting small messages and comparing with the received ciphertexts does not find a match. However, in some lunch break she overhears Alice bragging about how much trust she gained with Bob and describing the above system (of course without revealing p_1).

Explain how Eve can use this information to compute p_1 from a single ciphertext in at most c steps, where c is an upper bound on the message m that Alice sent. You may assume that c is small enough that c^e can be computed as an integer

Hint: This question leaves open what a "step" is, that is what you need to find, but you can deduce from the question that this is a computation involving guesses for m. | 8 points |