Mastermath
Spring 2017
Exam Selected Areas in Cryptology
Tuesday, 13 June 2017

Name :

Student number and home university :

Exercise	1	2	3	4	5	total
points						

Notes: Please hand in this sheet at the end of the exam. You may keep the sheets with the exercises.
This exam consists of 5 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 requires usage of a particular algorithm other solutions will not be accepted even if they give the correct result.
All answers must be submitted on paper provided by the university; 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 allowed to use any books and notes, e.g. your homework. You are not allowed to use the textbooks of your colleagues.
You are allowed to use a calculator without networking abilities. Usage of laptops and cell phones is forbidden.

1. This exercise is about the NTRU encryption system. Remember that all computations take place in $R = \mathbb{Z}[x]/(x^n - 1)$ and are done modulo 3 or modulo q. The secret key consists of $f(x), g(x) \in R$, where f is invertible in $R_q = R/q$ and R_3, and f has exactly d_f coefficients equal to 1 and $d_f - 1$ coefficients equal to -1 for some integer d_f. Similarly, g has d_g coefficients equal to 1 and the same number equal to -1. The public key is $h = 3g/f$ in R_q.

   To encrypt $m \in R$ with coefficients in $[-1, 1]$ pick random, sparse $r \in R$ with d_r coefficients equal to -1 and the same number equal to 1. Then compute the ciphertext $c \equiv r \cdot h + m \bmod q$; move all coefficients to $[-q/2, q/2]$ to get a unique representative of c.

   To decrypt $c \in R_q$ compute $a = f \cdot c \bmod q$, again moving all coefficients to $[-q/2, q/2]$ (hence we use $=$ instead of $\equiv$) and compute $m = a/f \bmod 3$ with coefficients in $[-1, 1]$.

   (a) Let $N = 3, p = 3$, and $f(x) = x^2 - x + 1$. Compute the inverse f_p of f in R_3. Then compute $f \cdot f_p$ in R_3 to verify that the result is indeed 1.
   Hint: this needs a XGCD computation. Make sure to document the steps or state how you did this computation. Do *not* simply state the result or just a verification of the result.

   6 points

   (b) Let $d_f = 4$, $d_g = 2$ and $d_r = 4$ and $N = 32$. Explain how decryption errors can happen and compute how large q has to be so that decryption is guaranteed to be correct, i.e., so that taking the coefficients of $a = f \cdot c$ in R_q as elements in $[-(q-1)/2, (q-1)/2]$ produces the correct message.
   Note: The parameter choices are different than in the lecture to ensure that you go through all steps of the argument. Make sure to justify all statements.

   6 points

2. This exercise is about code-based cryptography.

   (a) The binary Hamming code $\mathcal{H}_3(2)$ has parity check matrix

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

   and parameters $[7, 4, 3]$.
   Correct the word $(0, 1, 1, 0, 0, 1, 0)$.

   2 points

(b) Let H be a parity-check matrix of an $[n, k, d]$ code and assume there exists a code word of Hamming weight d. Explain in your own words how the algorithm by Lee and Brickell finds a codeword of weight d given H. Assume that there is only a single such word.
State the steps performed and the probability of success for each step and how often it is repeated on average.
Note: This exercise is not specific to the H in the previous part of the exercise.
12 points

(c) Let G be the generator matrix of a code with parameters $[n, k, d]$ and let $d = 2t + 1$, i.e. d is odd. McEliece's original encryption system uses G to encrypt $m \in \mathbb{F}_2^k$ as $c = mG + e$, where e is a random element of $\mathbb{F}_2^n$ of Hamming weight exactly t.

For simplicity assume that the code is such that decoding $mG + e'$ with e' of weight $> t$ will return a different m' and assume that decryption uses some validity check so that m' would be caught as a decryption error.

Assume further that Alice will decrypt every ciphertext she receives and send back a notification of an error if decryption gives an invalid ciphertext. If decryption works she does not react. She does not check whether e has t or fewer errors and her decoder works for $\leq t$ errors.

Explain how Eve can use this behavior of Alice to decrypt ciphertext c that Bob sent to Alice, i.e., show how Eve can send somewhat modified ciphertexts to Alice to learn the correct mG and thus m from observing Alice's reactions.
State how many such queries Eve has to send to Alice to make your attack work.
8 points

3. This exercise is about differential cryptanalysis of the same toy cipher from the lectures. Using key $(k_1, k_2, k_3, k_4, k_5) \in (\{0,1\}^{16})^5$ it encrypts a plaintext $P = P_1||\ldots||P_{16} \in \{0,1\}^{16}$ as follows. Let S be the current state, we start with $S = P$. Rounds $i = 1, 2, 3$ perform key mixing

$$S \leftarrow S \oplus k_i,$$

substitution using a Sbox (Table 2)

$$S \leftarrow Sbox(S_1 \ldots S_4)||\ldots||Sbox(S_{12} \ldots S_{16}),$$

and finally applies permutation π_P (Table 1) on the state bits:

$$S \leftarrow S_{\pi_P(1)}||\ldots||S_{\pi_P(16)} = S_1||S_5||S_9||\ldots||S_{12}||S_{16}.$$

Round 4 applies key mixing with round key k_4, substitution using the sbox and finally applies another key mixing with round key k_5. After round 4, the cipher outputs the current state S as the ciphertext C.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\pi_P(i)$	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

Table 1: State bit permutation

In contrast to the lecture notes, we use the following SBox:

in	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
out	14	6	4	1	9	3	12	8	11	0	15	7	13	10	5	2

Note most significant bit is left most bit, so 12 represents '1100' in binary.

Table 2: Sbox

This SBox has the following Difference Distribution Table (Table 3:

		out														
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0																
	1		0	0	0												
	2		0	0	0												
	3			2	0	0	0	0	0	0	0	0	0	2	0	0	10
	4		0	0	0	0	4	2	2	2	2	4	0	0	0	0	0
	5			2	0	0	0	0	0	0	0	0	0	2	8	0	2
	6			8	0	0	0	0	0	0	0	0	0	0	4	4	0
	7			0	0	0	4	2	2	2	2	4	0	0	0	0	0
in	8		0	0	0	2	2	4	0	0	4	2	2	0	0	0	0
	9			0	4	0	0	0	0	0	0	0	0	0	4	8	0
	10			2	0	0	0	0	0	0	0	0	0	2	0	0	2
	11			0	0	2	2	4	0	0	4	2	2	0	0	0	0
	12			2	8	0	0	0	0	0	0	0	0	2	0	0	2
	13			0	0	4	0	2	2	2	2	0	4	0	0	0	0
	14			0	0	4	0	2	2	2	2	0	4	0	0	0	0
	15			0	4	0	0	0	0	0	0	0	0	8	0	4	0

Table 3: Sbox difference distribution table

(a) Complete the DDT. You only have to write down the missing numbers in a table. Do not just enter the numbers in this table, but make a copy on the exam paper. 6 points

(b) Construct a differential for this cipher over the first three rounds with only one active SBox in the third round and compute its estimated probability. 8 points

(c) Consider the boomerang with input plaintext difference

$$\Delta P = (0000\ 0110\ 0000\ 0000)$$

and output ciphertext difference

$$\Delta C = (0000\ 0000\ 1111\ 0000),$$

then a quartet $(P^{(1)}, P^{(2)}, P^{(3)}, P^{(4)})$ satisfies this boomerang if

$$P^{(1)} \oplus P^{(2)} = \Delta P, \quad P^{(3)} \oplus P^{(4)} = \Delta P, \quad \text{and}$$

$$C^{(1)} \oplus C^{(3)} = \Delta P, \quad C^{(2)} \oplus C^{(4)} = \Delta C.$$

Compute the total success probability of finding such quartets over all round 1 & 2 differentials with the given ΔP and all round 3 & 4 differentials with the given ΔC. (Hint: in round 2 each Sbox has either input

difference 0 or 4 (0100), so every *active* round 2 Sbox contributes a term $4 \times (2/16)^2 + 2 \times (4/16)^2$. Likewise, in round 3 each active Sbox has output difference 2 (0010).) | 8 points

(d) Consider all 3-round differentials that have only 1 active Sbox in round 1 and where in round 3 only the third and/or fourth Sbox (S_{33} and/or S_{34}) may be active. Prove that all such 3-round differentials are impossible differentials, i.e., they have probability 0. (Hint: the only allowed output differences for active round 2 Sboxes are 0, 1, 2, 3 (0000, 0001, 0010, 0011).) | 8 points

4. This exercise is about the Merkle-Hellman scheme, explained in the following. The scheme is a public-key scheme in which both the secret and public key contain a sequence of ℓ different numbers and the secret key also includes information on how to get from the secret sequence to the public one. The following describes key generation followed by steps to encrypt and to decrypt a message.

Key generation: Pick ℓ integers $a_0, a_1, \ldots, a_{\ell-1}$ such that $a_0 > 0$ and $a_i > \sum_{j=0}^{i-1} a_j$ for $i > 0$. The a_i form a so-called *super increasing sequence*. Pick a positive integer M, the modulus, with $M > \sum_{i=0}^{\ell-1} a_i$ and a positive integer $q < M$ with $\gcd(M, q) = 1$. Compute $b_i \equiv a_i \cdot q \bmod M$, for $0 \leq i < \ell$, and represent b_i by an integer in $[0, M-1]$. Sort the b_i by size and remember the permutation P from the original order to the sorted sequence $(c_0, c_1, \ldots, c_{\ell-1})$ with $c_i = b_{P(j)}$ for some j.

The secret key is $((a_0, a_1, \ldots, a_{\ell-1}), M, q, P)$. The public key is $(c_0, c_1, \ldots, c_{\ell-1})$.

Encryption: To encrypt a number $0 \leq n < 2^\ell$ compute the binary representation of $n = \sum_{i=0}^{\ell-1} n_i 2^i$ with $n_i \in \{0, 1\}$ and compute the ciphertext $N = \sum_{i=0}^{\ell-1} n_i c_i$.

Decryption: To decrypt N compute $K \equiv N/q \bmod M$, using one modular inversion (which could be precomputed) and a multiplication modulo M. Then determine which a_j of the private key were summed up to reach K. Apply P to learn which c_i were included and thus the n_i. Compute the plaintext $n = \sum_{i=0}^{\ell-1} n_i 2^i$.

(a) Encrypt message $n = 12$ to a user with public key $(3, 9, 29, 31)$. | 2 points

(b) Compute the public key for secret key starting with $\ell = 4$,
$(a_0, a_1, a_2, a_3) = (2, 3, 7, 13), M = 29, q = 11$.
Compute the permutation P.
Decrypt ciphertext $N = 31$ sent to this public key. | 4 points

(c) Explain why the system works, i.e., why does decryption recover the plaintext and why decryption can be computed efficiently. | 6 points

(d) Explain what problem the attacker faces and write explicitly what system of equations the attacker would need to solve to find the secret key for the public key used in part 4a.
Note: You are not expected to break the system. | 4 points

5. This exercise is about password recovery. Let $h : \{0,1\}^* \to \{0,1\}^{512}$ be a fixed 512-bit hash function. A website stores for each user a username string u and a 512-bit hash $a = h(p)$ of the user's password string p.

   (a) Let $\mathcal{P}$ be the set of all alphabetic (i.e., 'a...z,A...Z') passwords of length 10. Compute the size of the set $\mathcal{P}$. 2 points

   (b) Explain how one can construct an efficient map $f : \{0,1\}^{512} \to \mathcal{P}$ from the hash space to the password space. It has to be approximately balanced, i.e., preimage sizes have to be approximately equal: $|f^{-1}(p_1)| \approx |f^{-1}(p_2)|$ for all $p_1, p_2 \in \mathcal{P}$. 6 points

   (c) Explain how to apply Hellman's time-memory trade-off attack to h to recover passwords from the given password space $\mathcal{P}$ with success probability about 0.8. 6 points

   (d) Assume an attacker can use a single high-end GPU for this attack that can compute 2^{31} evaluations of $f \circ h$ per second. Estimate the offline and online runtime complexity in wall clock time (days, hours, seconds) for this attack using this single high-end GPU as well as the storage requirements. Disregard the effect from 'false alarms' and assume RAM and GPU memory size are not an issue. 6 points