Mastermath course: Cryptology, Spring 2015
Examples of exercises

Here is what the exam will say as a foreword. This set of exercises is much more than 3h worth of work.
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.
Do not write in red or with a pencil.
You are allowed to use any books and notes. 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. Exercises on code-based crypto:

   (a) The binary Hamming code $\mathcal{H}_4(2)$ has parity check matrix

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

   and parameters $[15, 11, 3]$.
   Correct the word $(0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1)$.

   (b) State the parameters of a Goppa code of length $n = 2^{20}$ using an irreducible polynomial of degree $s = 70$.

   (c) Let C be a binary code, given by parity-check matrix H. Stern's method searches for low-weight words in C. Assume that we know that there exists a word of weight t. In the algorithm we shuffle the columns of H and turn the rightmost $n - k$ vectors into an identity matrix.
   In one of the following steps we search for partial collisions, i.e., collisions on ℓ positions, in two column vectors. One vector is the sum of p columns in X and the other vector is the sum of p columns in Y, where X and Y form a partition of those k columns of H which are not part of the identity matrix. Compute the probability that the sum of these $2p$ vectors of length $n - k$ has value 0 in those (randomly chosen) ℓ positions and compute the conditional probability that this sum has Hamming weight $t - 2p$, given that those ℓ positions are all 0.

   (d) Make sure to read how Stern's attack (and the simpler ones) work. Try a small example.

(e) Make sure to understand why the school-book version of McEliece is not a good idea and that we need to have messages with some prescribed format. (Remember the attack in which you ask for decryption of $c + e_i$, for e_i the i-th unit vector.)

2. This exercise is about hash-based signatures.

   (a) Take Lamport's one-time signature scheme. Let messages be of length n and assume that Alice has published $2n$ hash results as her public key and knows $2n$ secret strings, which are her private keys, which lead to those $2n$ hash results. Alice uses this signature system multiple times with the same key. Analyze the following two scenarios for your chances of faking a signature on m: 1. You get to see signatures on random messages. 2. You get to specify messages that Alice signs. You may not ask Alice to sign m in the second scenario. How many signatures do you need on average in order to construct a signature on m? How many signatures do you need on average to be able to sign any message? Answer these questions in both scenarios.

   (b) The Winternitz one-time signature scheme is another one-time signature scheme. The advantage is that instead of having two hash values per bit, we sign strings of k bits at once: Starting from a secret value v_0 we compute $v_1 = H(v_0), v_2 = H(v_1) = H^2(v_0), \dots, v_{2^k-1} = H^{2^k-1}(v_0) = p$. We publish p as public key and keep v_0 as secret key. To sign a message of k bits, we take this message as an integer m in $[0, 2^k - 1]$ (just read the bits as binary expansion of a number) and reveal v_m. The verifier can then check that $H^{2^k-m-1}(v_m) = p$. Note that so far this system could be abused: knowing v_m means knowing v_{m+1}, v_{m+2}, dots., so we need some protection against this. For that we add another signature which signs $2^k - m - 1$ using the same system.

   To sign a message $(m_0, m_1, \dots, m_{j-1})$, $m_i \in [0, 2^k - 1], i \in [0, j - 1]$, of kj bits, use j such signatures with public values $p_0, p_1, \dots p_{j-1}$, compute $\sigma = \sum_{i=0}^{j-1}(2^k - m_i - 1)$ and sign σ using the same method (using $\lceil \log_2(\sigma)/k \rceil$ more signatures).

   Check out `http://www.cdc.informatik.tu-darmstadt.de/~dahmen/papers/hashbasedcrypto.pdf` for a detailed description.

   Let $k = 4$ and $j = 16$. Draw the dataflow and compute how many extra signatures (beyond the j) are needed for the checksum signatures.

   (c) Read up on how to design few-times signatures from one-time signatures.

3. This exercise is about the NTRU encryption system. The system has three general public parameters: namely positive integers N, p, and q, where $\gcd(p, q) = 1$ and q is much larger than p. For this exercise we use $p = 3, q = 101$, and $N = 7$.

   All computations take place in $R = \mathbb{Z}[x]/(x^N - 1)$, i.e. all elements are represented by polynomials of degree $< N$. Some computations additionally reduce modulo p or modulo q.

The private key of user Alice is a polynomial $f(x) \in R$ which satisfies that f is invertible in $R/p = (\mathbb{Z}/p)[x]/(x^N - 1)$ and in $R/q = (\mathbb{Z}/q)[x]/(x^N - 1)$.

To generate her public key, Alice picks a polynomial $g(x) \in R$ and computes $f_p = f^{-1}$ in R/p, $f_q = f^{-1}$ in R/q and $h = f_q \cdot g$ in R/q. Alice's public key is h along with the public parameters p, q, and N.

To encrypt message $m(x) \in R$ (with coefficients in $[-(p-1)/2, (p-1)/2]$) to a user with public key h take a random polynomial $\phi(x) \in R$ and compute $c = p \cdot \phi \cdot h + m$ in R/q.

To decrypt ciphertext $c \in R/q$ use private key f and compute $a = f \cdot c$ in R/q, choosing coefficients in $[-(q-1)/2, (q-1)/2]$. [If you're a mathematician, lift a to R, i.e. forget about the reduction modulo q]. Then compute $m' = a \cdot f_p$ in R/p, taking coefficients from $[-(p-1)/2, (p-1)/2]$.

(a) Let $f(x) = x^6 - x^3 + x \in R$.
Compute $(91x^6 + 35x^5 + 52x^4 + 28x^3 + 42x^2 + 63x + 94) \cdot (x^6 - x^3 + x)$ in R/q to verify that $f_q = 91x^6 + 35x^5 + 52x^4 + 28x^3 + 42x^2 + 63x + 94$.

(b) Compute the inverse of $f = x^6 - x^3 + x$ in R/p.
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.

(c) Your secret key is $f = x^6 - x^3 + x$; you have computed
$f_p = -x^6 + x^5 + x^4 - x^3 + 1$ in setting up your key.
Somebody sends you ciphertext $c = 6x^5 + 4x^4 + 3x^3 + 92x + 99$.
Compute m'.

(d) Show that the system correctly recovers the message, i.e. $m = m'$ if f, g, and ϕ are very sparse and small, i.e. have very few non-zero coefficients chosen from $\{-1, 1\}$ and m has coefficients in $[-(p-1)/2, (p-1)/2]$.
For the parameters given here, g and ϕ each have one coefficient equal to 1 and one equal to -1.

4. The GGH encryption system (see e.g. the slide deck for lecture 1 by Thijs Laarhoven) looks very similar to the McEliece cryptosystem. Explain the differences.

See also `http://larc.usp.br/~pbarreto/PQC-3.pdf` for Peikert's scheme and relations.