KTH Computer Science and Communication

Homework III, Foundations of Cryptography 2014

Before you start:

1. The deadlines in this course are strict. This homework set is due as specified at `http://www.csc.kth.se/DD2448/krypto14/deadlines` and is April 28 at 15.00.
2. Read the detailed homework rules at `http://www.csc.kth.se/DD2448/krypto14/rules`.
3. Read about I and T-points, and how these translate into grades, in the course description at `http://www.csc.kth.se/DD2448/krypto14/handouts/course_description.pdf`.
4. Note that in problems with subproblem, the first number given is the total number of points for the problem and later there is information how this total is distributed over the subproblems.

The problems are given in no particular order. If something seems wrong, then visit `http://www.csc.kth.se/DD2448/krypto14/handouts` to see if any errata was posted. If this does not help, then email `johanh@csc.kth.se`. Don't forget to prefix your email subject with `Krypto14`.

1 (6I+4T) Implement Schnorr's signature scheme. Choose the parameter sizes so that you expect them to withstand attacks for at least 20 years from attackers willing to spend a total of 10^7 USD to buy hardware. A sound motivation for your parameters is worth 4T. Choose all parameters needed (public and private for one user) and sign the message "Send more money" for an additional 6I.

Your code should be handed in as an attachment to an email to sangxia@csc.kth.se with subject line "krypto14-sch". The attachment should consist of a single file named ''`firstname_lastname.tar.gz`'', such that the command ''`tar xvfz firstname_lastname.tar.gz`'' produces a single directory named ''`firstname_lastname`'' containing your files. If you write your solution in C/C++, the directory must contain a Makefile such that `make` builds your program. Regardless of the programming language used, your directory must contain a file `README` containing a brief description of how your program can be invoked, including a few concrete examples that execute in a reasonable amount of time. Your code should obviously be structured and documented. You may use pre-made subroutines as discussed in the heading of this problem set.

2 (10T) A oracle mapping $2n$ bits to n bits is simply a black-box function that for each $2n$ bit input string returns n bits. In a random oracle, h for each $x \in \{0,1\}^{2n}$ the value $h(x)$ is picked in $\{0,1\}^n$ with uniform probability (and independently of all other answers) and fixed. Please prove lower bounds (in the form of number of calls to the black-box) for the problems below. If possible also give (almost) matching upper bounds. The probability of success needs to be analyzed in terms over "a random h".

2a (5T) To find a collision, i.e. to find $x_1 \neq x_2$ such that $h(x_1) = h(x_2)$.

2b (5T) To break the following symmetric cryptosystem. The secret key is given by $K \in \{0,1\}^{2n}$ and the clear-text is divided into blocks of length n, P_i, $1 \leq i \leq L$ and the i'th cipher block is defined by

$$C_i = P_i \oplus h(K+i)$$

where $\oplus$ is bit-wise exclusive-or. In the inputs to h, we identify bit-strings of length $2n$ and integers in the range 0 to $2^{2n}-1$. The addition is defined modulo 2^{2n}. By "break" we here mean to find any information about the clear text apart from its length.

3 (5I+4T) Let us consider elliptic curves.

3a (4T) Study the curve defined by

$$y^2 \equiv x^3 + 2x + 6 \bmod 7$$

by hand. Calculate all points and give a complete addition table.

3b (5I) Let p_0 be a six digit number giving your date of birth in the order YYMMDD and let p be the smallest prime, $p \geq p_0$ and let q be the smallest prime $q > p$. Find an elliptic on the form

$$y^2 \equiv x^3 + Ax + B \bmod p$$

such that the corresponding elliptic curve group has order q. Find a point on the curve that generates the full group. Motivate your answer why it generates the full group. If you cannot find a curve with order q try to get as close as possible and describe your efforts.

Hint: You might simply try random A and B and do not forget the point at infinity when counting points.

4 (12T) Suppose n is a product of two primes, and x is a number that is relatively prime with n, i.e. $\gcd(x,n) = 1$. Consider the following interactive protocol for proving that x is a non-square modulo n, i.e. that the equation $y^2 \equiv x \bmod n$ is not solvable.

1. The verifier picks a random bit b and a random number r, $1 \leq r \leq n$ that is relatively prime with n. If $b = 0$ then the verifier sends $r^2 \bmod n$ to the prover and if $b = 1$ it sends $xr^2 \bmod n$.
2. The prover responds with a bit b'.
3. The verifier accepts iff $b' = b$.

Prove that this protocol is complete and sound. To prove that it is complete you should prove that if x is a non-square then an all powerful prover can convince a verifier of this fact. Prove that it is efficient in that if the prover knows the factorization of n then it can be implemented efficiently. To prove that it is sound you need to prove that a prover cannot fool an honest verifier too often. To be more precise prove that if x is a square modulo n then, for any strategy of the prover, the probability that a verifier following the protocol accepts is bounded by $1/2$.

Finally discuss whether this protocol has the zero-knowledge property. Argue that if the verifier follows the protocol then in fact it learns nothing new. Discuss what can be said if the verifier deviates from the protocol.

5 (10I) Implement the SHA-256 hash function (5I). A detailed description is found on Kattis. `https://kth.kattis.scrool.se/problems/sha256`. Feel free to read from different sources on how to make an efficient implementation, but any borrowed ideas should be explained briefly in the solutions submitted on paper. You must also be prepared to explain in detail what you did and why at the oral exam. Make sure that your code is commented and well structured. Up to 5I points may be subtracted if this is not the case.

5a (5I) Use your implementation to find $x \neq x'$ under SHA-256 such that the k initial bits of SHA-256(x) and SHA-256(x') agree for as large value of k as possible. Your score on this problem is $\min(5, \max(k/2 - 20, 0))$. An extra 5I is given to the student with the highest value of k (divided into equal shares in case of a tie). You can only solve this subproblem if implementation passed the first part of this problem. Please provide a description of your approach and discuss what problems encountered.