KTH Computer Science
and Communication

Homework IV, Foundations of Cryptography 2016

Before you start:

1. The deadlines in this course are strict. This homework set is due as specified at `https://www.kth.se/social/course/DD2448/subgroup/vt-2016-krypto16/page/deadlines-16`.
2. Read the detailed homework rules at `https://www.kth.se/social/files/5686fcd8f276542387729c18/solution_rules.pdf`.
3. Read about I and T-points, and how these translate into grades, in the course description at `https://www.kth.se/social/files/5692df7bf2765405aca1825f/course_description.pdf`.
4. You may only submit solutions for a nominal value of 25 points in total (summing I and T points). The total number of points below may be larger and this should be interpreted as giving you a way to choose problems you like.

The problems are given in no particular order. If something seems wrong, then visit `https://www.kth.se/social/course/DD2448/subgroup/vt-2016-krypto16/page/handouts-10` to see if any errata was posted. If this does not help, then email `dog@kth.se`. Don't forget to prefix your email subject with `Krypto16`.

We may publish hints on the homepage as well if a problem appears to be harder than expected.

1 Read about the *Dual Elliptic Curve Deterministic Random Bit Generator* proposed by NIST in the original document `http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf`. Read other sources you find online as well.

1a (1T) Briefly summarize the controversy regarding this PRG.

1b (1T) Why do you think it was obvious to most researchers that something was not right with this construction even before the backdoor was made public?

1c (2T) More generally it is worthwhile to consider how a good elliptic curve is chosen. What is the purpose of the *million dollar curve* and what is special about how it is generated?

2 You are given a non-singular elliptic curve over a prime order field $\mathbb{Z}_q$ on Weierstrass normal form, i.e., $E : y^2 = f(x)$, where $f(x) = x^3 + ax + b$.

2a (2T) Construct an efficiently computable invertible injection $\{0,1\}^k \to E$, i.e., describe: (1) an algorithm Encode that takes a bitstring as input and outputs an element in the curve, and (2) an algorithm Decode that takes a group element and outputs a bitstring.

2b (2T) Prove that your construction satisfies $\mathsf{Decode}(\mathsf{Encode}(m)) = m$ for all $m \in \{0,1\}^k$ and explain how big you can make k relative to q.

2c (1T) It turns out that you may need to allow your encoding algorithm to be probabilistic, so it suffices to prove (under reasonable assumptions) that the *expected* running time (over the randomness of your algorithms) for any fixed input is bounded by a polynomial in $\log q$. What is the polynomial?

3 (4T) You are given a pseudo-random function $F_n = \{f_{n,\gamma}\}_{\gamma\in\Gamma_n}$, where $n \in \mathbb{N}$ is the security parameter and Γ_n is a set of possible keys for the security parameter n. Suppose that $f_{n,\gamma} : \{0,1\}^n \to \{0,1\}^{\log n}$ for every $\gamma \in \Gamma_n$. Can you construct a pseudorandom function $F'_n = \{f'_{n,\gamma}\}_{\gamma\in\Gamma'_n}$ such that $f'_{n,\gamma} : \{0,1\}^n \to \{0,1\}^n$? Prove that it works in that case, or explain informally why you think it is not possible if you think it is not possible.

4 (2T) Consider SHA-256 as a random oracle. What would you do if you needed a function in practice that you could consider to be (almost) a random oracle $\{0,1\}^* \to \{0,1\}^{3000}$? What is the collision resistance of your function?

5

5a (1T) Investigate how randomness for cryptographic use is generated for software written in JavaScript in at least two open source browsers, and: (1) include a link to the code that does this, (2) explain briefly the cryptographic construction, and (3) write a minimal example of how to use it. (It does not have to be executable code, a snippet suffices.)

5b (1T) Investigate how randomness for cryptographic use is generated for software written in OracleJDK, and: (1) include a link to the code that does this, (2) explain briefly the cryptographic construction, and (3) write a minimal example of how to use it. (It does not have to be executable code, a snippet suffices.)

Rigorous proofs

The following was covered in class so your task is to give *rigorous* proofs, i.e., the expectation of the quality of your solution is higher than for other solutions.

6 (4T) You are given a pseudo-random generator such that $\mathsf{PRG} : \{0,1\}^n \to \{0,1\}^{n+1}$ for every security parameter $n \in \mathbb{N}$. Construct a pseudo-random function PRG' such that $\mathsf{PRG}' : \{0,1\}^n \to \{0,1\}^{2n}$ for every $n \in \mathbb{N}$, and prove that it is a pseudo-random generator.

7 (4T) You are given a pseudo-random generator such that $\mathsf{PRG} : \{0,1\}^n \to \{0,1\}^{2n}$ for every security parameter $n \in \mathbb{N}$. Construct a pseudo-random function $F_n = \{f_{n,\gamma}\}_{\gamma\in\Gamma_n}$ such that $f_{n,\gamma} : \{0,1\}^{\log n} \to \{0,1\}^n$, where $n \in \mathbb{N}$ is the security parameter and Γ_n is a set of possible keys for the security parameter n, and prove that it is a pseudo-random function.