KTH Computer Science and Communication

Homework II, Foundations of Cryptography 2017

Before you start:

1. The deadlines in this course are strict and stated on the course homepage.
2. Read the detailed homework rules at the course homepage.
3. Read about I and T-points and grades in the course description.

The problems are given in no particular order. If something seems wrong, then visit the course homepage 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 `Krypto17`. We may publish hints on the homepage as if a problem appears to be harder than expected.

IMPORTANT! Use java.math.BigInteger for implementation exercises instead of GMP to avoid problems with Kattis

Definition 1 *The RSA assumption states that if $N = pq$, where p and q are randomly chosen primes with the same number of bits, $e \in \mathbb{Z}^*_{\phi(N)}$, $e > 1$, and g is randomly chosen in $\mathbb{Z}^*_N$, then for every polynomial time algorithm A, $\Pr[A(N, e, g) = \beta \wedge \beta^e = g \bmod N]$ is negligible.*

Definition 2 *The Strong RSA assumption states that if $N = pq$, where p and q are randomly chosen primes with the same number of bits and g is randomly chosen in $\mathbb{Z}^*_N$, then for every polynomial time algorithm A, $\Pr[A(N, g) = (e, \beta) \wedge \beta^e = g \bmod N \wedge e > 1]$ is negligible.*

1 This is an old problem. Be the first to solve it completely and correctly!

In class we considered the RSA signature scheme, i.e., RSA with full domain hash. In this problem we develop a different scheme based on the strong RSA assumption. Our construction is similar to some efficient *provably secure* signature schemes, but we only consider a simplified scheme and analyze its security in the random oracle model.

The private key of our scheme consists of two random $n/2$-bit safe[1] primes p and q. The public key consists of the modulus $N = pq$ and a random element g from the subgroup QR_N of quadratic residues in $\mathbb{Z}_N$. Suppose that $H : \{0,1\}^* \to \mathcal{P} \cap \{0,1\}^{n/3}$ is a random oracle, where $\mathcal{P}$ denotes the set of odd primes. A signature s of a message m is computed as $s = g^{1/H(m)} \bmod N$, where $1/H(m)$ should be understood as $H(m)^{-1} \bmod \frac{1}{2}(p-1)(q-1)$. To verify a signature s, one simply checks that $s^{H(m)} \bmod N = g$.

1a (1T) For a standard RSA modulus we do not require that p and q are safe. Prove that this does not change the hardness of factoring N in any essential way. Hint: Use the prime number theorem to estimate heuristically the probability that a randomly chosen prime is safe by chance.

[1] A prime p is safe if $(p-1)/2$ is prime as well.

1b (1T) Prove that if the strong RSA assumption holds, then the standard RSA assumption holds. (The opposite direction is unknown.)

1c (1T) Prove that the signature scheme is correct for every message m.

1d (1T) Let $p_1, \ldots, p_k \in \mathcal{P} \cap \{0,1\}^{n/3}$ be primes and let $g' \in \mathsf{QR}_N$ be randomly chosen. Prove that if we define $g = (g')^{\prod_{i=1}^{k} p_i} \bmod N$, then g is randomly distributed in QR_N. Thus, given a random element g' we can construct another random element g of which we can take any p_ith root modulo N.

1e (1T) Suppose that there exists a polynomial time algorithm A such that for random keys $(pk, sk) = ((N, g), (p, q))$ and random H,

$$\Pr[A^{\mathsf{Sign}_{sk}(\cdot), H(\cdot)}(pk) = (m, s) \wedge \mathsf{Verify}_{pk}(m, s) = 1 \wedge \forall i : m_i \neq m] \geq \delta \ ,$$

where m_i is the ith query to the signature oracle $\mathsf{Sign}_{sk}(\cdot)$ and δ is non-negligible, i.e., A breaks the signature scheme. (In the literature the random oracle is often implicit. Here we make it explicit.)

Prove that without loss of generality we may assume that A never asks the same query twice and that it always evaluates the random oracle H on the message m of its output.

1f (1T) Use the above to prove that given a random RSA modulus N and a random element $g' \in \mathsf{QR}_N$ you can generate a public key $pk = (N, g)$ such that you can simulate (without the secret key sk) a signature oracle $\mathsf{Sign}'(\cdot)$ and a random oracle $H(\cdot)$ such that

$$\Pr[A^{\mathsf{Sign}'(\cdot), H(\cdot)}(pk) = (m, s) \wedge \mathsf{Verify}_{pk}(m, s) = 1 \wedge \forall i : m_i \neq m] \geq \delta - \epsilon \ ,$$

where m_i is the ith query to the "signature oracle" $\mathsf{Sign}'(\cdot)$ and ϵ is exponentially small.

1g (1T) Prove that if A has polynomial running time $T(n)$ and j is randomly chosen in $\{1, 2, \ldots, Q\}$, $Q \leq T(n)$, where Q is the number of queries made by A to $H(\cdot)$, then

$$\begin{aligned}&\Pr[A^{\mathsf{Sign}'(\cdot), H(\cdot)}(pk) = (m, s) \wedge \mathsf{Verify}_{pk}(m, s) = 1 \wedge \forall i : m_i \neq m \wedge m = m'_j] \\ &\geq \delta / T(n) - \epsilon'\end{aligned}$$

where m_i is the ith query to the signature oracle and m'_j is the jth query to the random oracle, and ϵ' is exponentially small.

1h (2T) Let N be an RSA modulus, let $g' \in \mathsf{QR}_N$, and define $g = (g')^{\prod_{i \neq j} p_i} \bmod N$. Prove that if (m, s) satisfies $\mathsf{Verify}_{pk}(m, s) = 1$ and $H(m) = p_j$, then we can find integers a and b such that $ap_j + b \prod_{i \neq j} p_i = 1$ and construct (β, ρ) such that $\beta^\rho \bmod N = g'$ and $\rho > 2$.

1i (3T) Use the above observatations to describe an algorithm A' that runs A as a subroutine and breaks the strong RSA assumption, i.e., A' takes an RSA modulus N and a random element $g' \in \mathsf{QR}_N$ as input and must use A to output (β, ρ) such that $\beta^\rho \bmod N = g'$ and $\rho > 2$ with non-negligible probability.

1j (2T) Suppose that we only wish to sign a polynomial $h(n)$ number of distinct messages known in advance (we can think of the messages as the integers $1, \ldots, h(n)$). Can you modify the signature scheme for this setting and prove its security without the random oracle?

2 In this problem we investigate some implications of using different groups.

2a (2T) Let q be an n bit prime and let A_q denote the additive group modulo q, i.e., $\mathbb{Z}_q$ as an additive group. Present an algorithm for solve discrete logarithms in A_q efficiently and explain why the running time of your algorithm implies that A_q is unsuitable for cryptographic applications.

2b (3T) In class we briefly discussed how the running times of discrete logarithm algorithms in multiplicative groups and elliptic curve groups influence how large security parameters must be.

Consider the following two parameterized types of groups:

1. Multiplicative group with a prime modulus p and prime order q subgroup G_q of $\mathbb{Z}_p^*$.
2. Elliptic curve group E_r of prime order r over a prime order field $\mathbb{Z}_s$ where $\log s = c \log r$ for a small constant c. (Assume that the curve has been chosen to avoid weak curves.)

State how big $\log p$, $\log q$, $\log r$, and $\log s$ must be to give a (conjectured) security level of 1024 against classical computers (not quantum computers).

Hint: Search for literature about recommended key sizes.

2c (2T) Describe why neither G_q nor E_r is suitable for cryptographic use if the adversary has access to a quantum computer with arbitrarily large entangled state.

2d (3T) It turns out that $\log q$ can be significantly smaller than $\log p$ without making discrete logarithms easier to solve in G_q. Explain why it is difficult to base the El Gamal cryptosystem on such groups and what you can do about it if you do not need the homomorphic property.

2e (3T) Suppose that $p = 2q + 1$ and we use the El Gamal cryptosystem defined over G_q. Suppose that k honest parties wish to encrypt messages $m_i \in \{0,1\}^{(\log q)/(2k)}$ using the El Gamal cryptosystem under a joint public key $pk = (g, y)$. Construct efficiently computable/invertible injections $\iota_i : \{0,1\}^{(\log q)/(2k)} \to G_q$ such that

$$\prod_{i=1}^{k} \mathsf{Enc}_{pk}(\iota_i(m_i)) = \mathsf{Enc}_{pk}\left(\prod_{i=1}^{k} \iota_i(m_i)\right) \ .$$

This is interesting, since it allows compacting multiple ciphertexts into one and later separate the plaintext submitted by different senders after decryption.

3 (4T) Read about Lamport's basic one-time signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ (where a message is signed as is) and consider the following modified scheme $(\mathsf{Gen}, \mathsf{Sign}', \mathsf{Verify}')$ defined by $\mathsf{Sign}'_{sk}(m) = \mathsf{Sign}_{sk}(\mathsf{SHA256}(m))$ and $\mathsf{Verify}_{pk}(\sigma, m) = \mathsf{Verify}_{pk}(\sigma, \mathsf{SHA256}(m))$.

Is this scheme secure against existential forgery attacks? (definition of security covered in class) If so, then prove it. If not, then prove that it is not.

4 (12I) Implement the arithmetic of an elliptic curve. A detailed description is found on Kattis. `https://kth.kattis.com/problems/kth.krypto.ellipticcurvearithm`. Make sure that your code is commented and well structured. Up to 12I points may be subtracted if this is not the case. Keep in mind that you must be able to explain your solution during the oral exam.

5 (7I) Implement the recovery phase of Feldman's verifiable secret sharing scheme. A detailed description is found on Kattis. `https://kth.kattis.com/problems/kth.krypto.feldman`. Make sure that your code is commented and well structured. Up to 7I points may be subtracted if this is not the case. Keep in mind that you must be able to explain your solution during the oral exam.

6 (10I) Implement the SHA-256 hash function. A detailed description is found on Kattis. `https://kth.kattis.com/problems/kth.krypto.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 10I points may be subtracted if this is not the case.

7 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.

7a (1T) Briefly summarize the controversy regarding this PRG.

7b (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?

7c (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?

8 (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.

9 (3T) Consider SHA-256 as a random oracle. What would you do if you needed a function in practice that you could consider to be a random oracle $\{0,1\}^* \to \{0,1\}^{3000}$? What is the collision resistance of your function?

10 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 significantly higher than for other solutions.

10a (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.

10b (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.

To ensure that your solution is a simplified version of the general construction we require that PRG is stateless and applied at most $O(\log n)$ times for any input.