Introduction to Modern Cryptography
Exercise Sheet #7

University of Amsterdam, Master of Logic, 2014
Lecturer: Christian Schaffner
TA: Malvin Gattinger

Monday, 13 October 2014
(to be handed in by Wednesday, 29 October 2014, 14:00)

Homework

1. **Key-Update:** A company is using the public RSA key (N, e). During a security update, the key is updated to (N, e'), i.e. the modulus N remains the same, but the exponent e is changed. However, it is made sure that e and e' are coprime, i.e. that $\gcd(e, e') = 1$. 10 p.

   A client is sending a message m encrypted under the old public key to the company. After getting notified of the update, he sends the same message again, now encrypted under the new key. An attacker reads both messages and gets $x = m^e \bmod N$ and $y = m^{e'} \bmod N$.

   (a) Show how the attacker can compute the message m in time polynomial in $\log(N)$. You may assume that $e, e' < N$.

   (b) For $N = 247, e = 11, e' = 17, x = 24$ and $y = 93$, compute m.

2. **(In-)Security of Textbook RSA Signatures for Weaker Security Notions:** Exercise 12.2 in [KL]. For each of the following variants of the definition of security for signatures, state whether textbook RSA is secure and prove your answer: 10 p.

   (a) In this first variant, the experiment is as follows: the adversary is given the public key pk and a random message m. The adversary is then allowed to query the signing oracle once on a single message that does not equal m. Following this, the adversary outputs a signature σ and succeeds if $\mathsf{Vrfy}_{pk}(m, \sigma) = 1$. As usual, security is said to hold if the adversary can succeed in this experiment with at most negligible probability.

   (b) The second variant is as above, except that the adversary is not allowed to query the signing oracle at all.

3. **Encoded RSA:** Exercise 12.4 in [KL]. Another approach (besides hashed RSA) to trying to construct secure RSA signatures is to use encoded RSA. Here, public and private keys are as in textbook RSA; a public encoding function enc is fixed; and the signature on a message m is computed as $\sigma := [\mathsf{enc}(m)^d \mod N]$. 25 p.

   (a) How is verification performed in encoded RSA?

   (b) Discuss why appropriate choice of an encoding function prevents the "no-message attack" described in Section 12.3.1.

   (c) Show that encoded RSA is insecure for $\mathsf{enc}(m) = 0\|m\|0^{\ell/10}$ (where $\ell = \|N\|$, $|m| = 9\ell/10 - 1$, and m is not the all-0 message).

(d) Show that encoded RSA is insecure for $\mathsf{enc}(m) = 0\|m\|0\|m$ (where $|m| = (\|N\| - 1)/2$ and m is not the all-0 message)

4. **Random Oracles:** Let $H : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ be a random oracle. Show that 15 p.

   (a) For $\ell(n) = 2n$, H behaves like a PRG.

   (b) For $\ell(n) = n$, H behaves like a one-way function (according to Definition 6.1 in [KL]).

   (c) For $\ell(n) = n/2$, H behaves like a collision-resistant hash function..

5. **Secure E-mail in Practice:** Send and receive encrypted and authenticated e-mail. Start at `http://www.gnupg.org/`. GnuPG runs on Linux but the website also includes links to the versions for Windows and Mac OS. See `https://www.enigmail.net/documentation/quickstart.php` for Mozilla Thunderbird or use whatever software makes sense for you. 15 p.

   (a) At `http://homepages.cwi.nl/~schaffne/courses/crypto/2014/pgp/` you can download several files. What can you tell us about these files?

   (b) Send an e-mail, encrypted and signed by your personal key, to both Malvin and Christian. Ideally, your public key should be on keyservers (see `https://sks-keyservers.net/`). If you don't want to upload it, please send it to us (in the same or a separate message).

6. **Bitcoin:** 15 p.

   (a) What can you tell us about `1GwLBrEojCo3cXgTnvUFFw7mdb1Eej2g9U`?

   (b) Consider the *bitcoin backbone protocol*, the abstract model described in the second hour of Marc's lecture, i.e. the q-bounded synchronous setting described in `http://eprint.iacr.org/2014/765` and in Marc's notes. In this model, as the number of nodes and number of hash queries q per round per node is fixed, the difficulty D is also fixed. In real bitcoin, the total hash power is growing and the difficulty D is adjusted every 2016 blocks accordingly.

   Suppose the *bitcoin backbone protocol* is run in the real bit-coin setting of variable computing power and the difficulty is adjusted to enforce an average mining time of 10 minutes per block. Then, an attacker could simulate the entire protocol where the hashing power stays unrealistically low. The attacker can simulate 10 minutes of time within a few seconds of real time and thereby eventually construct a valid blockchain that is longer than the one of the real network. After broadcasting this new chain, it will be adopted by the network.

   Suggest a way to prevent this attack. Why does this attack not work against the real bitcoin protocol?