Solution

Cryptography – Homework 5

Discussed on **Tuesday**, 29th January, 2019.

Exercise 5.1

Suppose Eve is given (N, e) the public key of an RSA cryptosystem. Show that she can efficiently factor $N = pq$ in each of the following cases:

(a) she can efficiently compute $\varphi(N)$ (Hint: What are the roots of the polynomial $X^2 - X(N + 1 - \varphi(N)) + N$?).

(b) she can efficiently compute an $0 \neq x \in \mathbb{Z}_N \setminus \mathbb{Z}_N^*$.

Solution:

(a) We have $q - (N + 1 - (p - 1)(q - 1)) + p = 0$. Multiplying by $q(\neq 0)$ yields the identity in the hint. Given this identity we see that once we know $\varphi(N)$ we can obtain q (and thus p) by solving a quadratic equation (over the reals!) which can be done in polynomial time.

(b) If $x \in \mathbb{Z}_N \setminus \mathbb{Z}_N^*$ then $\mathsf{gcd}(x, N) > 1$ and thus $\mathsf{gcd}(x, N) = p$ or $\mathsf{gcd}(x, N) = q$. $\mathsf{gcd}(x, N)$ can of course be computed efficiently using Euclid's Algorithm.

Exercise 5.2 Collision resistance of the DLP-CCF

In the lecture we have seen the proof sketch that the DLP-CCF is collision-resistant, given that the DLP relative to $\mathsf{Gen}\mathbb{QR}_{\text{safe}}$ is hard, i.e.:

Assuming we have a collision attack $\mathcal{A}$ on the DLP-CCF we build from it the following algorithm $\mathcal{B}$ for computing discrete logarithms. Define $\mathcal{B}$ as:

- Input: (p, q, g) and $r = g^x \bmod p$ for some secret $x \overset{u}{\in} \mathbb{Z}_q$.
- If $r = 1$, output $x = 0$.
- Otherwise, pass (p, q, g, r) to $\mathcal{A}$ to obtain $(a, b) \neq (u, v)$.
- If $h_I(a, b) \neq h_I(u, v)$, output any element in $\mathbb{Z}_q$.
- Otherwise return $(a - u) \cdot (v - b)^{-1} \mod q$.

Complete the proof bz determining the probability that $\mathcal{B}$ succeeds in computing a logarithm of r modulo p. Why is it important that q is prime?

Solution: Let $p_{\texttt{coll}}$ be the probability that $\mathcal{A}$ finds a collision (i.e. $(a, b) \neq (u, v)$ with $h_I(a, b) \neq h_I(u, v)$).

We will show that $\mathcal{B}$'s success probability is at least $p_{\texttt{coll}}$ (actually slightly larger because if $\mathcal{A}$ fails we still have the chance of guessing).

If $\mathcal{A}$ finds a collision $(a, b) \neq (u, v)$ then we have $h_I(a, b) = h_I(u, v)$ which implies $g^a \cdot r^b \equiv g^u \cdot r^v \bmod p$ and thus $g^{(a-u)} \equiv_p r^{(v-b)}$ which gives us: $g^{(a-u)(v-b)^{-1}} \equiv_p r = g^x$. Thus $(a-u)(v-b)^{-1}$ is the discrete logarithm of r (which is found by $\mathcal{B}$ with probability at least $p_{\texttt{coll}}$).

It is important that q is prime—otherwise $(v - b)^{-1}$ might not exist!

Remark: The public information/parameter $I = (p, q, g, r)$ is the same as the public key in the Elgamal signature scheme we will see later.

Exercise 5.3

Let f be a OWP with hardcore bit $\mathsf{hc}(x)$. Show that $G_l(x) := f^l(x)||\mathsf{BM}^l(x)$ is a PRG of *fixed stretch* for every fixed l polynomial in n.

- Discuss the advantages/disadvantages of outputting also $f^l(x)$.
- In particular, consider the case when a TDP is used for f and the resulting PRG is used within the prOTP.

Solution: Consider the proof of the Blum-Micali construction where we turned a predictor for $\mathsf{BM}^l(x) = \mathsf{hc}(f^{l-1}(x))|| \ldots ||\mathsf{hc}(x)$ into a an algorithm $\mathcal{A}$ to compute $\mathsf{hc}(x)$ from $f(x)$:

Define $\mathcal{A}$ so that it simulates the prediction experiment for $x' = f^{i-l}(x)$:

- Input: $f(x)$.
- Choose $i \overset{u}{\in} [l(n)]$.
- Compute $y_i' \overset{r}{:=} \mathcal{P}(1^n, \mathsf{BM}^{i-1}(f(x)))$.
- Output: y_i'.

We simply change this construction so that also $f^i(x)$ is passed to $\mathcal{P}$:

- Input: $f(x)$.
- Choose $i \overset{u}{\in} [l(n)]$.
- Compute $y_i' \overset{r}{:=} \mathcal{P}(1^n, f^{i+1}(x)||\mathsf{BM}^{i-1}(f(x)))$.
- Output: y_i'.

The argument is then almost the same as in the original proof of the BM construction. $\mathcal{P}$ still needs to predict $\mathsf{hc}(x)$, but now i is not chosen uniformly over the lenght $n + l(n)$ of the whole output $f^l||\mathsf{BM}^l(x)$, but only within the last $l(n)$ bits. But the first n bits are unpredictable anyways, as with $x \overset{u}{\in} \{0,1\}^n$ also $f(x) \overset{u}{\in} \{0,1\}^n$ (f is a permutation).

As soon as $f^l(x)$ is made public, no further bits can be extracted using the Blum-Micali construction, so the stretch is fixed, but we get the n bits $f^l(x)$ "for free" in exchange.

The real advantage is with a TDP like the RSA problem. Here we have $f(x) := (x^e \bmod N)$ for (N, e) as required for the RSA problem. Any sender can choose $x \overset{u}{\in} \{0,1\}^n$ in secret, then compute $\mathsf{BM}^l(x)$ with $l = |m|$ as long as required by a given message, and then send $f^l(x)||(\mathsf{BM}^l(x) \oplus m)$ to the receiver.

Note that with $\mathsf{BM}^l(x)$ also $\mathsf{BM}^l(x) \oplus m$, and thus also $f^l(x)||(\mathsf{BM}^l(x) \oplus m)$ is pseudorandom. So this is essentially the prOTP but with the secret key x chosen for every message anew.

As the receiver can compute f^{-1}, he can recover $\mathsf{BM}^l(x)$ from $f^l(x)$ and thus recover m.

As x is chosen anew for every message, the resulting PKES can be shown to be CPA-secure: essentially, the concatenation of all ciphertexts send over the public channel is one big prOTP again.