HAI709I : Fondements cryptographiques de la sécurité, Université de Montpellier, 2024

09/12/2024. Lecture 12.

1 One-way functions.

In the class we defined the fundamental cryptographic notions: a one-way function and its vriations.

Definition 1. A function $f : \{0,1\}^* \to \{0,1\}^*$ is called a (strong) *one way function* (*fonction à sens unique*) if
(1) there is a deterministic algorithm that given $x \in \{0,1\}^*$ computes the value of $f(x)$ in time $\mathrm{poly}(n)$, and
(2) it is impossible to in invert f in polynomial time. To make the second condition more precise, we consider the following experiment:

- we choose uniformly a random $x \in \{0,1\}^n$ and compute $y = f(x)$
- an opponent's algorithm $\mathcal{INV}$ takes 1^n and y and returns a string $x' \leftarrow \mathcal{INV}(y, 1^n)$

We say that the opponent *succeeds* if $f(x') = y$. (Thus, the inversion algorithm $\mathcal{INV}$ is allowed to find not the initial value x but possibly some other x' such that $f(x') = f(x)$.) The condition (2) of the definition (non-invertibility of f) means that for every polynomial-time (deterministic or randomized) algorithm $\mathcal{INV}$, its probability to succeed is negligibly small, i.e., for every $c > 0$ and for all large enough n this probability becomes smaller than $1/n^c$.

Hypothesis: there exist functions f that satisfy the definition above.

It is believed in the cryptographic community that one-way functions do exist. Moreover, there are several specific candidates of such functions. However, this is still a hypothesis and not a mathematically proven theorem. If there is no one-way function, then a large part of the modern cryptography would crash.

Exercise 1. Show that every pseudo-random generator is a one-way function.

Exercise 2. Show that if $\mathrm{P} = \mathrm{NP}$ then one-way functions do not exist.

Most known constructions of functions that are believed to be one-way are based on the following definition.

Definition 2. A function $f : \{0,1\}^* \to \{0,1\}^*$ is called a *weak one way function* if
(1) there is a deterministic algorithm that given $x \in \{0,1\}^*$ computes the value of $f(x)$ in time $\mathrm{poly}(n)$, and
(2) for a significant fraction of x, it is impossible to in invert $f(x)$ in polynomial time. To make the second condition more precise, we consider the same experiment as in the previous definition:

- we choose uniformly a random $x \in \{0,1\}^n$ and compute $y = f(x)$
- an opponent's algorithm $\mathcal{INV}$ takes 1^n and y and returns a string $x' \leftarrow \mathcal{INV}(y, 1^n)$

We say that the opponent *succeeds* if $f(x') = y$. The condition (2) of the definition (weak non-invertibility of f) means that there exists a polynomial $q(n)$ such that for every polynomial-time (deterministic or randomized) algorithm $\mathcal{INV}$, for all large enough n the probability of opponent's success is not greater than $1 - 1/|q(n)|$.

Exercise 3. Show that every strong one-way function is also a weak one-way function.

Definition 3. A function $f : \{0,1\}^* \to \{0,1\}^*$ is called *length-preserving* if for every $x \in \{0,1\}^*$ the length (the number of bits) of $f(x)$ is equal to the length of x. A function is called *one-to-one* (or *injective*) if it maps distinct elements of the domain to distinct images. In some applications we may need one-way functions that are length-preserving and one-to-one.

Example 1. One of the most famous candidate that possibly satisfies the definition of a weak one-way function is the product of integers:

$$f : [x, y] \mapsto x \cdot y$$

The input of this function is a concatenation of binary expansions of two natural numbers x, y (if the input has length n, then the first $n/2$ bits represent x and the last $n/2$ bits represent y). It is believed that given the product of two prime numbers $n = x \cdot y$ it is (typically) hard to reconstruct the factors x, y. However, the problem of factorisation may be much simpler of x and y are not prime.

Exercise 4. ~~Show that the function~~

$$\cancel{f : [x, y] \mapsto x \cdot y}$$

~~is *not* a strong one-way function.~~

In the previous lecture we have seen that among all k-bit integers the set of prime numbers take a fraction of size $\geq \frac{const}{k}$. It follows that among all strings of length n, the set of inputs for f representing a pair of numbers (x, y) where both x and y are prime (with $n/2$ bits for each number) is $\geq \frac{const}{n^2}$. This explains why the function f is believed to be hard to invert for at least a fraction $1/\text{poly}(n)$ of all inputs of length n. (Once again, this is an unproven conjecture and not a theorem.)

Remark 1. There exist a deterministic polynomial-time algorithm (Agrawal–Kayal–Saxena) that tests primality of a natural number. Thus, given a pair (x, y) we can verify whether both of them are prime. We can modify the function from Example 1 and let

$$f([x, y]) = \begin{cases} x \cdot y & \text{if } x < y,\ x \text{ and } y \text{ are prime numbers and their binary expansions} \\ & \text{contain the same number of binary digits} \\ [x, y] & \text{otherwise.} \end{cases}$$

It is conjectured that such a "regularized" version of the product function is a still a weak one-way function.

Example 2. Another example of a function that seems to be a good candidate for a weak one way function is

$$f : [x, g, n] \mapsto [g^x \mod n, g, n],$$

where $[x, g, n]$ is a code of a triple of natural numbers (the exponent modulo n). The inversion of this function (discrete logarithm) seems to be hard, at least in the case when n is a prime number and the series of powers of g

$$g \mod n,\ g^2 \mod n,\ g^3 \mod n, \ldots$$

includes all non-zero elements modulo n. It is believed therefore that for a non-negligible fraction of possible inputs, this function is hard to invert, and it satisfies the definition of a weak one-way function.

Remark 2. Similarly to Remark 1, we can modify the definition of the modular exponentiation so that the function is non-trivial only on the triples $[x, g, n]$ with a "regularly behaving" of n and g (for which the discrete logarithm is believed to be intractable), and make the function be identity for other inputs. Such a modified version is still believed to be a weak one way-function; moreover, such a function (with a reasonably chosen encoding of inputs and outputs) can be made length preserving and one-to-one weak one-way permutation, see [1] for details.

Theorem 1. *If there exists a weak one-way function f, then there exists a (strong) one-way function f'.*

In the class we did not prove this theorem but we mentioned that the strong one-way function f' can be constructed as follows. We choose a suitable polynomial $k(n) = \mathrm{poly}(n)$ and for each n and for all $x_i \in \{0,1\}^n$ ($i = 1, \ldots, k(n)$) we let

$$f'(x_1 \circ x_2 \circ \ldots \circ x_k) := f(x_1) \circ \ldots \circ f(x_k)$$

(where $\circ$ denotes concatenation). This constructive is pretty intuitive: if the initial function $f(x)$ is hard to invert on a *non-negligible* fraction of inputs x, than in a typical lists of inputs $(x_1, \ldots, x_k)$ there is at least one x_i such that $f(x_i)$ is hard to invert. Thus, if f is hard to invert on a small (but not negligible) fraction of inputs, then f' is hard to invert on an *overwhelming* fraction of all possible inputs. However, a formal proof of this fact is more involved.

Example 3. Let us re-discuss the problem of factorisation. We believe that the function

$$[x, y] \mapsto x \cdot y$$

is hard to invert for a non-negligible fraction of n-bit integers x, y. So it seems plausible to assume that for $k(n) = n^2$, in a randomly chosen list of pairs

$$(x_1, y_1), \ldots, (x_k, y_k)$$

(where all x_i and all y_i are integer numbers between 2^n and 2^{n+1}) with an overwhelming probability at least one pairs (x_i, y_i) consists of two prime numbers. Thus, the mapping

$$[(x_1, y_1), \ldots (x_k, y_k)] \mapsto [x_1 \cdot y_1, \ldots, x_k \cdot y_k]$$

is hard to invert for *almost all inputs* (since typically there is at last one component $(x_i \cdot y_i)$ for which the factorisation is hard).

Remark 3. Using a length-preserving one-to-one weak one-way function one can construct a length-preserving one-to-one strong one-way functions.

In some cryptographic applications it is not enough to have a (strong) one-way functions f; we my need to assume that some specific "piece of information" in x cannot be reconstructed from $y = f(x)$. This idea is captured in the following definition.

Definition 4. A *one-way functions with a hard code predicate* is a pair of functions $f : \{0,1\}^* \to \{0,1\}^*$ and $h : \{0,1\}^* \to \{0,1\}$ such that
(1) there are deterministic algorithms that given an $x \in \{0,1\}^n$ compute the values of $f(x)$ and $h(x)$ in time $\mathrm{poly}(n)$, and
(2) given y, it is impossible to compute in polynomial time the value b such that for some x' such that $f(x') = y$ and $h(x') = b$. To make the second condition more precise, we consider the following experiment:

- we choose uniformly a random $x \in \{0,1\}^n$ and compute $y = f(x)$
- an opponent's algorithm $\mathcal{ADV}$ takes 1^n and t and computes a bit value $b \leftarrow \mathcal{ADV}(y, 1^n)$

We say that the opponent *succeeds* if there exists an x' such that $b = h(x')$ and $f(x') = y$. The condition (2) of the definition technically means that for every polynomial-time (deterministic or randomized) algorithm $\mathcal{ADV}$ there is a negligibly small functions $g(n)$ such that the probability to succeed is less than $\frac{1}{2} + g(n)$.

Theorem 2. *If the exists a one-way function f, then there exists a one-way function f' with a hard core predicate h.*

The proof of this theorem is pretty involved. In the class we only discussed the construction of suitable f' and h (the construction proposed by Goldreich and Levin). It is known that if f is a one-way function, then

$$f'(x_1 \ldots x_n \circ r_1 \ldots r_n) = f(x_1 \ldots x_n) \circ [r_1 \ldots r_n]$$

with

$$h(x_1 \ldots x_n \circ r_1 \ldots r_n) = x_1 r_1 + \ldots x_n r_n \mod 2$$

(where all x_i and r_i are zeros or ones) provide an example of a one-way function with a hard-core predicate.

Theorem 3. *If there exists a length preserving one-to-one one-way function f with a hard core bit h, then there exists a pseudo-random generator g.*

The proof of this theorem is also pretty involved. But the idea of the construction can be sketched as follows. If f is a length-preserving one-way function with a hard-core bit h, then to construct a pseudo-random generator $g : \{0,1\}^n \to \{0,1\}^{n+1}$, we may let

$$g(x) := f(x) \circ h(x).$$

If we need a pseudo-random generator $g : \{0,1\}^n \to \{0,1\}^m$ for some $m = n + \ell$ (in most applications we need a generator with m much greater than $n + 1$), we iterate this construction ℓ times. If $\ell = \text{poly}(n)$, this construction gives a pseudo-random generator (we omit the details).

In conclusion we observe that if the problem of integer factorisation or the problem of descrete logarithm is indeed computationally hard, then we have an explicit example of a weak one-way function. Starting from this function, we can construct a strong one-way function, a one-way function with a hard-core predicate, and eventually a pseudo-random generator.

It is known that one can construct a pseudo-random generator given an arbitrary one-way function (not necessary a length preserving bijection), see [2]. However, this alternative construction is extremely complicated.

2 Bit commitment : how to play *heads or tails* online.

In the class we discussed two protocols of *bit commitment* that allow for a pair of users (with polynomial computational resources) to play the game *heads ot tails* via a communication channel. The first protocol was based a one-way permutation with a hard-core bit; the second protocol was based on a pseudo-random generator $g : \{0,1\}^n \to \{0,1\}^{3n}$.

Using any of these constructions of bit commitment, one can implement an electronic version of the system of *zero-knowledge proofs* that we discussed in Lecture 6 (October 14).

Bibliography

[1] Goldreich, Oded, Leonid A. Levin, and Noam Nisan. ”On constructing 1-1 one-way functions.” In Studies in complexity and cryptography: miscellanea on the interplay between randomness and computation, pp. 13-25. 2011.

[2] Håstad, J., Impagliazzo, R., Levin, L.A. and Luby, M., 1999. A pseudorandom generator from any one-way function. SIAM Journal on Computing, 28(4), pp.1364-1396.