HAI709I : Fondements cryptographiques de la sécurité, Université de Montpellier, 2023

23/10/2023. Lecture 7.

1 Semantic definition of security

In the class we re-discussed the definitions of security that we have seen before: the definition of (absolute) security from Lecture 1 and the definition of computational security (see Section 5 from from Lecture 5). The last definition may seem a bit artificial: it claims that a scheme must be resistant against a very specific and somewhat unusual attack[1].

We proved that this definition implies a series of pretty natural properties that could be expected from a "reasonably good" cryptographic scheme. More specifically, we proved that resistance to the attack from the definition of a computationally secure scheme implies resistance to many other types of attacks. We proved in the class the following facts.

Proposition 1. *Let $\Pi = \langle Gen, Enc, Dec\rangle$ be a poly-time computable encryption scheme and let i be an integer number. For each $n \geq i$ we consider the following experiment:*

- *Alice (the Sender) chooses at random an clear message $m = x_1 \dots x_n \in \{0,1\}^n$ (with the uniform distribution, i.e., each message $m \in \{0,1\}^n$ is chosen with the probability $1/2^n$), produces a random secret key $k \leftarrow Gen(1^n)$, and computes the encrypted message $e = Enc(m,k)$.*
- *Adversary gets the encrypted message e and tries to guess the first bit of the clear message m_1 using a poly-time computable (deterministic or randomised) algorithm Adv:*

$$Adv(1^n, e) \text{ produces a bit } j.$$

The success of the Opponent is defined as

$$\textbf{\textit{success}} = \begin{cases} 1, & \text{if } j = m_1 \\ 0, & \text{if } j \neq m_1 \end{cases}$$

We claim that if the scheme Π is computationally secure, then the difference

$$\left|\mathrm{Prob}[\textbf{\textit{succes}} = 1] - \frac{1}{2}\right|$$

is negligibly small. In other words, the Opponent cannot extract from the encrypted message e the i-th bit of the clear message m with a probability substantially bigger than $1/2$.

Remark 1. The proposition remains true if we Adversary tries to guess not the first bit of the clear message x_1 but the bit x_2, or x_3, …, or any other bits x_ℓ for a fixed position ℓ.

[1] Let us recall that in this attack Adversary chooses a pair of clear messages $m_a, m_b \in \mathcal{M}$, and then Sender choses at random one of these two messages $m \in \{m_a, m_b\}$ at random, samples a random secret key $k \leftarrow \mathcal{K}(1^n)$, and encodes the chosen clear message $e = \mathrm{Enc}(m,k)$. On the final stage, Adversary gets the encrypted messages $m$and tries to guess which clear message was encrypted, m_a or m_b.

An encryption scheme is *computationally secure* if the probability of the correct guess differs from $1/2$ by a negligibly small function, for all algorithms of Adversary computable in polynomial time.

Proposition 2. *Let $\Pi = \langle Gen, Enc, Dec\rangle$ be a poly-time computable encryption scheme and let i be an integer number. For each $n \geq 4$ we consider the following experiment:*

- *Alice (the Sender) chooses at random (with the uniform distribution) an clear message $m = x_1 \dots x_n \in \{0,1\}^n$, produces a random secret key k with the algorithm $Gen(1^n)$, and computes the encrypted message $e = Enc(m, k)$.*
- *Adversary gets the encrypted message e and the first 56 bits of the clear message $x_1 \dots x_{56}$, and tries to guess the last bit of the clear message m_{57} using a poly-time computable (deterministic or randomised) algorithm Adv:*

$$Adv(1^n, e, x_1 \dots x_{56}) \textit{ produces a bit } j.$$

The success of the Opponent is defined as

$$\textbf{\textit{success}} = \begin{cases} 1, & \textit{if } j = x_{57} \\ 0, & \textit{if } j \neq x_{57} \end{cases}$$

We claim that if the scheme Π is computationally secure, then the difference

$$\left| \mathrm{Prob}[\textbf{\textit{succes}} = 1] - \frac{1}{2} \right|$$

is negligibly small. In other words, given the encrypted message e and the first bits of the clear message $x_1 \dots x_{56}$, Adversary cannot compute the next bit of the clear message x_{57} with a probability substantially bigger than $1/2$.

We proved both these proposition by a reduction: we assumed that there exists an algorithm Adv that succeed in the experiments suggested in each of this propositions, and then used it to construct an algorithm that succeeds in the experiments from the definition of computational security. This is how we obtain a contradiction with the assumption that the scheme is computationally secure.

In fact, these two propositions (as well as the exercises from the homework) are only special cases of a much more general statement which is called *semantic security*. The following theorem presents this property in formal terms.

Theorem 1. *Let $\Pi = \langle Gen, Enc, Dec\rangle$ be a poly-time computable encryption scheme and let F and G be functions computable in polynomial time. For each natural number n we consider the following experiment:*

- *Alice (the Sender) chooses at random (using a randomised algorithm $Sampler(1^n)$ that is computable in polynomial time) a clear message $m \in \{0,1\}^n$, produces a random secret key $k \leftarrow Gen(1^n)$, and computes the encrypted message $e = Enc(m,k)$.*
- *The Opponent gets the encrypted message e, the value $F(m)$ (which can be interpreted as a partial information on the clear message m), and tries to guess the value $G(m)$ using a polynomial time computable (deterministic or randomised) algorithm Adv:*

$$Adv(1^n, e, F(m)) \text{ produces } j.$$

We say that the Opponent succeeds, if $j = G(m)$.

We claim that if the scheme Π is computationally secure, then there exists another algorithm Adv' computable in polynomial time and a negligibly small function $f(n)$ such that

$$\left|\mathrm{Prob}[Adv(1^n, e, F(m)) = G(m)] - \mathrm{Prob}[Adv'(1^n, F(m)) = G(m)]\right| < f(n).$$

In other words, given the partial information on the clear text presented by the value $F(m)$, the Opponent can use the algorithm $Adv'(1^n, F(m))$ and guess the values $G(m)$ with almost the same probability as $Adv(1^n, e, F(m))$. Thus, in practice, the encrypted message e does not provide any useful information on the clear message m (whatever Adversary does given e, can be done without e).

(We did not prove this theorem in the class.)

2 Diffie–Hellman key exchange

In the class we discussed the classical protocol of key exchange suggested by Diffie and Hellman. In this protocol two parties (Alice and Bob) want to agree on a common secret key. To this end, they communicate via an open communication channel. Adversary (an eavesdropper) can intercept all the messages sent by Alice and Bob to each other. We assume that the computational resources of the opponent are bounded, and the security of the protocol is based on the assumption that the problem of *discrete logarithm* (see below) cannot be resolved efficiently.

The public parameters of the protocol of Diffie and Hellman are a prime number p and an integer number g such that the degrees of this numbers

$$g \mod p,\ g^2 \mod p,\ g^3 \mod p,\ g^4 \mod p, \ldots \tag{1}$$

cover the entier set of all possible non zero residues modulo p. In other words, for every integer number $1 \le y < p$ there exists an integer x such that

$$y = g^x \mod p$$

The protocol is organised as follows:

- Alice chooses at random an integer number a co-prime with p and sends the number $m_a = g^a \mod p$ to Bob
- Bob chooses at random an integer number b co-prime with p and sends the number $m_b = g^b \mod p$ to Alice
- Alice computes $k = (m_b)^a \mod p$
- Alice computes $k = (m_a)^b \mod p$

Observe that Alice and Bob end up with the same number

$$k = (g^a)^b \mod p = (g^b)^a \mod p.$$

This number is taken as the secret key.

Observe that the eavesdropper can access the numbers $g^a \mod p$ sent by Alice and $g^b \mod p$ sent by Bob. It is believed in the cryptographic community that it is computationally hard to obtain $(g^{ab} \mod p)$ given two numbers $(g^a \mod p)$ and $(g^b \mod p)$. More precisely, it is conjectured that no algorithm (deterministic or randomised) for a classical computer can do this task in polynomial time. However, this conjecture remains unproven.

It is known that there exists an algorithm for a *quantum computer* that computes in polynomial time the discrete logarithm, i.e., given (p, g, y) it finds an x such that $g^x = y \mod p$. Using such an algorithm, an eavesdropper could attack the protocol of Diffie–Hellman. This is why in *post-quantum cryptography* the researchers explore possible alternative to the protocol of Diffie and Hellman (these alternative would become necessary in case if quantum computers are eventually constructed).

Security of the protocol of Diffie and Hellman depends on the fact that there are *many* different $y \in \{1, 2, \ldots, p-1\}$ that can be represented as $y = g^x \mod p$ for a fixed g. Ideally, *all* values $g^x \mod p$ for $x = 1, 2, \ldots, p-1$ should be different. Indeed, let us consider the case when the list (1) consists only of k different elements and $k \ll p - 1$. Given the values m_a and m_b, the eavesdropper can find their *discrete logarithms* a and b (the numbers such that $g^a = m_a \mod p$ and $g^b = m_b \mod p$) by brute force search, by trying all values $(g^x \mod p)$ in (1) for $x = 1, 2, \ldots, k$ one by one. The smaller is k, the faster the brute force search works. Thus, for a small k, the task of the attacher is simpler. This is why we prefer to use in this protocol a number g such that the list (1) is maximally long (i.e., there are $(p-1)$ different elements in this list).

It remains to show that for every prime number p there exists an integer g such that the list (1) consists of $(p-1)$ numbers. We will prove in the next lecture. In the class we discussed one example: we looked for a g generating all non-zero elements modulo the prime number 17.

References

[1] J. Katz, Y. Lindell. Introduction to modern cryptography, CRC Press, 2021

[2] B. Martin. Codage, cryptologie et applications. PPUR presses polytechniques, 2004