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

14/10/2024. Lecture 6.

1 Pseudo-random generators: several simple properties.

Reminder 1:

Definition 1. A family of functions

$$G_n \ : \ \{0,1\}^{\ell(n)} \to \{0,1\}^n$$

is called a *pseudo-random generator* if

- $\ell(n) < n$
- $G_n(x)$ is computed (by a deterministic algorithm) in time $\mathrm{poly}(n)$
- for every poly-time algorithm TEST (deterministic or randomised) the difference

$$\left|\mathrm{Prob}_{x\in_R\{0,1\}^{\ell(n)}}[\mathrm{TEST}(G_n(x))=1] - \mathrm{Prob}_{y\in_R\{0,1\}^n}[\mathrm{TEST}(y)=1]\right|$$

  is negligibly small.

In the class we discussed several simple properties that follow from the definition of a pseudo-random generator.

Example 1. If a function

$$G_n \ : \ \{0,1\}^{\ell(n)} \to \{0,1\}^n$$

is defined so that for every $x \in \{0,1\}^{\ell(n)}$ the first bit of the value $y = G_n(x)$ is equal to 1, then G_n is *not* a pseudo-random generator.

Proof. Let us define

$$\mathrm{TEST}_1(y) = \begin{cases} 1, & \text{if the first bits of } y \text{ is equal to } 1, \\ 0, & \text{otherwise.} \end{cases}$$

Obviously, $\mathrm{TEST}_1()$ is computable in polynomial time. Moreover, for every $x \in \{0,1\}^{\ell(n)}$ we have $\mathrm{TEST}_1(G_n(x)) = 1$. Hence,

$$\mathrm{Prob}_{x\in_R\{0,1\}^{\ell(n)}}[\mathrm{TEST}_1(G_n(x))=1] = 1.$$

On the other hand, for a random $y \in \{0,1\}^n$, only in the half of all possible cases the first bit of y is equal to 1. Therefore,

$$\mathrm{Prob}_{y\in_R\{0,1\}^n}[\mathrm{TEST}_1(y)=1] = \frac{1}{2}.$$

Thus,

$$\left|\mathrm{Prob}_{x\in_R\{0,1\}^{\ell(n)}}[\mathrm{TEST}_1(G_n(x))=1] - \mathrm{Prob}_{y\in_R\{0,1\}^n}[\mathrm{TEST}_1(y)=1]\right| = \frac{1}{2},$$

which is *not* negligible. This contradicts the definition of a pseudo-random generator. □

Example 2. If a function

$$G_n \ : \ \{0,1\}^{\ell(n)} \to \{0,1\}^n$$

is defined so that for every $x \in \{0,1\}^{\ell(n)}$ the string $y = G_n(x)$ is a palindrome, then G_n is *not* a pseudo-random generator.

Proof. The argument is similar to Example 1. It is enough to define the test

$$\text{TEST}_2(y) = \begin{cases} 1, & y \text{ is a palindrome,} \\ 0, & \text{otherwise,} \end{cases}$$

and plug it into the definition of a pseudo-random generator. We omit the details. □

Example 3. Let

$$G_n \ : \ \{0,1\}^{\ell(n)} \to \{0,1\}^n$$

be a family of functions and let `InImage()` be the predicate defined as follows: for every $y \in \{0,1\}^n$

$$\texttt{InImage}(y) = \begin{cases} 1, & \text{if there is an } x \in \{0,1\}^{\ell(n)} \text{ such that } G_n(x) = y, \\ 0, & \text{otherwise} \end{cases}$$

In the class we proved that if `InImage()` is computable in polynomial time, then G_n is a not a pseudo-random generator (we omit the details). Using this fact we also showed that G_n cannot be a pseudo-random generator if $\ell(n) = \log_2 n$ (if $\ell(n)$ is so small, we can tun through all $x \in \{0,1\}^{\ell(n)}$ in polynomial time, so `InImage()` is efficiently computable).

2 From a pseudo-random generator to an encryption scheme secure against and adversary computable in polynomial time

Reminder 2: the Vernam encryption scheme. $\Pi_{\text{Vernam}} = \langle Gen, Enc, Dec \rangle$ is defined as follows. We define $\mathcal{M} = \mathcal{E} = \mathcal{K} = \{0,1\}^n$ (the spaces of clear texts, cyphertexts, and the secret keys), and let

- the algorithm $Gen()$ take a random $k \in \{0,1\}^{\ell(n)}$
- the algorithm $Enc(m,k)$ compute a bitwise XOR of the clear message m and k
- the algorithm $Dec(e,k)$ compute a bitwise XOR of the encrypted message e and k

This scheme is (absolutely!) secure. In particular, it is secure against any adversary computable in polynomial time[1]. The only disadvantage of the Vernam scheme is the large size of the secret key.

[1]This claim sound as a trivial observation: *absolute security* implies *security against bounded adversary*. However, speaking formally, the definitions of absolute security and security against bounded adversary are very different, and the relation between these definitions is not so obvious. However, in Lecture 3 we proved that the definition of absolute security implies indeed the definition of security against adversary computable in polynomial time.

Remainder 3: an attack with a chosen pair of clear messages. Let $\Pi = \langle \mathrm{Gen}(), \mathrm{Enc}(), \mathrm{Dec}() \rangle$ be an encryption scheme, where $\mathcal{M}, \mathcal{E}, \mathcal{K}$ are the spaces of *clear messages*, *encrypted messages*, and *secret key* respectively. Let us consider the following game between an adversary and Alice.

- Adversary uses an algorithm $Adv_1(\underbrace{11\ldots1}_{n})$ that chooses two clear messages $m_a, m_b \in \mathcal{M}_n$;
- Alice chooses at random $i \in \{a, b\}$ (with equal probabilities), samples a secret key $k \leftarrow \mathrm{Gen}(\underbrace{11\ldots1}_{n})$, and computes the encrypted message $e = \mathrm{Enc}(m_i, k, \underbrace{11\ldots1}_{n})$;
- Adversary computes $j \in \{a, b\}$ using another algorithm $j \leftarrow Adv_2(e, m_a, m_b, \underbrace{11\ldots1}_{n})$.

The success of the adversary is defined as follows:

$$\mathbf{success} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{otherwise.} \end{cases}$$

In words: the adversary prepares a pair of messages m_a, m_b; Alice decides which message to encrypt; then the adversary tries to understand which of the two messages was encrypted.

Definition 2. An encryption scheme $\Pi = \langle \mathrm{Gen}(), \mathrm{Enc}(), \mathrm{Dec}() \rangle$ is called *secure against an adversary computable in polynomial time*, if for the game between Adversary and Alice following the protocol of the *attack with a pair of chosen clear messages* (explained above) where the adversary's algorithms Adv_1 and Adv_2 are computable in polynomial time, the gap

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

is a negligible function.

The main result of the section: We proved earlier that an encryption scheme cannot be secure against an adversary with unbounded computational resources if the secret keys are shorter than the clear messages. However, we show below that we can make secret keys much shorter than the clear messages and maintain the property of security against the adversaries *computable in polynomial time*.

Theorem 1. *We fix $G_n : \{0,1\}^{\ell(n)} \to \{0,1\}^n$ and define an encryption scheme $\Pi = \langle Gen, Enc, Dec \rangle$ as follows. We let $\mathcal{M}_n = \mathcal{E}_n = \{0,1\}^n$ (the spaces of clear texts and cyphertexts), and $\mathcal{K}_n = \{0,1\}^{\ell(n)}$ (the space of secret keys), and assume that*

- *the algorithm $Gen(\underbrace{11\ldots1}_{n})$ samples a random $k \in \{0,1\}^{\ell(n)}$*
- *the algorithm $Enc(m, k)$ computes a bitwise XOR of the clear message m and $k' = G_n(k)$*
- *the algorithm $Dec(e, k)$ computes a bitwise XOR of the encrypted message e and $k' = G_n(k)$*

If G_n is a pseudo-random generator (PRG), then this scheme Π is secure against any adversary computable in polynomial time.

Proof. We need to prove that

G_n is a PRG $\Longrightarrow$ $\langle Gen, Enc, Dec\rangle$ is secure against any adversary computable in polynomial time.

It is helpful to rephrase this property in a counter-positive form:

$\langle Gen, Enc, Dec\rangle$ is **not** secure against any adversary computable in polynomial time $\Longrightarrow$ G_n is **not** a PRG.

Let us prove this implication. The conditions

$\langle Gen, Enc, Dec\rangle$ is **not** secure against any adversary computable in polynomial time

means that there exist algorithms Adv_1 and Adv_2, both converging in polynomial time, such that in the attack

$$\begin{cases} \bullet\ Adv_1(\underbrace{11\ldots1}_{n}) \text{ produces } m_a, m_b \in \{0,1\}^n; \\ \bullet \text{ Alice chooses at random } i \in \{a,b\} \text{ (with equal probabilities), samples a secret key } k \in \{0,1\}^{\ell(n)}, \\ \quad \text{and computes the encrypted message } e = m_i \oplus G_n(k); \\ \bullet\ Adv_2(e, m_a, m_b) \text{ produces } j \end{cases}$$

the probability of success is $\mathrm{Prob}[j = i] = \frac{1}{2} + \delta_n$, where δ_n is some *non-negligible* function (e.g., δ_n can be a positive constant, or $1/\log n$, or $1/n^2$, but δ_n cannot be equal to $1/2^n$).

Since k and m_a, m_b are sampled independently, we can rephrase this process as follows: we sample first a random $x \in \{0,1\}^{\ell(n)}$, define $y = G_n(x)$, and run the procedure

$$\textbf{TEST}(y) \begin{cases} \bullet\ Adv_1 \text{ produces } m_a, m_b \in \{0,1\}^n; \\ \bullet \text{ Alice chooses at random } i \in \{a,b\} \text{ (with equal probabilities),} \\ \quad \text{and computes the encrypted message } e = m_i \oplus y; \\ \bullet\ Adv_2(e, m_a, m_b) \text{ produces } j \\ \bullet \text{ return 1 if } j = i; \text{ return 0 otherwise} \end{cases}$$

which return 1 with probability $\frac{1}{2} + \delta_n$ for some non-negligible function δ_n. In other words,

$$\mathrm{Prob}_{x\in\{0,1\}^{\ell(n)}}[\textbf{TEST}(G_n(x)) = 1] = \frac{1}{2} + \delta_n$$

We claim that this **TEST**() can be used to "discredit" the generator G_n. Indeed, we constructed is so that it is computable in polynomial time. Further, if we sample a truly random $y \in \{0,1\}^n$ and run **TEST**(y), we actually simulate the attack with two clear messages against the classical Vernam scheme Π_{Vernam} (with a random key y whose length is equal to the length of the clear message). Due to Theorem 3 from Lecture 3 we know that in such an attack the probability of success is always exactly $\frac{1}{2}$, as if the adversary just tosses a coin to produce the final answer. Thus,

$$\mathrm{Prob}_{x\in\{0,1\}^{\ell(n)}}[\textbf{TEST}(G_n(x)) = 1] - \mathrm{Prob}_{y\in\{0,1\}^{n}}[\textbf{TEST}(G_n(x)) = 1] = \left(\frac{1}{2} + \delta_n\right) - \frac{1}{2} = \delta_n.$$

As δ_n is non-negligible, we arrive to a contradiction with the definition of a pseudo-random generator. □

3 Zero knowledge proof with physical gadgets

In the class we discussed a protocol of *zero knowledge proof* for the problem of 3-colorability of a graph and its cryptographic interpretation: *Prover* can convince *Verifier* that Prover knows a "secret password" (3-coloring of the given graph) without divulging any information on this coloring. We discussed a protocol that uses physical gadgets: caps that hid colors assigned to the vertices. Later we will discuss an "electronic" version of this protocol, with a possibility to perform a zero knowledge proof online (without physical cups).

References

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

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