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

07/10/2024. Homework for Lecture 5.

Exercise 1. Let $\Pi = (Gen, Enc, Dec)$ be a computationally secure encryption scheme with $\mathcal{M}_n = \{0,1\}^n$.

Informal version of the question: Prove that an adversary cannot find in polynomial time the last bit of the clear message even given access to the encrypted message together with the XOR of all bits of the clear message.

Formal version of the same question: We consider the following experiment:

- Alice produces a random secret key k for the security parameter n,
  $k \leftarrow Gen(1^n)$
- Alice produces a random open messages $m = x_1 \dots x_n$ of length n bits (with the uniform distribution, i.e., each message can be chosen with the probability $1/2^n$)
- Alice computes an encrypted message $e = Enc(m, k)$
- Adversary obtains the encrypted message e and one more bits that is equal to the parity of all bits of the open message, i.e., $b := x_1 \oplus x_2 \oplus \dots \oplus x_n$, and tries to guess x_n,
  $j \leftarrow Adv(1^n, e, b)$.

The success of Adversary is defined as

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

Prove that for every polynomial time computable algorithm Adv

$$|\mathrm{Prob}[\mathbf{success} = 1] - 1/2|$$

is a negligibly small function.

Exercise 2 (more difficult)**.** Let $\Pi = (Gen, Enc, Dec)$ be a computationally secure encryption scheme with $\mathcal{M}_n = \{0,1\}^n$.

Informal version of the question: Given access tot he encrypted message, a polynomial-time computable adversary cannot predict the first *two* bits of the clear message with a probability significantly better than $1/4$.

Formal version of the question: We consider the following experiment.

- Alice produces a random secret key k for the security parameter n,
  $k \leftarrow Gen(1^n)$
- Alice produces a random open messages $m = x_1 \dots x_n$ of length n bits (with the uniform distribution, i.e., each message can be chosen with the probability $1/2^n$)
- Alice computes an encrypted message $e = Enc(m, k)$

- Adversary obtains the encrypted message e, and tries to guess x_1x_2,
  $j \leftarrow Adv(1^n, e)$, where $j \in \{00, 01, 10, 11\}$.

The success of Adversary is defined as

$$\mathbf{success} = \begin{cases} 1, & \text{if } j = x_1x_2 \\ 0, & \text{otherwise.} \end{cases}$$

Prove that for every poly-time computable algorithm Adv

$$|\mathrm{Prob}[\mathbf{success} = 1] - 1/4|$$

is a negligibly small function.

Exercise 3. Let $G_n : \{0,1\}^{0.5n} \to \{0,1\}^n$ be a pseudo-random generator.

(a) Show that there exists a deterministic (*not* polynomial time) algorithm `Rev` such that for every $y \in \{0,1\}^n$ in the range of the function G_n this algorithm returns an $x \in \{0,1\}^{0.5n}$ such that $G_n(x) = y$.

(b) Show that there exists a randomized polynomial time algorithm `Rev`$'$ such that for every $y \in \{0,1\}^n$ in the range of the function G_n with probability $2^{-n/2}$ this algorithm returns an $x \in \{0,1\}^{0.5n}$ such that $G_n(x) = y$ and with probability $1 - 2^{-n/2}$ returns the symbol $\perp$ (*failure*).

(c) Show that there is no randomized polynomial time algorithm `Rev`$''$ such that for every $y \in \{0,1\}^n$ in the range of G with a probability $\geq 3/4$ this algorithm returns an $x \in \{0,1\}^{0.5n}$ such that $G_n(x) = y$.

Explication informelle: It is always easy to invert any pseudo-random generator deterministically in exponential time. It is always easy to invert a pseudo-random generator in polynomial time but with an exponentially small probability. However, it must be hard to invert it in polynomial time with a non-negligible probability.

Exercise 4. Let $G_n : \{0,1\}^{\sqrt{n}} \to \{0,1\}^n$ be a function computable by a deterministic algorithm in polynomial time.

(a) Assume that there exists deterministic polynomial time algorithm `InImage` such that for every $y \in \{0,1\}^n$

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

Use the definition of a pseudo-random generator and prove that G_n cannot be a pseudo-random generator.

(b) Show that if $\mathrm{P} = \mathrm{NP}$ than G_n cannot be a pseudo-random generator. *Hint:* use the conjecture $\mathrm{P} = \mathrm{NP}$ and construct `InImage` form (a).

(c) Assume that there exists randomized polynomial time algorithm `ProbablyInImage` such that for every $y \in \{0,1\}^n$ with probability 0.99 this algorithm returns

$$\begin{cases} 1, & \text{if there exists an } x \in \{0,1\}^{\sqrt{n}} \text{ such that } G_n(x) = y \\ 0, & \text{otherwise.} \end{cases}$$

(and with probability 0.01 returns the opposite answer). Use the definition of a pseudo-random generator and prove that in this case G_n cannot be a pseudo-random generator.

(d) Assume that there exists randomized polynomial time algorithm `Rev` such that for every $y \in \{0,1\}^n$ in the range of G_n this algorithm returns an $x \in \{0,1\}^{\sqrt{n}}$, and with probability ≥ 0.99 we have $G_n(x) = y$. Use the definition of a pseudo-random generator and prove that in this case G_n cannot be a pseudo-random generator. *Hint:* use the algorithm `Rev` to simulate `ProbablyInImage` form (c).

Exercise 5. Let $G_n : \{0,1\}^{\sqrt{n}} \to \{0,1\}^n$ be a function computable by a deterministic algorithm in polynomial time. Assume that there exists randomized polynomial time algorithm `Rev` that for every $y \in \{0,1\}^n$ in the range of G_n returns with probability $\geq 1/10$ an $x \in \{0,1\}^{\sqrt{n}}$ such that $G_n(x) = y$.

(a) [error amplification] Prove that there exists another polynomial time algorithm `Rev`$'$ that for every $y \in \{0,1\}^n$ in the range of G_n returns with probability $\geq 9/10$ an $x \in \{0,1\}^{\sqrt{n}}$ such that $G_n(x) = y$.

(b) [distinguishing from true randomness] Show that such a G_n cannot be a pseudo-random generator.

Hint: We are given that (i) G_n can be computed in polynomial time and that (ii) for every element in the range of G_n we can compute in polynomial time its pre-image. Combining these two facts, one can show that for $y \in \{0,1\}^n$ we can verify in polynomial time whether y has a pre-image or not. So we can apply the statement from Exercise 4.