Complexity Theory and Cryptography

Mihai Prunescu*

Recall the complexity classes P and NP. The class P consists of the sets $A \subseteq \{0,1\}^*$ such that the caracteristic function 1_A is computable in polynomial time. This means that there is a polynomial $p(n)$ such that for every $x \in \{0,1\}^*$, the value $1_A(x)$ is deterministically computed in at most $p(|x|)$ steps. The class NP consists of the sets $B \subseteq \{0,1\}^*$ such that there is a set $A \in$ P and a polynomial $q(n)$ with the property that:

$$\forall x \in \{0,1\}^* \quad x \in B \longleftrightarrow \exists\, y \in \{0,1\}^* \quad |y| = q(|x|) \wedge (y,x) \in A.$$

Also, a set C belongs to coNP if and only its complement $\{0,1\}^* \setminus C$ belongs to NP.

The question of P = NP is one of the most important and the most difficult question of mathematics, and is open.

There are various connections between this questions and the cryptographic security. Most of the public key algorithms rely on the supposition that the problem called FACTORING or the problem called DISCRETE LOGARITHM are not solvable in polynomial time. Let us take a closer look at FACTORING:

The function called multiplication $(x,y) \rightsquigarrow xy$ is polynomial time computable in the length of the pair (x,y) written binary. It is not known if for prime numbers $p, q \in \mathbb{N}$, the reverse operation $pq \rightsquigarrow (p,q)$ is computable in polynomial time. In contrast, the set of COMPOSITE numbers belongs to NP. To prove that a number x is composite, it is enough to guess a number y with $|y| < |x|$ and to perform the division $x : y$ hoping that the remainder will be 0. (Observe that division with remainder can be performed in polynomial time as well.) The complement of the set COMPOSITE is the set PRIME which consequently belongs to coNP.

There was a serious earthquake in 2003 when Manindra Agrawal, Neeraj Kayal and Nitin Saxena shown that the decision problems PRIME - and so also COMPOSITE - are in P. So for a product of two primes pq we can decide in polynomial time that the number is composite, but we don't know yet if we can find in polynomial time one of its nontrivial divisors p and q. However, to find such a divisor means to find a certificate proving that pq belongs to the set COMPOSITE, as above. How sure can we be that the computational problem is not solvable in polynomial time, while the decision problem is solvable in polynomial time?

The goal of this lecture is to explore some other connections between complexity theory and cryptography.

Theorem 1 *Suppose* P = NP. *Let* (Enc, Dec) *be a polynomial-time computable encryption scheme with the keys shorter then the messages. Then there is a deterministic polynomial-time algorithm* A *such that for every input-length* m *there is a pair of messages* $x_0, x_1 \in \{0,1\}^m$ *satisfying:*

$$Pr[A(Enc_k(x_b)) = b] \geq \frac{3}{4},$$

where $b \in \{0,1\}$, $k \in \{0,1\}^n$ *and* $n < m$.

*This belongs to the content of the chapter "Cryptography" from the book *Computational Complexity* by Arora and Barakcombined with some definitions and commentaries introduced by me.

Proof: Let $S \subset \{0,1\}^*$ be the set of encryptions of 0^m by all possible keys of length m. So $y \in S$ if and only if $y = Enc_k(0^m)$ for some k. If P = NP then the membership to S can be deterministically verified in polynomial time.

We define the algorithm A as follows: $A(y) = 0$ if $y \in S$ and $A(y) = 1$ otherwise. We set $x_0 = 0^m$. We claim that there is a x_1 such that the hypothesis holds.

For some message $x \in \{0,1\}^m$ let D_x be the set $\{Enc_k(x) \,|\, k \in \{0,1\}^n\}$. Of course $D_{x_0} = D_{0^m} = S$. This means that:

$$Pr[A(D_{x_0}) = 0] = 1.$$

We observe that:

$$Pr[A(Enc_k(x_b)) = b] = \frac{1}{2}Pr[A(D_{x_0}) = 0] + \frac{1}{2}Pr[A(D_{x_1}) = 1] = \frac{1}{2} + \frac{1}{2}Pr[A(D_{x_1}) = 1],$$

so it would be sufficient to show that $\exists\, x_1 \in \{0,1\}^m$ such that $Pr[A(D_{x_1}) = 1] \geq \frac{1}{2}$.

By complementarity this would be sufficient to show that for some $x_1 \in \{0,1\}^m$,

$$Pr_k[Enc_k(x_1) \in S] \leq \frac{1}{2}.$$

Suppose otherwise that for all $x_1 \in \{0,1\}^m$:

$$Pr_k[Enc_k(x_1) \in S] > \frac{1}{2}.$$

Let $\Omega = \{1, \ldots, u\}$ be a finite space of probability, let $p : \Omega \to [0,1]$ be the corresponding probability and let $X : \Omega \to \mathbb{R}$ be some random variable. We recall that the expectation of X is the sum:

$$E\ X := p(1)X(1) + \cdots + p(u)X(u).$$

Consider the random variable $S : \{0,1\}^m \times \{0,1\}^k \to \mathbb{R}$ given by:

$$S(x,k) = \begin{cases} 1, & Enc_k(x) \in S, \\ 0, & Enc_k(x) \notin S. \end{cases}$$

We define:

$$T = E_{x,k}\ S(x,k).$$

From our supposition, it follows $T > \frac{1}{2}$. But on the other hand:

$$T = E_k\ [E_x\ S(x,k)] \leq \frac{1}{2},$$

because for all $k \in \{0,1\}^n$, the function $x \rightsquigarrow Enc_k(x)$ is injective, hence at most $2^n \leq 2^m/2$ many x can be mapped to the set S, which has itself size $\leq 2^n$. This is a contradiction. □

Definition: A non-deterministic Turing machine (probabilistic Turing machine) is a Turing machine which has two different transition functions $\delta_0, \delta_1 : \Sigma \times \Gamma \to \Gamma$ and outputs only 1 (accept) or 0 (reject). At every step, one of the two function is chosen and the step is performed. □

Definition: For $T : \mathbb{N} \to \mathbb{N}$ and $L \subseteq \{0,1\}^*$ we say that a probabilistic Turing machine M decides L in time $T(n)$ if for every $x \in \{0,1\}^*$, M stops in $T(|x|)$ steps regardless of its random choices and $Pr[M(x) = L(x)] \geq 2/3$. Here the constant $2/3$ is not relevant and can be replaced by any real number which is strictly bigger than $1/2$. We let BPTIME($T(n)$) be the class of languages decided in time $O(T(n))$ and define BPP $= \bigcup_c$ BPTIME(n^c). □

Definition: A function $\varepsilon : \mathbb{N} \to [0,1]$ is called **negligible** if and only if for all $c \in \mathbb{N}$ and all sufficiently large n, $\varepsilon(n) < n^{-c}$. □

Definition: Let $f : \{0,1\} \to \{0,1\}$ a function which is computable in polynomial time. The function f is called a **one way function** if for every polynomial time probabilistic algorithm A there is a negligible function $\varepsilon : \mathbb{N} \to [0,1]$ such that for all n:

$$Pr_{|x|=n, y=f(x)}[f(A(y)) = y] < \varepsilon(n).$$

Conjecture: *One-way functions exist.*

Possible examples:

Multiplication: $x = (x_1, x_2) \rightsquigarrow \text{bin}(\text{num}(x_1)\ \text{num}(x_2))$. The inverse would be the problem FACTORING. Moreover, numbers have different decompositions as a product, only for product of primes one has a unique decomposition if the factors are sorted by absolute value.

RSA function: Given $N \in \mathbb{N}$, $e \in \mathbb{N}$ relatively prime with $|\mathbb{Z}_N^\times|$ and $t \in \mathbb{Z}_N^\times$ arbitrary, the function $t \rightsquigarrow t^e$ is known to be one-to-one and hard to invert. In fact, this is easy to invert if the prime decomposition of N is known.

Rabin's function: $N = pq$ where $p, q = 1 \bmod 4$. Then the function $t \in QR_N \to QR_N$ given by $t \rightsquigarrow t^2 \bmod N$ is one to one and hard to invert. This is also easy to invert if the factorisation of N is known.

Levin's Universal One-Way Function: The name is justified by the fact that if one-way functions exist, then it is known that the following function is a one-way function. We denote by M_i the i-th Turing machine. For a Turing machine M, let M^t be the output of M on input x if M needs at most t steps to stop, and $0^{|x|}$ if not.

An input x of length n is considered to be a list $x_1 \dots x_{\log n}$ of strings of length $n/\log n$. Then we define the universal one-way function f_U to be:

$$f_U(x) = M_1^{n^2}(x_1) \dots M_{\log n}^{n^2}(x_{\log n}).$$

Definition: Let (Enc, Dec) be some encryption system for $|k| = n$ and $|x| = m$. This system is called computationally secure if for every probabilistic polynomial time algorithm A there is a negligible function $\varepsilon : \mathbb{N} \to [0,1]$ such that:

$$Pr[A(Enc_k(x)) = (i, b) \mid x_i = b] \leq \frac{1}{2} + \varepsilon(n).$$

It is known that if one-way functions exist, then for every $c \in \mathbb{N}$, $c \neq 0$, there is a computationally secure encryption method with $|k| = n$ and $|x| = n^c$. Our goal is to understand this fact to some extent.

Definition: Let $G : \{0,1\}^* \to \{0,1\}^*$ and $l : \mathbb{N} \to \mathbb{N}$ functions computable in polynomial time such that $l(n) > n$. G is a function of stretch $l(n)$ if for all $x \in \{0,1\}^*$, $|G(x)| = l(|x|)$. Such a function is a secure pseudorandom generator if for every probabilistic polynomial time algorithm there is a negligible function $\varepsilon : \mathbb{N} \to [0,1]$ such that:

$$|Pr_{|x|=n}[A(G(x)) = 1] - Pr_{|y|=l(n)}[A(y) = 1]| < \varepsilon(n).$$

It is a known fact that if one-way functions exist then for all $c \in \mathbb{N}$ there is a secure pseudorandom generator with stretch $l(n) = n^c$. We will prove this only in the special case in which the one-way function is a permutation. This is not really a strong restriction because using the Feistel net method, every one-way function can be transformed in a one-way permutation.

Lemma 2 *Suppose there is an injective one-way function $f : \{0,1\}^* \to \{0,1\}^*$ such that $|f(x)| = |x|$. Then for all c, there is a pseudorandom generator of stretch n^c.*

Before we proceed in this direction, we introduce an alternative definition of the computationally secure pseudorandom generator.

Definition: Let $G\{0,1\}^* \to \{0,1\}^*$ be a polynomial time computable function of stretch $l(n)$. G is called **unpredictable** if for every polynomial time probabilistic algorithm B there is $\varepsilon : \mathbb{N} \to [0,1]$ negligible function such that:

$$Pr_{y=G(x),1\leq i\leq l(n)}[B(1^n, y_1 \ldots, y_{i-1}) = y_i] \leq \frac{1}{2} + \varepsilon(n),$$

where x is a random word of vlength n. □

With other words, to predict the next bit is difficult even if you have all the previous bits. Of course, pseudorandom implies unpredictable. But the converse is also true.

Theorem 3 *(Yao, 1982) Let $G : \{0,1\}^* \to \{0,1\}^*$ and $l : \mathbb{N} \to \mathbb{N}$ functions computable in polynomial time such that $l(n) > n$. G is a function of stretch $l(n)$ if for all $x \in \{0,1\}^*$, $|G(x)| = l(|x|)$. If G is unpredictable, then G is a secure pseudorandom generator. Moreover: If for every probabilistic polynomial time algorithm A there is a probabilistic polynomial time algorithm B such that from $\forall\, n \in \mathbb{N}$, $\forall\, \varepsilon > 0$, if:*

$$|Pr_{|x|=n}[A(G(x)) = 1] - Pr_{|y|=l(n)}[A(y) = 1]| \geq \varepsilon$$

then:

$$Pr_{y=G(x),1\leq i\leq l(n)}[B(1^n, y_1 \ldots, y_{i-1}) = y_i] \geq \frac{1}{2} + \frac{\varepsilon}{l(n)},$$

Proof: We observe that the "moreover" part implies the whole theorem. So let us prove it.

Let A be some polynomial time probabilistic algorithm that is supposed to more likely output 1 on inputs from $G(\{0,1\}^n)$ then on inputs from $\{0,1\}^{l(n)}$. We construct a predictor algorithm B as follows:

B chooses $z_i, \ldots, z_{l(n)}$ random,

B computes $a = A(y_1, \ldots, y_{i-1}, z_i, \ldots, z_{l(n)})$,

If $a = 1$, B outputs z_i,

If $a = 0$, B outputs $1 - z_i$.

Let us show that:

$$Pr_{y=G(x),1\leq i\leq l(n)}[B(1^n, y_1 \ldots, y_{i-1}) = y_i] \geq \frac{1}{2} + \frac{\varepsilon}{l(n)}.$$

We fix n and let $l = l(n)$. We define following distributions of probability $D_0, \ldots, D_l$ over $\{0,1\}^l$. For some i, D_i is constructed as follows: Take $x \in \{0,1\}^n$ arbitrary, $y = G(x)$, and output $y_1, \ldots, y_i, z_{i+1}, \ldots, z_l$ where all z_k are chosen randomly in $\{0,1\}$. Then $D_0 = \{0,1\}^l$ with the uniform probability distribution and $D_l = G(\{0,1\}^n)$ is the probability distribution on the image of G, supposing that the inputs from $\{0,1\}^n$ are equally probable.

Denote with $p_i := Pr[A(D_i) = 1]$. Observe that $p_l - p_0 \geq \varepsilon$. Observe also that:

$$p_l - p_0 = (p_l - p_{l-1}) + (p_{l-1} - p_{l-2}) + \cdots + (p_1 - p_0) \geq \varepsilon.$$

It follows that:

$$E_{1\leq i\leq l}[p_i - p_{i-1}] \geq \frac{\varepsilon}{l}.$$

We show that:

$$Pr_{y=G(x),1\leq i\leq l(n)}[B(1^n, y_1 \ldots, y_{i-1}) = y_i] \geq \frac{1}{2} + (p_i - p_{i-1}).$$

B predicts y_i correctly if and only if $(a = 1 \wedge y_i = z_i) \vee (a \neq 1 \wedge y_i = 1 - z_i)$, so this happens with probability:

$$\frac{1}{2} Pr[a = 1 \,|\, z_i = y_i] + \frac{1}{2}(1 - Pr[a = 1 \,|\, z_i = 1 - y_i]).$$

If $z_i = y_i$, B invokes A with the distribution D_i so $Pr[a = 1 \mid z_i = y_i] = p_i$. If not, then B invokes A with the distribution D_{i-1}. Hence:

$$p_{i-1} = Pr[a = 1] = \frac{1}{2}Pr[a = 1 \mid z_i = y_i] + \frac{1}{2}Pr[a = 1 \mid z_i = 1 - y_i] = \frac{1}{2}p_i + \frac{1}{2}Pr[a = 1 \mid z_i = 1 - y_i].$$

So B predicts y_i with probability $\frac{1}{2} + p_i - p_{i-1}$. □

Here we recall some technical facts:

Lemma 4 *If $a_1, a_2, \ldots, a_n \geq 0$ are numbers whose average is c, then the fraction of a_i's that are at least kc is at most $1/k$.*

Lemma 5 *(Markov) Any nonnegative random variable X satisfies:*

$$Pr[\ X \geq kE(X)\ ] \leq \frac{1}{k}.$$

Definition: For a random variable X we consider the random variable $(X - E\,X)^2$. Its expectation is denoted:

$$Var(X) = E\,(X - E\,X)^2$$

and is called the variance of X. □

The following facts are easy to prove:

$$Var(X) \geq 0,$$
$$Var(X) = E\,X^2 - (E\,X)^2.$$

If $X_1, \ldots, X_n$ are **pairwise** independent random variables, then:

$$Var(\sum X_i) = \sum Var(X_i).$$

Definition: The quantity $\sqrt{Var(X)}$ is called the standard deviation of X.

Lemma 6 *(Chebyshev) If X is a random variable with standard deviation σ, then for every $k > 0$:*

$$Pr[\ |X - E\,X| > k\sigma\ ] \leq \frac{1}{k^2}.$$

Proof: Apply Markov's inequality to the random variable $(X - E\,X)^2$, and its expectation is σ^2. □

Now we can finally formulate our main reasult:

Theorem 7 *(Goldreich - Levin) Let $f : \{0,1\}^* \to \{0,1\}^*$ an injective one-way function with $|f(x)| = |x|$. Then for every probabilistic polynomial time algorithm A there is a negligible function $\varepsilon : \mathbb{N} \to [0,1]$ such that:*

$$Pr_{x,r}[A(f(x), r) = \langle x, r\rangle] \leq \frac{1}{2} + \varepsilon(n)$$

where $x, r \in \{0,1\}^n$ and $\langle x, r\rangle = (\sum_{i=1}^{n} x_i r_i) \bmod 2$.

This means that $G(x, r) = f(x), r, \langle x, r\rangle$ is a secure pseudorandom generator that extends its input by one bit, so has stretch $l(n) = n + 1$.

Proof: For every probabilistic polynomial time algorithm A that violates the conclusion, we construct a probabilistic polynomial time algorithm B that invert the permutation f, contradicting the hypothesis that f is a one-way permutation. Specifically, if for some n,

$$Pr_{x,r}[A(f(x), r) = \langle x, r\rangle] \geq \frac{1}{2} + \varepsilon$$

then B will run in time $O(n^2/\varepsilon^2)$ and inverts the one-way permutation f on inputs of length n with probability at least $\Omega(\varepsilon)$.

According to the hypothesis, by average arguments, for at least a $\varepsilon/2$ fraction of the x's (with x fixed), the probability that $A(f(x), r) = \langle x, r\rangle$ is at least $1/2 + \varepsilon/2$. We call such an x good. We show that B with high probability inverts $f(x)$ for x good.

For example, if $Pr_r[A(f(x), r) = \langle x, r\rangle] = 1$ then it is easy to recover x from $f(x)$. Just run $A(f(x), 10\ldots0)$, $A(f(x), 01\ldots0)$, $\ldots$, $A(f(x), 00\ldots1)$ to get the first bit of x, the second bit of x, $\ldots$, the last bit of x. We find them to be exactly the products $\langle x, e_i\rangle$.

A special case:

In order to ilustrate the general idea, we first show the particular case with:

$$Pr_r[A(f(x), r) = \langle x, r\rangle] \geq 0.9.$$

In this case we cannot trust anymore the result of $A(f(x).e_i)$ because e_i could belong to the $2^n/10$ values of r for which the answer is false. But we observe that:

$$Pr_r[A(f(x), r) \neq \langle x, r\rangle \vee A(f(x), r \oplus e_i) \neq \langle x, r \oplus e_i\rangle] \leq 0.2$$

But $\langle x, r \oplus e_i\rangle = \langle x, r\rangle + \langle x, e_i\rangle$, so if r is chosen at random, and we compute $z = \langle x, r\rangle$ and $z' = \langle x, r \oplus e_i\rangle$ then $z \oplus z'$ is equal to x_i with probability at least 0.8. We can amplify this probability by taking majorities.

Algorithm B_1:

1. Choose r^1, $\ldots$, r^m independently and random in $\{0,1\}^n$. The number m is still to determine.

2. For every $1 \leq i \leq n$:

- Compute the values $z_1 = A(f(x), r^1)$, $z'_1 = A(f(x), r^1 \oplus e_i)$, $\ldots$, $z_n = A(f(x), r^m)$, $z'_m = A(f(x), r^m \oplus e_i)$.

- Guess that x_i is the majority value among $\{z_j \oplus z'_j\}$ with $1 \leq j \leq m$.

We will see that if $m = 200n$ then for every i this value will be correct with probability at least $1 - 1/(10n)$ so the word x is correct with probability at least 0.9. To prove this, let the random variable Z_j be 1 if $A(f(x), r^j) = \langle x, r^j\rangle \wedge A(f(x), r^j \oplus e_i) = \langle x, r^j \oplus e_i\rangle$, and 0 if not. Observe that the random variables Z_1, $\ldots$, Z_m are independent and $E(Z_j) \geq 0.8$ for every j. It is enough to show that with probability of $1 - 1/(10n)$ more than $m/2$ of the Z_j are equal with 1. This is equivalent for $Z = Z_1 + \cdots + Z_m$ that:

$$Pr[\ Z \leq m/2\ ] \leq 1/(10n).$$

But we know that $E(Z) \geq 0.8m$ so we need only a bound for $Pr[\ |Z - E(Z)| \geq 0.3m\ ]$. By the Chebychev's inequality:

$$Pr[\ |Z - E(Z)| \geq k\sqrt{Var(Z)}\ ] \leq 1/k^2.$$

As the random variables Z_j are independent, $Var(Z) = \sum Var(Z_j)$. As they are 0 - 1 variables, $Var(Z_j) \leq 1$, so $\sqrt{Var(Z)} \leq \sqrt{m}$. So for $k = 0.3\sqrt{m}$ one has:

$$Pr[\ |Z - E(Z)| \geq 0.3m\ ] \leq 1/(0.3\sqrt{m})^2,$$

which is smaller than $1/(10n)$ by our choice of $m = 200n$.

The general case:

If the success probability is only $1/2 + \varepsilon/2$, which is much smaller than 0.75, the idea from above seems not to work anymore.

But it does. The new idea is to show how to pick r^1, $\ldots$, r^m **pairwise** independent such that we already "know" each $\langle x, r^i\rangle$. Let k be such that $m \leq 2^k - 1$.

1. Choose k strings $s^1, \ldots, s^k$ independently at random from $\{0,1\}^m$.

2. For every $1 \leq j \leq m$ we associate a unique nonempty set $T_j \subset \{1, \ldots k\}$ in some fashion and define $r^j = (\sum_{t \in T_j} s^t) \bmod 2$.

It can be shown that the strings r^j are now pairwise independent. Moreover, $\langle x, r^j \rangle = \sum_{t \in T_j} \langle x, s^t \rangle$. So if we know the k values $\langle x, s^t \rangle$, then we can deduce the m values $\langle x, r^m \rangle$. Since $2^k = O(m)$ we can enumerate over all possible guesses for $\langle x, s^t \rangle$ in polynomial time. This leads to the following algorithm to invert f:

Algorithm B: Input $y \in \{0,1\}^m$, where $y = f(x)$ for unknown x. We suppose that x is good because it does not matters what happens if x is not good.

Operation: Let $m = 200n/\varepsilon^2$ and k be the smallest such that $m \leq 2^k - 1$. Choose $s^1, \ldots, s^k$ independently at random in $\{0,1\}^k$ and define $r^1, \ldots, r^m$ as previously. For every $w \in \{0,1\}^k$ do the follwing:

- Run the algorithm B_1 under the assumption that $\langle x, s^t \rangle = w_t$ for all $1 \leq t \leq k$. For every $1 \leq i \leq n$ we compute our guess z_j for $\langle x, r^j \rangle$ by setting $z_j = \sum_{t \in T_j} w_t$. We compute the guess $z'_j = A(y, r^j \oplus e_i)$.
- For every i, the guess for x_i is the majority value among $\{z_j \oplus z'_j\}$ with $1 \leq j \leq m$.
- For $x = x_1 \ldots x_m$ we test if $f(x) = y$. When we find such an x, we stop.

□

After having constructed a pseudo-random generator of stretch $f(n) = n+1$, we can now construct pseudorandom generators of arbitrary polynomial stretch:

Theorem 8 *If f is a one-way permutation and $c \in \mathbb{N}$ then the function G that maps $x, r \in \{0,1\}^n$ to:*

$$r, \langle f^l(x), r \rangle, \langle f^{l-1}(x), r \rangle, \ldots \langle f(x), r \rangle$$

where $l = n^c$ is a secure pseudorandom generator of stretch $l(2n) = n + n^c$.

□