Lecture 2: Computational Secrecy

Notes by Yael Kalai

MIT - 6.5620
Lecture 2 (September 8, 2025)

Warning: This document is a rough draft, so it may contain bugs. Please feel free to email me with corrections.

Recap

Recall that in the last lecture we covered the following:

1. Defined the notion of a perfect secure encryption scheme. We saw two equivalent security definitions:

    - **Shannon security:** For any probability distribution M over the plaintext space $\mathcal{M}$ and every plaintext $m \in \mathcal{M}$ and ciphertext $c \in \mathcal{C}$,

    $$\Pr[M = m] = \Pr_{k \leftarrow \mathcal{K}}[M = m | \mathsf{Enc}(k, M) = c].$$

    - **Perfect indistinguishability:** For every message $m_0, m_1 \in \mathcal{M}$ and every ciphertext $c \in \mathcal{C}$

    $$\Pr_{k \leftarrow \mathcal{K}}[\mathsf{Enc}(k, m_0) = c] = \Pr_{k \leftarrow \mathcal{K}}[\mathsf{Enc}(k, m_1) = c].$$

    Equivalently, for every $m_0, m_1 \in \mathcal{M}$ and for every adversary $\mathcal{A}$,

    $$\Pr[A(\mathsf{Enc}(k, m_b) = b] = 1/2.$$

2. Presented the One-Time Pad construction which achieves this definition. In this construction

    $$\mathcal{K} = \mathcal{M} = \mathcal{C} = \{0,1\}^n$$

    and

    $$\mathsf{Enc}(k, m) = k \oplus m \quad \text{and} \quad \mathsf{Dec}(k, c) = k \oplus c.$$

3. We argued that this scheme is only one-time secure, since given $\mathsf{Enc}(k, m_0)$ and $\mathsf{Enc}(k, m_1)$ one can learn $m_0 \oplus m_1$.

    This is a security breach since if the adv happened to know m_0 then he automatically learns m_1.[1]

4. Proved Shannon's theorem: Any scheme that achieves this definition has the drawback that $|\mathcal{K}| \geq |\mathcal{M}|$; namely, the key must grow with the number of bits encrypted!

[1] This is not only a theoretical attack, but actually similar attacks have been executed in reality.

Overcoming Shannon's Conundrum

Recall that Shannon's theorem was proved by showing that if $|\mathcal{K}| < |\mathcal{M}|$ then there is an attack on the perfect security of the scheme, but this attack takes time roughly $|\mathcal{K}|$, which is huge![2]

[2] Recall that in the attack given a ciphertext ct, we computed all the possible messages it could have encrypted, i.e. $\{\mathsf{Dec}(\mathsf{ct}, k)\}_{k \in \mathcal{K}}$, and checked if m_0 or m_1 was in this set. If both were in the set then we guessed randomly, but if only m_b is in the set then we guessed b.

Key idea: Consider only computationally bounded adversaries! One option is to limit the runtime of the adversary by, say at most 2^{256} bit operations, and construct an encryption scheme where the key consists of much more than 256 bits. Such a definition is perfectly reasonable, and in fact, this is how practitioners think about security. But it turns out that to provide theoretical guarantees, it is much easier to work with an *asymptotic* definition. Indeed, in complexity theory the notion of efficiency is asymptotic (efficient $\equiv$ polynomial time). So, instead of trying to design schemes that have some concrete level of security, we design schemes whose security level is governed by a parameter known as the *security parameter*, which we will denote by λ.[3] The idea is that the security of the scheme should increase with a larger security parameter.

[3] Often the security parameter is also denoted by n.

For now, we will think of the secret key as being a random λ-bit value, i.e., $k \leftarrow \{0,1\}^\lambda$. Later we will disentangle the security parameter from the secret key, and also allow the key to be structured and generated via a "key generation algorithm."

We restrict our adversaries to run in probabilistic polynomial time (PPT) in the security parameter λ, following the philosophy in complexity theory that polynomial-time algorithms are "feasible" while any super-polynomial-time algorithm is "infeasible."[4] More generally, we model our adversaries as polynomial-size circuits, which are equivalent to polynomial-time Turing machines with non-uniform advice. This is more general than PPT adversaries since the non-uniformity allows us to hardwire the "best" randomness for the adversary. Sometimes, we refer to poly-size adversaries as "efficient adversaries."[5]

[4] Of course, this notion of "feasible" is far from what is feasible in reality!

[5] We note that in practice, people are concerned with concrete efficiency where the security parameter is fixed to say $\lambda = 256$ and the adversary is restricted to perform at most 2^{128} bit operations.

Our cryptographic schemes will also be restricted to run in time polynomial in the security parameter (as otherwise they will be considered infeasible to implement). We always consider adversaries that are more powerful than the honest players. For example, we may consider honest players that run in time λ^2 but require security to hold against all PPT adversaries.

Definition 1 (Take 1:). A *computationally secure* encryption scheme is associated with a key space $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda \in \mathbb{N}}$, a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$, and a ciphertext space $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$, where there exists a polynomial $\ell : \mathbb{N} \to \mathbb{N}$ such that for every $k \in \mathcal{K}_\lambda$, $m \in \mathcal{M}_\lambda$ and $c \in \mathcal{C}_\lambda$ it holds that $|k|, |m|, |c| \leq \ell(\lambda)$.[6]

[6] Often $\mathcal{K}_\lambda = \{0,1\}^\lambda$. In addition, often $\mathcal{M}_\lambda$ does not depend on λ. For example, as we will see, for some of our schemes $\mathcal{M} = \{0,1\}$, independent of λ.

It is also associated with two polynomial time algorithms $\mathsf{Enc} = \{\mathsf{Enc}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathsf{Dec} = \{\mathsf{Dec}_\lambda\}_{\lambda\in\mathbb{N}}$, such that:

1. $\mathsf{Enc}_\lambda : \mathcal{K}_\lambda \times \mathcal{M}_\lambda \to \mathcal{C}_\lambda$.
2. $\mathsf{Dec}_\lambda : \mathcal{K}_\lambda \times \mathcal{C}_\lambda \to \mathcal{M}_\lambda$.

It satisfies the following two properties:

1. **Correctness:** For every $\lambda \in \mathbb{N}$, every $k \in \mathcal{K}_\lambda$, and every $m \in \mathcal{M}_\lambda$
$$\mathsf{Dec}_\lambda(k, \mathsf{Enc}_\lambda(k, m)) = m.$$
2. **Computational security:** For every poly-size adversary $\mathcal{A}$, every $\lambda \in \mathbb{N}$ and every $m_0, m_1 \in \mathcal{M}_\lambda$,
$$\Pr[\mathcal{A}(\mathsf{Enc}_\lambda(k, m_b)) = b] = 1/2.$$

Remark. We often omit the security parameter from Enc_λ and Dec_λ, as we can assume without loss of generality that it can be inferred from the key k.

Remark. Notice that the security definition requires security to hold for every two messages $m_0, m_1 \in \mathcal{M}_\lambda$, even adversarially chosen. We let the adversary have control of as many things as possible to make our security definition as strong as possible.

Is this definition achievable for $|\mathcal{M}_\lambda| > |\mathcal{K}_\lambda|$? It turns out that the it is not! In fact, it is subject to the same Shannon impossibility result! Namely, the adversary $\mathcal{A}$ given a ciphertext $c \in \mathcal{C}_\lambda$ will choose a random $k \leftarrow \mathcal{K}_\lambda$, and compute $m = \mathsf{Dec}(k, c)$. If there exists $b \in \{0, 1\}$ such that $m = m_b$ then output b, and otherwise output a random $b' \leftarrow \{0, 1\}$. This adversary will guess b correctly with probability $\geq 1/2 + 2^{-\lambda}$, as long as
$$\{\mathsf{Enc}(k, m_0) : \ k \in \{0, 1\}^\lambda\} \neq \{(\mathsf{Enc}(k, m_1) : \ k \in \{0, 1\}^\lambda\}$$
and by the correctness of the encryption scheme, and assuming that $|\mathcal{M}_\lambda| > |\mathcal{K}_\lambda|$, there must exist $m_0, m_1 \in \mathcal{M}_\lambda$ that satisfy the equation above.

To get around this impossibility result we just need to allow the adversary to have a tiny advantage.

Definition 2. A function $\mu : \mathbb{N} \to \mathbb{N}$ is said to be negligible if for every polynomial $p : \mathbb{N} \to \mathbb{N}$ there exists $\Lambda \in \mathbb{N}$ such that for every $\lambda \geq \Lambda$,
$$\mu(\lambda) < \frac{1}{p(\lambda)}.$$

Intuitively, an event that occurs with negligible probability looks to a poly-time algorithm like it never occurred.

Examples

1. $\mu(\lambda) = 2^{-\lambda}$ is negligible.

2. $\mu(\lambda) = 1/\lambda^2$ is non-negligible.

Definition 3. A *computationally secure* encryption scheme is defined as above but where computational security is defined as follows:

Computational security: For every poly-size adversary $\mathcal{A}$ there exists a negligible function $\mu : \mathbb{N} \to \mathbb{N}$ such that for every $\lambda \in \mathbb{N}$, every $m_0, m_1 \in \mathcal{M}_\lambda$,

$$\Pr[\mathcal{A}(\mathsf{Enc}(k, m_b) = b] \leq 1/2 + \mu(\lambda).$$

Remark. We often refer to μ, which is the difference between the adversary's success probability and $1/2$, as the adversary's advantage.

Notice that the runtime and success probability are measured as a function of the security parameter. Importantly:

- Honest algorithms run in fixed polynomial time in λ.
- Adversaries may run in (arbitrary) polynomial time in λ, and should have only negligible advantage.
- Honest algorithms are uniform, whereas the adversary is allowed to be non-uniform.

Can we construct an encryption scheme that achieves this definition??[7]

[7] We will construct an encryption scheme that achieves an even stronger definition, where the messages $(m_{b,1}, \ldots, m_{b,\ell})$ can be chosen adaptively depending on the ciphertexts. Namely, We allow the adversary to have black-box access to $\mathsf{Enc}(k, \cdot)$ and guarantee that for every m_0, m_1 that the adversary did not query it still holds that he cannot distinguish between $\mathsf{Enc}(k, m_0)$ and $\mathsf{Enc}(k, m_0)$, even with all the ciphertexts it obtained from the oracle.

Not if $\mathsf{NP} = \mathsf{P}$! So, if we construct a computationally secure encryption scheme it will immediately imply that $\mathsf{NP} \neq \mathsf{P}$, so this seems out of reach. Instead, we will construct such a scheme and prove that it is secure under some computational assumption. Namely, we will only prove conditional security.

One could hope to prove the security of our scheme under the assumption that $\mathsf{NP} \neq \mathsf{P}$. We do not know how to that. But we do have schemes that are secure assuming the existence of *one-way functions*, which is known to be equivalent to the existence of a computationally secure encryption scheme.

One-way functions

Intuitively, a one-way function is a function that is easy to compute, but hard to invert. Namely, you can compute it in one direction, but not the other.

Definition 4. A function $f\{0,1\}^* \to \{0,1\}^*$ is one-way if it satisfies the following two conditions:

- **Easy to compute:** There exists a poly-time algorithm $\mathcal{B}$ such that for every $x \in \{0,1\}^*$, $\mathcal{B}(x) = f(x)$.
- **Hard to invert:** For every poly-size adversary $\mathcal{A}$ there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$

$$\Pr_{x \leftarrow \{0,1\}^\lambda}[\mathcal{A}(f(x)) = x' \text{ s.t. } f(x') = f(x)] \leq \mu(\lambda)$$

Our first cryptographic tool: Pseudo Random Generator (PRG)

Our goal is to encrypt messages of length n with a key of length $\lambda << n$. Suppose that we lived in a magical world where we could take one secret key $k \leftarrow \{0,1\}^\lambda$ and stretch it into a random key of length n! Then we could use this stretched key as a one-time pad, and hence securely encrypt a message of length n!

You may think that it is impossible to generate randomness out of thin air! But using cryptographic magic it is possible! This is one of the most beautiful and magical gifts that cryptography bestowed upon us.