CAS CS 538. Lecture Notes on Indistinguishability (September 10, 2019 lecture by Leo Reyzin)

1 First attempt at relaxing perfect secrecy: allow a probability gap

As we said in the last class, if we want to allow for shorter keys, we need to relax the requirement of perfect secrecy. But how? Here's our first attempt.

Define an *advantage* or *bias* of an adversary $\mathcal{A}$ against two libraries $\mathcal{L}_{\mathsf{left}}$ and $\mathcal{L}_{\mathsf{right}}$ as the positive difference between the probabilities that the adversary outputs 1 when linked with the two libraries:

$$|\Pr[\mathcal{A} \diamond \mathcal{L}_{\mathsf{left}} \Rightarrow 1] - \Pr[\mathcal{A} \diamond \mathcal{L}_{\mathsf{right}} \Rightarrow 1]|\,.$$

For example, consider the one-time-pad but with 0^λ key disallowed. Consider the following adversary: query two different message (m_L, m_R) and output 1 if the output is m_L. It is not hard to show that the advantage of such an adversary is $\epsilon = 1/(2^\lambda - 1)$. This is the strongest one-query adversary we can build (this requires a proof, which is a good exercise). With more queries, the adversary can increase its advantage (think about how — another good exercise).

The notion of *interchangeability* [Ros19, Definition 2.5] says that the for every adversary, the advantage is 0. We can relax this notion and allow a small advantage. The smaller the advantage, the more secure the scheme. Advantage 1 means no security at all.

Definition 1. Let $\mathcal{L}_{\mathsf{left}}$ and $\mathcal{L}_{\mathsf{right}}$ be two libraries with a common interface. We say $\mathcal{L}_{\mathsf{left}} \approx_\epsilon \mathcal{L}_{\mathsf{right}}$ if for all programs $\mathcal{A}$ that output a single bit,

$$|\Pr[\mathcal{A} \diamond \mathcal{L}_{\mathsf{left}} \Rightarrow 1] - \Pr[\mathcal{A} \diamond \mathcal{L}_{\mathsf{right}} \Rightarrow 1]| \le \epsilon\,.$$

Definition 2. A symmetric-key encryption scheme Σ is ϵ-insecure if $\mathcal{L}^{\Sigma}_{\mathsf{ots\text{-}L}} \approx_\epsilon \mathcal{L}^{\Sigma}_{\mathsf{ots\text{-}R}}$.

Unfortunately, this relaxation doesn't buy you much: instead of key length being at least equal to message length, it will be at least half the message length, as the following theorem shows.

Theorem 1. *Let Σ be encryption scheme for which* $\mathrm{Enc}(k, m)$ *is deterministic. If Σ is ϵ-insecure for $\epsilon < 1$, then $|\Sigma.\mathcal{K}| \ge \sqrt{|\Sigma.\mathcal{M}|}$.*

Thus, for example, if key length is λ, then $|\Sigma.\mathcal{K}| = 2^\lambda$ and thus $|\Sigma.\mathcal{M}|$ cannot be more than $2^{2\lambda}$, and thus the maximum message length is at most 2λ. So to encrypt a gigabyte of message, you need a half-gigabyte key.

The proof for encryption schemes with randomized Enc is more complicated, and we will not do it here. Note that we have not studied any such schemes so far (picking a random key is essential for encryption, but is part of KeyGen, not of Enc).

We now prove Theorem 1.

Proof. Suppose, for purposes of contradiction, that $|\Sigma.\mathcal{K}| < \sqrt{|\Sigma.\mathcal{M}|}$, i.e., $|\Sigma.\mathcal{K}|^2 < |\Sigma.\mathcal{M}|$.

The idea is to strengthen the proof of Shannon's impossibility theorem form last class, so that $\Pr[\mathcal{A} \diamond \mathcal{L}^{\Sigma}_{\mathsf{ots\text{-}L}} \Rightarrow 1] = 1$, while $\Pr[\mathcal{A} \diamond \mathcal{L}^{\Sigma}_{\mathsf{ots\text{-}R}} \Rightarrow 1]$ remains at 0. We will do so by replacing c_0

with all possible ciphertexts that could be encryptions of m_L. So, pick an arbitrary m_L. Let $C_0 = \{\mathrm{Enc}(k, m_L) \,|\, k \in \Sigma.\mathcal{K}\}$. C_0 contains every possible ciphertext that m_L could produce. Note that $|C_0| \leq |\Sigma.\mathcal{K}|$, because $\mathrm{Enc}(k, m_L)$ is deterministic, and produces one ciphertext for each value k. Let $P = \{\mathrm{Dec}(k, c) \,|\, k \in \Sigma.\mathcal{K}, c \in C_0\}$. P contains every possible plaintext that could end up in C_0. Observe that $|P| \leq |C_0| \cdot |\Sigma.\mathcal{K}| \leq |\Sigma.\mathcal{K}|^2 < |\Sigma.\mathcal{M}|$. So let m_R be a message in $\Sigma.\mathcal{M} - P$.

Build a distinguisher $\mathcal{A}$ as follows: query (m_L, m_R) to get c; output 1 if $c \in C_0$ and 0 otherwise. Then

$$\Pr[\mathcal{A} \diamond \mathcal{L}^{\Sigma}_{\mathsf{ots\text{-}L}} \Rightarrow 1] = 1\,,$$

because C_0 contains every possible ciphertext m_L could produce.

At the same time,

$$\Pr[\mathcal{A} \diamond \mathcal{L}^{\Sigma}_{\mathsf{ots\text{-}R}} \Rightarrow 1] = 0\,,$$

because an encryption of m_R could never end up in C_0 by correctness of encryption, because nothing in C_0 decrypts to m_R.

Thus, the advantage of $\mathcal{A}$ is 1, which contradicts the ϵ-insecurity of Σ with $\epsilon < 1$. □

Thus, a probability difference alone is not enough.

2 Second Attempt: add a running time bound

While the distinguisher from the proof of Theorem 1 achieves a very good advantage, it also has a potentially very high running time and code size, because it needs to consult an exponential-size set C_0. In reality, no attacker has unlimited time. Perhaps we can salvage something if we limit the attacker's running time.

Definition 3. We say $\mathcal{L}_{\mathsf{left}} \approx_{t,\epsilon} \mathcal{L}_{\mathsf{right}}$ if for all $\mathcal{A}$ whose running time (plus code size) is at most t,

$$|\Pr[\mathcal{A} \diamond \mathcal{L}_{\mathsf{left}} \Rightarrow 1] - \Pr[\mathcal{A} \diamond \mathcal{L}_{\mathsf{right}} \Rightarrow 1]| \leq \epsilon\,.$$

In the previous section, in Theorem 1, we showed that allowing nonzero advantage alone (without limiting running time) is not very useful for overcoming impossibility results. Similarly, a running-time bound alone (without allowing a nonzero advantage) is not very useful for overcoming impossibility results, because the attacker from Shannon's impossibility theorem from last class has a very small running time (it just needs to test equality of two ciphertexts) and achieves a non-zero advantage. So we need to both limit the running time and allow a nonzero advantage to avoid the impossibility results.

The Joy of Cryptography defines indistinguishability [Ros19, Definition 4.5] by saying that the distinguishing advantage of every polynomial-time algorithm is negligible. It's a good definition, but sometimes you want to be more precise. Our Definition 3 above is a more precise definition of indistinguishability, which takes into account the specific running time t (instead of any polynomial) and distinguisher advantage ϵ (instead of any negligible function). As we will discuss later in the course, precision is particularly important when you are deciding on setting parameters such as key lengths, because you have to choose a specific length in order to provide security against a specific class of adversaries. However, [Ros19, Definition 4.5] is easier to work with, especially for initial intuition, because you don't have to write down ϵ and t all the time.

References

[Ros19] Mike Rosulek. *The Joy of Cryptography.* 2019. `http://web.engr.oregonstate.edu/~rosulekm/crypto/`.