Computational Security + Pseudorandomness

Recommended reading:
KL, Sections 3.1, 3.2.1, 3.3, 6.8

Last time: OTP is perfectly secure, but requires large keys!
Also, we can't do better!

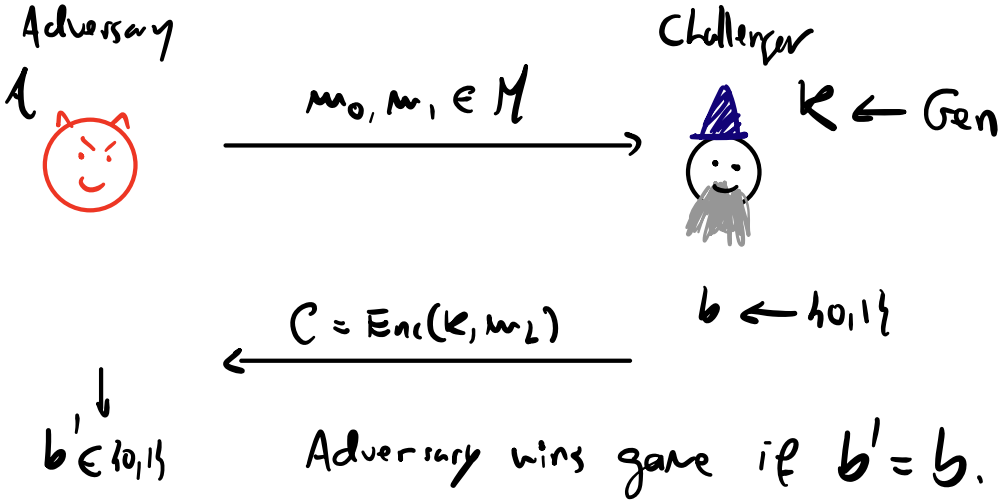
Let's rethink our notion of security for encryption.

Perfect security holds against <u>any</u> adversary, running for however long they need.

But we'd be fine with security against all adversaries that run in "time" at most t, for some possibly very large t

How do we formalize this?

A game-based perspective on security



Thm: (Enc, Dec, Gen) is perfectly secure if and only if for any adversary A we have $Pr(b'=b) = 1/2$

Can't do better than guessing at random

Proof: Homework!

Here's a possible definition of security for "computationally-bounded" adversaries.

An encryption scheme is (t, ε)-secure if any adversary "running in time t" wins the game above with probability at most $\frac{1}{2} + \varepsilon$.

What does "running in time t" mean?

Many reasonable ways to define it. Formally, we'll take this to mean a Turing machine, with access to a tape filled with random bits, running in t steps.

But for this course it's perfectly fine if you think about it as a program in your favorite language, with access to randomness.

In this course we'll take an asymptotic approach to security.

It helps us focus on the most interesting foundational aspects.

Nevertheless, concrete notions of security are very relevant too!

Asymptotic security

Introduce a security parameter $n \in \mathbb{N}$. Everything will be parametrized by n.

"Efficient adversary" = PPT algorithm → runs in time $\leq p(k)$ on input of length k, for some polynomial p.

- PPT = probabilistic polynomial-time
- ↗ matches complexity theory
- ↙ adversaries still much more powerful than good guys!!

"Negligible attack success probability" = decays faster than $1/p(n)$, for all polynomials p, as $n \to \infty$.

We redefine encryption schemes in terms of n.

→ Gen is a PPT algorithm that receives n in unary as input and outputs a key $k \leftarrow Gen(1^n)$

↳ technicality to ensure Gen runs in time polynomial in n

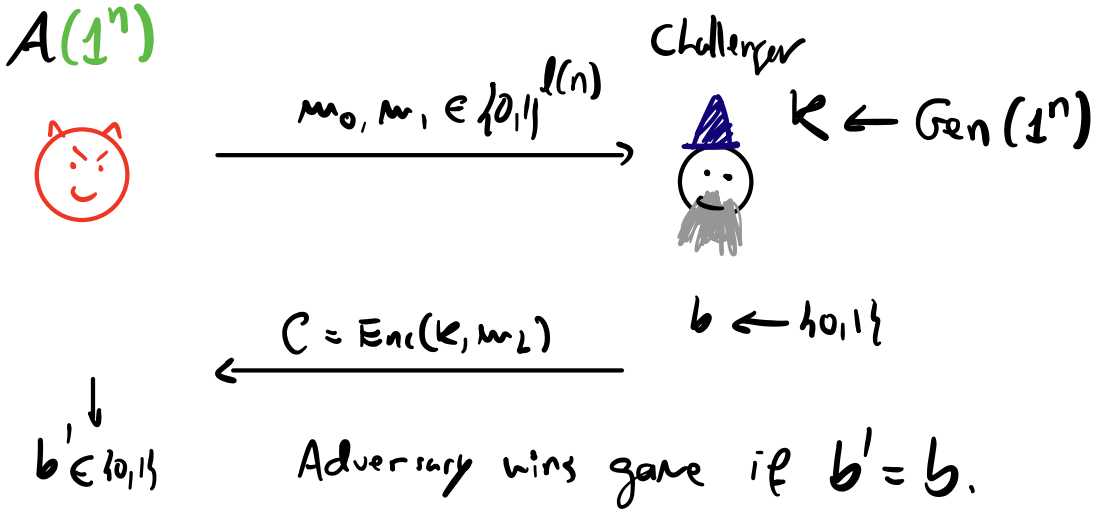
→ Enc is a PPT algorithm that receives a message $m \in \{0,1\}^{\ell(n)}$ and k, and outputs $c = Enc(k, m)$.

→ Dec is a PPT algorithm that receives k and c, and outputs $m = Dec(k, c)$.

Correctness: Exactly as before.

(Computational) Security: For any PPT adversary $\mathcal{A}$ there exists a negligible function $\varepsilon(n)$ such that $\mathcal{A}$ wins the following game with probability at most $1/2 + \varepsilon(n)$.

- negligible function: for any polynomial p there is n^* s.t. $\varepsilon(n) \leq \frac{1}{p(n)}$ $\forall n \geq n^*$.
- for example, 2^{-n}, $2^{-\sqrt{n}}$, $n^{-\log n}$ are all negligible



Adversary wins game if $b' = b$.

It's not obvious, but this definition of computational security ensures that no PPT adversary can learn any "non-trivial" property of the message from the ciphertext → this is called semantic security

see KL, Section 3.2.2

Let's establish a baby version of semantic security.

Thm: Suppose that (Enc, Dec, Gen) is computationally-secure. Then, for any PPT algorithm A and index i there is a negligible function $\varepsilon(n)$ such that

$$Pr\left[A(1^n, Enc(k, m)) = m_i\right] \leq \frac{1}{2} + \varepsilon(n),$$

where $k \leftarrow Gen(1^n)$, $m \leftarrow \{0,1\}^{\ell(n)}$.

Proof: By reduction!

Suppose that there is an index i and a PPT algorithm A^* such that $Pr\left[A^*(1^n, Enc(k, m)) = m_i\right] \geq \frac{1}{2} + \frac{1}{p(n)}$ for some polynomial p, with $k \leftarrow Gen(1^n)$, $m \leftarrow \{0,1\}^{\ell(n)}$.

We will use A^* to break computational security of (Enc, Dec, Gen).

Intuition: Choose random m_0, m_1 s.t. $m_{0,i} = 0 \neq 1 = m_{1,i}$, use A^* to guess $m_{b,i} = b$.

Consider the following adversary A for the "distinguishing ciphertexts" game:

- $A(1^n)$ samples $m_0 \leftarrow I_0 = \{m : m_i = 0\}$, $m_1 \leftarrow I_1 = \{m : m_i = 1\}$ and sends (m_0, m_1) to the challenger.
- Once it receives ciphertext c back, runs $A^*(1^n, c)$.
- If $A^*(1^n, c)$ outputs b^*, then A outputs $b' = b^*$.

First, note that A is PPT if A^* is PPT.

Second, we have

$$\begin{aligned}
\Pr(A \text{ wins game}) &= \Pr(b' = b) \\
&= \Pr(b^* = b) \\
&= \frac{1}{2} \Pr_{m_0 \leftarrow I_0}(A^*(1^n, \mathrm{Enc}(k, m_0)) = 0) + \frac{1}{2} \Pr_{m_1 \leftarrow I_1}(A^*(1^n, \mathrm{Enc}(k, m_1)) = 1) \\
&= \Pr_{m \leftarrow \{0,1\}^{p(n)}}(A^*(1^n, \mathrm{Enc}(k, m)) = m_i) \\
&\geq \frac{1}{2} + \frac{1}{p(n)}.
\end{aligned}$$

This contradicts the fact that the scheme is computationally secure.

∎

Before we try to construct computationally-secure schemes, let's discuss pseudorandomness. → KL, Section 3.3 and 6.8

There are notions of "indistinguishability" present in the definitions and results above.

Let's define a more general notion.

Def. (computational indistinguishability)

Let $(X_n)_{n\in\mathbb{N}} = X_1, X_2, \ldots$ and $(Y_n)_{n\in\mathbb{N}} = Y_1, Y_2, \ldots$ be two sequences of random variables.

We say that (X_n) and (Y_n) are computationally indistinguishable and write $X_n \approx Y_n$ if for any PPT algorithm D there is a negligible $\varepsilon(n)$ such that

$$|\Pr(D(1^n, X_n)=1) - \Pr(D(1^n, Y_n)=1)| \leq \varepsilon(n).$$

Now, something is "pseudorandom" if it is computationally indistinguishable from the uniform distribution

Here's how we'll construct encryption schemes with shorter keys:
We use a short key to generate a long pseudorandom string, and then use this string as the OTP key.

For this we need an object that expands randomness into longer pseudorandomness

PSEUDORANDOM GENERATORS

Def. A pseudorandom generator (PRG) is a function $G: \{0,1\}^* \to \{0,1\}^*$ computed by a deterministic polynomial time algorithm such that:

- G maps n-bit strings to $\ell(n)$-bit strings, for $\ell(n) > n$.
- If $U_n \leftarrow \{0,1\}^n$ and $U_{\ell(n)} \leftarrow \{0,1\}^{\ell(n)}$, then $G(U_n) \approx U_{\ell(n)}$.

Encryption with shorter keys using PRG with expansion factor $\ell(n)$

- $Gen(1^n)$ samples $k \leftarrow \{0,1\}^n$
- For $m \in \{0,1\}^{\ell(n)>n}$, $Enc(k,m) = G(k) \oplus m$
- $Dec(k,c) = G(k) \oplus c$.

Thm: If G is a PRG, then (Enc, Dec, Gen) is computationally-secure.

Proof: By reduction!

Suppose there is a PPT adversary A that breaks (Enc, Dec, Gen), in the sense that A wins the distinguishing game with prob. $\geq \frac{1}{2} + \frac{1}{p(n)}$ for some polynomial $p(n)$.

We use A to create a PPT distinguisher D such that $|Pr[D(G(U_n))=1] - Pr[D(U_{\ell(n)})=1]| \geq \frac{1}{p(n)}$, a contradiction.

On input 1^n, w, D proceeds as follows:

- Runs $A(1^n)$ to get messages $m_0, m_1 \in \{0,1\}^{\ell(n)}$
- Samples $b \leftarrow \{0,1\}$ & computes $c = Enc(w, m_b)$.
- Runs $b' \leftarrow A(1^n, m_0, m_1, c)$.
- If $b' = b$, outputs 1, else outputs 0.

If $w \leftarrow U_{\ell(n)}$, then $P_r(b'=b) = 1/2$, since $Enc(w, m_b)$ is independent of b.

If $w \leftarrow G(U_n)$, then $Enc(w, m_b) = G(k) \oplus m_b$ is the real encryption, and so

$$P_r(b'=b) = Pr(\text{A wins real dist. game}) \geq \frac{1}{2} + \frac{1}{p(n)}$$

This means that

$$Pr\left(D(1^n, G(U_n)) = 1\right) - Pr\left(D(1^n, U_{\ell(n)}) = 1\right)$$

$$\geq \left(\frac{1}{2} + \frac{1}{p(n)}\right) - \frac{1}{2}$$

$$= \frac{1}{p(n)}$$ ▨

So now all that we need to do is construct PRGs!

Hmm.....