Today we introduce computation-based cryptography. We define security of encryption in the computational model, consider a password example, and introduce pseudo-random bit generators.

## 1 Secure Encryption

According to the Shannon bound, a perfect encryption requires that the key space be as big as the message space. But you can’t reuse the key (for example, consider a one-time pad), so you need one key per message.

MACs and pairwise independent hash functions are effective for sending a single message. But for a sequence of messages, they’re not enough. For example, if you use pairwise independent hash functions to authenticate 2 messages, then an $ma + b$ code can be broken.

To get around these problems, we use computation-based cryptography and make certain assumptions on time and space of available computation. Let us define Secure Encryption as requiring that the statistical distribution of any two messages should be the same to an adversary who is “computationally bounded.” That is, given security parameter $\lambda$, the adversary is computationally bounded by a function in $\lambda$. Obviously, security increases as $\lambda$ increases (assuming our function of $\lambda$ is an increasing function).

Consider an encryption scheme defined by the following three algorithms:

- **Key Generation Algorithm:** $k \leftarrow_R \text{KEYGEN}(1^\lambda)$: This could be as simple as just generating a random bitstring of a certain length.
- **Encryption Algorithm:** $c \leftarrow_R E_k(1^\lambda, m)$.
- **Decryption Algorithm:** $m \leftarrow_R D_k(1^\lambda, c)$.

All these algorithms run in polynomial time on inputs $(\lambda, m)$, and may be probabilistic or deterministic. As usual, we have the correctness condition: $\forall \lambda \in \mathbb{N}$, for all possible outputs of $\text{KEYGEN}(1^\lambda)$, for all (or almost all) $m \in M$, $D(1^\lambda, E_k(1^\lambda, m)) = m$.

We can now define security with these functions via an attack game. There are 2 players: the user $A$ and the environment $\text{ENV}$. Both know the security parameter $1^\lambda$, which is part of the input. $A$, using some (possibly probabilistic) strategy, chooses one of two messages $m_0, m_1$, and sends it to $\text{ENV}$. In a $b = 0$ game, $\text{ENV}$ returns the encryption of $m_0$; in a $b = 1$ game, it returns the encryption of $m_1$. (Only $\text{ENV}$ knows if it is a $b = 0$ game or a $b = 1$ game.)
We require that no efficient \( A \) should be able to determine whether this is a \( b = 0 \) game or a \( b = 1 \) game. More precisely: For \( b = 0, 1 \) define

\[
\gamma_{A,b}(\lambda) = \Pr(\mathcal{A} \text{ outputs } 1 \text{ in game } b \text{ for given } \lambda).
\]

Define the distinguishing advantage of \( A \), \( \text{Dist}_A(\lambda) = \|\gamma_{A,0}(\lambda) - \gamma_{A,1}(\lambda)\| \). We can now view security in this framework as the requirement that \( \text{Dist}_A(\lambda) \) is negligible to a player \( A \) with efficient computation.

We’ll define computational efficiency and negligible distance momentarily, but first we note a useful alternative characterization of this model: define a single game between \( A \) and \( \text{Env} \). As before, each player get \( 1^\lambda \), and the environment chooses \( k \) at random via \( \text{KeyGen} \). But in this game, the environment selects \( b \) at random, encrypts \( c^* = E_k(1^\lambda, m_b) \), and outputs \( \hat{b} \in \{0, 1\} \).

For this new characterization, we define the guessing advantage of \( A \) to be \( \text{Guess}_A = \frac{1}{2} - \Pr[b = \hat{b}] \), and we can show that \( \text{Guess}_A = 1/2 \text{ Dist}_A \).

Now we define efficient and negligible.

- Efficient: computable in probabilistic polynomial time (more formally, by a BPP algorithm).
- Given \( f : \mathbb{N} \to \mathbb{R} \), the following are equivalent definitions of negligible functions:
  \[
  f < 1/poly(). \text{ That is, } f \text{ goes to } 0 \text{ faster than the inverse of any polynomial, i.e. } \forall g \text{ polynomial, } \lim_{x \to \infty} fg = 0.
  \]
  \[
  \forall c > 0, \exists n_0 \in \mathbb{N} \text{ s.t. } \forall n > n_0, \|f(n)\| < n^{-c}.
  \]
  \[
  \forall q, q \text{ poly, } f(n) < 1/q(n) \text{ for all sufficiently large } n.
  \]

If \( f, g \) are negligible then \( f, g, fg \) are negligible. If \( p \) is any polynomial, then \( fp \) is also negligible.
2 A Password Example

A user selects a password from some distribution $D$. She encrypts her password using a secure encryption scheme. An eavesdropper gets the encrypted password and attempts to login.

Security should imply that an eavesdropper’s success probability is not much more than that of an adversary who knows nothing about the password.

Stated more rigorously: User $U$ chooses a password $\text{passwd} \in_R D$, and generates a ciphertext $c = E_k(\text{passwd})$. An eavesdropper $A$ sees the ciphertext, and uses it to take a guess ($\text{passwd}'$) as to what the true password is. He then tries to convince a third party - perhaps a server - that he knows the password. Here, $\text{SUCC}_A(\lambda)$ is the probability that $A$ will have a successful login.

Let’s go back to the two game model, and consider its relationship to game switching: take game 0, and transform it into a new game 1, where instead of encrypting the user’s passwd, we encrypt some dummy string. Let $\text{SUCC}'_A(\lambda)$ = success probability in game 1 (where success is defined the same way as before). We can make two claims concerning this game:

- Claim 1: $|\text{SUCC}_A(\lambda) - \text{SUCC}'_A(\lambda)|$ is negligible.
- Claim 2: The success probability in game 1 of $A$ is min-entropy. That is, it is at most $\max_p \Pr_D(p) = \gamma(D)$.

Combining Claim 1 and 2 will imply that $\text{SUCC}_A \leq \gamma(D) + \mu$, where $\mu$ is a negligible function.

Claim 2 is obvious. We prove claim 1 by constructing a hybrid adversary, composed of bits and pieces of game 0 and game 1. We will construct a new adversary $\tilde{A}$, and show that $|\text{SUCC}_A(\lambda) - \text{SUCC}'_A(\lambda)| = \text{DIST}_A(\lambda)$ where $\tilde{A}$ runs in time ”not much more” than $A$. Since $A$ is a poly-time adversary to a secure encryption system, its distinguishing advantage is negligible.

How does $\tilde{A}$ work? See the figure.
\[ m_0 \leftarrow_D R \]
\[ m_1 \leftarrow_d \text{dummy} \]
\[ \text{run A on C} \]
\[ \text{and compute passw'} \]
\[ \text{output 1 if passw = passw', 0 otherwise} \]

\[ \tilde{A} \text{ chooses } m_0 \leftarrow_R D, m_1 \leftarrow \text{“dummy”}, \tilde{A} \text{ sends } m_0, m_1, \text{ receives } c. \tilde{A} \text{ runs algorithm A on input } c, \text{ computing passwd’}. \text{ It outputs 1 if passwd = passwd’, 0 otherwise.} \]

Let \( \gamma_{\tilde{A},0} = \text{Succ}_A, \gamma_{\tilde{A},1} = \text{Succ}'_A \).

Using \( A \) as a subroutine, assuming that \( A \) was polynomial time, we have \( \tilde{A} \) is also polynomial time.

\( A \) was poly-time implies \( \tilde{A} \) is also poly-time, as required.

We have \( \| \text{Succ}_A - \text{Succ}'_A \| = \| \gamma_{\tilde{A},0} - \gamma_{\tilde{A},1} \| = \text{Dist}_A(\lambda) \) is negligible, and the proof is complete.

### 3 Pseudo-Random Bit Generators

How does one build a secure encryption scheme? We know that by Shannon’s theorem, we can’t get by with short keys if we want information-theoretic security. But if we only want computational security, we’re in luck.

To encrypt, we need to ‘stretch’ a short random bitstring into a ‘long’ pseudorandom bitstring that looks random to a computationally bounded observer. Somewhat more formally, we define a pseudorandom bit generator (PRBG) to be a function \( G: \{0,1\}^t \mapsto \{0,1\}^{t'} \).

We consider two distributions:

- \( P = G(s), s \leftarrow_R \{0,1\}^t \)
- uniform distribution \( R = U_{\nu} \) on \( \{0,1\}^{t'} \)

The probability that a random element of \( R \) is in \( P \) is at most \( 2^{t-t'} \), by a counting argument.

More formally, we introduce a more general notion of computationally indistinguishability.
Let \( \{X_\lambda\}_{\lambda \in \mathbb{N}} \) and \( \{Y_\lambda\}_{\lambda \in \mathbb{N}} \) be families of distributions, where \( X_\lambda \) and \( Y_\lambda \) take values in \( \{0, 1\}^{\leq m(\lambda)} \), and \( m(\lambda) \) is bounded by a polynomial in \( \lambda \).

We say that \( X = \{X_\lambda\} \) and \( Y = \{Y_\lambda\} \) are computationally indistinguishable if for every probabilistic polynomial-time adversary \( A \), \((X, Y) \leftarrow \{0, 1\}^{m(\lambda)}\),

\[
\| \Pr(A(X_\lambda) = 1) - \Pr(A(Y_\lambda) = 1) \|
\]

**Definition 1.** A pseudo-random bit generator consists of a function \( G(1^\lambda, s) \), \( s \in \{0, 1\}^{l(\lambda)} \), \( G(1^\lambda, s) \in \{0, 1\}^{l'(\lambda)} \) where \( l \) and \( l' \) are bounded by a polynomial in \( \lambda \).

We define \( P_\lambda \) as the output of \( G \) on input \((1^\lambda, s)\) where \( s \leftarrow_R \{0, 1\} \), and we define \( R_\lambda \) to be the uniform distribution on \( \{0, 1\} \).

Security means these two distributions \( P_\lambda, R_\lambda \) are computationally indistinguishable.

Now we consider the properties of computational indistinguishability. Let \( X, Y \) be families of probability distributions on bitstrings. Write \( X \equiv Y \) if \( X \) and \( Y \) are computationally indistinguishable.

We claim that computational indistinguishability is an equivalence relation. The reflexive and symmetric properties are clear. To establish transitivity, we need to show \( X \equiv Y, Y \equiv Z \Rightarrow X \equiv Z \). We have, by the triangle inequality,

\[
\| \Pr(A(X) = 1) - \Pr(A(Y) = 1) \| + \| \Pr(A(Y) = 1) - \Pr(A(Z) = 1) \| \\
\leq \| \Pr(A(Z) = 1) - \Pr(A(X) = 1) \|
\]

The sum of two negligible functions is negligible, so we have established transitivity.

Next few lectures: We consider using PRBG’s as one-time pads, and how to build PRBG’s from other more primitive objects for which assumptions are more reasonable. One-way functions, one-way permutations, and hardcore bits, Pseudo-random permutations, pseudo-random functions, etc.