Comp Security and Pseudo-Random Generators
Where do we stand?

- We defined the notion of perfect secrecy
- We proved that the one-time pad achieves it!
- We proved that the one-time pad is optimal!
  - i.e. we cannot improve the key length
- We saw drawbacks of 1-time pad
- If we want to do better we need to relax the definition
  - But in a meaningful way...
Perfect Secrecy

- Requires that **absolutely no information** about the plaintext is leaked, even to eavesdroppers with **unlimited computational power**
  - Seems unnecessarily strong
Computational secrecy

- Would be ok if a scheme leaked information with tiny probability to eavesdroppers with bounded computational resources

- i.e. we can relax perfect secrecy by
  - Allowing security to “fail” with tiny probability
  - Restricting attention to “efficient” Eves.
Roadmap

- We will give an alternate (but equivalent) definition of perfect secrecy
  - Using a Game!
Roadmap

- We will give an alternate (but equivalent) definition of perfect secrecy
  - Using a Game!. **Warning:** Most math games are not fun :-(

- That definition has a natural relaxation
**Formal Def of Perfect Indistinguishability**

$\Pi = (\text{GEN}, \text{ENC}, \text{DEC})$ is an enc sch. Message space $\mathcal{M}$.

**Game:** Alice and Eve are the players. Alice has full access to $\Pi$.

1. Eve chooses $m_0, m_1 \in \mathcal{M}$

2. Alice computes $k \leftarrow \text{GEN}(1^n)$

3. Alice chooses $m \in \{m_0, m_1\}$ and $c \leftarrow \text{ENC}_k(m)$

4. Alice sends $c$ to Eve.

5. Eve outputs $m_0$ or $m_1$, hoping that her output is $\text{DEC}_k(c)$.

6. Eve wins if she is right.

Can Eve win this game with Probability over $\frac{1}{2}$?
Formal Def of Perfect Indistinguishability

- Easy to succeed with probability $\frac{1}{2}$

- $\Pi$ is perfectly indistinguishable if for all Eve (algorithms), it holds that

  $$\Pr[\text{Eve Wins}] = \frac{1}{2}$$

- Note: No time or space limits on Eve.
Perfect Indistinguishability

- **Fact:** $\Pi$ is perfectly indistinguishable $\iff$ $\Pi$ is perfectly secret

- i.e. perfect indistinguishability is just an alternate definition of perfect secrecy
Example or Counterexample: Shift

Does Shift have Perfect Ind? Discuss, Vote. Y, N, Un.
Example or Counterexample: Shift

Does Shift have Perfect Ind? Discuss, Vote. Y, N, Un.NO

Eve’s strategy:

1. Pick the two strings $aa$ and $ab$. $N$ large.
2. When get $c = xy$ if $x = y$ then guess its $aa$, if $x \neq y$ then guess $ab$.

Note: Eve did not need lots of time for this, so will later see that Shift is not comp-ind either.

Note: Affine, Gen Sub, any 1-letter-sub not perfect-ind or comp-ind.
Example or Counterexample: Randomized Shift

Does Rand Shift have Perfect Ind? Discuss, Vote. Y, N, Un.

Eve's strategy:
1. Pick the two strings $a^N$ and $(abcd \cdots z)^N/26$. $N$ large.
2. When get $c^i = ((r_1; \sigma_1), \ldots, (r_N; \sigma_N))$ find the $r_i$ that occurs the most often. If all have same $\sigma_i$ then guess $a^N$ and prob right. If not then guess $(a \cdots z)^N/26$ and definitely right.

Informally: Prob that every time $r_i$ comes up its the same place mod 26 is very small. Hence Prob Eve wins is large – bigger than $1/2$. 
Example or Counterexample: Randomized Shift

Does Rand Shift have Perfect Ind? Discuss, Vote. Y, N, Un.NO

Eve’s strategy:

1. Pick the two strings \( a^N \) and \( (abcd \cdots z)^{N/26} \). \( N \) large.
2. When get \( c = ((r_1; \sigma_1), \ldots, (r_N; \sigma_N)) \) find the \( r_i \) that occurs the most often. If all have same \( \sigma_i \) then guess \( a^N \) and prob right. If not then guess \( (a \cdots z)^{N/26} \) and definitely right.

Informally: Prob that every time \( r_i \) comes up its the same place mod 26 is very small. Hence Prob Eve wins is large – bigger than \( \frac{1}{2} \).
Pick $m$ items out of $n$ types at random until you get $k$ of the same type. Want do succeed with high prob.

We know from earlier work that $m$ is poly on $n$. Let $N$ be that poly in alphabet size.

Let $m$ be such that if pick $m$ elements out of $\{0, \ldots, 25\}$ the prob that they are all $\equiv \pmod{26}$ is small. Can take $m = 26$ (actually much less).
Computational secrecy?

- Idea: Relax perfect indistinguishability
  1. Allow Eve to win with probability slightly more than $\frac{1}{2}$.
  2. Bound how powerful Eve is.

- Two approaches
  - Concrete security (we omit)
  - Asymptotic security
Asymptotic security

- Introduce security parameter $n$
  - For now, can view as the key length
  - Fixed by honest parties at initialization
    - Allows users to tailor the security level
  - Known by Eve

- Measure running times of all parties, and the success probability of Eve, as functions of $n$
Comp. Indistinguishability (asymptotic)

- Computational indistinguishability
  - Security may fail with probability negligible in n
  - Eve’s algorithm is allowed to be randomized but must stop in poly time. PPT=Prob Poly Time.

Note: A Randomized Algorithm is allowed to flip coins but has a small prob of being wrong. Considered a good definition of efficiency.
Definitions

- A function $f : \mathbb{Z}^+ \to \mathbb{Z}^+$ is (at most) **polynomial** if there exists $c$ such that $f(n) < n^c$ for large enough $n$

- A function $f : \mathbb{Z}^+ \to [0, 1]$ is **negligible** if for every polynomial $p$ it holds that $f(n) < \frac{1}{p(n)}$ for large enough $n$

  - Typical example: $f(n) = poly(n) \cdot 2^{-cn}$

**Notation:** Denote polynomial by $poly$. Denote negligible by $neg$. 
Why these choices?

- Taking Efficient to mean PPT is standard.

- poly and neg both have convenient closure properties
  - poly * poly = poly
    - Poly-many calls to PPT subroutine (with poly-size inputs) is PPT
  - poly * neg = neg
    - Poly-many calls to subroutine that fails with neg probability fails with neg probability overall
Eve is Less Powerful. What About Alice and Bob?

When we first defined an encryption scheme \((GEN, ENC_k, DEC_k)\) we did not mention that in order for Alice and Bob to use the scheme:

1. \(GEN\) had to be fast to compute.
2. \(ENC_k\) had to be fast to compute.
3. \(DEC_k\) had to be fast to compute.

This was informally true of Shift, . . . , RSA. But could not formalize since we did not have a security parameter \(n\).

We will now redefine Encryption Scheme with a security parameter \(n\) and formalize that \(GEN, ENC_k, DEC_k\) must be fast to compute.
A private-key encryption scheme is defined by three PPT algorithms (GEN, ENC, DEC):

- **GEN**: takes as input $1^n$; outputs $k$. (assumed $|k| \geq n$).
- **ENC**: takes as input a key $k$ and a message $m \in \{0, 1\}^x$; outputs ciphertext $c$
  $$c \leftarrow ENC_k(m)$$
- **DEC**: takes key $k$ and ciphertext $c$ as input; outputs a message $m$ or “error”
Comp. Indistinguishability

\( \Pi = (\text{GEN}, \text{ENC}, \text{DEC}) \) an enc sch. Message space \( \mathcal{M} \).

**Game:** Alice and Eve are the players. Alice has full access to \( \Pi \).

1. Eve chooses \( m_0, m_1 \in \mathcal{M}, |m_0| = |m_1| \)
2. Alice computes \( k \leftarrow \text{GEN}(1^n) \)
3. Alice picks \( m \in \{m_0, m_1\} \)
4. Alice encodes \( m \): \( c \leftarrow \text{ENC}_k(m) \)
5. Alice sends \( c \) to Eve.
6. Eve outputs \( m_0 \) or \( m_1 \), hoping that her output is \( \text{DEC}_k(c) \).
7. Eve **wins** if she is right.

Can Eve win this game with probability over \( \frac{1}{2} \).
Π is **computationally indistinguishable** (aka **EAV-secure**) if for all **PPT** Eves, there is a **neg function** $\epsilon(n)$ such that

$$\Pr[\text{Eve Wins}] \leq \frac{1}{2} + \epsilon(n)$$

**EAV** stands for **Encryption Against eAVesdropper**
**Example of A Strategy for Eve**

Eve has $O(t(n))$ time.

**Eve’s Algorithm**

1. Input $m_0, m_1, c$. (Eve wants $b$ such that $DEC_k(c) = m_b$.)
2. Eve randomly selects $t(n)$ $k$’s, computes $ENC_k(m_0)$ and $ENC_k(m_1)$. If ever get $c$ then know the answer!
3. If didn’t find answer then guess.

$\Pr$ Eve wins is

\[
\Pr(\text{one of } t(n) \text{ correct}) + \Pr(\text{none of } t(n) \text{ correct but final flip correct})
\]

\[
= \frac{t(n)}{2^n} + \left(1 - \frac{t(n)}{2^n}\right)\frac{1}{2} = \frac{1}{2} + \frac{t(n)}{2^{n-1}}
\]

If this is the only possible algorithm and $t(n)$ is poly then Scheme is EAV-secure.
Computational secrecy

From now on, we will assume the computational, asymptotic, setting by default.
Intuition Behind Example We Will Do

Recall: The only Encryption scheme with perfect security was 1-time pad. But that is hard to do (randomness expensive).


Pseudorandomness: Rather than demand perfect randomness for 1-time pad we settle for pseudorandomness.

1. Pseudorandom: Intuitively means that to a poly-bounded Eve the sequence looks random.

2. Using pseudorandom instead of random:
   2.1 PRO: Practical! Being Used!
   2.2 CON: Won’t get perfect secrecy.
Pseudorandomness
What does “uniform” mean?

Which of the following is a uniform string?

- 0101010101010101
- 0010111011100110
- 0000000000000000
- 00000000000000000000
What does “uniform” mean?

Which of the following is a uniform string?

- 0101010101010101
- 0010111011100110
- 0000000000000000

Trick Question! There is no such think as a uniform string. There is the uniform Distribution.

Def: The uniform dist on \( \{0, 1\}^n \) picks each string with prob \( \frac{1}{2^n} \).
Pseudorandom generators (PRGs)

- A PRG is an efficient, deterministic algorithm that expands a short, uniform seed into a longer, pseudorandom output
  - Useful whenever you have a “small” number of true random bits, and want lots of “random-looking” bits
Definition of PRGs

Let $G : \{0, 1\}^n \rightarrow \{0, 1\}^{p(n)}$ be an efficient, deterministic algorithm.

Game: Alice and Eve are the players. Both have access to $G$.

1. Alice picks $x \in \{0, 1\}^n$ unif, computes $y = G(x) \in \{0, 1\}^{p(n)}$.
2. Alice picks $z \in \{0, 1\}^{p(n)}$ unif.
3. Alice gives $y, z$ to Eve
4. Eve tries to determine which one is $G(x)$ and which one is not.
5. Eve says which is which. If she is right she wins!

Can Eve win this game with probability over $\frac{1}{2}$?
Definition of PRGs

$G$ is a PRG if for all PPT Eves, there is a neg function $\epsilon(n)$ such that

$$\Pr[\text{Eve Wins}] \leq \frac{1}{2} + \epsilon(n)$$
Candidate for a PRG

$D_n$ will be the strings in $\{0, 1\}^{n^2}$ that come out of the following process.

1. Pick safe prime $p$, length $n$ ($\{0, \ldots, p - 1\}$ has $\sim 2^n$ elts).
2. Find a generator $g$ for $p$ of length $n$.
3. Compute $(g^1, g^2, \ldots, g^{n^2})$ all mod $p$.
4. View $(g^1, g^2, \ldots, g^{n^2})$ as $n$-bit strings.
5. Let $b_i$ be the $\left\lceil \frac{n}{2} \right\rceil$th bit of $g^i$.
6. Output $b_1 b_2 \cdots b_{n^2}$

Not known if this is really PRG.
Assuming Discrete Log is hard.
Candidate for a PRG

$D_n$ will be the strings in $\{0, 1\}^{n^2}$ that come out of the following process.

1. Pick safe prime $p$, length $n$ ($\{0, \ldots, p - 1\}$ has $\sim 2^n$ elts).
2. Find a generator $g$ for $p$ of length $n$.
3. Compute $(g^1, g^2, \ldots, g^{n^2})$ all mod $p$.
4. View $(g^1, g^2, \ldots, g^{n^2})$ as $n$-bit strings.
5. Let $b_i$ be the $\left\lceil \frac{n}{2} \right\rceil$th bit of $g^i$.
6. Output $b_1 b_2 \cdots b_{n^2}$

Not known if this is really PRG. Assuming Discrete Log is hard still not known.
Candidate for a PRG

$D_n$ will be the strings in $\{0, 1\}^{n^2}$ that come out of the following process.

1. Pick safe prime $p$, length $n$ ($\{0, \ldots, p - 1\}$ has $\sim 2^n$ elts).
2. Find a generator $g$ for $p$ of length $n$.
3. Compute $(g^1, g^2, \ldots, g^{n^2})$ all mod $p$.
4. View $(g^1, g^2, \ldots, g^{n^2})$ as $n$-bit strings.
5. Let $b_i$ be the $\lceil n^2/2 \rceil$th bit of $g^i$.
6. Output $b_1 b_2 \cdots b_{n^2}$

Not known if this is really PRG.
Assuming Discrete Log is hard still not known.
But thought to be PRG.
Candidate for a PRG

$D_n$ will be the strings in $\{0, 1\}^{n^2}$ that come out of the following process.

1. Pick safe prime $p$, length $n$ ($\{0, \ldots, p - 1\}$ has $\sim 2^n$ elts).
2. Find a generator $g$ for $p$ of length $n$.
3. Compute $(g^1, g^2, \ldots, g^{n^2})$ all mod $p$.
4. View $(g^1, g^2, \ldots, g^{n^2})$ as $n$-bit strings.
5. Let $b_i$ be the $\lceil \frac{n^2}{2} \rceil$th bit of $g^i$.
6. Output $b_1 b_2 \cdots b_{n^2}$

Not known if this is really PRG. Assuming Discrete Log is hard still not known. But thought to be PRG. At least by me.
Do PRGs exist?

1. We don’t know
Do PRGs exist?

1. We don’t know ... Would imply $P \neq NP$
2. We will *assume* certain algorithms are PRGs
3. Can *construct* PRGs from weaker assumptions (Chap 7)
Using Pseudo one-time pad

- Let $G$ be a deterministic algorithm, with $|G(k)| = p(|k|)$
- $Gen(1^n)$: output uniform $n$-bit key $k$
  - Security parameter $n \Rightarrow$ message space $\{0,1\}^{p(n)}$
- $Enc_k(m)$: output $G(k) \oplus m$
- $Dec_k(n)$: output $G(k) \oplus c$
- correctness is obvious
Theorem: Pseudo-OTP is comp secure.
Proof Sketch: Can show that if not comp secure then $G$ is not PRG. We omit details.
Stepping back

- **Proof** that the pseudo OTP is secure . . .
- . . . with some caveats
  - Assuming G is a pseudorandom generator
  - Relative to our definition
- The *Only* way the scheme can be broken is:
  - If a weakness is found in G
  - If the definition isn’t sufficiently strong . . .
Have we gained anything?

- YES: the pseudo-OTP has a key shorter than the message
  - $n$ bits vs. $p(n)$ bits

- The fact that the parties \textit{internally} generate a $p(n)$-bit string to encrypt/decrypt is irrelevant
  - The \textit{key} is what the parties share \textit{in advance}
  - In real-world implementation, could avoid storing entire $p(n)$-bit temporary value
Recall . . .

- Perfect secrecy has two limitations/drawbacks
  - Key as long as the message
  - Key can only be used once

- We have seen how to circumvent the first

- the pseudo OTP still has the second limitation (for the same reason as the OTP)

- How can we circumvent the second? Yes, but we omit.