Introduction to Modern Cryptography
Lecture 2

October 22, 2013

Instructor: Benny Chor

Teaching Assistant: Nir Bitansky

School of Computer Science
Tel-Aviv University

Fall Semester, 2013–14, 15:00–18:00
Dan David 201

Course site: http://tau-crypto-f13.wikidot.com/

Lecture 2: Plan

- Cyclic groups and multiplicative generators.
- The mod operation over integers.
- The ring $(Z_N, +_{\bmod N}, \cdot_{\bmod N})$.

- Perefect Ciphers, revisited.
- Complexity Theoretic Assumptions.
- Indistinguishability of Distributions.
- Symmetric Encryption.
- Pseudo random generators imply stream ciphers.
- Stream ciphers (synchronous mode).

Perfect Cipher: Equivalent Definition

- Plaintext (message) space – $\{0,1\}^n$
- For the sake of simplicity, assume the ciphertext space is also $\{0,1\}^n$.
- Let $M_1, M_2 \in \{0,1\}^n$ be any two plaintexts and $C \in \{0,1\}^n$ be any ciphertext.
- The probabilities that $E_k(M_1) = C$ and $E_k(M_2) = C$ are exactly the same.
- Probability over what?

Perfect Cipher: Equivalent Definition, cont.

- Probability over what?
- Over the key space $\{k\}$ (alone!).
- In a probabilistic language, for every M_1, M_2, C:

$$Pr_k[E_k(M_1) = C] = Pr_k[E_k(M_2) = C] \ .$$

- In daily language: Knowing the ciphertext gives absolutely no information towards knowing the plaintext.
- Important: Requirement does not depend on any distribution of plaintexts.

Example – One Time Pad

- Plaintext space – $\{0,1\}^n$
- Key space – $\{0,1\}^n$. The key k is chosen at random and indep. of P.
- The scheme is symmetric, $\oplus$ stands for bit-wise XOR:
  $E_k(P) = C = P \oplus k$
  $D_k(C) = C \oplus k = P$

Example – One Time Pad, cont.

- The ciphertext space is $\{0,1\}^n$ as well.
- Key space – $\{0,1\}^n$. The key k is chosen at uniformly at random over $\{0,1\}^n$.
- $E_k(P) = C = P \oplus k$ and thus each bit of the ciphertext equals 1 with probability $1/2$, and 0 with probability $1/2$.
- Different ciphertext bits are mutually independent.
- So, for each plaintext, the ciphertext is uniformly distributed in $\{0,1\}^n$.

- Thus one time pad is a perfect cipher.
- Unfortunately, keys must be as long as plaintext to achieve such perfect security (by Shannon's theorem).

Example – One Time Pad, cont. cont.

- Thus one time pad is a perfect cipher.
- By Shannon's theorem, keys must be as long as plaintext to achieve such perfect security.

- We will explore systems employing shorter keys.
- The price to pay is that security will no longer be perfect.
- Instead, security will depend on complexity assumptions, and will hold only wrt computationally bounded adversaries.

Reusing a One Time Pad

- Given two plaintexts $P_1, P_2 \in \{0,1\}^n$.
- We chose a key uniformly at random over $k \in \{0,1\}^n$.
- And happily compute $E_k(P_1) = C_1 = P_1 \oplus k$, $E_k(P_2) = C_2 = P_2 \oplus k$ (happily because we just cut the random bits generation by one half, and deserve a bonus!)
- And send C_1, C_2 to our agent at the other side of the globe.

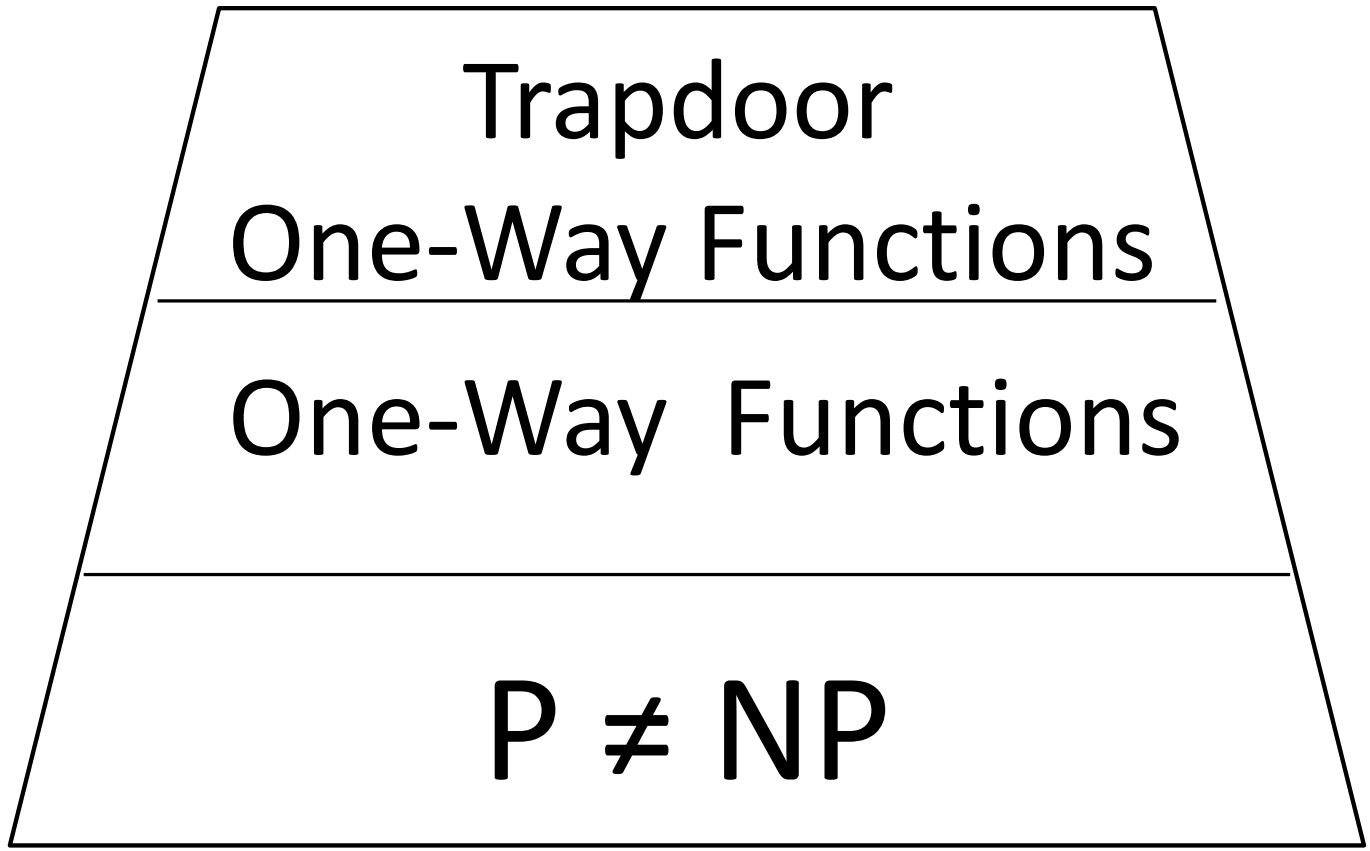
Is this really a good idea?
What can Eve infer about P_1, P_2?

Cryptography and Computation Complexity

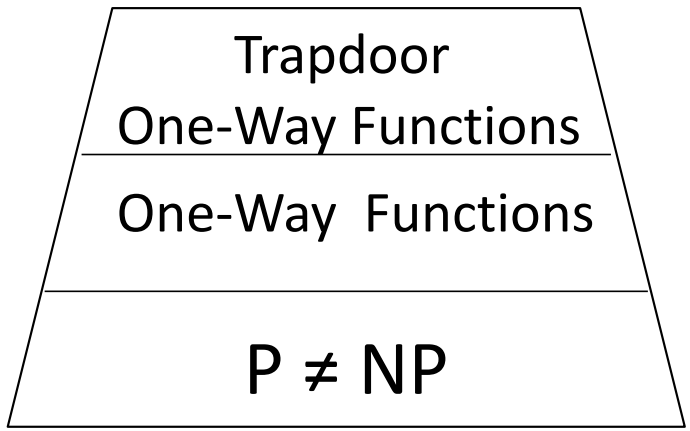
Modern cryptographic research and modern complexity theory have advanced "hand in hand", often fertilizing the other domain considerably.

We will explore "what type of crypto" is doable under various assumption.

The Crypto Assumptions "Pyramid"



The Crypto Assumptions "Pyramid"



We will see that OWF (one way functions) are equivalent to PRG (pseudo number generators).
And PRGs enable private key encryptions.

Perfect Indistinguishability

Let A_n and B_n be two probability distributions on $\{0,1\}^n$.
A distinguisher is a machine, K, that on input string x outputs either a 0 or a 1.
Notations: $x \stackrel{R}{\leftarrow} C$ means "x is chosen according to the distribution C".

$$p_{K,A}(n) = \text{Prob}(K(x) \text{ outputs } 1 : x \stackrel{R}{\leftarrow} A_n)$$
$$p_{K,B}(n) = \text{Prob}(K(x) \text{ outputs } 1 : x \stackrel{R}{\leftarrow} B_n)$$

We say that the distributions A_n and B_n are indistinguishable if for every distinguisher and for all $n \geq n_0$, $p_{K,A}(n) = p_{K,B}(n)$.

Perfect Indistinguishability: A Simple Example

Let A_n be the uniform probability over strings in $\{0,1\}^n$.
Let B_n be the uniform probability over strings in $0\{0,1\}^{n-1}$.

These two probability distributions are over strings in $\{0,1\}^n$.

It is easy to design (do it!) a machine, K, which tells A_n and B_n apart. Namely $p_{K,A}(n) \neq p_{K,B}(n)$.

Perfect vs. Computational Indistinguishability

Let A_n and B_n be two probability distributions on strings in $\{0,1\}^n$.
In the next definition, we will relax the perfect indistinguishability requirement in two aspects:

- The machine, K, is required to run in polynomial time.
- We relax $p_{K,A}(n) = p_{K,B}(n)$ to $\sim$, *i.e.* not too far apart.

Computational Indistinguishability (Goldwasser Micali '82)

Let A_n and B_n be two probability distributions on strings in $\{0,1\}^n$.

A polynomial distinguisher is a polynomial time machine, D, that on input a string x outputs either a 0 or a 1.

$x \stackrel{R}{\leftarrow} C$ means "x is chosen according to the distribution C".

Denote

$$p_{D,A}(n) = \text{Prob}(D(x) \text{ outputs } 1 : x \stackrel{R}{\leftarrow} A_n)$$
$$p_{D,B}(n) = \text{Prob}(D(x) \text{ outputs } 1 : x \stackrel{R}{\leftarrow} B_n)$$

We say that the distributions A_n and B_n are polynomially indistinguishable if for every $\varepsilon > 0$ and every polynomial distinguisher D there is an n_0 such that for all $n \geq n_0$,

$$\mid p_{D,A}(n) - p_{D,B}(n) \mid < \varepsilon \ .$$

Pseudo Random Distributions

Let A_n be a probability distributions on strings in $\{0,1\}^n$.
We say that the distribution A_n is pseudo random if it is polynomially indistinguishable from the uniform distribution on $\{0,1\}^n$.

Computational vs. Perfect Indistinguishability: Example

Let q be a large prime and G an easy to describe group with q elements. Each non unit element $g \in G, g \neq e$ is a multiplicative generator of G (follows from Lagrange theorem and also from the elective assignment). Let $1 \leq a, b, c \leq q-1$ be random and independent.

- Let $\mathcal{A}$ be the uniform distribution on G[1], g, g^a, g^b, g^{ab} (exponentiations are iterated multiplications in G).
- Let $\mathcal{B}$ be the uniform distribution on G, g, g^a, g^b, g^c.

$\mathcal{A}$ and $\mathcal{B}$ are distinguishable (computing discrete logarithms in G tells them apart).

[1] namely a succinct description of G

Computational Indistinguishability: Example

Let q be a large prime and G an easy to describe group with q elements. Each non unit element $g \in G, g \neq e$ is a multiplicative generator of G (follows from Lagrange theorem and also from the elective assignment). Let $1 \leq a, b, c \leq q-1$ be random and independent.

- Let $\mathcal{A}$ be the uniform distribution on p, g, g^a, g^b, g^{ab} (exponentiations are iterated multiplications in G).
- Let $\mathcal{B}$ be the uniform distribution on p, g, g^a, g^b, g^c.

$\mathcal{A}$ and $\mathcal{B}$ are believed to be computationally indistinguishable This is called the decisional Diffie–Hellman assumption (DDH).

Notice that if discrete logarithm in G is easy, DDH is easy.

Discussion on board (and later in the recitation).

Pseudo Random Generators

A pseudo random generator is a polynomial time computable function $G : \{0,1\}^n \mapsto \{0,1\}^{p(n)}$ (on input of length n it produces an output of length $p(n)$), where $p(n) > n$ is a polynomial in n, which satisfies:

The output of G is polynomial time indistinguishable from truly random strings of length $p(n)$.

Notice that the output of such G cannot be truly random!

Further explanation on the board.

One Way Functions

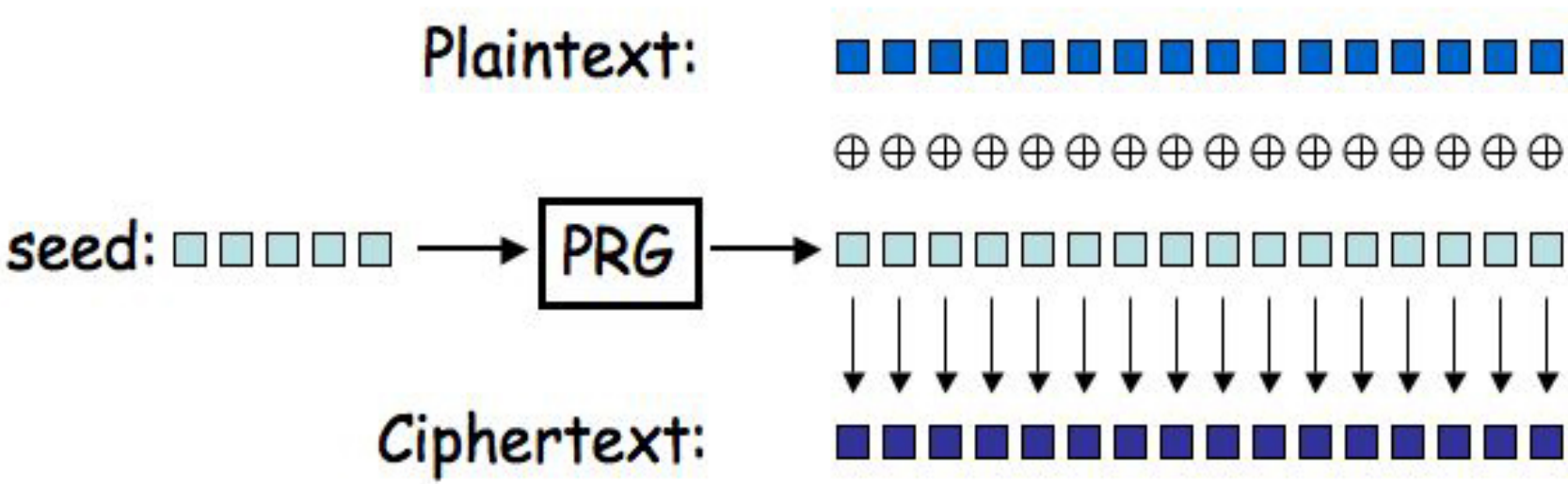
A one way function is a polynomial time computable function $f : \{0,1\}^n \mapsto \{0,1\}^n$ (on input of length n it produces an output of length n), which satisfies: The output of f cannot be inverted in polynomial time. Every (probabilistic) poly time machine fails to invert with probability $\geq 1 - \varepsilon$.

Further explanation on the board.

Remarks: For crypto application we sometime require that, in addition, f is a permutation of $\{0,1\}^n$.

Synchronous Stream Ciphers ("imitating" one-time pad)

- Start with a secret, random key ("seed"). Generate (online) a keying stream by applying the PRG, G, to the seed. The i-th bit of the keying stream is the i-th bit of G's output.
- Combine the keying stream by bitwise XORing with the plaintext, to produce the ciphertext.
- This type of stream cipher is called synchronous (why?).
- Decryption is done in the same manner (XORing ciphertext with keying stream).



PRGs in Practice[2]

- Typical parameters: seed length $n = 128$ or 256 bits; output length: $\ell = 2^{20}$.
- security: time complexity of adversary up to $T(n) \approx 2^{n/2}$ steps, and distinguishing probability $\varepsilon \approx 2^{-n/3}$.
- Concrete implementations are fast but lack theoretical basis.
- Passing public scrutiny is a good measure, but not a sufficient one.
- We'll see an example in a few slides.

[2] slide curtesy of Benny Appelbaum

Real Synchronous Stream Ciphers

- Provide concrete implementations, each with fixed length key (seed) and fixed (maximum) output length.
- Formally there is nothing asymptotic, hence cannot be PRGs.
- Still, with a large key length n one hopes that the best way to break the code is by exhaustive search, 2^n, or close to it.
- Concrete implementations usually have no theoretical foundations.

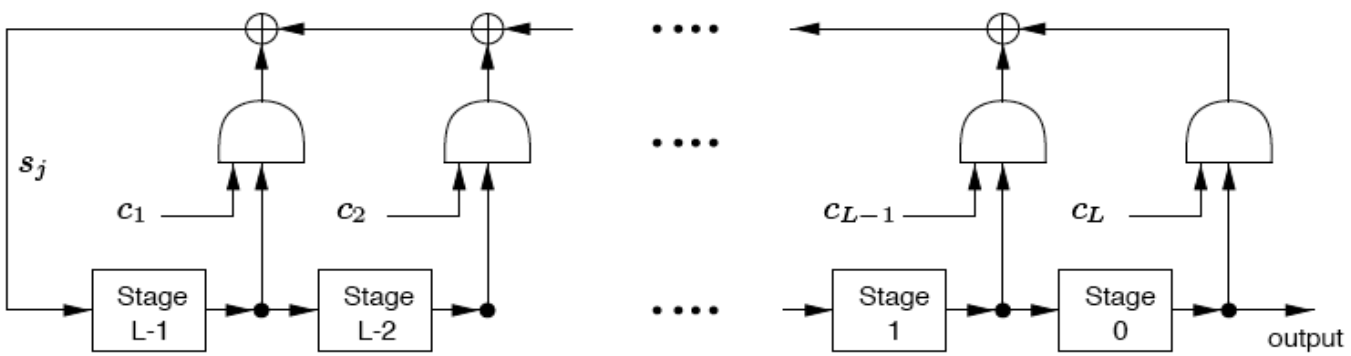
Real Synchronous Stream Ciphers

- Most pre-WWII machines

- German Enigma
- Linear Feedback Shift Register (LFSRs)
- A5 – encrypting GSM handset to base station communication
- RC4 – Ron's (Rivest) Code

Linear Feedback Shift Registers

- An LFSR is a function that produces a binary output stream.
- The device in the picture (from Menezes, Oorschot and Vanstone's book) has L stages (or delay elements).
- The $c_i \in \{0, 1\}$ in the picture are the hardwiring of the device. They are constant, assumed known, and $c_L = 1$.
- The initial state is $[s_{L-1}, \ldots, s_1, s_0]$, which is the secret seed.
- The output sequence (stream) $s_0, s_1, s_2, \ldots$ is determined by the recursion $s_j = \sum_{i=1}^{L} c_i s_{j-i} \mod 2$.



Linear Feedback Shift Registers and Stream Ciphers

- LFSRs have been investigated extensively.
- They have extremely fast implementations in hardware and even in software.
- Closely elated to irreducible polynomials over Z_2.
- With correct choice of wiring and initialization, output stream has a very long period.
- However, they are way too weak for cryptographic use – a relatively short output stretch allows to determine initial seed efficiently.
- Multiplexing or combining several LFSRs, and adding non-linear components, do produce good stream ciphers.