CSC 591 Cryptography 09/05/2018

Lecture COMPUTATIONAL SECURITY

Lecturer: Alessandra Scafuro *Scribe: SURAJ SIDDHARUDH*

Computational Security and PRG

In the last lecture, we discussed about Perfect Secrecy and intuitively defined an encryption scheme is perfectly secure if the ciphertexts generated by the encryption algorithm do not provide any additional information to an adversary with respect to the message that was actually encrypted, even when the adversary has unlimited computational power. Also, we saw the limitations of Perfect Secrecy i.e key space K must be $\geq$ the message space M.

In this lecture, we introduced the concept of Computational Security and Pseudo-Random Generator (PRG).

Definition

The basic idea of Computational Security comes from Kerckhoffs' principle: *A [cipher] must be practically, if not mathematically, indecipherable.*
The computational approach incorporates two relaxations of the notion of perfect security:

1. *Security is only guaranteed against efficient adversaries that run for some feasible amount of time*

2. *Adversaries can potentially succeed (i.e security can potentially fail) with some very negligible probability.*

We define Asymptotic approach to Computational Security in following terms:

1. Running Time + Negligible Probability
   We have an security parameter (denoted by n) that parameterizes both cryptographic schemes as well as all involved parties (viz. honest parties and the adversary). We assume that the security parameter n is known to adversary and we see that the running time of the adversary, as well as its success probability, as functions of n. Then:

   (a) *Adversary is bounded.* This means that there is some polynomial p such that the adversary runs for time at most $p(n)$ when the security parameter is n.

   (b) Adversary can win with "small probability of success" - success probabilities smaller than any inverse polynomial in n. i.e. $\forall c$, Adversary's Running time is n^c, and Adversary's success probability $\approx 2^{-n}$. Such probabilities are called *negligible.*

   Let PPT stand for "probabilistic polynomial-time". A definition of asymptotic security thus takes the following general form: *A scheme is secure if every PPT adversary*

succeeds in breaking the scheme with only negligible probability.
Note: n is the security parameter and can be tuned by the Honest parties based on availability of computational power. As n increases $\Rightarrow$ More time for Adversary to break the scheme $\Rightarrow$ Success Probability is negligible (as its 2^{-n}) $\Rightarrow$ Harder job for Adversary to break.

Definition 1 *Negligible Function*
A function f from the natural numbers to the non-negative numbers is negligible if for every positive polynomial p there is an integer N such that for all integers $n > N$ it holds that $f(n) < \frac{1}{p(n)}$.

Proposition 2 *Let $negl_1$ and $negl_2$ be negligible functions.Then,*

(a) The function $negl_3$ defined by $negl_3(n) = negl_1(n) + negl_2(n)$ is negligible.

(b) For any positive polynomial p, the function $negl_4$ defined by $negl_4(n) = p(n) \cdot negl_1(n)$ is negligible. (Also, $negl_4(n) = negl_1(n) \cdot negl_2(n)$)

Pseudo Randomness Vs Randomness
Historically, a string is considered pseudo-random if it passes Statistical tests. Examples of some of these tests are:

- *The probability of the first bit being 1 should be roughly 1/2*
- *Parity of the bits should be roughly 1/2*
- *There shouldn't be any repeated pattern*

In Modern Cryptography, A string is pseudo-random if it passes ***Any*** Efficient Statistical test.

Assumption

Pseudo-Random Generator
A pseudo-random Generator (PRG) is a deterministic polynomial time algorithm G, for transforming short, uniform string called seed into a longer, "uniform-looking"(or "pseudo random") output string. i.e. $G : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$.
Based on our Intuition, we can define a good PRG as, for any efficient statistical test (or distinguisher) D, the probability that D returns 1 when given the output of PRG should be close to the probability that D returns 1 when given a uniform string of the same length.*Informally*, G is a PRG if no efficient distinguisher can detect whether it is given a string output by G or a string chosen uniformly at random.

Definition 3 *Let l be a polynomial and let G be a deterministic polynomial time algorithm such that for any n and any input $s \in \{0,1\}^n$, the result G(s) is a string of length l(n). We say that G is a PRG if the following conditions hold:*

1. *(Expansion:) $\forall$ n, it holds that $l(n) > n$*

2. *(Pseudorandomness:) $\forall$ PPT distinguisher D, there exists a negligible function $\epsilon(n)$* $|Pr[D_{s\leftarrow\{0,1\}^n,\{y\leftarrow G(s)\}}(y)=1] - Pr[D_{y\leftarrow\{0,1\}^{l(n)}}(y)=1]| \leq \epsilon(n)$

Example 1 *Check whether the function $G(s) \rightarrow s\| \oplus_{i=1}^{n}(s_i)$, where $s = s_1, \cdots, s_n$, is a PRG.*

Proof:

Step 1: Find a Distinguisher D, which takes the input y and computes,

1. *Parse $y = y_1, y_2 ..... y_n, y_{n+1}$*
2. *If $y_{n+1} = \oplus_{i=1}^{n}(y_i)$, then output 1 else output 0*

Step 2: Analysis

1. *Case 1: $y \leftarrow G(s)$, i.e. when y is the output of G(s), $y_{n+1} = \oplus_{i=1}^{n}(y_i)$. Distinguisher D predicts correctly and outputs 1.*
   $Pr[D_{s\leftarrow\{0,1\}^n,\{y\leftarrow G(s)\}}(y)=1] = 1$
2. *Case 2: $y \leftarrow U^{n+1}$ i.e. when y is the output of truly random function. As this is uniform distribution, the probability of $y_{n+1} = \oplus_{i=1}^{n}(y_i)$ is $\approx \frac{1}{2}$*
   $Pr[D_{y\leftarrow\{0,1\}^{n+1}}(y)=1] = \frac{1}{2}$

From our Definition, $|Pr[D_{s\leftarrow\{0,1\}^n,\{y\leftarrow G(s)\}}(y)=1] - Pr[D_{y\leftarrow\{0,1\}^{n+1}}(y)=1]| \leq |1-\frac{1}{2}| = \frac{1}{2}$, which is a non negligible value. An Adversary can distinguish between the output from the PRG and output from the Uniform distribution with a non-negligible value.
So, This is not a PRG. □

Note: Follow these steps if you want to show a function is not a PRG

1. Design an Distinguisher, D.
2. Show the analysis using the distinguisher D i.e D is able to distinguish between the output from PRG and uniform distribution.
3. Work on the Probability Equation and show that the difference in probabilities is Non-Negligible.

Observation 4 *How to Construct PRG*

1. Practical: PRG from Block Cipher (Ex: AES, 3-DES, ChaCha20, Salsa20)
2. Theoretical: PRG from Mathematical Hard Problem (Ex: DDH Assumption)
3. Theoretical: PRG from One way Function.

Scheme

Pseudo One Time Pad
A PRG provides a natural way to construct a secure, fixed length encryption scheme with a key shorter than the message. As for security, the intuition is that a pseudorandom string "looks random" to any PPT Adversary. So, the adversary cannot distinguish between a message encrypted with One Time Pad or a message encrypted using Pseudo-One Time Pad.
Let G : $\{0,1\}^n \rightarrow \{0,1\}^{2n}$ be a PRG. Let $M = \{0,1\}^{2n}$ be the Message space and $K = \{0,1\}^n$ be the Key Space.
We can use PRG as the step to generate the key k, which can be XORed with the message m.We define Pseudorandom Encryption Scheme as follows:

- Key Gen : Choose a Seed s, from the uniform distribution K, $s \leftarrow \{0,1\}^n$
  Use this Seed as the input to the PRG to get Pseudorandom key k, $k = G(s)$
- Encryption: On a input key $k \in \{0,1\}^{2n}$ and a message $m \in \{0,1\}^{2n}$, output the ciphertext c. $Enc(k,m) : c = m \oplus k$
- Decryption: On a input key $k \in \{0,1\}^{2n}$ and a message $c \in \{0,1\}^{2n}$, output the message m $Dec(k,c) : m = c \oplus k$