Alin Tomescu, CSE408
Tuesday, February 8th, Lecture #3

Pseudo Random Number Generators

Expansion functions

As we've discussed before, the one-time-pad when used with a random key that is as large as the plaintext is a secure encryption algorithm.

However, in the real world, keys that are as large as the data to be encrypted are very impractical. Therefore, we would like to be able to expand a smaller key k to a larger "pseudo-random" key. We will now focus on an expansion function G whose purpose is to expand smaller keys into larger keys.

Properties an expansion function G should have:

1. $G(k)$ needs to be pseudo-random (look like a randomly generated string)
2. G needs to be hard or impossible to invert
3. Given n bits of $G(k)$, an attacker should not be able to correctly guess the $(n+1)$th bit
4. G needs to be an injective function: $x_1 \neq x_2 \Rightarrow f(x_1) \neq f(x_2)$
5. G needs to be efficient, and not take too long to execute
6. G needs to be hard to recognize

A good story

Two armies are planning to coordinate an attack. They are either attacking at dawn or dusk. The enemy knows they'll get attacked at either dawn or dusk. The message between the armies has been encrypted with the output of G. The enemy sees $E(m)$. He can take the ciphertext and XOR it with "attack at dawn" and "attack at dusk" getting two different outputs of G. Suppose he can tell G could never generate one of those outputs. That would be bad. He could then tell that the matching plaintext was wrong, and he will know the time of the attack.

The big concept

Consider the (probability) distribution of random numbers of a certain length. Now consider the distribution of $G(k)$ of the same length. Obviously, this one is a smaller distribution than the distribution of random numbers: it will have fewer elements.

The enemy gets to see samples over these distributions. The enemy shouldn't be able to tell whether you're getting samples from distribution A or distribution B.

Simple example

You have two black boxes.

- one box, you hit a button, it flips n coins and gives you a random number on n bits
- one box, you hit a button, it flips 128 coins, gives you a pseudo-random number on n bits

The goal is to never be able to tell (realistically in a couple of years) which box the generated n-bit number came from.

Notion: The security of a system is parameterized by how well an attacker can do in a certain amount of time.

- if you can break my ciphertext in a 1000 years, be my guest
- if you can break my ciphertext in a couple of days, I'd better find a more secure algorithm

Alin Tomescu, CSE408
Tuesday, February 8th, Lecture #3

The concept of advantage

Definition

Let D_0 and D_1 be distributions on some space X. Then D_0 and D_1 are $(\boldsymbol{t}, \boldsymbol{\varepsilon})$**-computationally indistinguishable** if $\forall$ algorithms A (A outputs 0 if input is from D_0 and outputs 1 if input is from D_1) running in time $\leq t$, then:

$$Advantage(A) = Adv\ A = |\Pr[A(x) = 0 | x \leftarrow D_0] - \Pr[A(x) = 0 | x \leftarrow D_1]| \leq \varepsilon$$

Note: ε is related to the time t. If I give you more time, your ability should improve.

Notation: $D_0 \overset{t}{\underset{\varepsilon}{\sim}} D_1$ means D_0 and D_1 are (t, ε)-computationally indistinguishable.

Example

Consider the following distributions.

	Probability it's a 0	Probability it's a 1
D_0	.5	.5
D_1	0	1

Case 1: Suppose we have an algorithm A that can distinguish (heuristically in this example) between D_0 and D_1.

Then it is only normal for $A(0) = 0$, since $\Pr[x \leftarrow D_1 | x = 0] = 0 \Rightarrow \Pr[x \leftarrow D_0 | x = 0] = 1$.

We will set $A(1) = 1$, since it is more likely for a 1 to have been picked from D_1 instead of D_0 because:

$$\Pr[x = 1 | x \leftarrow D_1] = 1 > \Pr[x = 1 | x \leftarrow D_0] = .5$$

Let's calculate the advantage of A:

$$Adv\ A = |\Pr[A(x) = 0 | x \leftarrow D_o] - \Pr[A(x) = 0 | x \leftarrow D_1]| = .5 - 0 = .5$$

Case 2: What happens if we let A always pick distribution D_0?

$$A(0) = 0\ and\ A(1) = 0$$

$$Adv\ A = |1 - 1| = 0$$

Case 3: What happens if we let $A(0) = 0$ and $A(1)$ output a 0 or a 1 with equal probability $\Pr[A(1) = 0] = \Pr[A(1) = 1] = .5$?

$$Adv\ A = |\Pr[A(x) = 0 | x \leftarrow D_o] - \Pr[A(x) = 0 | x \leftarrow D_1]| = ?$$

$$\Pr[A(x) = 0 | x \leftarrow D_o] = \Pr[A(x) = 0 | x = 0] \Pr[x = 0 | x \leftarrow D_0] + \Pr[A(x) = 0 | x = 1] \Pr[x = 1 | x \leftarrow D_0] =$$

$$= 1 \times .5 + .5 \times .5 = 0.75$$

$$\Pr[A(x) = 0 | x \leftarrow D_1] = \Pr[A(x) = 0 | x = 0] \Pr[x = 0 | x \leftarrow D_1] + \Pr[A(x) = 0 | x = 1] \Pr[x = 1 | x \leftarrow D_1] =$$

$$= 1 \times 0 + 1 \times .5 = .5$$

So therefore $Adv\ A = |.75 - .5| = .25$

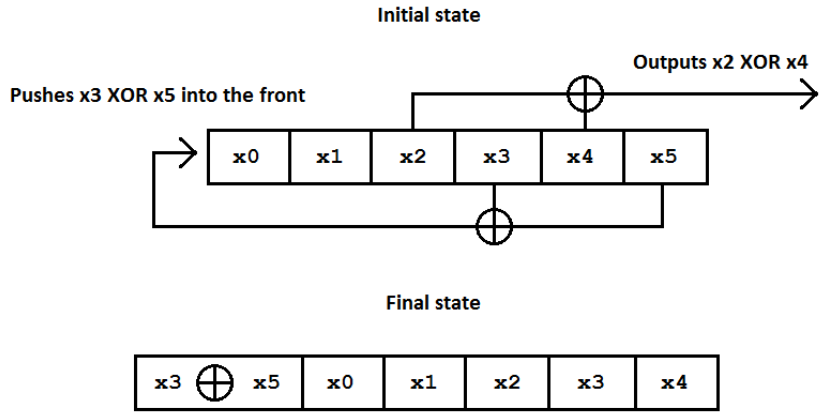
Alin Tomescu, CSE408
Tuesday, February 8th, Lecture #3

Definition

A pseudo-random generator $G: \{0,1\}^l \to \{0,1\}^L$ is $(\boldsymbol{t}, \boldsymbol{\varepsilon})$**-secure** if $G(U_l) \overset{t}{\underset{\varepsilon}{\sim}} U_L$, where $U_k = k$-bit uniform random strings. This definition pretty much states that a pseudo-random generator is (t, ε)-secure if its output is (t, ε)-indistinguishable from the uniform distribution of random strings.

A bad example of a pseudo-random generator

A Linear Feedback Shift Register (LFSR) is a type of PRNG. The following example would be a broken LFSR.



The LFSR is initialized with some bits $x_1, x_2, \ldots x_5$ (the seed value) and then a pseudo-random bit y_0 is generated as follows:

$$y_0 = x_2 \oplus x_4$$

The register will change its data as follows

- First $x_0 = x_3 \oplus x_5$, and then the shifting is done (the old x_0 is used to do the shifting)
- Shifting is done: $\forall k, 0 \leq k \leq 4, x_{k+1} = x_k$
- x_5 gets discarded

The reason this LFSR is insecure is because the output bits would give you a system of linear equations which could be easily solved to find the seed value consisting of the $x_1, x_2, \ldots x_5$ bits.

You would just have to solve

$$y_0 = x_2 \oplus x_4$$

$$y_1 = x_1 \oplus x_3$$

$$y_2 = x_0 \oplus x_2$$

$$y_3 = x_3 \oplus x_5 \oplus x_1$$

So for instance XORing $y_1 \oplus y_3$ would output x_5. With this method, once you have enough output bits you can determine $x_1, x_2, \ldots x_5$.

Alin Tomescu, CSE408
Tuesday, February 8th, Lecture #3

Data processing inequality or DPI

Theorem

If $D_0 \overset{t}{\underset{\varepsilon}{\sim}} D_1$ and f is a function running in time t' then $f(D_0) \overset{t-t'}{\underset{\varepsilon}{\sim}} f(D_1)$.

What this says is that if you have distributions and you can't tell them apart, then a simple operation such as squaring them won't help you to tell them apart either. This is almost like a reduction.