CSE 5852: Lecture 14

Benjamin Fuller

October 17, 2016

1 Last Class

Last class we finished our proof that the following two definitions are equivalent:

Next-bit unpredictability Define the following experiment $\texttt{prg} - \texttt{predict}$, parameterized by n:

1. Select random s of length $n(k)$.
2. Compute $y = G(s)$.
3. Run $\mathcal{A}(1^n)$, giving it bits of y in response to each *next* request.

If $\mathcal{A}$ stops after $i \leq m(n)$ stages and outputs $b = y_i$ we that that $\mathcal{A}$ wins $\texttt{prg} - \texttt{predict}$ and it outputs 1.

An attacker can win the above experiment with probability 1/2 by guessing a random bit and ignoring the bits it is being given. Similarly to security of encryption we have the following definition.

Definition 1. *[BM84]. A function $G_n(s) : \{0,1\}^k \to \{0,1\}^m$ is a pseudorandom generator* satisfying next bit unpredictability *if for all PPT $\mathcal{A}$,*

$$\Pr[\texttt{prg} - \texttt{predict}^{G,\mathcal{A}} = 1] \leq 1/2 + \epsilon(n)$$

where $\epsilon(n)$ is a negligible function of n.

All efficient tests That is consider two experiments: $\texttt{exp} - \texttt{pr}$ and $\texttt{exp} - \texttt{r}$. Let T be some PPT test that outputs either 1 or 0.

Experiment $\texttt{exp} - \texttt{pr}^{G,T}$:	**Experiment** $\texttt{exp} - \texttt{r}^{T}$:
Select random s of length n.	Select random y of length m
Compute $y = G(s)$	Run $T(y)$ and output whatever it does.
Run $T(y)$ and output whatever it does.	

Definition 2. *[Yao82] G passes all statistical tests if for all PPT T, there exists negligible function $\epsilon(n)$ such that for all n,*

$$\left|\Pr[\texttt{exp} - \texttt{pr}^{G,T} = 1] - \Pr[\texttt{exp} - \texttt{r}^{T} = 1]\right| \leq \epsilon(n).$$

We also showed part of the proof that the two definitions are equivalent.

Theorem 1. *An algorithm G passes all statistical tests if and only if it is next bit unpredictable.*

2 Using a pseudorandom generator

In the previous classes we introduced a candidate pseudorandom generator (based on the discrete logarithm problem), showed is secure if the DL problem is hard, and showed the equivalence of two definitions. We also showed that a pseudorandom generator can be used to construct a one-time secure computational encryption scheme. In this class we will start to explore the power of a pseudorandom generator. It can be used to construct a lot of powerful objects. We'll show that a pseudorandom generator can be used to create an encryption scheme that is secure for sending multiple messages (with a single key) and is sufficient to prevent adversary tampering (again with a single key).

We'll start with small jump which is that the length of the output of a pseudorandom generator doesn't matter as long as it expands the output.

Theorem 2. *Let $G : \{0,1\}^n \to \{0,1\}^m$ be a pseudorandom generator (passing all efficient statistical tests) where $m > n$. Let $m' = \mathsf{poly}(n)$ where $m' > m$. Then, there exists $G' : \{0,1\}^n \to \{0,1\}^{m'}$ that is a pseudorandom generator.*

Before presenting how G' works lets talk a little bit about the implication. We called the difference between m and n $(m-n)$ the expansion factor of a PRG. This theorem says that the expansion factor doesn't matter, we can have any expansion factor we want once we have some expansion factor. Furthermore, we can always decrease the expansion factor as well.

Lemma 1. *Let $G : \{0,1\}^n \to \{0,1\}^m$ be a pseudorandom generator (passing all efficient statistical tests) where $m > n$. Let $m' < m$. Then, there exists $G' : \{0,1\}^n \to \{0,1\}^{m'}$ that is a pseudorandom generator.*

Proof. Just take the first m' bits of G and call that G'. $\square$

Together these statements say that the expansion factor of a PRG doesn't matter we can expand or shrink this factor. (At some cost to security and efficiency of the PRG.)

Let's think about how to build a PRG with a better expansion factor than G. In the previous class we said that the output of G was as good as random. The idea was that any algorithm that could tell between the output of G and random would be a distinguisher and could be used to create an efficient statistical test. So as a group lets come up with a mechanism to double the expansion factor of G.

Theorem 3. *Let $G : \{0,1\}^n \to \{0,1\}^m$ be a PRG. Let $y_1 ... y_m$ denote the output of G. Then function $G' = G(y_1 ... y_n) y_{n+1} ... y_m$ is a PRG with expansion $2(m-n)$.*

Proof. Denote by $m' = 2m - n$ which is the output length of G'. We proceed by the contrapositive. Assume that G' is not a pseudorandom generator, we will use this to contradict the fact that G is a PRG. Since G' is not a PRG, there exists some PPT A and some polynomial p such that,

$$\Pr[A(G'(S)) = 1] - \Pr[A(U_{m'}) = 1] > 1/p(n).$$

As in the previous class we will use the *hybrid* argument. We will come up with an intermediate experiment between these two. Define $exp - 0$ as the one that has the output of G' and $exp - 2$ as the output of $U_{m'}$. Then,

$$|\Pr[A(exp-0)=1]-\Pr[A(exp-2)=1]|>1/p(n).$$

We'll define an intermediate experiment $exp-1$ as the following:

- Sample random $u_1 ... u_n$.
- Construct $x_1 ... x_m = G(u_1 ... u_n)$
- Sample random $z_1 ... z_m$.
- Output $x_1 ... x_{m-n} z_1 ... z_m$.

As before we can use the triangle inequality to show that

$$|\Pr[A(exp-0)=1]-\Pr[A(exp-2)=1]| \leq$$
$$|\Pr[A(exp-0)=1]-\Pr[A(exp-1)=1]| + |\Pr[A(exp-1)-\Pr[A(exp-2)=1]|.$$

As before we can argue that one of these terms must be greater than $1/2p(n)$. (What happens if the are both less than $1/2p(n)$?)

We know need to show we can break the security of G in either case. We'll built a new test A' to break G in either situation.

First case $|\Pr[A(exp-0)=1]-\Pr[A(exp-1)=1]|>1/2p(n)$ This is the easier direction of the proof. We need to construct a A' in this case. Recall that we receive m bits as input that are either random or pseudorandom. We do the following:

1. Receive $x_1 ... x_m$.
2. Create $y_1 ... y_m = G(x_1 ... x_n)$.
3. Output $A(y_1 ... y_m || x_{n+1} ... x_m)$.

Note that if the bits x we got as input were pseudorandom then the input we provide to A is exactly as generated in $exp-0$. If x is random then the input we provide to A is exactly as generated in $exp-1$. That is,

$$|\Pr[A'(G(S))=1]-\Pr[A'(U_m)=1]| = |\Pr[A(exp-0)=1]-\Pr[A(exp-1)=1]| > 1/2p(n).$$

as required. This is a contradiction.

Second case $|\Pr[A(exp-1)=1]-\Pr[A(exp-2)=1]|>1/2p(n)$ This is the easy case. Note that the last $m-n$ bits are random in both experiments. So our distinguisher, A' is very simple:

1. Receive $y_1 ... y_m$.
2. Flip random bits $x_{m-n+1} .. x_m$.
3. Output $A(y_1 ... y_m || x_{m-n+1} ... x_m)$.

Note that if the bits y we got as input were pseudorandom then the input we provide to A is exactly as generated in $exp-1$. If y is random then the input we provide to A is exactly as generated in $exp-2$. That is,

$$|\Pr[A'(G(S))=1]-\Pr[A'(U_m)=1]| = |\Pr[A(exp-1)=1]-\Pr[A(exp-2)=1]| > 1/2p(n).$$

as required. This is a contradiction.

Since we have a contradiction in both cases we have a contradiction overall. Thus, G' is secure if G is secure. $\square$

3 Encryption secure with multiple messages

What could we do with a truly random function (who's description is unknown to the adversary)?

- First let's and build an encryption scheme. An attacker even with knowing the input to the function has no idea what the output will be. So the output of the function is actually random. We'll assume a function $f : \{0,1\}^n \to \{0,1\}^\ell$. Consider the following encryption scheme:

  1. Input function f and message m.
  2. Select a random $r \leftarrow \{0,1\}^n$.
  3. Output $c = f(r) \oplus m, r$.

  And the corresponding decryption:

  1. Input function f and ciphertext $c = c', r$.
  2. Compute $m = f(r) \oplus c'$.

- Second, let's try and build a MAC scheme. Lets recall what a strongly universal function provided. It said that on any two points the output of the function was truly random. What if we had a truly random function instead? Then the output would be random on every point. This suggests the following MAC scheme.

  1. Input function f and message/ciphertext m.
  2. Output $t = f(m)$.

Let's recall our definition of multi-message secure MAC scheme: Our definition models the situation where messages are sent in sequence. The adversary should only have to modify a single message to win. This leads us to the following experiment.

$\texttt{Mac} - \texttt{forge}^{\mathcal{A},\mathsf{Mac},k}$:

1. Key $k \leftarrow \mathsf{Gen}()$.
2. The adversary, $\mathcal{A}$ outputs m_1 and is given $t_1 \leftarrow \mathsf{Mac}(k, m)$.
3. This process repeats with the adversary outputting m_i and being given t_i. Note that m_i can depend on $t_1, ..., t_i$.
4. After k iterations the adversary outputs (m', t').
5. The output of the experiment is 1 if $m' \neq m_i$ for any i and $\mathsf{Vfy}(k, m', t') = 1$. Otherwise the output is 0.

Definition 3. *A MAC scheme is k-chosen message unforgeable if* $\Pr[\texttt{Mac} - \texttt{forge}^{\mathcal{A},\mathsf{Mac},k}] < \epsilon$.

Theorem 4. *The MAC scheme presented above is k-chosen message unforgeable for $\epsilon = 1/2^\ell$.*

The encryption scheme is slightly more complicated. The encryption scheme works well as long as the value r is not repeated between messages. Thus, we'll present security in the computational world (even though the scheme is information-theoretically secure).

We first present a definition for indistinguishable encryptions:

Definition 4. *An encryption scheme* $(K, \mathsf{Enc}, \mathsf{Dec})$ *has k-indistinguishable encryptions if for all PPT A for all vectors $m_1 = m_{1,1}, ..., m_{1,k}$ and $m_2 = m_{2,1}, ..., m_{2,k}$ there exists some negligible ϵ such that the following is true,*

$$\begin{aligned} &|\Pr[A(\mathsf{Enc}(key, m_{1,1}), \mathsf{Enc}(key, m_{1,2}), ..., \mathsf{Enc}(key, m_{1,k})) = 1] \\ &- \Pr[A(\mathsf{Enc}(key, m_{2,1}), \mathsf{Enc}(key, m_{2,2}), ..., \mathsf{Enc}(key, m_{2,k})) = 1] < \epsilon. \end{aligned}$$

Theorem 5. *The above encryption scheme is k-message unforgeable for $\epsilon \approx \frac{k}{2^{\sqrt{n}}}$.*

However, neither of these schemes can be efficiently instantiated because storing and computing a random function takes an exponential amount of space/time.

Definition 5 (Informal Definition)**.** *A* pseudorandom *function is a function $f : \{0,1\}^s \times \{0,1\}^n \to\to \ell$ such that for a fixed (but unknown) key key no PPT A can tell the difference between interacting with f_s and a truly random function.*

Theorem 6. *If PRGs exist then so do pseudorandom functions (PRF)s.*

References

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM journal on Computing*, 13(4):850–864, 1984.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *FOCS*, pages 80–91, 1982.