CSE 5852: Lecture 13

Benjamin Fuller

October 12, 2016

1 Last Class

Last class we finish our proof that the Blum-Micali PRG satisfies next-bit unpredictability assuming that the discrete logarithm problem is hard in $\mathbb{Z}_p^*$. We started to look at two definitions:

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.

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

2 Computational Security of Encryption

Before finishing our proof of computational security I want to make formal our claim that if we have a PRG that passes all statistical tests then it gives a secure encryption scheme.

Theorem 1. *Let $G : \{0,1\}^n \to \{0,1\}^m$ be a function that passes all efficient statistical tests. Then $\mathsf{Enc}(s,m) = G(s) \oplus m$ is an encryption scheme with indistinguishable encryptions.*

Proof. Last class I said this proof was pretty simple but lets be formal about it. We had the notion that if something passes all statistical tests then we can substitute the output of the function for random. How are we going to do this proof? For once the answer is not by contradiction or contrapositive.

Lemma 2. *If G passes all statistical tests, then for all constant c, $G(s) \oplus c$ passes all statistical tests.*

This lemma is on Problem Set 4. Let A be some indistinguishable encryption distinguisher for messages m_1, m_2. Let U_m be the uniform random variable of length m.

By the above definition we know that for any there exists some negligible $\epsilon_1(n)$ such that

$$|\Pr[A(G(S) \oplus m_1) = 1] - \Pr[A(U_m \oplus m_1) = 1]| < \epsilon_1(n).$$

Similarly,

$$|\Pr[A(G(S) \oplus m_2) = 1] - \Pr[A(U_m \oplus m_2) = 1]| < \epsilon_2(n).$$

We also know by Shannon secrecy of the one-time pad that

$$\Pr[(U_m \oplus m_1) = c] = \Pr[(U_m \oplus m_1) = c]$$

And thus,

$$|\Pr[A(U_m \oplus m_1) = 1] - \Pr[A(U_m \oplus m_2) = 1]| = 0.$$

The triangle inequality says that $d(A,B) \leq d(A,C) + d(C,B)$. We can apply that to the difference between two probabilities. We can add an intermediate point, that is,

$$\begin{aligned}
&|\Pr[A(G(S) \oplus m_2) = 1] - \Pr[A(G(S) \oplus m_2) = 1]| \\
&\qquad \leq |\Pr[A(G(S) \oplus m_1) = 1] - \Pr[A(U_m \oplus m_1) = 1]| \\
&\qquad + |\Pr[A(U_m \oplus m_1) = 1] - \Pr[A(U_m \oplus m_2) = 1]| \\
&\qquad + |\Pr[A(G(S) \oplus m_2) = 1] - \Pr[A(U_m \oplus m_2) = 1]| \\
&\qquad \leq \epsilon_1(n) + 0 + \epsilon_2(n)
\end{aligned}$$

By Problem Set 3 the sum of two negligible functions is negligible, thus $\epsilon_1 + \epsilon_2$ is negligible and the scheme satisfies indistinguishable encryptions. □

3 Definitional Equivalence

Lemma 3. *An algorithm G passes all statistical tests if it is next bit unpredictable.*

Proof. As is standard we are going to proceed by contradiction. We're going to assume that there exists a polynomial time computable statistical test that G does not pass. We need to build a next bit predictor.

How can we hope to construct such a thing? We can run our statistical test until we have all of the bits. However at this point we've already failed the next bit predictor. There is nothing left to predict. How, can we hope to use this statistical test?

The key to this proof is something called the hybrid argument. Consider some statistical test T that distinguishes the two settings. Let the random variable Y represent the output of the pseudorandom generator and U represent a uniform random string. We know there is some statistical test such that

$$|\Pr[T(Y) = 1] - \Pr[T(U) = 1]| > 1/p(n).$$

For some polynomial function $p(n)$. Note that T is just getting some sequence of bit either drawn from Y or U. Let's use the same triangle inequality we used in the previous proof.

$$\begin{aligned}
&|\Pr[T(Y_1....Y_m) = 1] - \Pr[T(U_1...U_m) = 1]| \\
&\quad \leq |\Pr[T(Y_1....Y_{m-1}Y_m) = 1] - \Pr[T(Y_1...Y_{m-1}U_m) = 1]| \\
&\quad + |\Pr[T(Y_1....Y_{m-1}U_m) = 1] - \Pr[T(U_1...U_m) = 1]|.
\end{aligned}$$

We can continue to expand this using the triangle inequality:

$$\begin{aligned}
&|\Pr[T(Y_1....Y_m) = 1] - \Pr[T(U_1...U_m) = 1]| \\
&\quad \leq |\Pr[T(Y_1....Y_{m-1}Y_m) = 1] - \Pr[T(Y_1...Y_{m-1}U_m) = 1]| \\
&\quad + |\Pr[T(Y_1....Y_{m-1}U_m) = 1] - \Pr[T(Y_1....Y_{m-2}U_{m-1}U_m) = 1]| \\
&\quad + |\Pr[T(Y_1....Y_{m-2}U_{m-1}U_m) = 1] - \Pr[T(Y_1....Y_{m-3}U_{m-2}U_{m-1}U_m) = 1]| \\
&\quad + ... \\
&\quad + |\Pr[T(Y_1U_2....U_m) = 1] - \Pr[T(U_1...U_m) = 1]|.
\end{aligned}$$

Each line here implicitly defines an intermediate experiment where we provide some pseudorandom bits and some random bits. These are called *hybrid* experiments. They represent smaller steps between the two experiments we are about and have components from both.

Furthermore there exists some i such that

$$|\Pr[T(Y_1...Y_{i-1}Y_iU_{i+1}...U_m) = 1] - \Pr[Y_1....Y_{i-1}U_iU_{i+1}...U_m = 1]| \geq \frac{1}{mp(n)}.$$

And note that $mp(n)$ is a polynomial. Lets assume for a moment that we know i and that

$$\Pr[T(Y_1...Y_{i-1}Y_iU_{i+1}...U_m) = 1] - \Pr[Y_1....Y_{i-1}U_iU_{i+1}...U_m = 1] \geq \frac{1}{mp(n)}.$$

Where we have removed the absolute values (the proof goes through if the opposite is true, we'll see how).

We now describe our bit predictor $\mathcal{A}$:

1. Say next until you have received $i-1$ bits $y_1....y_{i-1}$.
2. Generate $m(n)-(i-1)$ random bits $r_i....r_m$.
3. Run $T(y_1...y_{i-1}r_i...r_m)$.
4. If T returns 1 output $b = r_i$ otherwise output $b = 1 - r_i$.

The idea of why $\mathcal{A}$ works is that if r_i happens to be the next bit of y we will be on the left side of our inequality and T is more likely to output 1, if r_i happens to be the wrong bit we are more likely to be on the right side of our inequality and T is more likely to output 0. Thus, when T outputs 0 we assume we got the guess wrong. Lets use g to denote the bit we guessed and let z be the input to T.

We have

$$\Pr[b = y_i] = \Pr[T(z) = 1 \wedge y_i = g] + \Pr[T(z) = 0 \wedge y_i = 1 - g].$$

Define z_1 to be the string of i bits of y followed by the remaining bits of u and z_2 be the string of $i-1$ bits of y followed by $1-y_i$ and then bits of r (and notice we're providing one of these inputs. Our input $z = z_1$ if $y_i = g$ and $z = z_2$ if $y_i = 1 - g$. We can rewrite the above probability as

$$\Pr[b = y_i] = \Pr[T(z_1) = 1 \wedge y_i = g] + \Pr[T(z_2) = 0 \wedge y_i = 1 - g].$$

Note that the probability we guessed y_i is exactly $1/2$, that is $\Pr[y_i = 1] = 1/2$ furthermore z_1, z_2 don't depend on g so we can split up this probability:

$$\begin{aligned}
\Pr[b = y_i] &= \Pr[T(z_1) = 1 \wedge y_i = g] + \Pr[T(z_2) = 0 \wedge y_i = 1 - g] \\
&= \Pr[T(z_1) = 1]\Pr[y_i = g] + \Pr[T(z_2) = 0]\Pr[y_i = 1 - g] \\
&= \frac{1}{2}\left(\Pr[T(z_1) = 1] + \Pr[T(z_2) = 0]\right) \\
&= \frac{1}{2}\left(\Pr[T(z_1) = 1] + 1 - \Pr[T(z_2) = 1]\right) \\
&= \frac{1}{2} + \frac{\left(\Pr[T(z_1) = 1] - \Pr[T(z_2) = 1]\right)}{2}
\end{aligned}$$

Our goal now is to transform these z_1, z_2 back into the experiments we describe above. Lets consider $\Pr[T(Y_1...Y_{i-1}Y_iU_{i+1}...U_m) = 1]$ this experiment always gets z_1 as input (the correct guess. So $\Pr[T(Y_1...Y_{i-1}Y_iU_{i+1}...U_m) = 1] = \Pr[T(z_1) = 1]$. On the other hand consider $\Pr[T(Y_1...Y_{i-1}U_iU_{i+1}...U_m) = 1]$ this experiment gets a random bit in the ith position so half the time it gets z_1 and half the time it gets z_2. That is,

$$\Pr[T(Y_1...Y_{i-1}U_iU_{i+1}...U_m) = 1] = 1/2\Pr[T(z_1) = 1] + \Pr[T(z_2) = 1].$$

Subtracting the two equations yields

$$1/2(\Pr[T(z_1) = 1] - \Pr[T(z_2) = 1]) =$$

$$Pr[T(Y_1...Y_{i-1}Y_iU_{i+1}...U_m) = 1] - \Pr[Y_1....Y_{i-1}U_iU_{i+1}...U_m = 1] \geq \frac{1}{mp(n)}.$$

That means that our predictor guesses the next bit with an inverse polynomial probability as desired. □

Notes: We did a couple of things that were a little confusing. The first is that we removed the absolute values and just assumed that the probability T output 1 when given i pseudorandom bits was higher. If this wasn't the case, we would have inverted the output. Note that the correct choice for which to do might depend on the security parameter but one choice will work for at least half of n which is enough to cause a contradiction. Second, we showed that there is some i where T has to have an advantage. How does P know which i to choose. The answer is that it doesn't, the right thing to do is to choose a random i. While our proof showed there is some i where the T has an advantage it is actually true for a random i. Not all i have to have the same advantage but the average over them is polynomial.

4 Extending a pseudorandom generator

So we've shown how to construct a pseudorandom generator based on one computational assumption. Lets assume that we have a pseudorandom generator $G : \{0,1\}^n \to \{0,1\}^m$. How can we get more bits out of this object? We can run it sequentially using its own output.

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.