CSE 5852, Modern Cryptography: Foundations Fall 2016

Lecture 12

Prof. Benjamin Fuller *Scribe: Tham Hoang*

1 Last Class

Last class we continued trying to build a pseudorandom generator. We showed that an algorithm that can predict if an exponent is greater than $x > p/2$ can be used to compute discrete logarithms. That is, consider the following function:

$$B(x) = \begin{cases} 1 \; if \; x \geq \frac{p}{2} \\ 0 \; otherwise \end{cases}$$

Lemma 1. If you can predict B(x) from g^x, you can compute discrete log efficiently.

Algorithm:

1. Computer lower order bit g^{2x} or g^{2x+1}

   Femmat's little theorem $a^{p-1} \equiv 1 \bmod p \; \forall$a

   $(g^{2x})^{\frac{p-1}{2}} = g^{x(p-1)} \equiv 1 \bmod p$

   $(g^{2x+1})^{\frac{p-1}{2}} = g^{x(p-1)}.g^{\frac{p-1}{2}} \equiv g^{\frac{p-1}{2}} \not\equiv 1 \bmod p$

2. If x is odd, then divide by g $\rightarrow$ set $y = \frac{g^x}{g}$

3. Compute square root of y for notational purposes we will assume that $y = g^x$. Then either $g^{x/2}$ or $g^{\frac{x}{2}+\frac{p-1}{2}}$

   2 issues:

   - can not distinguish which one it is

   - can not go back to Step 1 directly

   $\rightarrow$ use Predictor for $B(x)$ here

4. Check which root using predictor for B

5. If necessary, compute $g^{x/2+(p-1)/2}g^{\frac{p-1}{2}} \equiv g^{x/2} \mod p$. Now we can go to Step 1 to predict the next bit until we got the entire sequence of x.

Run time: We can obtain x if there exists a predictor of B(x) and the algorithm runs in polynomial time.

2 Pseudorandom Generator (PRG)

Now lets recall our pseudorandom generator:

$G(p, g, x) =$

1. Input g, p, x.
2. Set $x_1 = x$.
3. For $i = 1$ to m:
   (a) Set $y_{n-i+1} = B(x_i)$.
   (b) Set $x_{i+1} = g^{x_i} \mod p$.

Theorem 1. *G is a good PRG if B is hard to predict.*

Proof. Show if we have predictor P for B, we can build next bit predictor for G.

Here, we rewrite the theorem:

Theorem If B is hard to predict then G is a good PRG

Proof As with the previous result we're going to show this by contradiction. We're going to assume there is some algorithm that breaks the next bit unpredictability of G. We need to show how to convert that to an algorithm that predicts B (for a single B). Assume exists A that is a next bit predictor of G, want to build predictor for B(x). So our goal is to build some $\mathcal{A}'$. Our $\mathcal{A}'$ will take $p, g, y = g^x$ as input and is trying to predict $B(x)$. We are allowed to call $\mathcal{A}$ to help us with this task (a polynomial number of times).

$\mathcal{A}'$ Input: g, p, g^x mod p

Note that the only thing is know about $\mathcal{A}$ is there is some i where it is good at predicting. But we don't know where this i is. So we're just going to guess an i at random between 1 and n.

So what we're going to do is try and compute a sequence for the attacker. As a reminder we can compute all previous elements of the sequence.

$$B(g^{g^{\cdots g^{g^x}}}) \ldots B(g^{g^x})B(g^x)$$

$$\ldots B(g^{g^x})B(g^x) \leftarrow g^x$$

If A makes guess at position i, output that guess. Otherwise output random bit.

Pr[P outputs B(x)] = Pr[P outputs B(X) | A outputs guess at position i]Pr[A outputs guess at location i]

+ Pr[P outputs B(X) | A outputs guess at other location]

+ $\frac{1}{2}$Pr[A outputs guess at other location]

$= \Pr[\text{P outputs B(X)} \mid \text{A outputs guess at location i}]\frac{1}{m} + \frac{1}{2}(\frac{m-1}{m})$

$= \sum_i Pr[\text{A guess next bit at the location i}]\frac{1}{m} + \frac{1}{2}(\frac{m-1}{m})$

$= (\frac{1}{2} + \frac{1}{p(n)})\frac{1}{m} + \frac{1}{2}(\frac{m-1}{m})$

$\geq \frac{1}{2} + \frac{1}{m(p(n))} \geq \frac{1}{2} + \frac{1}{q(n)}$ for polynomial q □

Corollary 2. *If discrete log is hard then G is good PRG.*

Now that we've shown a single definition of a pseudorandom generator (which seems very artificial), lets consider some other definitions.

Other definitions:

1) Can't guess y_m from $y_1 \dots y_{m-1}$

2) $\forall$ PPT D $|Pr[D(y_1 \dots y_m) = 1] - Pr[D(r_1 \dots r_m) = 1]| < \epsilon$

$r_1 \dots r_m$ is true random sequence.

We'll work with the second of these definitions and make it a bit more formal. 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. Compute $y = G(s)$ Run $T(y)$ and output whatever it does.	Select random y of length m Run $T(y)$ and output whatever it does.

Definition 1. *[Yao82] G passes all statistical tests if for all PPT T, there exists negligible function $\epsilon(n)$ such that for all n,*

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

Theorem 3. *If G passes all efficient statistical tests then $Enc(s, m) = G(s) \oplus m$ has indistinguishable encryptions.*

Theorem 4. *G is next bit unpredictable if and only if G passes all statistical tests.*

Some examples of efficient tests:

1) Number of 1's

2) Max number of 1's

3) Average run length

4) Value of $g^{y_1 \cdots y_m}$

From now, G is just an arbitrary G.

Lemma If G passes all statistical tests then G is next bit unpredictable.

Proof Let $\boldsymbol{A}$ be a next bit predictor for G

Pr[A is correct on $r_1 \dots r_m$] $= \frac{1}{2}$

Pr[A is correct on $y_1 \dots y_m$] $> \frac{1}{2} + \frac{1}{p(n)}$

We will now built an efficient A'. A' will call A as subroutine to build an efficient distinguisher to point out which one of $r_1 \dots r_m$ and $y_1 \dots y_m$ is a sequence by G or random.

A' get $x_1 \dots x_m$

1) Run A until it outputs b

2) Check if $b = x_{i+1}$ output 1 if correct, 0 otherwise

3) If $\mathcal{A}$ does not guess a bit, output a random bit.

If $\mathcal{A}$ predicts b with probability $1/2 + \epsilon(n)$, then $\Pr[\mathtt{exp} - \mathtt{pr}^{G,\mathcal{A}'} = 1] = 1/2 + \epsilon(n)$. For a truly random string the predictor cannot succeed so $\Pr[\mathtt{exp} - \mathtt{r}^{T} = 1] = 1/2$. Thus the difference between these is $\epsilon(n)$ which is non-negligible by assumption, thus $\mathcal{A}'$ is a statistical test that G does not pass. This completes the proof of the lemma.

References

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