CS 290G (Fall 2014) – Introduction to Cryptography Oct 14th, 2014

Lecture 3: One-Way Functions

Instructor: Rachel Lin *Scribe: Asad Ismail*

1 Recap

Last time, we started moving towards coming up with a concrete way of providing Computational Security. In order to do this formally, we first needed to define a few key terms:

- *Model of Computation:* We define a legitimate and efficient algorithm to be a Probabilistic Polynomial-time Turing Machine (PPT). Additionally, we define an adversarial attack as feasible if it is a Non-uniform Probabilistic Polynomial-time Algorithm which is a family of PPT algorithms (Can be thought of as running a different algorithm for inputs of different length).
- *What is 'costly'?* Any attack that cannot be modeled using a non-uniform PPT machine is said to be costly.
- *Security Goals:* We can have multiple security goals formally defined. One such goal could be One-Way Functions (OWF), i.e. Our goal is that we want to come up with a function which is easy to compute but hard to invert.

Having defined these things, when we say a particular crypto system is *secure*, it just means that for any adversary, breaking the system is *costly*, and if we can prove this to be true, than our system is said to be secure.

Before moving on to formally define OWFs, we could very well ask why we've chosen the particular model of computation to be a PPT TM and the adversary to be a non-uniform PPT TM. We could argue that the adversary actually formulates a C program as an attack and launches it against the honest parties. Alternatively, we might have an adversary that doesn't formally come up with an algorithm, but *thinks* through the problem by analysing the transcript and thus somehow breaks the crypto system (human aided instead of just an algorithm). It is true that our definition does not encompass such an adversary, but our definition is still general enough for the purposes of this course in order to give us a good starting point.

2 One-Way Functions

2.1 Strong OWF

We first go on to define Strong OWFs, which are characterized by the following two properties:

- Easy to Compute: $\exists PPT$ M $\forall n$ and $\forall x$, s.t.
  $Pr[x \leftarrow \{0,1\}^n,\ M(x) \rightarrow f(x)] = 1$
- Hard to Invert: $\forall A$, $\exists$ negligible function ϵ s.t.
  $Pr[x \leftarrow \{0,1\}^n, y = f(x), x^{'} \leftarrow A(y) : f(x^{'}) = y] <= \epsilon(n)$

In this class, we play around with this definition and see whether we can construct functions which can partially (or fully) fulfill it.

2.2 Worst-Case OWF

Looking at the Strong OWF, we see that it is quite strict in its formulation (Inverting *any* input is possible only with probability $< \epsilon(n)$). If we loosen up our security criterion, we can come up with the *Worst-Case OWF* definition. The Worst-Case OWF is defined as follows:

$\exists x$ s.t. $Pr[y = f(x), x \leftarrow A(y) : f(x^{'}) = y] < 1$

Looking at this definition, there two very glaring weaknesses that we would ideally like to overcome (Both of which are solved by Strong OWFs, as we will see). First, the Worst-Case OWF definition only needs a *single* input for which the adversary cannot invert. Second, we are only guaranteed that the adversary will invert with a probability of < 1. This probability could still be arbitrarily close to 1, for example 0.99.

3 How strong is the Strong OWF?

To answer this question, we will state a lemma and go on to prove it. Note that both the weaknesses we identified with Worst-Case OWFs will be solved here.

Lemma: If function f is a Strong OWF, then $\forall A$, $\exists \epsilon^{'}(n)$ s.t. $\forall n$, $\exists$ an overwhelmingly large fraction $(1 - \epsilon^{'}(n))$ of $x \in \{0,1\}^n$ for which:

$Pr[y = f(x), x^{'} \leftarrow A(y) : f(x^{'}) = y] < \epsilon^{'}(n)$

Proof: We have our premise, which is the definition of the Strong OWF stated in Section 2.1. Let us set $\epsilon^{'}(n) = \sqrt{\epsilon(n)}$. Now we will prove that the lemma statement holds through proof by contradiction. Let's suppose the statement is false, i.e.

$\exists$ atleast a $\epsilon^{'}(n)$ fraction of $x \in \{0,1\}^n$ (we call this set S_{bad}) s.t.

$Pr[y = f(x), x^{'} \leftarrow A(y) : f(x^{'}) = y] > \epsilon^{'}(n)$

Now lets compute what the probability is for A to be able to invert a random input (Using conditional probability):

$Pr[\text{A inverts for } x \leftarrow \{0,1\}^n] >=$
$Pr[\text{A inverts for } x \leftarrow \{0,1\}^n | x \in S_{bad}].Pr[x \leftarrow \{0,1\}^n, x \in S_{bad}]$

which turns out to be:
$Pr[\text{A inverts for } x \leftarrow \{0,1\}^n] >= \epsilon^{'}(n).\epsilon^{'}(n)$

$Pr[\text{A inverts for } x \leftarrow \{0,1\}^n] >= \epsilon(n)$

which clearly contradicts the premise/definition we start out with of having a Strong OWF ($Pr[Ainverts] <= \epsilon(n)$), thus the Lemma holds.

Thus, using this lemma, we've established two things about the strength of the Strong OWF. First, it holds for an overwhelmingly large fraction of the input (weakness no. 1 of Worst-Case OWF). Second, for any of those input, the probability of inversion being successful is negligible (As opposed to < 1, weakness no. 2 of the Worst-Case OWF).

4 Construction of OWFs

Now that we've discussed the strength of the Strong OWF, we want to know whether it is possible to construct such a Strong OWF (or atleast come close to it). To start with though, lets start with an easier OWF, and see whether we can construct a Worst-Case OWF. For this, we take the help of a theorem which states:

Theorem: If $NP \neq BPP$ (a more general form of $NP \neq P$), this implies that there exists a Worst-Case OWF

We will not prove this theorem here, but just remark that this does make some intuitive sense for $NP \neq P$ to imply atleast worst-case One-Way behaviour. However, as we previously saw, Worst-Case guarantees are not strong enough for our purposes, we need something stronger. We therefore need to move from worst-case hardness to a notion of average-case hardness. We can think of this as having a distribution from which if we sample some problem, we won't be able to solve it with a certain probability.

This leads us to the idea of Factoring, and that factoring is hard. Now ideally, we'd want that if we assume factoring to be hard, that should imply that we're able to get a Strong OWF. Thinking about how factoring works, we somehow want to make the hardness of factoring equatable to the hardness of inversion. i.e. if inverting is analogous to factoring, then going forward should be analogous to the opposite of factoring, which is Multiplication.

Thus if we define a function, f_{mult} which works as follows (We take care of the trivial case of factoring with 1 and the product itself using this definition):

$$f_{mult}(x,y) = \begin{cases} 1 & \text{if x or y} = 1 \\ x.y & \text{otherwise} \end{cases}$$

But just directly looking at this definition, we know we can't get a Strong OWF because any time either x or y or both are even (Probability of 3/4), the adversary can invert very easily. To solve this problem, let's look at our basic assumption again and try understanding when it is applicable, when is factoring hard anyway?

4.1 Factoring

When we say factoring is hard, what we mean is that for any two primes picked at random multiplied together, factoring the resulting product is hard. More formally, assuming factoring is hard means:

Let $\Pi_n = \{q : q < 2^n \text{ and q is a prime } \}$

$\forall A$, $\exists \epsilon(n)$ s.t. $Pr[p \leftarrow \Pi_n, q \leftarrow \Pi_n, N = p.q : A(N) = \{p,q\}] <= \epsilon(n)$

Like we asked with the model of computation, we can very well ask why we take the assumption "Factoring is hard" to be true anyway? There are a number of reasons based on empirical evidence that suggests this is a reasonable assumption to make.

- The state-of-the-art factoring algorithm has a worst-case running time of $2^{\sqrt{N}}$.
- Heuristic solvers for factoring take time in the order of $2^{O(\sqrt[3]{N})}$
- In 2010, a 768 bit product was factored successfully using 1500 years worth of compute time using a 2.2 GHz AMD CPU with 2 GB of RAM.

Given this definition of factoring, we come back to the question of what kind of Strong OWF we can build out of this factoring assumption. Considering we've established that the assumption only holds in the case of primes, the next logical question would be to ask how frequently we encounter prime numbers in the wild, if we were picking at random.

For this purpose, let $\pi(m)$ = Number of primes smaller than m

We have two famous mathematical theorems which can help us here, one by Chebyshev which gives us the lower bound on $\pi(m)$, and one known as the Prime Number Theorem.

Theorem (Chebyshev): $\pi(m) > \frac{m}{2log(m)}$

Prime Number Theorem: $\lim_{m\to\infty} \pi(m) = \frac{m}{log(m)}$

Using these theorems, we can answer the question of how many primes there are in $\{0,1\}^n$.

$|\Pi_n| = \pi(2^n) >= \frac{2^n}{2log(2^n)} = \frac{2^n}{2n}$

Which implies that the fraction of primes in $\{0,1\}^n >= \frac{1}{2n}$, which is a polynomial fraction of primes in an n-bit string.

4.2 Building an OWF from Factoring (Second Try)

Having calculated the fraction of primes, lets now try to formulate a OWF again (starting from the definition of Worst-Case OWF which we made using the function f_mult).

$Pr[x, y \leftarrow \{0,1\}^n, z = f_{mult}(x,y) : x^{'}, y^{'} \leftarrow A(z) \text{ s.t. } f_{mult}(x^{'}, y^{'}) = z^{'}] <= 1 - \frac{1}{p(n)}$ where p(n) is a polynomial

Now lets assume that factoring is impossible ($Pr[factoring] = 0$) as opposed to simply being hard ($Pr[factoring] <= \epsilon(n)$), then our inversion probability would be:

Pr[A inverts] $<= 1 - (\frac{1}{2n})(\frac{1}{2n})$ ... where we have two $\frac{1}{2n}$'s for both x and y.

Pr[A inverts] $<= 1 - (\frac{1}{4n^2})$

This leads us to the definition of a *Weak OWF*. This solves one of the weaknesses of Worst-Case OWF (Applies to all inputs). It is characterized by the following:

- Weakly hard to invert: $Pr[x \leftarrow \{0,1\}^n, y = f(x), x^{'} \leftarrow A(y) : f(x^{'} = y] <= 1 - \frac{1}{p(n)}$

Which leads us to the following theorem:

Theorem: Factoring is hard implies that f_{mult} is a weak OWF

This is in-line with our general goal during this goal, taking an atomic hard problem (factoring being hard) and achieving a high level security goal using it (Existence of Weak OWF).

Proof: We will do a proof of reduction, i.e. We will prove the theorem using counter-position. We will try to show that if f_{mult} is NOT a weak OWF, that would imply that factoring is NOT hard. More concretely, if $\exists A$ that violates the weak OWF property with large probability, then there exists an algorithm C, that is able to factor with polynomial probability.

We want to show f_{mult} is a weak OWF w.r.t. $p(n) = 8n^2$, i.e.

$Pr_{x,y \leftarrow \{0,1\}^n}[z = f_{mult}(x,y) : \text{A inverts z}] <= 1 - \frac{1}{8n^2}$

Suppose not, lets say that $Pr_{x,y \leftarrow \{0,1\}^n}[z = f_{mult}(x,y) : \text{A inverts z}] >= 1 - \frac{1}{8n^2}$

Given this, what we want is to build an algorithm C that factors with some polynomial probability p^* ($\frac{1}{8n^2}$ in this case). More concretely:

$p^* = Pr_{p,q \leftarrow \Pi_n}[C(z^{'} = p.q) \text{ finds either p or q}] >= \frac{1}{8n^2}$

To do this, we should leverage the ability of A to break the weak OWF properly. Lets define our algorithm C to do the following:

- Run A on input $z^{'}$ to get $x^{'}$ and $y^{'}$ $(A(z^{'}) \rightarrow x^{'}, y^{'})$
- Simply output $x^{'}$ and $y^{'}$

Thus now, $p^* = Pr_{p,q \leftarrow \Pi_n}[z^{'} = f_{mult}(p,q) : A(z^{'}) \rightarrow x^{'}, y^{'}$ s.t. $f_{mult}(x^{'}, y^{'}) = z^{'}]$

$= Pr_{x,y \leftarrow \{0,1\}^n}[z^{'} = f_{mult}(x,y) : A(z^{'}) \rightarrow x^{'}, y^{'}$ s.t. $f_{mult}(x^{'}, y^{'}) = z^{'} |$ x,y are prime]

From Conditional Probability, where $Pr[A] = Pr[A|B].Pr[B] + Pr[A|B^C].Pr[B^C]$, we can write this as:

$Pr[$A inverts $z^{'}] = Pr[$A inverts $z^{'}|$x,y are primes$].Pr[$x,y are primes$] + Pr[$A inverts $z^{'}|$x,y are not primes$].Pr[$x,y are not primes$] >= 1 - \frac{1}{8n^2}$

The $Pr[$A inverts $z^{'}|$x,y are not primes] can be replaced by $<= 1$ (We know nothing about whether the adversary is able to invert this or not), and $Pr[$x,y are not primes] is replaced by $<= \frac{1}{4n^2}$ (using the Chebyshev Theorem). We also know that $Pr[$A inverts $z^{'}|$x,y are primes] is just p^*, which is what we want to find. Given all this, we are left with:

$p^*.Pr[$x,y are primes$] >= (1 - \frac{1}{8n^2}) - \frac{1}{4n^2}$

$p^*.Pr[$x,y are primes$] >= \frac{1}{8n^2}$

Which tells us that atleast, $p^* >= \frac{1}{8n^2}$, which completes the proof.

5 Going towards Strong OWF

Having achieved all this, we still havn't found the elusive Strong OWF we wan't to make. Fortunately, we can make a strong OWF using the following theorem:

Theorem: Existence of a Weak OWF implies the existence of a Strong OWF

We won't prove this right now, but we'll try constructing a strong OWF (function f) using Weak OWFs as building blocks (function $f^{'}$). Intuitively, if we define our Strong OWF to be as follows (Apply bit-wise Weak OWF to each bit of the Strong OWF function, so to speak):

$f(x_1, x_2, \ldots, x_m) = f^{'}(x_1) f^{'}(x_2) \ldots f^{'}(x_m)$

Then if *one* Weak OWF is bounded by $<= 1 - \frac{1}{p(n)}$, then a multiplication of m such functions would ideally be bounded by $<= (1 - \frac{1}{p(n)})^m$, which would be great. However, this will only be true if the m different elements of the pre-image computed by the adversary are independent, which will not always be the case. So even though our approach had merit, we cannot simply string together Weak OWFs to obtain a Strong OWF. We will discuss a better way of achieving Strong OWFs in the next class.