Topic: A Universal One-way Function
Lecturer: Wei-Kai Lin (TA: Arup Sarker)

Date: Oct 1, 2024
Scriber: Eric Xie, Jinye He

1 A Universal One-Way Function

"Cryptographers Seldom Sleep Well"

—— Silvio Micali, personal communication to Joe Kilian, 1988

In the last lecture, we constructed a one-way function (OWF) assuming that factoring is hard. Unfortunately, we do not know how to prove factoring is hard, and Shor's algorithm [Wik] seems to be evidence that factoring might be easy in the future. So, cryptographers want to construct a OWF based on weaker assumptions. Fortunately, if we assume that a valid OWF exists, it is possible for us to construct a OWF solely based on that assumption. That is the universal OWF.

1.1 Universal OWF Construction

The idea of constructing a universal one-way function (OWF) is to construct a function that can computes any efficiently computable function. Since any OWF must be easy to compute with a constant size Turing machine, we can use random strings to sample such a Turing machine. Even though we may not know the exact machine, we can still achieve this with almost constant probability.

Theorem 1. *If there exists a OWF, then the following polynomial-time computable function $f_{universal}$ is a weak OWF.*

Proof. We will construct the function $f_{universal}$ and show that it is a weak OWF. As shown in the last lecture, weak OWFs can be converted to strong OWFs.

Construction of $f_{universal}$. For an input y, where the length of the input $|y| = n$.

1. Represent y as $M|x$, where y is interpreted as a pair (M, x) consisting of a Turing machine M and a bitstring x, with $|M| = \log n$ and $|x| = n - \log n$;
2. Run M on x for $T = n^2$ steps;
3. If M halts within T steps, output $(M, M(x))$. Otherwise, output $\perp$.

Now, assume that g is a one-way function (OWF), though we do not know the exact function. There exists a Turing machine M_g that computes g in polynomial time. Let $|M_g|$ be the description length of M_g, and note that we can pad the description of any Turing machine with a special $\perp$ symbol to arbitrary length. For sufficiently large n, the first random $\log n$ bits of y will correspond to M_g with probability $1/n$. Hence, the output of $f_{universal}(y)$ is exactly $g(x)$.

We now claim that $f_{universal}$ is a weak OWF. For simplicity, we omit the subscript of the universal OWF in the following part of the proof. Suppose f is not a weak OWF. Then, there exists a non-uniform probabilistic polynomial-time (NUPPT) algorithm $\mathcal{A}$ such that, for every polynomial q and infinitely many n,

$$\Pr_{y \leftarrow \{0,1\}^n}[\mathcal{A}(1^n, f(y)) \in f^{-1}(f(y))] \geq 1 - 1/q(n).$$

In particular, we can take $q(n) = n^2$. For $n \geq 2^{|M_g|}$ and random n-bit input $y = M \| x$, the probability of that M represents M_g is $\frac{1}{2^{\log n}} = \frac{1}{n}$. Let $x' \leftarrow \{0,1\}^{n-\log n}$, $z' = g(x')$ and $M' \leftarrow \{0,1\}^{\log n}$. Now we construct an NUPPT $\mathcal{B}$ to invert g as follows:

1. Run $\mathcal{A}(1^n, z')$ to output y;
2. Interpret y as (M, x);
3. If $z' = g(x)$, output x; otherwise output $\perp$.

Notice that

$$\begin{aligned}
\frac{1}{n^2} &\geq \Pr_{\substack{x' \leftarrow \{0,1\}^{n-\log n} \\ M' \leftarrow \{0,1\}^{\log n}}}[\mathcal{A}(1^n, f(M' \| x')) \notin f^{-1}(f(M' \| x'))] \\
&\geq \Pr_{\substack{x' \leftarrow \{0,1\}^{n-\log n} \\ M' \leftarrow \{0,1\}^{\log n}}}[\mathcal{A}(1^n, f(M' \| x')) \notin f^{-1}(f(M' \| x')) | M' = M_g] \Pr_{M' \leftarrow \{0,1\}^{\log n}}[M' = M_g] \\
&= \Pr_{x' \leftarrow \{0,1\}^{n-\log n}}[\mathcal{A}(1^n, f(M_g \| x')) \notin f^{-1}(f(M_g \| x'))] \Pr_{M' \leftarrow \{0,1\}^{\log n}}[M' = M_g] \\
&= \Pr_{x' \leftarrow \{0,1\}^{n-\log n}}[\mathcal{A}(1^n, f(M_g \| x')) \notin f^{-1}(f(M_g \| x'))] \frac{1}{n}.
\end{aligned}$$

Then we have

$$\Pr_{x' \leftarrow \{0,1\}^{n-\log n}}[\mathcal{B}(1^n, g(x')) \notin g^{-1}(g(x'))] = \Pr_{x' \leftarrow \{0,1\}^{n-\log n}}[\mathcal{A}(1^n, f(M_g \| x')) \notin f^{-1}(f(M_g \| x'))] \leq n \cdot \frac{1}{n^2} = \frac{1}{n}$$

which implies that g is not a OWF, resulting in a contradiction. □

We omitted an explanation in the previous proof for why we set the running time T as n^2. The following lemma clarifies that there exists an OWF computable in $O(n^2)$ time, assuming the existence of an OWF.

Lemma 2. *If there exists a OWF g computable in time n^c, then there exists then there is some OWF g' computable in time $O(n^2)$.*

Proof. For any input $x \in \{0,1\}^{n^c}$, interpret $x = x_1 \| x_2$ s.t. $|x_1| = n^c - n$, and then define $g'(x) = g'(x_1 \| x_2) := x_1 \| g(x_2)$. Let $m = n^c$ be the input size of g'. It is easy to see that g' is computable in $O(m^2)$ time, and it follows by standard reduction that g' is hard to invert if g is a OWF. □

Although this construction is theoretically sound, it is highly inefficient in practice. For instance, suppose there exists a one-way function that can be computed by a Turing machine with a description length of 1000 bits. To ensure the universal OWF is hard to invert, we would need the input size n such that $\log n \geq 1000$. This implies that the universal OWF would only be secure for inputs of size $n = |x| \geq 2^{1000}$, which is impractically large.

2 From OWF to PRG

Next, we aim to demonstrate that a PRG can be constructed from OWF. We first recap and compare the properties of PRGs and OWFs.

PRG: efficient, expanding, pseudorandom.

OWF: efficient, hard to invert.

Differences between OWF and PRG:

- Output of OWF can be shorter or longer, but PRG must be expanded.
- OWF just needs to be hard to invert, doesn't need to be pseudorandom.

Properties of OWF:

- The output of an OWF, when given a uniformly random input, must exhibit sufficient randomness. (Otherwise, one could guess randomly and have a reasonable probability of finding the preimage.)
- It must be difficult to predict the input x from $f(x)$, even if the OWF f is a one-to-one function, meaning x is uniquely determined by $f(x)$. The OWF introduces additional pseudorandomness.

References

[Wik] Wikipedia. Shor's algorithm.