CS 6222 Graduate Cryptography September 23, 2025

Lecture 9: One-Way Functions and Factoring

Lecturer: Jack Doerner *Scribe: Matthew Lucio*

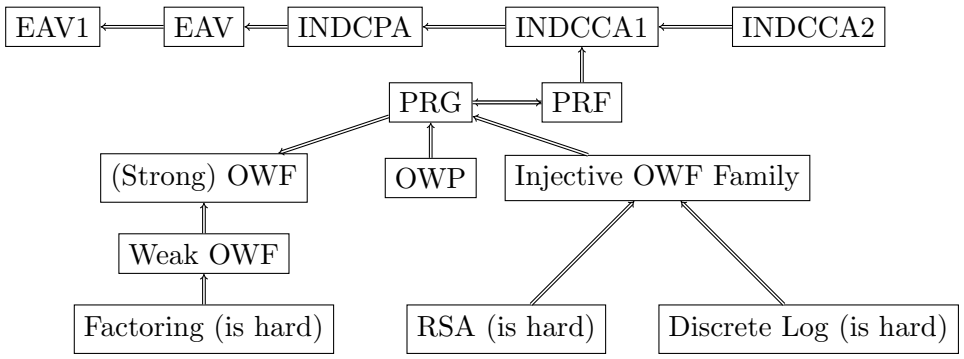
1 Topics Covered

- Putting Things Into Perspective
- Three Flavors of One-Way Functions
- Weak One-Way Functions from the Hardness of Factoring
- Hardness Amplification Preview

2 Putting Things Into Perspective

Last time we proved $PRG \Rightarrow PRF$. Now, we will introduce One Way Functions (OWF), which are the fundamental primitive of modern cryptography. This will provide a bridge from concrete mathematical assumptions, which concern specific mathematical functions that you can implement, to the general definitions we discussed before, which then give us IND-CPAsecure encryption.

The diagram below illustrates the implications we will show over the next few classes, as well as those we have already proven. Note that there are other implications we will not prove. Most notably, Håstad et al. [2] proved that we can construct a PRG from any OWF, but the proof is rather intense, so we will not go through it in class.



3 One Way Functions in Three Flavors

Unlike PRG and PRF, we don't need any concept of computational indistinguishability. Instead, we are bridging the gap from fundamental math to cryptography.

Definition 1 (Worst-Case OWF) *A function* $f : \{0,1\}^* \to \{0,1\}^*$ *is a* worst-case *one-way function if*

1. f *is PPT*
2. $\forall$ *NUPPT* $\mathcal{A}$, $\exists n \in \mathbb{N}$, $x \in \{0,1\}^n$ *such that*

$$\Pr[f(x') = f(x) : x' \leftarrow \mathcal{A}(1^n, f(x))] < 1$$

Note that functions of the above type certainly exist if $P \neq NP$, but there is no guarantee that we can easily find the input x that is hard to invert! Unfortunately, we do not know how to use such one-way functions to build cryptography. We need a stronger notion of one-wayness. Note also that we must give the adversary the security parameter in this game, because there is no guarantee that f outputs any particular number of bits.

Definition 2 (Strong (Average-Case) OWF) *A function* $f : \{0,1\}^* \to \{0,1\}^*$ *is a strong OWF if*

1. f *is PPT*
2. $\forall$ *NUPPT* $\mathcal{A}$ $\exists$ *negligible function* ε *s.t.* $\forall n \in \mathbb{N}$

$$\Pr[f(x') = f(x) : x \leftarrow \{0,1\}^n, x' \leftarrow \mathcal{A}(1^n, f(x))] < \varepsilon(n)$$

This function is useful for cryptography, but unfortunately we don't know how to construct it *directly* from concrete mathematical assumptions. For example, we think it's hard to factor the product of two large primes. But if we suppose I sample two random numbers x and y (not necessarily prime) and give the product z to the adversary to factor, $\frac{3}{4}$ of the time one of the inputs will be an even number, and the adversary can trivially factor z into 2 and $\frac{z}{2}$. We need to capture this definition of hardness: many inputs produce easy-to-invert outputs, yet it is also easy to find inputs that produce hard-to-invert outputs.

Definition 3 (Weak (Average-Case) OWF) *A function* $f : \{0,1\}^* \to \{0,1\}^*$ *is a* weak *OWF if*

1. f *is PPT*
2. $\exists$ *polynomial* μ *s.t.* $\forall$ *NUPPT* $\mathcal{A}$ $\exists$ $n_0 \in \mathbb{N}$ *s.t.* $\forall n > n_0$

$$\Pr[f(x') = f(x) : x \leftarrow \{0,1\}^n, x' \leftarrow \mathcal{A}(1^n, f(x))] < 1 - \frac{1}{\mu(n)}$$

Notice that in the above definition, the probability that the adversary inverts f actually approaches 1 as n increases. The main thing is that it doesn't approach 1 too fast: in particular, it is not overwhelming. In a future class, we will prove that Weak OWFs $\Rightarrow$ Strong OWFs. The other direction is trivial.

4 Weak OWFs from the Hardness of Factoring

Our first putative one-way function is a function you have likely seen before.

Definition 4 (Multiplication OWF)

$$f_{\mathsf{mult}} : x, y \mapsto \begin{cases} 1 & \text{if } x = 1 \vee y = 1 \\ x \cdot y & \text{otherwise} \end{cases}$$

Notice that the first of the two cases in the above function ensures that $(1, z)$ is never a valid preimage of the output $z \neq 1$ under f.

Definition 5 (The Factoring Assumption) *Let* $\Pi_n = \{q \in \mathbb{N} : q < 2^n$ *and* q *is prime*$\}$. $\forall$ *NUPPT* $\mathcal{A}$ $\exists$ *negligible* ε *s.t.*

$$\Pr[\mathcal{A}(N) \in \{p, q\} : p, q \leftarrow \Pi_n, N := p \cdot q] < \varepsilon(n)$$

In other words, the probability that a NUPPT adversary can factor a product of two random primes is negligible. Obviously real-world factoring algorithms exist—the problem has been studied in some guise since at least Eratosthenes—so how do they compare to this bound? How hard is factoring, concretely, when the best currently-known algorithms are used? The most efficient algorithm with a rigorously proven[1] running time can factor biprimes in time $2^{O(\sqrt{nlogn})}$ [3]; this result has not been improved much since the 1980s. There is also a heuristic algorithm called the General Number Field Sieve, which runs in time $2^{O(n^{1/3}log^{2/3}(n))}$. However, sufficiently-advanced quantum computers have the potential to factor large biprimes easily.[2]

In order to argue for the hardness of f_{mult} under the factoring assumption, we must prove that we have a good chance of sampling two primes if we sample inputs for f_{mult} at random. In other words, we need a lower bound on the density of prime numbers.

Definition 6 (The Prime Counting Function) *Let* $\pi(x) = |\{q : q < x \text{ and } q \text{ is prime}\}|$.

Thus, for example, $\pi(2^n) = |\Pi_n|$. An approximate bound is given by the prime number theorem, but we need a strict lower bound. Many strict bounds are known, but for our purposes Chebyshev's is sufficient.

Fact 1 (Prime Number Theorem) *As* $n \to \infty$ $\pi(n) \approx \frac{n}{\ln(n)}$.[3]

Fact 2 (Chebyshev) *For* $x > 1, \pi(x) > \frac{x}{2\log_2(x)}$.

Corollary 1 $\Pr[p \text{ is prime} : p \leftarrow \{0,1\}^n] > \frac{1}{2n}$.

We will construct a reduction which performs primality testing. Consequently, it's important to extablish that primality testing is efficient.

[1] Note that it assumes the Generalized Riemann Hypothesis is true.

[2] In 1994 Peter Shor [4] found a quantum algorithm that runs in time $O(n^3)$ but needs $O(n)$ logical qubits. The number of physical qubits required is orders of magnitude more, due to error correction.

[3] Gauss actually conjectured this as a teenager!

Fact 3 (AKS [1]) *Primality testing is in* P.

Finally, we are to prove that f_{mult} is weakly one-way.

Theorem 1 *If the factoring assumption is true, then* f_{mult} *is a weak one-way function. In particular,* $\forall$ *NUPPT* $\mathcal{A}$, $\exists n_0$ *s.t.* $\forall n \geq n_0$

$$\Pr[w \in \{p,q\} : p,q \leftarrow \{0,1\}^n, N := f_{\mathsf{mult}}(p,q), w \leftarrow \mathcal{A}(1^{2n}, N)] < 1 - \frac{1}{8n^2}$$

Note that in the above theorem statement, we have defined the security parameter to be the length of each putative prime p, q, rather than the total input length of f_{mult}. Consequently, we need to scale the adversary's security parameter by a factor of 2.

Proof: Assume towards contradiction $\exists \mathcal{A}$ that violates Theorem 1. That is, $\mathcal{A}$ succeeds in inverting f_{mult} with probability $\geq 1 - \frac{1}{8n^2}$ for infinitely many $n \in \mathbb{N}$.

Now we will construct a reduction $\mathcal{R}(N)$ that plays the factoring game (i.e. it plays the role of $\mathcal{A}$ in Definition 5).

Construction 1 ($\mathcal{R}(N)$)

1. *Samples* $x, y \leftarrow \{0,1\}^n$
2. *If* x, y *are both prime, set* $N' := N$. *Otherwise, set* $N' := x \cdot y$.[4]
3. $w \leftarrow \mathcal{A}(1^{2n}, N')$
4. *Output* w *if* x *and* y *are both prime, otherwise output* 0

First we must argue that our reduction is a valid adversary for the factoring game, and that $\mathcal{A}$ behaves as it does in the one-way function game. The following claims can be verified by inspection.

Claim 1 *Since primality testing* $\in \mathsf{P}$ *(see Fact 3) and* $\mathcal{A}$ *is NUPPT,* $\mathcal{R}$ *is NUPPT.*

Claim 2 *The distribution of* (N', w) *in Construction is identical to the distribution of* (N, w) *in Theorem 1.*

Now we will bound the failure probability of our reduction. There are two ways it could fail: becausde it does not set $N' := N$, or because the adversary fails to factor N'. The next claim follows from Corollary 1:

Claim 3 $\Pr[N' \neq N] < 1 - \frac{1}{4n^2}$

By our assumption on the success probability of $\mathcal{A}$, we have:

Claim 4 *For infinitely many* $n \in \mathbb{N}$ $\Pr[w$ is **not** a nontrivial factor of $N'] < \frac{1}{8n^2}$

[4] This instruction ensures that we give the adversary exactly the input distribution it expects in the one-way function game for f_{mult}, which comprises the products of two random (not necessarily prime) numbers.

By taking a union bound on Claims 3 and 4, we have:

Claim 5 *For infinitely many* $n \in \mathbb{N}$

$$\Pr[w \text{ is } \textbf{not} \text{ a nontrivial factor of } N] < 1 - \frac{1}{4n^2} + \frac{1}{8n^2} = 1 - \frac{1}{8n^2}$$

And from Claim 5 it follows that for infinitely many $n \in \mathbb{N}$, we have

$$\Pr[\mathcal{R}(N) \in \{p, q\} : p, q \leftarrow \Pi_n, N := p \cdot q] > \frac{1}{8n^2}$$

Thus, we have shown that if such an adversary $\mathcal{A}$ existed, then our construction $\mathcal{R}$ would contradict the factoring assumption. Therefore, no such adversary $\mathcal{A}$ can exist, and Theorem 1 holds. □

We have proven that multiplication gives us a weak one-way function, but what about strong one-way functions? In a later class, we will prove that we can *amplify* the hardness of any weak OWF to create a strong OWF. For now, let's take an informal look at how we might do that for multiplication, specifically.

5 Hardness Amplification Preview

How many pairs of random numbers do we need to sample in order to ensure that at least one pair contains two primes with overwhelming probability? Corollary 1 tells us:

Corollary 2

$$\Pr[\exists i \in [4n^3] \text{ s.t. } p_i, q_i \text{ are prime } : p_i, q_i \leftarrow \{0,1\}^n \ \forall i \in [4n^3]] \geq 1 - \frac{1}{e^n}$$

Proof: By Corollary 1 the probability that no such i exists is

$$\left(1 - \frac{1}{4n^2}\right)^{4n^3} = \left(1 - \frac{1}{4n^2}\right)^{4n^2 \cdot n} < e^{-n}$$

for all $n > 1$. □

What if we constructed a one-way function that performed $4n^3$ multiplications in parallel, instead of just one? That is, consider the following putative one-way function:

$$f_{\mathsf{smult}} : (x_1, y_1, \ldots, x_{4n^3}, y_{4n^3}) \mapsto (f_{\mathsf{mult}}(x_1, y_1), \ldots, f_{\mathsf{mult}}(x_{4n^3}, y_{4n^3}))$$

To prove that the above function is secure, we would need to modify Construction 2 (the reduction we constructed for the proof of Theorem 1) to solve a single instance of the factoring problem, given an adversary that inverts $4n^3$ multiplications simultaneously. If the reduction samples $4n^3$ pairs of numbers internally, Corollary 2 guarantees that with overwhelming probability there will be at least one pair of primes among them, and the reduction can substitute its challenge N for the product of the first such pair (analogously to

Step 2 in Construction 2). Because the reduction has an overwhelming probability of finding a place to plug in its challenge, it has negligible loss in advantage relative to the adversary, and thus the hardness of factoring implies that f_{smult} is a *strong* one-way function.

This is a simple version of a technique known as *hardness amplification*. In the next lecture we will show that you can amplify the hardness of *any* weak one-way function to create a strong one.

References

[1] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Annals of Mathematics*, 160:781–793, 2004.

[2] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. doi:10.1137/S0097539793244708.

[3] Hendrik Lenstra and Carl Pomerance. A rigorous time bound for factoring integers. *J. Amer. Math. Soc. 5*, 5, 09 1992. doi:10.2307/2152702.

[4] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994. doi:10.1109/SFCS.1994.365700.