One-way Functions

601.642/442: Modern Cryptography

Fall 2018

Today's Agenda

- Learning the crypto language
  - Modeling "real-world" adversaries
  - Defining security against such adversaries
- One-way Functions (OWFs)
  - Motivation
  - Definitions: Strong and Weak OWFs
  - Candidate OWF

How to model the adversary?

- **Adversary's Resources:** Is the adversary an all powerful entity or does it have bounded computational resources?
- **Adversary's Strategy:** Can the adversary use its resources anyway it likes or is it restricted to certain strategies?

Adversary Model

Goal: Model real-world adversaries

- In practice, *everyone* has bounded computational resources.
- Therefore, it is reasonable to model the adversary as such an entity
- However, we do not make any assumptions about adversarial strategy. Adversary can use its bounded computational resources however intelligently it likes.

Adversary Model (contd.)

- Turing machines capture all types of computations that are possible.
- So our adversary will be a computer program or an algorithm, modeled as a Turing machine.
- Our adversary will also be *efficient* (captured via its running time)
- Computational Security: Security against efficient adversaries
- Information-theoretic Security: Security against inefficient adversaries
- In this course, we will mostly focus on computational security

Algorithms and Running Time

Definition (Algorithm)

An *algorithm* is a deterministic Turing machine whose input and output are strings over the binary alphabet $\Sigma = \{0, 1\}$.

Definition (Running Time)

An algorithm $\mathcal{A}$ is said to run in time $T(n)$ if for all $x \in \{0,1\}^n$, $\mathcal{A}(x)$ halts within $T(|x|)$ steps. $\mathcal{A}$ runs in **polynomial time** if there exists a constant c such that $\mathcal{A}$ runs in time $T(n) = n^c$.

An algorithm is *efficient* if it runs in polynomial time.

Randomized Algorithms

Definition (Randomized Algorithm)

A *randomized algorithm*, also called a probabilistic polynomial time Turing machine (PPT) is a Turing machine equipped with an extra *randomness* tape. Each bit of the randomness tape is uniformly and independently chosen.

- Output of a randomized algorithm is a distribution.
- This notion captures what we can do efficiently *ourselves*. (uniform TMs)

The Adversary

- The adversary could be more tricky...
- For example, the adversary might posses a *different* algorithm for each input size, each of which might be efficient.
- This still counts efficient since he is using polynomial time resources!
- We call this a *non-uniform* adversary since the algorithm is not uniform across all input sizes.

Non-Uniform PPT

Definition (Non-Uniform PPT)

A *non-uniform probabilistic polynomial time Turing machine* A is a sequence of probabilistic machines $A = \{A_1, A_2, \ldots\}$ for which there exists a polynomial $p(\cdot)$ such that for every $A_i \in A$, the description size $|A_i|$ and the running time of A_i are at most $p(i)$. We write $A(x)$ to denote the distribution obtained by running $A_{|x|}(x)$.

- Our adversary will usually be a non-uniform PPT Turing machine. (most general)

One Way Functions

- A function is one-way if it "easy to compute," but "hard to invert"
- Necessary for the existence of most cryptographic primitives (e.g., encryption, digital signatures)
- Also sufficient for some cryptographic primitives (e.g., pseudorandom functions, digital signatures)
- OWFs are at the bottom of the cryptographic "complexity zoo;" hence the natural first primitive for study

Defining One Way Functions: Attempt 1

Attempt 1: A function $f : \{0,1\}^* \to \{0,1\}^*$ is a one-way function (OWF) if it satisfies the following two conditions:

- **Easy to compute:** there is a polynomial-time algorithm $\mathcal{C}$ s.t. $\forall x \in \{0,1\}^*$,
$$\Pr\left[\mathcal{C}(x) = f(x)\right] = 1.$$
- **Hard to invert:** for every non-uniform PPT adversary $\mathcal{A}$, for any input length $n \in \mathbb{N}$

Probability of Inversion is small

Defining One Way Functions: Attempt 1

Attempt 1: A function $f : \{0,1\}^* \to \{0,1\}^*$ is a one-way function (OWF) if it satisfies the following two conditions:

- **Easy to compute:** there is a polynomial-time algorithm $\mathcal{C}$ s.t. $\forall x \in \{0,1\}^*$,
$$\Pr\left[\mathcal{C}(x) = f(x)\right] = 1.$$
- **Hard to invert:** for every non-uniform PPT adversary $\mathcal{A}$, for any input length $n \in \mathbb{N}$
$$\Pr\left[\mathcal{A} \text{ inverts } f(x) \text{ for random } x\right] \leqslant \textit{small}.$$

This is called **average-case** hardness.

Defining One Way Functions: Attempt 1

Attempt 1: A function $f : \{0,1\}^* \to \{0,1\}^*$ is a one-way function (OWF) if it satisfies the following two conditions:

- **Easy to compute:** there is a polynomial-time algorithm $\mathcal{C}$ s.t. $\forall x \in \{0,1\}^*$,
$$\Pr\left[\mathcal{C}(x) = f(x)\right] = 1.$$
- **Hard to invert:** for every non-uniform PPT adversary $\mathcal{A}$, for any input length $n \in \mathbb{N}$
$$\Pr\left[x \xleftarrow{\$} \{0,1\}^n;\ \mathcal{A} \text{ inverts } f(x)\right] \leqslant \textit{small}.$$

Defining One Way Functions: Attempt 1

Attempt 1: A function $f : \{0,1\}^* \to \{0,1\}^*$ is a one-way function (OWF) if it satisfies the following two conditions:

- **Easy to compute:** there is a polynomial-time algorithm $\mathcal{C}$ s.t. $\forall x \in \{0,1\}^*$,
$$\Pr\left[\mathcal{C}(x) = f(x)\right] = 1.$$
- **Hard to invert:** for every non-uniform PPT adversary $\mathcal{A}$, there exists a fast decaying function $\nu(\cdot)$ s.t. for any input length $n \in \mathbb{N}$
$$\Pr\left[x \xleftarrow{\$} \{0,1\}^n;\ \mathcal{A} \text{ inverts } f(x)\right] \leqslant \nu(n).$$

Negligible Function

Definition (Negligible Function)

A function $\nu(n)$ is negligible if for every c, there exists some n_0 such that for all $n > n_0$, $\nu(n) \leqslant \frac{1}{n^c}$.

1. Negligible function decays faster than all "inverse-polynomial" functions
2. That is, $n^{-\omega(1)}$

Defining One Way Functions: Attempt 1

Attempt 1: A function $f : \{0,1\}^* \to \{0,1\}^*$ is a one-way function (OWF) if it satisfies the following two conditions:

- **Easy to compute:** there is a polynomial-time algorithm $\mathcal{C}$ s.t. $\forall x \in \{0,1\}^*$,
$$\Pr\left[\mathcal{C}(x) = f(x)\right] = 1.$$
- **Hard to invert:** for every non-uniform PPT adversary $\mathcal{A}$, there exists a *negligible* function $\nu(\cdot)$ s.t. for any input length $\forall n \in \mathbb{N}$:
$$\Pr\left[x \xleftarrow{\$} \{0,1\}^n;\ \mathcal{A} \text{ inverts } f(x)\right] \leqslant \nu(|x|).$$

One Way Functions: Definition

Definition (One Way Function)

A function $f : \{0,1\}^* \to \{0,1\}^*$ is a *one-way function* (OWF) if it satisfies the following two conditions:

- **Easy to compute:** there is a polynomial-time algorithm $\mathcal{C}$ s.t. $\forall x \in \{0,1\}^*$,
$$\Pr\left[\mathcal{C}(x) = f(x)\right] = 1.$$
- **Hard to invert:** there exists a *negligible* function $\nu : \mathbb{N} \to \mathbb{R}$ s.t. for every non-uniform PPT adversary $\mathcal{A}$ and $\forall n \in \mathbb{N}$:
$$\Pr\left[x \leftarrow \{0,1\}^n, x' \leftarrow \mathcal{A}(1^n, f(x)) : f(x') = f(x)\right] \leqslant \nu(n).$$

- Note that adversary receives n as an input in unary. Think: Why?
- The above definition is also called **strong** one-way functions.

Injective OWFs and One Way Permutations (OWP)

- **Injective or 1-1 OWFs**: each image has a *unique* pre-image:

$$f(x_1) = f(x_2) \implies x_1 = x_2$$

- **One Way Permutations (OWP)**: 1-1 OWF with the additional conditional that "each image has a pre-image"

  (Equivalently: domain and range are of same size.)

Existence of OWFs

- Do OWFs exist? NOT Unconditionally — proving that f is one-way requires proving (at least) $\mathbf{P} \neq \mathbf{NP}$.
- However, we can construct them ASSUMING that certain problems are hard.
- Such constructions are sometimes called "candidates" because they are based on an assumption or a conjecture.

Factoring Problem

- Consider the **multiplication** function $f_\times : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$:

$$f_\times(x, y) = \begin{cases} \perp & \text{if } x = 1 \vee y = 1 \\ x \cdot y & \text{otherwise} \end{cases}$$

- The first condition helps exclude the trivial factor 1.
- Is $f_\times$ a OWF?
- **Clearly not!** With prob. 1/2, a random number (of any fixed size) is *even*. I.e., xy is even w/ prob. $\frac{3}{4}$ for random (x, y).
- Inversion: given number z, output $(2, z/2)$ if z is even and $(0, 0)$ otherwise! (succeeds 75% time)

Factoring Problem (continued)

- Eliminate such trivial small factors.
- Let Π_n be the set of all **prime** numbers $< 2^n$.
- Choose numbers p and q randomly from Π_n and multiply.
- This is unlikely to have small trivial factors.

Assumption (Factoring Assumption)

For every (non-uniform PPT) adversary $\mathcal{A}$, there exists a negligible function ν such that

$$\Pr\left[p \xleftarrow{\$} \Pi_n; q \xleftarrow{\$} \Pi_n; N = pq : \mathcal{A}(N) \in \{p, q\}\right] \leqslant \nu(n).$$

Factoring Problem (continued)

- Factoring assumption is a well established conjecture.
- Studied for a long time, with no known polynomial-time attack.
- Best known algorithms for breaking Factoring Assumption:

$$2^{O\left(\sqrt{n \log n}\right)} \qquad \text{(provable)}$$
$$2^{O\left(\sqrt[3]{n \log^2 n}\right)} \qquad \text{(heuristic)}$$

- Can we construct OWFs from the Factoring Assumption?

Back to Multiplication Function

- Let's reconsider the function $f_\times : \mathbb{N}^2 \to \mathbb{N}$.
- Clearly, if a random x and a random y happen to be prime, no PPT $\mathcal{A}$ can invert (except with negligible probability). Call it the GOOD case.
- If GOOD case occurs with probability $> \varepsilon$,

  $\Rightarrow$ every PPT $\mathcal{A}$ must fail to invert $f_\times$ with probability at least ε.
- Now suppose that ε is a noticeable function (Definition of noticeable function later; for now think of it as inverse polynomial.)

  $\Rightarrow$ every $\mathcal{A}$ must fail to invert $f_\times$ with noticeable probability.
- This is already useful!
- Usually called a **weak** OWF.

Weak One Way Functions

Definition (Weak One Way Function)

A function $f : \{0,1\}^* \to \{0,1\}^*$ is a *weak one-way function* if it satisfies the following two conditions:

- **Easy to compute:** there is a polynomial-time algorithm $\mathcal{C}$ s.t. $\forall x \in \{0,1\}^*$,
$$\Pr\left[\mathcal{C}(x) = f(x)\right] = 1.$$
- **Somewhat hard to invert:** there is a noticeable function $\varepsilon : \mathbb{N} \to \mathbb{R}$ s.t. for every non-uniform PPT $\mathcal{A}$ and $\forall n \in \mathbb{N}$:
$$\Pr\left[x \leftarrow \{0,1\}^n, x' \leftarrow \mathcal{A}(1^n, f(x)) : f(x') \neq f(x)\right] \geqslant \varepsilon(n).$$

Noticeable means $\exists c$ and integer N_c s.t. $\forall n > N_c$: $\varepsilon(n) \geqslant \frac{1}{n^c}$.

Back to Multiplication

- Can we prove that $f_\times$ is a weak OWF?
- Remember the GOOD case? Both x and y are prime.
- If we can show that GOOD case occurs with noticeable probability, we can prove that $f_\times$ is a weak OWF.

Theorem

Assuming the factoring assumption, function $f_\times$ is a weak OWF.

- Proof Idea: The fraction of prime numbers between 1 and 2^n is noticeable!
- Chebyshev's theorem: An n bit number is a prime with probability $\frac{1}{2n}$
- Next time: Formal proof by reduction

What about strong OWFs?

- Can we construct strong OWFs from the Factoring Assumption?
- Even better:
  Can we construction strong OWFs from ANY weak OWF?
- Yes! Yao's Hardness Amplification Theorem.
- Details Next time!