One-way Functions

600.442: Modern Cryptography

Fall 2016
Today’s Agenda

- Learning the crypto language
  - Modeling “real-world” adversaries
  - Defining security against such adversaries
- Definition of One-way functions
- Candidate One-way function
Modeling the adversary

- In practice, *everyone*, including the adversary has some bounded computational resources.
- Adversary can use these computational resources however intelligently he likes, but it is still bounded by these resources.
- **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.
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 $A$ is said to run in time $T(n)$ if for all $x \in \{0, 1\}^n$, $A(x)$ halts within $T(|x|)$ steps. $A$ runs in polynomial time if there exists a constant $c$ such that $A$ runs in time $T(n) = n^c$.

An algorithm is *efficient* if it runs in polynomial time.
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.
Definition (Non-Uniform PPT)

A non-uniform probabilistic polynomial time Turing machine is a 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)
Attempt 1: A function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ is a one-way function (OWF) if it satisfies the following two conditions:

- **Easy to compute:** there is a PPT algorithm $C$ s.t. $\forall x \in \{0, 1\}^*$,

$$\Pr[C(x) = f(x)] = 1.$$ 

- **Hard to invert:** for every non-uniform PPT adversary $A$, for any input length $n \in \mathbb{N}$

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

- **Easy to compute**: there is a PPT algorithm $C$ s.t. $\forall x \in \{0, 1\}^*$,

  $$\Pr[C(x) = f(x)] = 1.$$

- **Hard to invert**: for every non-uniform PPT adversary $A$, for any input length $n \in \mathbb{N}$

  $$\Pr[A \text{ inverts } f(x) \text{ for random } x] \leq \text{small}.$$
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 PPT algorithm $C$ s.t. $\forall x \in \{0, 1\}^*$,

  $$\Pr[C(x) = f(x)] = 1.$$ 

- **Hard to invert:** for every non-uniform PPT adversary $A$, for any input length $n \in \mathbb{N}$

  $$\Pr[x \leftarrow \{0, 1\}^n; A \text{ inverts } f(x)] \leq \text{small}.$$
Attempt 1: A function \( f : \{0, 1\}^* \rightarrow \{0, 1\}^* \) is a one-way function (OWF) if it satisfies the following two conditions:

- **Easy to compute:** there is a PPT algorithm \( C \) s.t. \( \forall x \in \{0, 1\}^* \),
  \[
  \Pr[ C(x) = f(x) ] = 1.
  \]

- **Hard to invert:** for every non-uniform PPT adversary \( A \), there exists a fast decaying function \( \nu(\cdot) \) s.t. for any input length \( n \in \mathbb{N} \)
  \[
  \Pr[ x \leftarrow \{0, 1\}^n ; \ A \text{ inverts } f(x) ] \leq \nu(n).
  \]
**Negligible Function**

**Definition (Negligible Function)**

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

1. Negligible function decays faster than all “inverse-polynomial” functions
2. That is, $n^{-\omega(1)}$
Attempt 1: A function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ is a one-way function (OWF) if it satisfies the following two conditions:

- **Easy to compute:** there is a PPT algorithm $C$ s.t. $\forall x \in \{0, 1\}^*$,
  \[
  \Pr[C(x) = f(x)] = 1.
  \]

- **Hard to invert:** for every non-uniform PPT adversary $A$, there exists a negligible function $\mu(\cdot)$ s.t. for any input length $\forall n \in \mathbb{N}$:
  \[
  \Pr[x \xleftarrow{\$} \{0, 1\}^n; A \text{ inverts } f(x)] \leq \nu(|x|).
  \]

Technical Problem: What is $A$’s input?
A’s Input

- Let’s write $y = f(x)$.
- **Condition 1:** $A$ on input $y$ must run in time $\text{poly}(|y|)$. 
- **Condition 2:** $A$ cannot output $x'$ s.t. $f(x') = y$.
- What if $|y|$ is much smaller than $n = |x|$?
  $\implies A$ cannot write the inverse even if it can find it!

Example: $f(x) = \text{first log } |x| \text{ bits of } x$.

It is trivial to invert: $f^{-1}(y) = y\underbrace{00\ldots0}_{n-\lg n}$ where $n = 2|y|$.

But it satisfies our Attempt 1 definition!
  - $f$ is easy to compute.
  - $A$ cannot invert in time $\text{poly}(|y|)$.
    It needs $2|y|$ steps just to write the answer!
Fixing the definition

- Give $A$ a long enough input.
- If $y$ is too short, pad it with 1s in the beginning.
- We adopt the convention to *always* pad it and write: $A(1^n, y)$.
- Now $A$ has enough time to write the answer.
A function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ is a one-way function (OWF) if it satisfies the following two conditions:

- **Easy to compute:** there is a PPT algorithm $C$ s.t. $\forall x \in \{0, 1\}^*$, 
  \[ \Pr[C(x) = f(x)] = 1. \]

- **Hard to invert:** there exists a negligible function $\mu : \mathbb{N} \rightarrow \mathbb{R}$ s.t. for every non-uniform PPT adversary $A$ and $\forall n \in \mathbb{N}$:
  \[ \Pr \left[ x \leftarrow \{0, 1\}^n, x' \leftarrow A(1^n, f(x)) : f(x') = f(x) \right] \leq \mu(n). \]

This definition is also called **strong** one-way functions.
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) $P \neq NP$.

- 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 $f_\times : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$:

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

- 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 $\Pi_n$ be the set of all \textbf{prime} numbers $< 2^n$.
- Choose numbers $p$ and $q$ randomly from $\Pi_n$ and multiply.
- This is unlikely to have small trivial factors.

**Assumption (Factoring Assumption)**

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

$$\Pr \left[ p \leftarrow \Pi_n; q \leftarrow \Pi_n; N = pq : A(N) \in \{p, q\} \right] \leq \nu(n).$$
Factoring assumption is a well established conjecture.

Studied for a long time, with no “good” attack.

Best known algorithms for breaking Factoring Assumption:

\[ 2^{O\left(\sqrt{n \log n}\right)} \] (provable)

\[ 2^{O\left(\frac{3}{n \log^2 n}\right)} \] (heuristic)

Can we construct OWFs from the Factoring Assumption?
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 $\mathcal{A}$ could invert. Call it the GOOD case.

If GOOD case occurs with probability $> \varepsilon$,

$\Rightarrow$ every $\mathcal{A}$ must fail to invert $f_\times$ with probability at least $\varepsilon$.

Now suppose that $\varepsilon$ is a noticeable function

$\Rightarrow$ every $\mathcal{A}$ must fail to invert $f_\times$ with noticeable probability.

This is already useful!

Usually called a weak OWF.
A function \( f : \{0, 1\}^* \rightarrow \{0, 1\}^* \) is a **weak one-way function** if it satisfies the following two conditions:

- **Easy to compute:** there is a PPT algorithm \( C \) s.t. \( \forall x \in \{0, 1\}^* \),
  \[
  \Pr[C(x) = f(x)] = 1.
  \]

- **Somewhat hard to invert:** there is a noticeable function \( \varepsilon : \mathbb{N} \rightarrow \mathbb{R} \) s.t. for every non-uniform PPT \( A \) and \( \forall n \in \mathbb{N} \):
  \[
  \Pr[x \leftarrow \{0, 1\}^n, x' \leftarrow A(1^n, f(x)) : f(x') \neq f(x)] \geq \varepsilon(n).
  \]

Noticeable means \( \exists c \) s.t. for infinitely many \( n \in \mathbb{N} \), \( \varepsilon(n) \geq \frac{1}{n^c} \).
Can we prove that $f_x$ 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 $f_x$ is a weak OWF.

**Theorem**

*Assuming the factoring assumption, function $f_x$ 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 $\frac{1}{2n}$
What about normal OWFs?

- Can we construct normal (a.k.a, strong) OWFs from the Factoring Assumption?
- Even better: Can we construction strong OWFs from ANY weak OWF?
- Yes! Yao’s theorem.
Weak to Strong OWFs

**Theorem (Yao)**

*Strong OWFs exist if and only weak OWFs exist.*

- This is called **hardness amplification**: convert a somewhat hard problem into a really hard problem.
- Hint: use many samples of the weak OWF as the output of the strong OWF.
- Proof by reduction: if $A$ can break your strong OWF, you can come up with an algorithm $B$ for breaking weak OWFs.