Introduction to Cryptography 1/23/18

Lecture 3: One Way Functions - I

Instructor: Vipul Goyal *Scribe: Alexander Litzenberger*

1 Review

1.1 Recall Shannon's Theorem

Theorem 1 *(Shannon's theorem)*

It is impossible to have a perfectly correct symmetric key scheme if $|\mathcal{K}|$ (the number of possible keys) is less than $|\mathcal{M}|$ (the number of possible messages).

1.2 Algorithms and Running Time

Definition 1 *(Probabilistic Turing machine)*

A probabilistic Turing machine is a non-deterministic Turing machine which chooses between the available transitions at each point according to some probability distribution. Or equivalently it is a deterministic Turing machine with an added tape full of random bits called the random tape.

Definition 2 *(Algorithm)*

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

Definition 3 *(Running time)*

An algorithm A is said to run in time $T(n)$ if for all $x \epsilon \{0,1\}^n$, $A(x)$ 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 4 *(Randomized algorithm)*

A randomized algorithm is a probabilistic Turing machine where each bit of the randomness tape is uniformly and independently chosen.

The runtime of a randomized algorithm may depend on random tape that is selected. The output of a randomized algorithm is a distribution.

1.3 Complexity Classes

Definition 5 *(Formal Language)*

A formal language L over an alphabet Σ is a subset of Σ^*; i.e., a subset of words of any length over the alphabet.

A language L is decided by a Turing machine M if M(I) accepts iff $I \epsilon L$.

Henceforth Formal languages will be refereed to simply as languages.

Definition 6 *(P: Polynomial Time)*

A Turing machine M is polynomial time if $\forall$ inputs I of length n for sufficiently large n there exist c_1, c_2 such that M(I) halts in less than $c_1 \cdot n^{c_2}$ steps.
A language L is in P if there exists a deterministic Turing machine M that decides L in polynomial time.

Definition 7 *(NP: Non-deterministic Polynomial Time)*

NP is set the set of decision problems that can be solved by a non-deterministic Turing machine in polynomial time. Equivalently NP is the set of problems for which a certificate can be created which allows a deterministic Turing machine to verify a solution to the problem in polynomial time. In general those problems which are NP-complete (Easier than another problem in NP by at most a polynomial factor) are conjectured to have exponential running time for deterministic Turing machines.

Definition 8 *(BPP: Bounded-error Polynomial Time)*

BPP is the set of languages that can be solved by a probabilistic Turing machine with a probability of greater than or equal to $\frac{2}{3}$ of accepting an input in the language and a probability of greater than or equal to $\frac{2}{3}$ of rejecting an input not in the language.

2 One Way Functions

2.1 Adversaries

A cryptosystem is perfectly secure if an adversary with infinite computational power cannot break it. The One-time pad is perfectly secure however it has the major drawback of requiring a key space that is the same size of the message space. In practice, adversaries have bounded computational resources allowing modern cryptography to deal with schemes which are good for practical purposes by making compromises on perfect security where it is hard for a bounded adversary. This allows the necessary key length to be less than the message length. Due to the Church-Turing thesis we know that any feasible computation can be modeled by a Turing machine and so that is how our adversary will be modeled.

Definition 9 *(Adversary)*

Let $y = f(x)$. The adversary A, performing inversion, gets $f(x)$. So the input length to A is $|f(x)| = |y|$. Let the following conditions hold good.

Condition 1 A must run in time polynomial in the length of the input.

Condition 2 A cannot output x' such that $f(x') = y$.

2.2 Probabilistic Polynomial Time

Definition 10 *(Probabilistic polynomial time)*

An algorithm $A(\cdot)$ is said to be a probabilistic polynomial time Turing machine (p.p.t.) if it is a probabilistic Turing machine and $\forall$Input x with length $\ell(x)$, $\exists$ polynomial $p(\cdot)$ such that the maximum runtime of $A(x)$ is $p(\ell(x))$ the running time of A(x) is $p(\ell(x))$ where p=polynomial.

Definition 11 *(Non-uniform probabilistic polynomial time)*

A non-uniform p.p.t. A is made up of a sequence of probabilistic Turing machines $A_1, A_2, ..., A_k$ where there exists polynomial $p(\cdot)$ such that $\forall A_i \epsilon A$, input x with length $\ell(x)$, the maximum runtime of $A_i(x) : A_i \epsilon A$ is $p(\ell(x))$

Definition 12 *(Negligible function)*

A negligible function $V(\cdot)$ is such that $V(x) < \frac{1}{p(x)}$ for every $p(*)$ and large enough x.

Examples:

2^{-x}: *suppose $\exists c$ such that $2^{-x} \geq x^{-c}$ let $x > \lg(x^{c+1}) \implies 2^{-x} < 2^{-\lg(x^{2c})} = x^{-2c} < x^{-c}$ contradiction.*

$x^{-\lg x}$: *suppose $\exists c$ such that $x^{-\lg x} \geq x^{-c}$ let $x > 2^{c+1} \implies x^{-\lg x} < x^{-2c} < x^{-c}$ contradiction.*

$x^{-\lg\lg\lg x}$: *suppose $\exists c$ such that $x^{-\lg x} \geq x^{-c}$: let $x > 2^{2^{2^{c+1}}} \implies x^{-\lg x} < x^{-2c} < x^{-c}$ contradiction.*

Not Examples:

x^{-c} *let $p = x^{2c}$ let $x > 1 \implies x^{-c} > x^{-2c}$ which is a contradiction.*

2.3 One-way functions

One-way functions are the most basic primitive abstractions in cryptography. Informally the two criteria for one way functions are that it is easy to compute the output given the input and hard to compute an input that would produce that output given the output. One-way functions find applications in creating pseudo-random generators, digital signature schemes, bit-commitment schemes, and message authentication codes.

Definition 13 *(Strong One-way functions)*

$$f : \{0,1\}^n \to \{0,1\}^m$$

1) *Easy to compute: $\exists$ p.p.t. algorithm A such that $f(x) = A(x)$, $\forall$ x, $\forall$ coins of A $Pr_a[A(x) = f(x)] = 1$ w.l.g. infact A runs in poly-time.*

2) *Hard to invert* $\forall$ *Non-Deterministic p.p.t. algorithms B,* $\exists$ *some negilible function* V *such that*

$$Pr_B[B \text{ given f(x) outputs x' s.t.} f(x') = f(x)] \leq V(n)$$

I.E. A Negligible chance of finding a valid pre-image

Definition 14 *(Weak one-way functions)*

1) *Easy to compute:* $\exists$ *p.p.t. algorithm A such that* $f(x) = A(x)$, $\forall x, \forall$ *coins of A* $Pr_a[A(x) = f(x)] = 1$ *w.l.g. infact A runs in poly-time. Just as in strong one-way functions.*

2) $\exists$ *Polynomial* $q(\cdot)$ *such that* $\forall$ *p.p.t. B.*

$$Pr_{\text{coins of B, choice of x}}[B \text{ given } f(x) \text{ fails to output x' s.t.} f(x) = f(x')] \geq \frac{1}{q(n)}$$

$$Pr[B \text{ succeeding}] \leq \frac{1}{q(n)}$$

Definition 15 *(One-to-one one-way functions)*

One-to-One One-way functions map each input to a unique output.

$$f(x) = f(x') \implies x = x'$$
$$x \neq x' \implies f(x) \neq f(x')$$

One-to-One One-way functions can also be characterized as

$$f(x) = f(x') \iff x = x'$$

Definition 16 *(One way permutations)*

Bijective One-to-One One-way functions i.e. every element of the range has an input which produces it.

1) *Is also a One-to-one OWF*

2) *Domain and range are the same*

One-way permutations can also be characterized as

$$f : \mathcal{A} \to \mathcal{B} \text{ such that } f(x) = f(x') \iff x = x' \land \mathcal{A} = \mathcal{B}$$

2.4 Problems with One-Way Functions:

1) Given any OWF f, construct OWF f' s.t. f' leaks at least half of the input.

$$f'(x_1||x_2) = x_1||f(x_2)$$

Exercise: Prove f' is a strong OWF if f is a strong OWF.

2) If P=NP, OWFs can't exist. → Inventing OWF is in NP, given y & x easy to compute it.

Exercise: Prove the above.

If P=BPP, OWF can't exist.