CSL 759 – Cryptography and Network Security Jan 07, 2014

Lecture 2

Instructor: Shweta Agrawal *Scribe: Nihal Srivastava*

We begin with a discussion on what building blocks are required for cryptography, and introduce one way functions, one way permutations and trapdoor permutations as simple cryptographic objects. We discuss the need to make the weakest possible assumptions in order to do useful things in cryptography. We discuss how notions such as *easy* for the legitimate player and *hard* for the attacker are formalized mathematically. We provide the definitions for one way functions, one way permutations and trapdoor permutations and note that none of these objects (also known as cryptographic *primitives*) exist unconditionally, since this would imply that P is not equal to NP. We look at some candidate instantiations for these.

1 Formalizing Easy and Hard

In cryptography, a fundamental asymmetry is required between what the honest players can learn versus what the adversary can learn. We require that the honest players be able to learn information – such as the message from the ciphertext, easily, while the attacker's chances of successfully learning the message are suitably small. The time taken to compute something, or the probability of success are mathematical functions of the problem size, also known as the *security parameter*. Intuitively, the security parameter captures the difficulty of the attacker in breaking the system. We will ask that the probability that a computationally bounded attacker learns anything useful be a vanishing (to be made precise in a moment) function of the security parameter, so that as we set the security parameter to be larger and larger, this probability decreases very quickly.

First we define what we mean by a computationally bounded adversary. We will assume that the attacker Eve is computationally bounded. We will also restrict the legitimate players (usually referred to as Alice and Bob) to be computationally bounded so as to not give them an unfair advantage.

Definition 1 (Probabilistic Polynomial Time (PPT)) *A randomized algorithm A is called probabilistic polynomial time, or PPT if upon input of size k bits, it runs in $O(k^c)$, where c is some constant.*

The notion of *easy* is captured by saying that the function can be computed by a PPT algorithm. We also refer to such algorithms as *efficient*.

We say an attacker is unlikely to win if the chances of its success are a *negligible* function of the security parameter. Intuitively, a function is negligible if it decreases faster than the

inverse of any polynomial. Such functions decay so quickly that we can safely disregard success probabilities that are negligible. Thus, for all practical purposes, negligible events are too unlikely to be considered.

Definition 2 (Negligible Function ($\mathsf{negl}(k)$)**)** *A function $\gamma(k)$ is called negligible if it satifies following*

$$\forall c\ \exists k' : \forall k > k', \gamma(k) \leq \frac{1}{k^c}.$$

Examples of negligible functions $\mathsf{negl}(k)$ are 2^{-k}, $2^{-\sqrt{k}}$, $n^{-\log k}$. These functions approach zero at different rates, but they are all negligible.

Note that it is not a win-win situation to choose the security parameter as high as possible, because the efficiency of the honest players also degrades (gracefully) with larger values of k. Usually, k is chosen small enough so that the system remains efficient but large enough so that it resists the best known attacks (plus some!). Usually, systems are designed with k as a parameter so that it can be set according to the best known attacks, and modified later if required.

2 Cryptographic Primitives

In this section we introduce some basic building blocks of cryptography, namely one way functions, one way permutations and trapdoor permutations.

2.1 One Way Function (OWF)

We say a function is one way if it is easy to compute and hard to invert. Formally,

Definition 3 (One Way Function) *A function $f : \{0,1\}^* \to \{0,1\}^*$ is one-way function if it satisfies the following two conditions:*

1. *$\forall x, \exists$ PPT algorithm which computes f(x) correctly.*
2. *$\forall$ PPT algoritm A,*
   $\Pr\left(f(z) = y \mid x \xleftarrow{Rand.} \{0,1\}^k, y = f(x), z \leftarrow A(y, 1^k)\right) \leq \mathsf{negl}(k)$

Here, $z \leftarrow A(y)$ means y was fed to algorithm A and z was output $)\leq \mathsf{negl}(k)$. Also, 1^k is unary representation of k i.e. $\underbrace{11111...1}_{k}$ times, which we use as input instead of k because the size of k in bits is $\log k$.
To build OWFs we will find it convenient to define a collection of OWF as follows.

Definition 4 *A collection of OWF is given by three PPT algorithms as stated follows:*

1. $\mathsf{Gen}(1^k)$: *Takes as input the security parameter and outputs a public key PK which implicitly defines the domain and range of function* f_{PK}.

2. $\mathsf{Sample}(PK)$: *Takes as input the public key and outputs a random element x from domain of* f_{PK}.

3. $\mathsf{Eval}(x, PK)$: *Takes as input an element x from the domain of* f_{PK} *and computes* $f_{PK}(x)$.

Security of a collection of OWFs is defined as follows: $\forall$ PPT algoritms A,

$$\Pr\left(f_{PK}(z) = y \mid PK \leftarrow \mathsf{Gen}(1^k), x \leftarrow \mathsf{Sample}(PK), y = f_{PK}(x), z \leftarrow A(PK, y, 1^k)\right) \leq \mathsf{negl}(k)$$

2.2 One Way Permutation (OWP)

A function is a OWP if it is a OWF and also a permutation.

Definition 5 *A function* $f : \{0,1\}^* \rightarrow \{0,1\}^*$ *is one-way permutation if it satisfies the following conditions:*

1. f *is a One Way Function*

2. f *is also a permutation*

We can define a collection of OWP the same way as a collection of OWF.

2.3 Trapdoor Permutation (TDP)

A function is a trapdoor permutation or TDP if it is a OWP but additionally has a trapdoor, i.e. given some secret information (known as trapdoor) the OWP becomes easy to invert.

Definition 6 *A Trapdoor Permutation is given by four PPT algorithms as stated follows:*

1. $\mathsf{Gen}(1^k)$*, whose output is a public key PK which implicitly defines the domain and range of function* $f_{PK}(x)$ *and a secret key SK which allows inverting* f_{PK} *efficiently.*

2. $\mathsf{Sample}(PK)$*, whose output is a random element x from domain of* f_{PK}.

3. $\mathsf{Eval}(x, PK)$*, that computes* $f_{PK}(x)$.

4. $\mathsf{Invert}(SK, y)$*, whose output is z such that* $f_{PK}(z) = y$

The security of TDP is identical to OWP, and asks that no efficient adversary can invert the function without the trapdoor secret key.