CSA E0 235: Cryptography Oct 2, 2019

Scribe for Lecture 14

Instructor: Arpita Patra *Submitted by: Bhargav Thankey*

1 Introduction

1.1 Recap

Recall that in the last lecture we defined one way functions (OWFs) and one way permutations (OWPs). Informally, a function (permutation) is said to be one way if it can be computed efficiently but is hard to invert. We noted that there is a strong belief in the cryptography community that OWFs and OWPs exist. We also saw a couple of candidate OWFs - these functions relied on the assumption of computational hardness of the subset sum problem and the integer factorization problem.

We also defined Hard-Core predicates for OWFs. Informally, if f is a OWF, then "some" information about x remains hidden even when $f(x)$ is known and hard-core predicates capture this "something" that remains hidden. We saw that given a OWF, it is not easy to come up with a hard-core predicate for that function.

We then saw a proof of a very simple case of the Goldreich-Levin theorem. This theorem states that given a OWF (OWP) one can come up with another OWF (OWP) and its corresponding hard-core predicate. Then, assuming the existence of OWPs and hard-core predicates, we proved that a PRG with one bit expansion exists. As we have seen in previous lectures, this implies that a PRG with polynomial expansion exists which in turn implies that a PRF exists. Hence, Goldreich-Levin theorem implies that if OWPs exist then so do PRGs and PRFs.

1.2 Today's Agenda

In today's class, we will first recall the proof of the simple case of the Goldreich-Levin theorem that we covered in the last class and then build on that proof to prove a slightly more general, but yet a special case of the Goldreich-Levin theorem. Then we will see the definition and applications of hash functions.

2 Partial proof of the Goldreich-Levin theorem

Recall that the Goldreich-Levin theorem states that:

Theorem 1 *Let f be a OWP and define g by $g(x,r) = (f(x), r)$, where $x = x_1x_2...x_n$, $r = r_1r_2...r_n$. Then g is a OWP and the boolean function $hc(x,r) = r_1x_1 \oplus ... \oplus r_nx_n$ is a hard-core predicate for the function g.*

Proving that g is a OWP is simple and is left as an exercise to the reader. In the last lecture we proved the following lemma:

Lemma 2 *If there exists a PPT adversary $\mathcal{A}$ such that for all $n \in \mathbb{N}$ and for all $x, r \in \{0,1\}^n$, $\Pr[\mathcal{A}(f(x), r) = hc(x, r)] = 1$, then there exists a PPT adversary $\mathcal{A}'$ such that for all n and $x \in \{0,1\}^n$, $\Pr[\mathcal{A}'(f(x)) = x)] = 1$.*

The lemma states that if there exists a PPT adversary that can always break the hard-core predicate hc, then f is not a OWP. Hence if f is a OWP, then no such PPT adversary can exist for hc. Note that this is a far cry from proving that there does not exist a PPT adversary $\mathcal{A}$ such that $\Pr_{x,r \in_R \{0,1\}^n}[\mathcal{A}(f(x), r) = hc(x, r)] \geq \frac{1}{2} + \frac{1}{p(n)}$.

Proof Let $r_i \in \{0,1\}^n$ be a string whose i^{th} bit is 1 and all other bits are 0. Consider an adversary $\mathcal{A}'$ that on input $f(x)$ queries $\mathcal{A}$ for $(f(x), r_1), ..., (f(x), r_n)$. As for all $x, r \in \{0,1\}^n$, $\Pr[\mathcal{A}(f(x), r) = hc(x, r)] = 1$, $\mathcal{A}(f(x), r_i) = (x_1 \cdot 0) \oplus ... \oplus (x_{i-1} \cdot 0) \oplus (x_i \cdot 1) \oplus (x_{i+1} \cdot 0) \oplus ... \oplus (x_n \cdot 0) = x_i$. Thus, $\mathcal{A}'$ gets $x_1 ... x_n$ which it outputs and hence $\Pr[\mathcal{A}'(f(x)) = x)] = 1$. As $\mathcal{A}'$ makes n queries to $\mathcal{A}$ and as $\mathcal{A}$ is a PPT adversary, so is $\mathcal{A}'$. ■

We now focus our attention on proving the following theorem:

Theorem 3 *If there exists a PPT adversary $\mathcal{A}$ and a polynomial p such that for infinitely many $n \in \mathbb{N}$, $\Pr_{x,r \in_R \{0,1\}^n}[\mathcal{A}(f(x), r) = hc(x, r)] \geq \frac{3}{4} + \frac{1}{p(n)}$, then there exists a PPT adversary $\mathcal{A}'$ such that for infinitely many n, $\Pr_{x \in_R \{0,1\}^n}[\mathcal{A}'(f(x)) = x)] \geq \frac{1}{4p(n)}$.*

Note that this theorem is stronger than lemma 2 as we are no longer assuming that $\mathcal{A}$ always succeeds in breaking hc. However, it is weaker than Theorem 1.

Let us first see whether we can use the proof of lemma 2 to prove this theorem. We immediately run into the following problems:

1. The adversary $\mathcal{A}$ does not always succeed in breaking hc. Hence, it is possible that for some n and x, it always fails for r_1. In this case, $\mathcal{A}'$ won't be able to retrieve x_1.
2. The only guarantee we have is that $\mathcal{A}$ can break hc (with a good probability) when queried on random r. However, in the proof of lemma 1, the choices of r are not random.

To prove theorem 2, we fix these problems.

Consider an adversary $\mathcal{A}'$ which on input $f(x)$ queries $\mathcal{A}$ for $(f(x), r)$ and $(f(x), r')$, where $r' = \overline{r_1} ... r_n$ and r (and thus r') is a string of length n picked uniformly at random. Hence, $\mathcal{A}$ returns the correct hc for each of the query with probability at least $\frac{3}{4} + \frac{1}{p(n)}$. If both hc are correct, then $\mathcal{A}'$ can obtain x_1 by computing an Ex-OR of the two hc and can repeat the same process for all $2 \leq i \leq n$ to retrieve $x_2, ..., x_n$. However, $\mathcal{A}$ may not return the correct hc for both queries for some $1 \leq i \leq n$. We overcome this problem by proving the following lemmas:

Lemma 4 *Let n be such that $\Pr_{x,r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) = hc(x,r)] \geq \frac{3}{4} + \frac{1}{p(n)}$ and let e_i be the string whose i^{th} bit is 1 and all other bits are 0. Then there exists $S \subseteq \{0,1\}^n$ of size $\frac{2^n}{2p(n)}$ such that for every $x \in S$ and every $1 \leq i \leq n$, $\Pr_{r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) = hc(x,r) \wedge \mathcal{A}(f(x), r \oplus e_i) = hc(x, r \oplus e_i)] \geq \frac{1}{2} + \frac{1}{p(n)}$.*

Lemma 5 *If there exists $S \subseteq \{0,1\}^n$ of size $\frac{2^n}{2p(n)}$ such that for every $x \in S$ and every $1 \leq i \leq n$, $\Pr_{r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) = hc(x,r) \wedge \mathcal{A}(f(x), r \oplus e_i) = hc(x, r \oplus e_i)] \geq \frac{1}{2} + \frac{1}{p(n)}$, then we can construct a PPT adversary $\mathcal{A}'$ such that $\Pr_{x\in_R\{0,1\}^n}[\mathcal{A}'(f(x)) = x] \geq \frac{1}{4p(n)}$.*

Note that proving these lemmas will prove theorem 3 as the hypothesis of the theorem asserts that for infinitely many n, $\Pr_{x,r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) = hc(x,r)] \geq \frac{3}{4} + \frac{1}{p(n)}$ and hence from lemmas 4 and 5 we will have that there exists a PPT adversary $\mathcal{A}'$ such that for infinitely many n, $\Pr_{x\in_R\{0,1\}^n}[\mathcal{A}'(f(x)) = x] \geq \frac{1}{4p(n)}$. We now prove these lemmas.

2.1 Proof of Lemma 5

The proof is divided into the following lemmas:

Lemma 6 *If there exists $S \subseteq \{0,1\}^n$ of size $\frac{2^n}{2p(n)}$ such that for every $x \in S$ and every $1 \leq i \leq n$, $\Pr_{r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) = hc(x,r) \wedge \mathcal{A}(f(x), r \oplus e_i) = hc(x, r \oplus e_i)] \geq \frac{1}{2} + \frac{1}{p(n)}$, then we can construct a PPT adversary $\mathcal{A}'$ such that $\Pr[\mathcal{A}'(f(x)) = x_i] \geq 1 - \frac{1}{2n}$ for every $x \in S$.*

Lemma 7 *If there exists a PPT adversary $\mathcal{A}'$ such that $\Pr[\mathcal{A}'(f(x)) = x_i] \geq 1 - \frac{1}{2n}$ for every $x \in S$, then we can construct a PPT adversary $\mathcal{A}'$ such that $\Pr[\mathcal{A}'(f(x)) = x] \geq \frac{1}{2}$ for every $x \in S$.*

Lemma 8 *If there exists a PPT adversary $\mathcal{A}'$ such that $\Pr[\mathcal{A}'(f(x)) = x] \geq \frac{1}{2}$ for every $x \in S$, then we can construct a PPT adversary $\mathcal{A}'$ such that $\Pr_{x\in_R\{0,1\}^n}[\mathcal{A}'(f(x)) = x] \geq \frac{1}{4p(n)}$.*

Proof of Lemma 7 From lemma 6 we have,

$$\Pr[\mathcal{A}'(f(x)) \neq x_i] \leq \frac{1}{2n}.$$

Hence by union bound,

$$\Pr[\mathcal{A}'(f(x)) \neq x_1] \vee \Pr[\mathcal{A}'(f(x)) \neq x_2] \vee ... \vee \Pr[\mathcal{A}'(f(x)) \neq x_n] \leq n\left(\frac{1}{2n}\right) = \frac{1}{2}.$$

Thus,

$$\Pr[\mathcal{A}'(f(x)) = x_1] \wedge \Pr[\mathcal{A}'(f(x)) = x_2] \wedge ... \wedge \Pr[\mathcal{A}'(f(x)) = x_n] \geq \frac{1}{2}.$$

■

Proof of Lemma 8

$$\begin{aligned}\Pr_{x\in_R\{0,1\}^n}[\mathcal{A}'(f(x))=x] &= \Pr_{x\in_R\{0,1\}^n}[\mathcal{A}'(f(x))=x|x\in S]\Pr_{x\in_R\{0,1\}^n}[x\in S]\\ &+\Pr_{x\in_R\{0,1\}^n}[\mathcal{A}'(f(x))=x|x\notin S]\Pr_{x\in_R\{0,1\}^n}[x\notin S]\\ &\geq \Pr_{x\in_R\{0,1\}^n}[\mathcal{A}'(f(x))=x|x\in S]\Pr_{x\in_R\{0,1\}^n}[x\in S]\\ &\geq \frac{1}{2}\cdot\frac{1}{2p(n)}\\ &= \frac{1}{4p(n)}\end{aligned}$$

■

To prove lemma 6, we will make use of the Chernoff bound.

Fact 9 *Fix $\varepsilon > 0$ and $b \in \{0,1\}$, and let $\{X_i\}_{1\leq i\leq m}$ be independent $0/1$-random variables with $\Pr[X_i = b] = \frac{1}{2}+\varepsilon$ for all i. The probability that their majority value is not b is at most $e^{-\varepsilon^2 m/2}$.*

Proof of Lemma 6 To prove this lemma we will use the probability boosting (probability amplification) technique. In this technique, one takes a PPT algorithm whose output is either 0 or 1 and which succeeds with probability $\frac{1}{2}+\frac{1}{p(n)}$ for some polynomial p, *independently* executes it polynomially many times and takes the majority output. Then one argues using concentration inequalities like Chernoff bounds that the majority output is correct with probability at least $1-\frac{1}{q(n)}$ for some polynomial q.

From the hypothesis, we have that there exists $S \subseteq \{0,1\}^n$ of size $\frac{2^n}{2p(n)}$ such that for every $x \in S$ and every $1 \leq i \leq n$, $\Pr_{r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) = hc(x,r) \wedge \mathcal{A}(f(x), r\oplus e_i) = hc(x, r\oplus e_i)] \geq \frac{1}{2}+\frac{1}{p(n)}$. Consider a PPT adversary $\mathcal{A}'$ which on input $f(x)$, queries $\mathcal{A}$ for $(f(x),r)$ and $(f(x),r')$, where $r' = \overline{r_1}...r_n$, and r is a string of length n picked uniformly at random. It *independently* repeats this process m times for some appropriately chosen parameter m and finally takes the majority output. Thus, we have that for any one of the repetitions,

$$\Pr[\mathcal{A}'(f(x))=x_i] = \Pr_{r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r)=hc(x,r)\wedge\mathcal{A}(f(x),r\oplus e_i)=hc(x,r\oplus e_i)] \geq \frac{1}{2}+\frac{1}{p(n)}$$

and hence, using Chernoff bound,

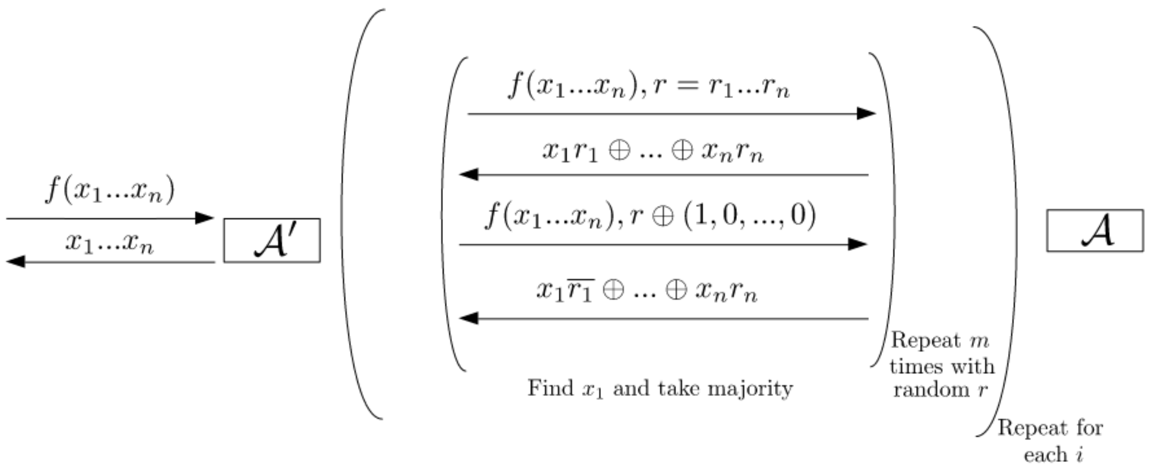
$$\Pr[\mathcal{A}'(f(x))\neq x_i] \leq e^{-m/2(p(n))^2}.$$

For $m = 2(\ln n+\ln 2)(p(n))^2$, $e^{-m/2(p(n))^2} = \frac{1}{2n}$ and hence,

$$\Pr[\mathcal{A}'(f(x))\neq x_i] \leq \frac{1}{2n}.$$

As m is polynomial in n and as $\mathcal{A}$ is a PPT algorithm, so is $\mathcal{A}'$. ■

The adversary $\mathcal{A}'$ is summarized in the following figure.



$f(x_1...x_n)$
$x_1...x_n$
$\mathcal{A}'$
$f(x_1...x_n), r = r_1...r_n$
$x_1r_1 \oplus ... \oplus x_nr_n$
$f(x_1...x_n), r \oplus (1,0,...,0)$
$x_1\overline{r_1} \oplus ... \oplus x_nr_n$
Find x_1 and take majority
Repeat m times with random r
Repeat for each i
$\mathcal{A}$

Figure 1: Adversary $\mathcal{A}'$

2.2 Proof of Lemma 4

In order to prove the lemma, we will prove the following two lemmas:

Lemma 10 *Let n be such that* $\Pr_{x,r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) = hc(x,r)] \geq \frac{3}{4} + \frac{1}{p(n)}$. *Then there exists $S \subseteq \{0,1\}^n$ of size $\frac{2^n}{2p(n)}$ such that for every $x \in S$,* $\Pr_{r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) = hc(x,r)] \geq \frac{3}{4} + \frac{1}{2p(n)}$.

Lemma 11 *If there exists $S \subseteq \{0,1\}^n$ of size $\frac{2^n}{2p(n)}$ such that for every $x \in S$,* $\Pr_{r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) = hc(x,r)] \geq \frac{1}{2}+\frac{1}{2p(n)}$, *then there exists $S \subseteq \{0,1\}^n$ of size $\frac{2^n}{2p(n)}$ such that for every $x \in S$ and every $1 \leq i \leq n$,* $\Pr_{r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) = hc(x,r) \wedge \mathcal{A}(f(x),r\oplus e_i) = hc(x,r\oplus e_i)] \geq \frac{1}{2}+\frac{1}{p(n)}$.

Proof of Lemma 11 From lemma 10, we have that, for any $x \in S$, $\Pr_{r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) \neq hc(x,r)] \leq \frac{1}{4} - \frac{1}{2p(n)}$ and $\Pr_{r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r\oplus e_i) \neq hc(x,r)] \leq \frac{1}{4} - \frac{1}{2p(n)}$. Hence, by union bound

$$\begin{aligned}\Pr_{r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) \neq hc(x,r) \vee \mathcal{A}(f(x),r\oplus e_i) \neq hc(x,r\oplus e_i)] &\leq \left(\frac{1}{4} - \frac{1}{2p(n)}\right) + \left(\frac{1}{4} - \frac{1}{2p(n)}\right)\\ &= \frac{1}{2} - \frac{1}{p(n)}\end{aligned} \tag{1}$$

and hence for any $x \in S$,

$$\Pr_{r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) = hc(x,r) \wedge \mathcal{A}(f(x),r\oplus e_i) = hc(x,r\oplus e_i)] \geq \frac{1}{2} + \frac{1}{p(n)}$$

■

Note that in the proof of lemma 11 we crucially used the fact that $\Pr_{x,r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) = hc(x,r)] \geq \frac{3}{4}+\frac{1}{p(n)}$. For if we had $\Pr_{x,r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) = hc(x,r)] \geq \frac{1}{2}+\frac{1}{p(n)}$, then equation (1) would have been

$$\begin{aligned}\Pr_{r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) \neq hc(x,r) \vee \mathcal{A}(f(x),r\oplus e_i) \neq hc(x,r\oplus e_i)] &\leq \left(\frac{1}{2}-\frac{1}{2p(n)}\right)+\left(\frac{1}{2}-\frac{1}{2p(n)}\right)\\ &= 1-\frac{1}{p(n)}\end{aligned}$$

which is not useful.

Proof of lemma 10 Let S be the set of all stings $x \in \{0,1\}^n$ such that $\Pr_{r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) = hc(x,r)] \geq \frac{3}{4}+\frac{1}{2p(n)}$. Then, we have that,

$$\begin{aligned}\frac{3}{4}+\frac{1}{p(n)} &\leq \Pr_{x,r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) = hc(x,r)]\\ &= \sum_{x'\in\{0,1\}^n} \Pr_{r\in_R\{0,1\}^n}[\mathcal{A}(f(x'),r) = hc(x',r) \wedge x = x']\\ &= \frac{1}{2^n}\sum_{x\in\{0,1\}^n} \Pr_{r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) = hc(x,r)]\\ &= \frac{1}{2^n}\sum_{x\in S} \Pr_{r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) = hc(x,r)] + \frac{1}{2^n}\sum_{x\notin S} \Pr_{r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) = hc(x,r)]\\ &\leq \frac{|S|}{2^n}+\frac{1}{2^n}\sum_{x\notin S}\left(\frac{3}{4}+\frac{1}{2p(n)}\right)\\ &\leq \frac{|S|}{2^n}+\left(\frac{3}{4}+\frac{1}{2p(n)}\right),\end{aligned}$$

where the second to last inequality follows from $\Pr_{r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) = hc(x,r)] \leq 1$ for all $x \in S$ and $\Pr_{r\in_R\{0,1\}^n}[\mathcal{A}(f(x),r) = hc(x,r)] \leq \frac{3}{4}+\frac{1}{2p(n)}$ for all $x \notin S$, while the last inequality follows from the fact that $|\{x : x \notin S\}| \leq 2^n$. Hence,

$$\frac{3}{4}+\frac{1}{p(n)} \leq \frac{|S|}{2^n}+\left(\frac{3}{4}+\frac{1}{2p(n)}\right)$$

which yields

$$\frac{2^n}{2p(n)} \leq |S|.$$

■

3 Hash Functions

In this section we will briefly (and informally) describe hash functions and their applications. Informally, a hash function is a many-to-one function mapping arbitrary length bit strings to fixed length bit strings. Generally the size of the domain of the function is very large compared to the size of the codomain. Hence collisions are bound to exist i.e. if h is a hash function, then $\exists x_1 \neq x_2$ such that $h(x_1) = h(x_2)$. We say that a hash function is a good cryptographic hash function if it is collision resistant i.e. it is infeasible to find collisions. We now list a few applications of hash functions:

1. Hash functions are used for domain extension of MACs.
2. A message digest of a file can be computed by applying a hash function to the contents of the file. If the hash function used is a good cryptographic hash function, then the probability of two distinct files having the same message digest is negligible. Hence, message digests can be used as unique identifier for files. This idea finds many applications, some of which are mentioned below.
3. File Integrity Check: If the hash of a downloaded file does not match the provided hash, then we know that the file has been corrupted in transmission.
4. Virus Fingerprinting: Virus scanners have to store information about a large number of known viruses. Storing this information in its raw form will use up a lot of space. So instead the virus scanners store the hashes of known viruses. When an email attachment or an application is downloaded from the web, its hash is compared with the hashes stored in the database of the virus scanner. If a match is found, then the attachment or application contains harmful code with a very high probability.
5. De-duplication: Consider a cloud storage that is shared by multiple users. These users may upload the same file to the server. Since all users can be served from a single copy of the file, storing multiple copies is a waste of precious space. Hence, the cloud servers compare the message digests of uploaded files to find and delete duplicates.
6. Password hashing: Instead of storing users' passwords in the open, computer systems store the hashes of passwords. When a user enters the password, its hash is computed and compared with the stored hash.

3.1 Collision Resistance Security

Let $\Pi = (\text{Gen}, h)$ be a hashing scheme and let n be the security parameter. Then, the collision resistance security experiment Hash-CR$_{\mathcal{A},\Pi}(n)$ is defined as follows.

$\mathbf{Hash - CR_{\mathcal{A},\Pi}(n)}$:

1. The challenger $\mathcal{C}$ picks a key k using the Gen algorithm and sends it to the adversary $\mathcal{A}$.
2. $\mathcal{A}$ sends a tuple of strings (x, x') to the challenger.

The output of the game is 1 and the adversary succeeds if $h(x) = h(x')$ and the output of the game is 0 and the adversary fails if $h(x) \neq h(x')$. Π is said to be a collision resistant hashing scheme if for every PPT adversary $\mathcal{A}$, there exists a negligible function *negl* such that

$$\Pr[\mathrm{Hash} - \mathrm{CR}_{\mathcal{A},\Pi}(n) = 1] \leq negl(n).$$

3.2 Merkle Trees

Now we will describe an application of hash functions to cloud data storage. Consider a situation where a client has a large number of files which they want to store on a cloud server. When they download the saved file from the server, they want to be able to verify that the file they got was correct. We will first describe a few candidate solutions and their drawbacks and then discuss a good solution. Throughout the discussion, we will assume that the client has files $X_1, ..., X_t$ where $t = 2^k$ for some $k \in \mathbb{N}$.

1. The client can store the hash of all files locally. Then, when they download a file, they can compare the hash of the downloaded file with the corresponding locally stored hash. If the two hashes are not the same, then the file has been corrupted. However, the drawback of this solution is that the client has to store the hashes of all the files and hence the local space used grows linearly with t. If t is large, then this approach will become infeasible.

2. The client can locally store a single hash $y = h(X_1, ..., X_t)$ i.e. a hash of all the files. Then, whenever they need a file X_i, they can download all the files, compute the hash and compare it with the locally stored hash. However, in this case, to verify the contents of a single file, the client has to download all the files and this will require a considerable amount of bandwidth if the number of files is large.

The solution to this problem is to use a Merkle tree. It is a trade-off between the above mentioned extreme approaches. This approach is an alternative to the Merkle-Damgard construction used for constructing arbitrary length hash functions. We now describe this approach.

Let H be a collision resistant hash function. Then given inputs $X_1, ..., X_t$, the Merkle tree is constructed by recursively dividing the files into two groups of equal size and then compute a hash of each of these groups. Figure 2 shows the construction of a Merkle tree when $t = 8$.

When the client wants to upload the files to the cloud storage, they first construct a Merkle tree and locally store the hash $h_{1,...,t}$. In our example as $t = 8$, the client locally stores the hash $h_{1,...,8}$. Upon receiving the files, the cloud storage also computes the Merkle tree and saves it along with the files. When the client wants to retrieve a file - say X_1, the server sends the file along with a proof of correct transmission. In our case, this proof is $X_1, H(X_3, X_4)$ and $h_{5,...,8}$. In general, the proof consists of values at the nodes in the Merkle tree adjacent to the path from the file (in our case X_1) to the root. Using this proof and X_1, the client computes $h_{1,...,8}$ and compares it with the locally stored value of $h_{1,...,8}$. If the

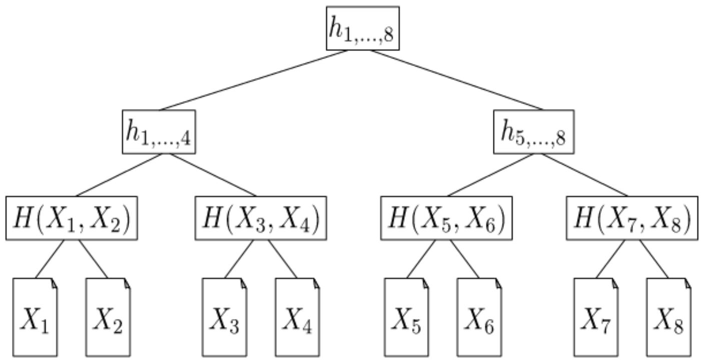


Figure 2: Merkle Tree

two values match, then since the hash function H is collision resistant, the client concludes that the server hasn't tempered with the files and the file obtained is correct. Note that in this approach, the local space used is constant and the communication between the client and server is of order $\log t$.

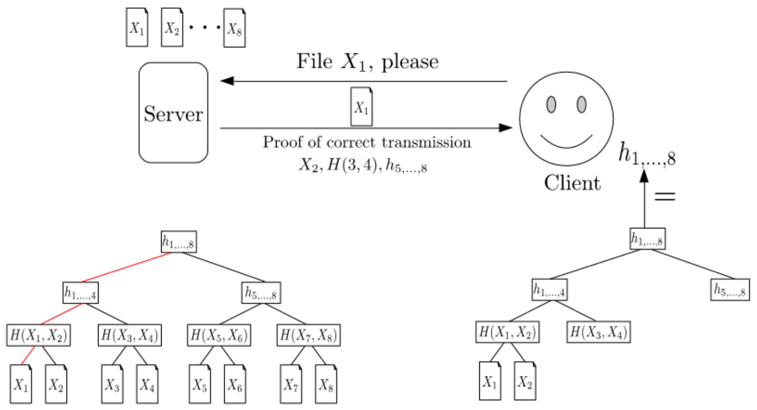

X_1 X_2 $\cdots$ X_8
Server
File X_1, please
X_1
Proof of correct transmission
$X_2, H(3,4), h_{5,\ldots,8}$
Client
$h_{1,\ldots,8}$
$=$

Figure 3: File retrieval using Merkle Trees

4 Conclusion

In today's class we saw a partial proof of Goldreich-Levin theorem and the definition and applications of hash functions.

References

[1] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition.* CRC Press, 2014.

[2] Arpita Patra. Csa e0-235: Cryptography (aug-dec 2019) lecture 14. `https://www.csa.iisc.ac.in/~cris/resources/e0_235/ppt/AP_Lecture14.pdf`.