CSE 594 : Modern Cryptography 03/07/2017

Lecture 12: Hash Functions

Instructor: Omkant Pandey *Scribe: FNU Gaurav, Hemant Pandey*

1 Last Class

In the last class we studied about message authenticating code, MAC using pseudo-random functions and the proof of unforgeability of MAC scheme. This lecture discusses about compressing long messages into short ones using hash functions.

2 Introduction

We have discussed earlier how pseudorandom functions can be used to extend the length of random strings while maintaining the computational indistinguishability. Now, what if we want to compress a long string to a short one? The basic idea is to store a part of set or small number of elements coming from a larger set. The goal is to find a deterministic method which quickly stores and accesses element with least amount of collisions. A common example is the concept of hashmap data structures used in programming languages which stores element in form of a $(key, value)$ pair.
The basic approach is to use a hash function such that : $\forall x, y$:

$$\Pr_{h \in \mathcal{H}}[h(x) = h(y)] \leq \frac{1}{|\mathcal{R}|}$$

where $h : \mathcal{D} \to \mathcal{H}$

These are specifically called universal hash functions and $\mathcal{H}$ is a family of universal hash functions. The applications of hash functions are not limited to cryptography, but have proven to be efficient in designing and optimizing data structures, randomness extraction and many more domains. However, these universal hash functions are not always good for crytpographic purposes since it is easy to find collisions in universal hash functions as can be seen below mathematically,

$$h_{a,b,p}(x) = (ax + b) \ mod \ p$$

$$h(x_1) = (ax_1 + b) = v$$

$$h(x_2) = (ax_2 + b)$$

$$\Rightarrow x_2 = (inv(a) * (v - b)) \ mod \ p$$

Thus, there can be values of a and b for which $x_2 = x_1$.

To be used for cryptography, hash functions should be designed in such a way that they compress efficiently (say to half length) and for a given h, it should be hard to find collisions. A collison is said to occur if for different x and y, $h(x) = h(y)$. The property of hash functions by which it is difficult to find collisions is called 'collision resistance'. We will discuss about collision resistance hashing now.

3 Collision-Resistant Hash Functions

Since a hash function h compresses its input, there certainly exists two inputs that comprise a collision i.e two or more inputs in the domains maps to same value in the range. To get around this issue, a family of hash functions called 'collision resistant hash functions' is introduced.

A set of functions $H = \{h_i : \mathcal{D}_i \rightarrow \mathcal{R}_i\}_{i \in I}$ is a family of collision resistant hash functions (CRHF) if :

- (Easy to sample) There is a PPT algorithm Gen s.t. $Gen(1^n) \in I$
- (Compression) $|\mathcal{R}_i| < |\mathcal{D}_i|$
- (Easy to evaluate) There is a PPT algorithm for evaluation (Eval) s.t. $\forall x \in \mathcal{D}_i$ and $\forall i \in \mathcal{I}$, $Eval(x, i) = h_i(x)$.
- (Collision resistance) $\forall$ non uniform PPT adversary A, there is a negligible function μ such that $\forall n \in \mathbb{N}$:

$$\Pr[i \leftarrow Gen(1^n), (x, x') \Leftarrow A(1^n, i) : x \neq x' \bigwedge h_i(x) = h_i(x')] \neq \mu(n).$$

We may note that compression is a relatively weak property and does not even guarantee that the output is compressed by a reasonable amount of bits. Ideally, we want the compressed length to be atleat half of original i.e $|h(x)| < \frac{|x|}{2}$. We may also realise that if h is collision resistant, then h is one way.

If given a case where the length of input string is more than the desired input length for hash function, merkle tree construction method is used. Merkle tree is defined as follow :
The below figure gives a visual idea of merkel tree. Here a, b, c and d are input blocks :

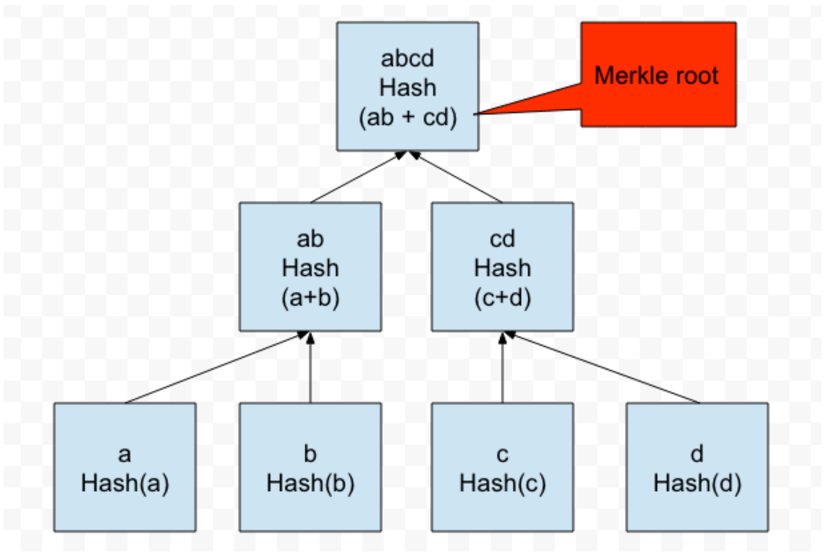

Figure 1: Merkle tree.

- Step 1 : The given string is broken down into blocks of length n i.e write string $x \in \{0,1\}^*$ as $x_1||x_2||x_3||x_4....$ where $|x_1| = |x_2| = |x_3|.... = n$

- Step 2 : The next level is obtained by pairing up nodes of previous level such that $y_1 = h(x_1||x_2, y_2 = h(x_3||x_4))....andsoon$. Note that the length gets halved at each level.
- Step 3 : The above steps are repeated again and again until we get the *root* with the desired length. This *root* denotes the final hash written $h(x)$.

Depth of a merkle tree can be found out by visualising the length of input string in the form of $2^i.2^k$ where $2^i \rightarrow$ desired length of output; $k \rightarrow$ depth of merkle tree. We arrive to this by simply taking $\log(2^k)$ for base 2 (because merkle tree is a binary tree and this is how to calculate the depth).

One major application of collision resistance hash function comes into play when they are used with MAC. CRHF's are used for extending the message space of MAC. Suppose we are given a secure MAC $I = (S, V)$ for short messages. Our goal is to build a new secure MAC I' for much longer messages. We do so using a collision resistant hash function: we compute a tag for a long message m by first hashing m to a short digest and then applying I to the digest. Thus, we use CRHF's to compress long messages to short, then apply MAC to authenticate the short hash value.

3.1 Are one way 1-1 injective permutations equivalent to CRHF?

Above stated properties may suggest that such one way permutations can be CRHF but that is not true as permutations have $\mathcal{D} = \mathcal{R}$ i.e. they don't compress. It may appear that they are the same but since they don't compress, the main functionality which led to adoption of hashing, they can't be treated as equivalent.

3.2 General attacks on CRHFs

Collision-resistance is a stronger property than one-wayness, so finding an attack on a collision-resistant hash functions is easier than finding an attack on a one-way function. Some popular attacks are :

- Enumeration attack :
  $|\mathcal{D}_i| = 2^d$: Length of Domain
  $|\mathcal{R}_i| = 2^n$: Length of Range
  x_1 and x_2 are chosen at random. Now, we need to determine the probability of collision between $h(x)$ and $h(x')$.

  In order to analyze this situation, we must count the number of ways that a collision can occur. Let P_y be the probability that h maps a element from the domain $y \in R_i$. The probability of collision at y is therefore P_y^2. Since the collision can occur at either y_1 or y_2, etc, the probability of a collision can be written as :

  $$\Pr[collision] = \sum_{y \in \mathcal{R}_i} P_{y^2}$$

  Since $\sum_{y \in \mathcal{R}_i} P_y = 1$, , by the Cauchy-Schwarz Inequality we have that :

  $$\sum_{y \in \mathcal{R}_i} P_{y^2} > \frac{1}{|\mathcal{R}_i|}$$

  The probability that x and x' are not identical is $\frac{1}{|\mathcal{D}_i|}$. Combining these two shows that the total probability of a collision is greater than $\frac{1}{|\mathcal{R}_i|}$. In other words, enumeration requires searching most of the range to find a collision.

- Birthday attack :
  Instead of enumerating pairs of values, consider a set of random values $x_1, ..., x_t$. Evaluate h on each x_i and look for a collision between any pair x_i and $x_{i'}$. By the linearity of expectations, the expected number of collisions is the number of pairs multiplied by the probability that a random pair collides. This probability is $\approx \frac{t^2}{|R_i|}$.

  As an example, consider the scenario in which a teacher with a class of 30 students asks for everybody's birthday (for simplicity, ignore leap years), to determine whether any two students have the same birthday (corresponding to a hash collision as described further). Intuitively, this chance may seem small. If the teacher picked a specific day (say September 16), then the chance that at least one student was born on that specific day is $1 - (364/365)^{30}$, about 7.9%. However, the probability that at least one student has the same birthday as any other student is around 70% for n = 30, from the formula $1 - 365!/((365 - n)! \cdot 365^n)$.

  Let Q(H) be the expected number of values we have to choose before finding the first collision. This number can be approximated by -

$$Q(H) \approx \sqrt{\frac{\pi}{2}H}.$$

3.3 Miscellaneous facts regarding CRHFs

Recently it has been shown that SHA-1 encryption has been broken successfully. The people concerned are yet to release details of how they achieved the first SHA-1 collision, but has already released a proof of concept that can be used to visualize the collision. It has two specially-crafted PDF files that have identical SHA-1 hashes but different content (the definition of a collision). Given is the link for the paper - http://eprint.iacr.org/2017/190 .

While once very popular, SHA-1 is now far less common, with experts having predicted the day the technology would be cracked quite some time ago. There has been a general move away from this form of encryption in recent years. It is hoped that such a practical attack on SHA-1 will increase awareness and convince the industry to quickly move to safer alternatives, such as SHA-256 or SHA-512 (both belonging to SHA-2 family).