Lecture 21: Cryptography: Hashing

In this lecture, we will be studying some basics of cryptography. Specifically, we will be covering

- Hash functions
- Random oracle model
- Desirable Properties
- Applications to security

1 Hash Functions

A hash function h maps arbitrary strings of data to fixed length output. The function is deterministic and public, but the mapping should look "random". In other words,

$$h : \{0,1\}^* \to \{0,1\}^d$$

for a fixed d. Hash functions do not have a secret key. Since there are no secrets and the function itself is public, anyone can evaluate the function. To list some examples,

Hash function	MD4	MD5	SHA-1	SHA-256	SHA-512
d	128	128	160	256	512

In practice, hash functions are used for "digesting" large data. For example, if you want to check the validity of a large file (potentially much larger than a few megabytes), you can check the hash value of that file with the expected hash. Therefore, it is desirable (especially for cryptographic hash functions covered here) that the function is collision resistant. That is, it should be "hard" to find two inputs m_1 and m_2 for hash function h such that $h(m_1) = h(m_2)$. Most modern hash functions hope to achieve security level of 2^{64} or better, which means that the attacker needs to test more than 2^{64} different inputs to find a collision. Unfortunately, MD4 and MD5 aimed to provide 2^{64} security, but has been shown to be broken using 2^6 and 2^{37} inputs respectively. SHA-1 aimed to provide 2^{80} security, but has been shown (at least theoretically) to be no more than 2^{61} security.

1.1 Random Oracle

The Random Oracle model is an ideal model of the hash function that is not achievable in practice. In this model, we assume there exists an oracle h such that on input $x \in \{0,1\}^*$, if h has not seen x before, then it outputs a random value as $h(x)$. Otherwise, it returns $h(x)$ it previously output. The random oracle gives a random value for all new inputs, and gives deterministic answers to all inputs it has seen before. Unfortunately, a random oracle does not exist since it requires infinite storage, so in practice we use pseudo-random functions.

1.2 Desirable Properties

There are many desirable properties of a hash function.

1. One-way (pre-image resistance): Given $y \in \{0,1\}^d$, it is hard to find an x such that $h(x) = y$.
2. Strong collision-resistance: It is hard to find any pair of inputs x, x' such that $h(x) = h(x')$.
3. Weak collision-resistance (target collision resistance, 2nd pre-image resistance): Given x, it is hard to find x' such that $h(x) = h(x')$.
4. Pseudo-random: The function behaves indistinguishable from a random oracle.
5. Non-malleability: Given $h(x)$, it is hard to generate $h(f(x))$ for any function f.

Some of the properties imply others, and some others do not. For example,

- 2 $\Rightarrow$ 3
- 1 $\not\Rightarrow$ 2, 3.

Furthermore, collision can be found in $O(2^{d/2})$ (using birthday paradox), and inversion can be found in $O(2^d)$.

To give more insight as to why some properties do not imply others, we provide examples here. Consider h that satisfies 1 and 2. We can construct a new h' such that h' takes in one extra bit of input, and XORs the first two bits together to generate an input for h. That is, $h'(a, b, x_2, \ldots, x_n) = h((a \oplus b), x_2, \ldots, x_n)$. h' is still one-way, but is not weak collision resistant. Now consider a different h'' that evaluates to $0||x$ if $|x| \leq n$, and $1||h(x)$ otherwise. h'' is weak collision resistant, but is not one-way.

1.3 Applications

There are many applications of hash functions.

1. Password Storage: We can store hash $h(p)$ for password p instead of p directly, and check $h(p)$ to authenticate a user. If it satisfies the property 1, adversary comprising $h(p)$ will not learn p.

2. File Authenticity: For each file F, we can store $h(F)$ in a secure location. To check authenticity of a file, we can recompute $h(F)$. This requires property 3.

3. Digital Signature: We can use hash functions to generate a signature that guarantees that the message came from a said source. For further explanation, refer to Recitation 11.

4. Commitments: In a secure bidding, Alice wants to bid value x, but does not want to reveal the bid until the auction is over. Alice then computes $h(x)$, and publicize it, which serves as her commitment. When bidding is over, then she can reveal x, and x can be verified using $h(x)$.

   It should be "binding" so that Alice cannot generate a new x that has the same commitment, and it should "hide" x so no ones learns anything before x is revealed. Furthermore, it should be non-malleable. To guarantee secrecy, we need more than the properties from the previous section, as $h'(x) = h(x)||MSB(x)$ actually satisfies 1, 2, and 5. In practice, this problem is bypassed by adding randomness in the commitment, $C(x) = h(r||x)$ for $r \in_R \{0,1\}^{256}$, and reveal both randomness and x at the end.

6.046J / 18.410J Design and Analysis of Algorithms
Spring 2015