Lecture - 33

Friday, 2 November 2016 (2:25 - 3:15)

Cryptographic Hash Functions

In this lecture, we revisited Hashing and discussed its use in cryptographic settings followed by a brief probabilisitic analysis of cryptographic hash functions.

1 Hash Function

A hash function is a function that takes a variable sized input (called *message*) and provides a fixed size output (called *hash digest*).

1.1 Applications of Hashing

1. **Searching:** We know that linear search takes $\mathcal{O}(n)$ time to search an element in an array having n elements, whereas binary search takes $\mathcal{O}(\log n)$ time to search. However, binary serach requires the elements to be sorted, which itself might take $\mathcal{O}(n \log n)$ time (Or at the least $\mathcal{O}(n)$ time in case the elements are uniformly distributed across a range). In such a scenario, hashing provides an efficient approach for quick searching. Although the pre-processing time for hashing is $\mathcal{O}(n)$, it still has an upper hand over other techniques due its $\mathcal{O}(1)$ time complexity for searching.
2. **Bloom Filters:** Bloom filter is a probabilistic data structure based on hashing that is rapid and takes very less memory to tell whether an element is present in a set or not. Assume we have a set $S = \{s_1, s_2, ..., s_m\}$ of m elements. A bloom filter consists of an array of n bits $A[0]$ to $A[n-1]$, initially set to 0 where $n \ll m$. This array A helps in quickly finding whether an element x is a part of array S or not. For that purpose, it uses k hash functions $h_1, h_2, ...h_k$ with range 0,...n-1. We set $A[h_i(s_j)] = 1$ if not 1 already, for all $1 \leq i \leq k$ and $1 \leq j \leq m$. Now, in order to know whether an element x belongs to S or not, we check whether $A[h_i(x)] = 1$ or not, $\forall 1 \leq i \leq k$. If not, clearly, x is not a member of S. However, if all $A[h_i(x)]$ are set to 1, we will assume that x is in S, although we could be wrong, because these values could have been set by the other elements. This shows that bloom filters may yield false positives. It has, however, been observed that this probability is not very high and hence can be covered by the amount of time and space efficiency that it provides.
3. **Accoustic Fingerprinting:** This is yet another application of hashing which uses *locality sensitive hash functions* such that $h(m_1)$ will be closer to $h(m_2)$ if m_1 is closer to m_2.
4. **Cryptographic Hash Functions:** Hash functions play a pivotal role in the field of cryptography. The next section discusses the properties of hash functions suitable for this field.

1.2 Properties of Crypographic hash functions

Apart from the fact that a hash function should generate hash values *Uniformly at random* by taking variable sized input and providing fixed sized output, a cryptographic hash function should have the following properties:

1. **Pre-Image Resistance:**

   This property means that it should be computationally hard to reverse a hash function. In other words, if a hash function h produced a hash value z, then it should be a difficult process to find any input value x that hashes to z. This property protects against an attacker who only has a hash value and is trying to find the input.

2. **Second Pre-Image Resistance:**

   This property means given an input and its hash, it should be hard to find a different input with the same hash. In other words, if a hash function h for an input x produces hash value h(x), then it should be difficult to find any other input value y such that h(y) = h(x). This property of hash function protects against an attacker who has an input value and its hash, and wants to substitute different value as legitimate value in place of original input value.

3. **Collision Resistance:**

   This property means it should be hard to find two different inputs of any length that result in the same hash. This property is also referred to as collision free hash function. In other words, for a hash function h, it is hard to find any two different inputs x and y such that h(x) = h(y). Since, hash function is compressing function with fixed hash length, it is impossible for a hash function not to have collisions. This property of collision free only confirms that these collisions should be hard to find. This property makes it very difficult for an attacker to find two input values with the same hash. Also, if a hash function is collision-resistant then it is second pre-image resistant.

1.3 Public Key Cryptography and Digital Signatures

Public Key Cryptography The most important properties of public key encryption scheme are:

1. Different keys are used for encryption and decryption. This is a property which set this scheme different than symmetric encryption scheme.
2. Each receiver possesses a unique decryption key, generally referred to as his private key.
3. Receiver needs to publish an encryption key, referred to as his public key.
4. The intelligent part of any public-key cryptosystem is in designing a relationship between two keys. Also, if P is the public key and S is the private key, then P(S(m)) = S(P(m)) = m.

Digital Signatures Digital signatures use public key cryptography in which we encrypt the hash of the message instead of the entire message. Using a public key algorithm such as RSA, one can generate two keys that are mathematically linked: one private and one public. To create a digital signature, the sender creates a one-way hash of the message to be signed. The private key is then used to encrypt the hash. The encrypted hash is the digital signature. The reason for encrypting the hash instead of the entire message or document is that a hash function can convert an arbitrary input into a fixed length value, which is usually much shorter and hence saves time.

The value of the hash is unique to the hashed message. Any change in the message, even changing or deleting a single character, results in a different value. This attribute enables others to validate the integrity of the data by using the sender's public key to decrypt the hash. If the decrypted hash matches a second computed hash of the same message, it proves that the message hasn't changed

since it was signed. If the two hashes don't match, the message has either been tampered with in some way (integrity) or the signature was created with a private key that doesn't correspond to the public key presented by the signer (authentication).

1.4 Probabilistic analysis

As already stated that a hash function should have the above stated characteristics, we use probability to try and break one of the above properties:

Assume M is the set of messages to be hashed and D be the set of digests, i.e. hash values which the messages in M map to. Further assume that M_0 is a small subset of M. The idea here is to compute the probability that given a digest d^*, if a person takes a subset of M, i.e. M_0 and follows a brute force method to try and find the message m^* corresponding to the digest d^*, what will be the probability of his success? We observe that,

Probability that the first message out of M_0 gets mapped to one of the D digests: $1/|D|$

Probability that the first message out of M_0 does not get mapped to one of the D digests: $1-1/|D|$

Probability that the none of the messages out of M_0 gets mapped to one of the D digests: $(1-1|/D|)^{|M_0|}$

i.e., probability of failure is : $(1-1/|D|)^{|M_0|}$

Hence, probability of success is : $1-(1-1/|D|)^{|M_0|}$