Engineering Blockchain Applications

CSE 598 : Spring B

Hashing : More reference

Agenda

- Topics to be Discussed
  - Understanding Problems
  - The birthday paradox and the birthday attack
  - Structure of cryptographically secure hash functions
  - SHA series of hash functions

WHEN IS A HASH FUNCTION CRYPTOGRAPHICALLY SECURE?

- A hash function is called cryptographically secure if the following two conditions are satisfied:
  - It is computationally infeasible to find a message that corresponds to a given hashcode. This is sometimes referred to as the one-way property of a hash function.

  - It is computationally infeasible to find two different messages that hash to the same hashcode value. This is also referred to as the strong collision resistance property of a hash function

Second pre-image resistance: Given an input m1 it should be difficult to find another input m2 such that m1 ≠ m2 and hash(m1)=hash(m2). Functions that lack this property are vulnerable to second-preimage attacks.

Collision resistance: It should be difficult to find two different messages m1 and m2 such that hash(m1)=hash(m2). Such a pair is called a cryptographic hash collision.

Preimage resistance: It is the property of a hash function that it is hard to invert, that is, given an element in the range of a hash function, it should be computationally infeasible to find an input that maps to that element.

- Hash functions that are not collision resistant can fall prey to birthday attack. More on that later.

If you use n bits to represent the hashcode, there are only 2^n distinct hashcode values.

Collision resistance refers to two different messages that possess certain basic structure so as to be meaningful not resulting in the same hashcode.

Simple Hash Function

Practically all algorithms for computing the hashcode of a message view the message as a sequence of n-bit blocks. The message is processed one block at a time in an iterative fashion in order to generate its hashcode

WHAT DOES PROBABILITY THEORY HAVE TO SAY ABOUT A RANDOMLY PRODUCED MESSAGE HAVING A PARTICULAR HASH VALUE ?

Assume that we have a random message generator and that we can calculate the hashcode for each message produced by the generator.

 Let's say we are interested in knowing whether any of the messages is going to have its hashcode equal to a particular value h.

 Let's consider a pool of k messages produced randomly by the message generator

- We pose the following question: What is the value of k so that the pool contains at least one message whose hashcode is equal to h with probability 0.5?

Let's say that the hashcode can take on N different but equiprobable values. – Say we pick a message x at random from the pool of messages.

Since all N hashcodes are equiprobable, the probability of message x having its hashcode equal to h is 1/N .

Since the hashcode of message x either equals h or does not equal h, the probability of the latter is (1 −1/N) .

If we pick, say, two messages x and y randomly from the pool, the events that the hashcode of neither is equal to h are probabilistically independent. That implies that the probability that none of two messages has its hashcode equal to h is $(1 - 1/N)^2$

Extending the above reasoning to the entire pool of k messages, it follows that the probability that none of the messages in a pool of k messages has its hashcodes equal to h is $(1 - 1/N)^k$

Therefore, the probability that at least one of the k messages has its hashcode equal to h is $1 - (1 - 1/N)^k$

The probability expression shown above can be considerably simplified by recognizing that as a approaches 0, we can write $(1 + a)^n \approx 1 + an$. Therefore, the probability expression we derived can be approximated by

$\approx 1 - (1 - k/N) = k/N$

So, given a pool of k randomly produced messages, the probability there will exist at least one message in this pool whose hashcode equals the given value h is k/N

 Let’s now go back to the original question: How large should k be so that the pool of messages contains at least one message whose hashcode equals the given value h with a probability of 0.5? We obtain the value of k from the equation k/N = 0.5. That is, k = 0.5N.

What Is the Probability That There Exist At Least Two Messages With the Same Hashcode?

Assuming that a hash algorithm is working perfectly, meaning that it has no biases in its output that may be induced by either the composition of the messages or by the algorithm itself, the goal of this section is to estimate the smallest size of a pool of randomly selected messages so that there exist at least two messages in the pool with the same hashcode with probability 0.5.

 Given a pool of k messages, the question “What is the probability that there exists at least one message in the pool whose hashcode is equal to a specific value?” is very different from the question “What is the probability that there exist at least two messages in the pool whose hashcodes are the same?”

 Raising the same two questions in a different context, the question “What is the probability that, in a class of 20 students, someone else has the same birthday as yours (assuming you are one of the 20 students)?” is very different from the question “What is the probability that there exists at least one pair of students in a class of 20 students with the same birthday?” The former question was addressed in the previous section. Based on the result derived there, the probability of the former is approximately 19/365

The latter question we will address in this section. As you will see, the probability of the latter is roughly the much larger value

((20*19)/2)/365

Strictly speaking, as you'll see, this calculation is valid only when the class size is very small compared to 365.] This is referred to as the birthday paradox — it is a paradox only in the sense that it seems counterintuitive.

A quick way to accept ! the ‘paradox’ intuitively is that for ‘20 choose 2’ you can construct C(20, 2) $=20C_2$

=20!/ 18!2!

= 20×19/2

= 190 different possible pairs from a group of 20 people. Since this number, 190, is rather comparable to 365, the total number of different birthdays,

 Given a pool of k messages, each of which has a hashcode value from N possible such values, the probability that the pool will contain at least one pair of messages with the same hashcode is given by

$$1 - \frac{N!}{(N-k)!N^k}$$

M 1 , in which we can construct a pool of k message with no duplicate hashcodes and the total number of ways,

M 2 , we can do the same while allowing for duplicates.

The ratio M 1 /M 2 then gives us the probability of constructing a pool of k messages with no duplicates.

Subtracting this from 1 yields the probability that the pool of k messages will have at least one duplicate hashcode.

Therefore, the total number of ways, M1, in which we can construct a pool of k messages with no duplications in hashcode values is

$$M_1 = N \times (N-1) \times \ldots \times (N-k+1) = \frac{N!}{(N-k)!}$$

Let’s now try to figure out the total number of ways, M2, in which we can construct a pool of k messages without worrying at all about duplicate hashcodes. Reasoning as before, there are N ways to choose the first message. For selecting the second message, we pay no attention to the hashcode value of the first message. There are still N ways to select the second message; and so on. Therefore, the total number of ways we can construct a pool of k messages without worrying about hashcode duplication is

$$M_2 = N \times N \times \ldots \times N = N^k$$

Therefore, if you construct a pool of k purely randomly selected messages, the probability that this pool has no duplications in the hashcodes is

$$\frac{M_1}{M_2} = \frac{N!}{(N-k)!N^k}$$

– We can now make the following probabilistic inference: if you construct a pool of k message as above, the probability that the pool has at least one duplication in the hashcode values is

$$1 - \frac{N!}{(N-k)!N^k}$$

$$1 \quad - \quad \frac{N \times (N-1) \times \ldots \times (N-k+1)}{N^k}$$

which is the same as

$$1 \quad - \quad \frac{N}{N} \times \frac{N-1}{N} \times \ldots \times \frac{N-k+1}{N}$$

and that is the same as

$$1 \quad - \quad \left[\left(1-\frac{1}{N}\right) \times \left(1-\frac{2}{N}\right) \times \ldots \times \left(1-\frac{k-1}{N}\right)\right]$$

 We will now use the approximation that $(1 - x) \leq e^{-x}$ for all $x \geq 0$ to make the claim that the above probability is lower-bounded by

$$1 - \left[e^{-\frac{1}{N}} \times e^{-\frac{2}{N}} \times \ldots \times e^{-\frac{k-1}{N}} \right]$$

Since 1 + 2 + 3 + . . . + (k − 1) is equal to k(k−1)/ 2 , we can write the following expression for the lower bound on the probability

$$1 - e^{-\frac{k(k-1)}{2N}}$$

 When k is small and N large, we can use the approximation $e^{-x} \approx 1 - x$ in the above formula and express it as

$$1 - \left(1 - \frac{k(k-1)}{2N}\right) = \frac{k(k-1)}{2N}$$

We will now estimate the size k of the pool so that the pool contains at least one pair of messages with equal hashcodes with a probability of 0.5. We need to solve

$$1 - e^{-\frac{k(k-1)}{2N}} = \frac{1}{2}$$

Simplifying, we get

$$e^{\frac{k(k-1)}{2N}} = 2$$

Therefore,

$$\frac{k(k-1)}{2N} = ln2$$

which gives us

$$k(k-1) = (2ln2)N$$

Assuming k to be large, the above equation gives us

$$k^2 \approx (2ln2)N$$

implying

$$\begin{aligned} k &\approx \sqrt{(2ln2)N} \\ &\approx 1.18\sqrt{N} \\ &\approx \sqrt{N} \end{aligned}$$

So our final result is that if the hashcode can take on a total N different values with equal probability, a pool of $\sqrt{N}$ messages will contain at least one pair of messages with the same hashcode with a probability of 0.5.

So if we use an n-bit hashcode, we have $N = 2^n$. In this case, a pool of $2^{n/2}$ randomly generated messages will contain at least one pair of messages with the same hashcode with a probability of 0.5.

Understand the similar question in different way:

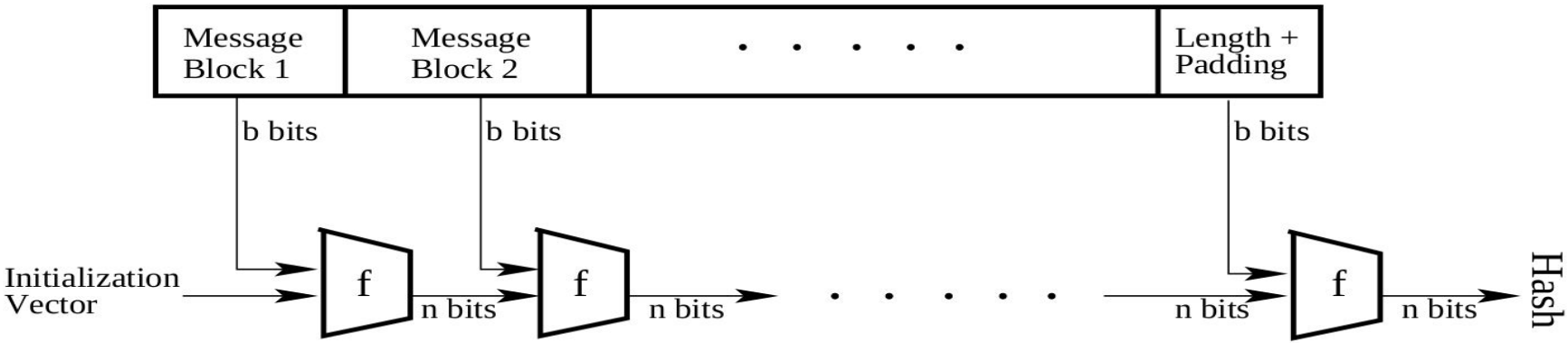
Alice has a dishonest assistant, Bob, preparing contracts for Alice's digital signature.

> Bob prepares the legal contract for a transaction and then proceeds to create a large number of variations of the legal contract without altering the legal content of the contract and computes the hashcode for each. These variations may be constructed by mostly innocuous changes such as the insertion of additional white space between some of the words, or contraction of the same; insertion or deletion of some of the punctuation, slight reformatting of the document, etc.

Next, Bob prepares a fraudulent version of the contract. As with the correct version,Bob prepares a large number of variations of this contract, using the same tactics as with the correct version.

- Now the question is: “What is the probability that the two sets of contracts will have at least one contract each with the same hashcode?”

Structure of Cryptographically Secure Hash function



SHA Family

Algorithm	*Message Size* (bits)	*Block Size* (bits)	*Word Size* (bits)	*Message Digest Size* (bits)	*Security* (ideally) (bits)
SHA-1	$< 2^{64}$	512	32	160	80
SHA-256	$< 2^{64}$	512	32	256	128
SHA-384	$< 2^{128}$	1024	64	384	192
SHA-512	$< 2^{128}$	1024	64	512	256