Authentication

Examples

Steven Gordon

1 Message Authentication Codes

As a simple illustration of how MACs work, consider the case of A sending a message to B, and a malicious user C intercepting the message.

The assumptions are:

- A and B have a shared secret key, *S*
- B can detect that a message

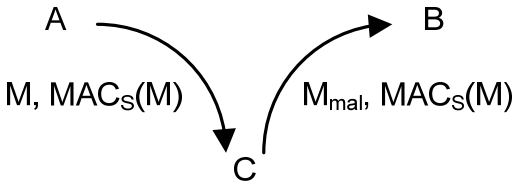
In the normal case, A sends a message, *M*, to B, as well as the MAC of that message, which is generated with the secret key *S*.

Upon receipt, B will calculate the MAC of the received message and compare it to the received MAC. If they are equal, B assumes the message:

- Has definitely come from A because only A has *S* (authentication)
- Has not been modified along the way since the MACs are equal (data integrity)

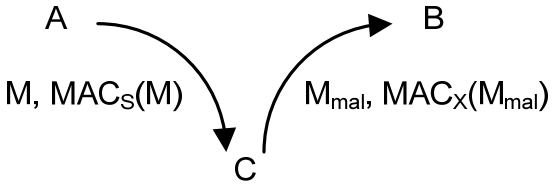
Now consider the cases of the malicious user C intercepting the message and modifying the message.

In the first scenario, C modifies the message to M_{mal}, but does not modify the MAC.



Upon receipt, B will calculate the MAC of the received message, M_{mal}, and compare it to the received MAC. B will determine they are not equal because applying a MAC function on two different messages using the same key *S* will produce two different answers. Hence, B has detected a problem and will ignore the message (and probably contact A some other means to inform them of the problem).

In the second scenario, C modifies the message to M_{mal}, and also recalculates the MAC of M_{mal}. Note that since C does not know *S*, they use their own key *X* to generate the MAC.



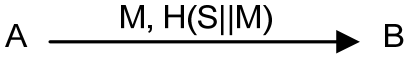
Upon receipt, B will calculate the MAC of the received message, M_{mal}, and compare it to the received MAC. B will determine they are not equal because applying a MAC function on the same message M_{mal} but using a different key (*S* and *X*) will produce two different answers. Hence, B has detected a problem and will ignore the message (and probably contact A some other means to inform them of the problem).

2 Requirements of Hash Functions

Two requirements of hash functions are the one way property and weak collision resistance. Lets consider two practical cases that demonstrate the need for these requirements (that is, if the properties do not hold, how does an attacker break the security?).

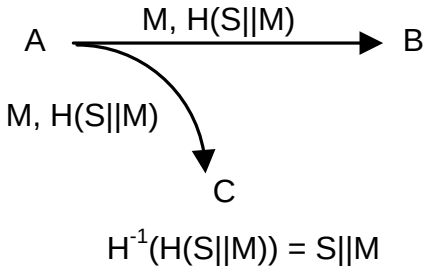
2.1 One-Way Property

One simple method of performing authentication is for A to send a secret, *S*, to B. Assuming only A and B know the secret, then upon receipt, B confirms that the message comes from A (because it contains the secret *S*). To do this so that other users cannot discover the secret S, A can send a Hash of S (instead of S). See below:



Upon receipt, B will calculate Hash of the received message M and the secret S (that it knows), and if it is equal to the received hash value, then A is authenticated.

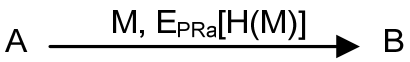
Lets consider what could happen if the one-way property did not hold. That is, given the hash value, the attacker C can calculate the original message. In other words, the attacker can calculate the inverse Hash function, H^{-1}.



Upon receipt, C calculates the inverse Hash function on the received hash value and obtains the secret S combined with the message M. Since the attacker also knows M, the attacker can discover the secret. This is one reason the one way property is a requirement of Hash functions for authentication.

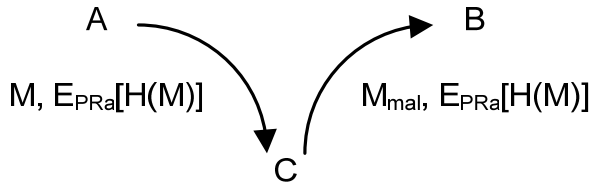
2.2 Weak Collision Resistance

Hash functions and RSA encryption are often combined to produce a digital signature for a message *M*. A hash value of the message is calculated and then encrypted using the Private Key of the sender A. Therefore, anyone who receives the message can confirm that the message came from A (since it will only successfully decrypt with the Public Key of A).



Upon receipt, B will decrypt with *PUa* and then compare the hash value with the result of Hashing the received message *M*.

Lets consider what would happen if the weak collision resistance property does not hold. That is, an attacker can find another message, M_{mal}, such that its hash value is the same as the hash value of *M*.



Malicious user C intercepts the message and finds another message Mmal with the same hash value as M. That is: H(M_{mal}) = H(M). C does not modify E_{PRa}[H(M)], but forwards it, along with M_{mal} to B.

Upon receipt, B will decrypt E_{PRa}[H(M)] using A's public key to get H(M). Then B will calculate the Hash of the received message, M_{mal}, to get H(M_{mal}). Since the two hash values are identical, B will think the message has not been modified, thereby the attack is successful. This is one reason weak collision resistance is a requirement of Hash functions for authentication and integrity.

3 Strength of Hash Functions

3.1 Collision Resistance

The aim of a hash function is that a unique hash value will be calculated for each unique input message. However, in practice this is impossible because we want our hash function to operate on arbitrarily large messages and produce fixed size, small hash values. If, for example, our hash value was to be 4 bits, then we could only have input messages that were four bits in length. Hence, it is possible that two different input messages will produce the same hash value – a *collision*.

A desirable property of secure hash functions is that it is difficult for the attacker to discover messages that produce colliding hash values. This is formulated as:

- Weak collision resistance: for a given message x, it should be hard for an attacker to find another message y such that H(y) = H(x).
- Strong collision resistance: it should be hard for an attacker to be able to choose two values, x and y, such that H(x) = H(y).

Comparing the two properties: strong collision resistance is a harder property to achieve in hash functions (in other words, it is easier for an attacker to find to such values x and y); weak collision resistance is an easier property to achieve in hash functions (in other words, it is harder for an attacker to find a value for y, if they are given x).

3.2 Birthday Attack

Although proving the relative strengths of the weak collision resistant and strong collision resistant property, you can gain an understanding by comparing to the statistics of the birthday attack.

Firstly, lets assume there are a limited set of messages, n.

Again, the weak collision resistance property can be re-stated as: given a message x with has a hash value, H(x), then what is the chance of finding another message y (out of the remaining n-1 messages) that has the same has value?

And the strong collision resistance property: of the n messages, what is the chance that we can find any pair of messages, x and y, that have the same hash values?

Now consider the problem with respect to a group of n people, and their birth days. Lets assume everyone has equal chance to be born on one of the 365 days in a year, ignoring leap years and

other real phenomena such as twins. Two people born on 12 January (irrespective of year) have the same birth day. Lets calculate:

1. Given one person *x* with a birth day on some date, then what is the chance of finding another person *y* (out of the remaining *n*-1 people) that has the same birth day? (Similar to weak collision resistance)
2. Of the *n* people in the group, what is the chance that we can find any two people, *x* and *y*, that have the same birth days? (Similar to strong collision resistance)

3.2.1 Pr(someone has same birth day as x)

This is the opposite of: probability that no-one has the same birth day as *x*. That is:

Pr(someone has same birth day as *x*) = 1 – **Pr(no-one has the same birth day as *x*)**

Pr(no-one has the same birth day as *x*)

If there are 2 people in the group ($n = 2$), and if *x* has a birth day on 1 January, then probability that the other person (*a*) does not have same birthday as *x* is 364/365 (since *a* can have a birth day on 364 of the possible 365 days, and not be the same as *x*).

If there are 3 people in the group ($n = 3$), and if *x* has a birth day on 1 January, then probability that *a* does not have same birthday as *x* is 364/365, and the probability that *b* does not have the same birthday as *x* is also 364/365. Hence, the probability that neither *a* nor *b* have the same birth day as *x* is: (364/365) x (364/365).

It is easy to show that in general, out of a group of *n* people (inclusive of *x*):

Pr(no-one has the same birth day as *x*) = $(364/365)^n$

Hence,

Pr(someone has same birth day as *x*) = 1 – $(364/365)^n$

3.2.2 Pr(any 2 people have the same birth day)

This is the opposite of: probability that no two people in the group have same birthday. That is:

Pr(any 2 people have the same birth day) = 1 – **Pr(no two people have same birth day)**

Pr(no two people have same birth day)

If there are 2 people in the group ($n = 2$), and if *x* has a birth day on 1 January, then probability that the other person (*a*) does NOT have a birth day on the same day is 364/365 (since a must have a birth day on 1 of the other 364 days).

If there are 3 people in the group ($n = 3$), and if *x* has a birth day on 1 January, then probability that *a* does not have a birth day on the same day is 364/365, and the probability that *b* does not have a birth day on the same day as *x* or *a* is 363/365. Hence the probability that none of the 3 have the same birth day is: (364/365) x (363/365)

In general, this can be extended to:

Pr(no two people have same birth day) = (364/365) x (363/365) x (362/365) … x ((365 – (n-1))/365)

= 365 x 364 x 363 x 362 x … x (365 – (*n*-1)) / 365^n

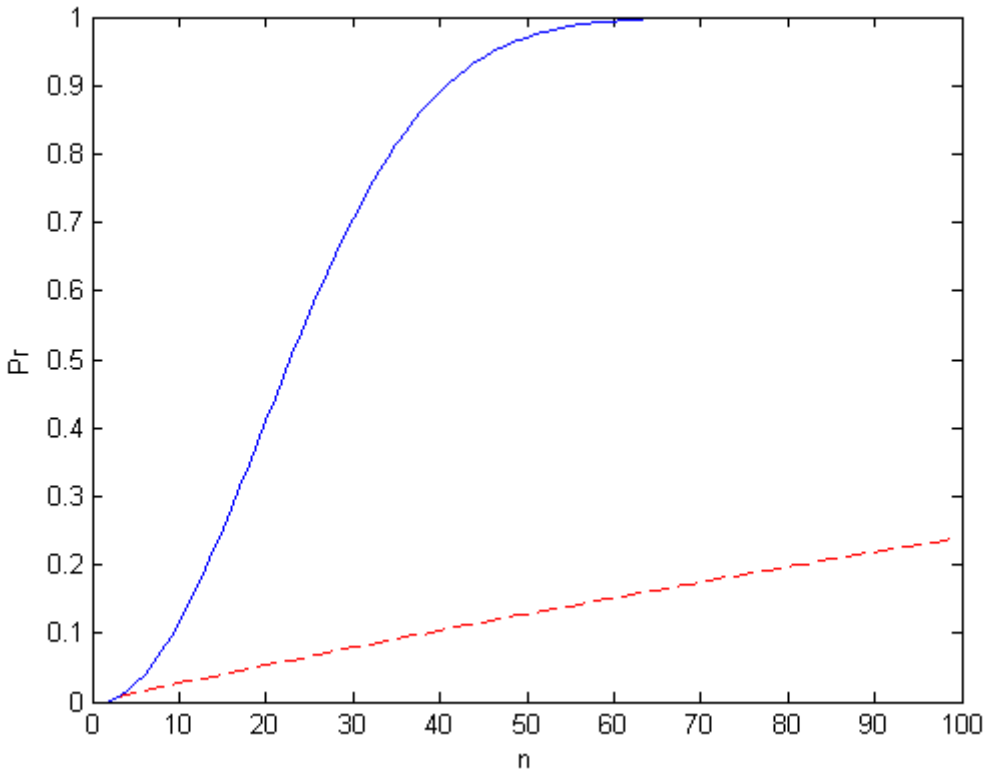
= 365! / (365^n x (365-*n*)!)

Hence,

Pr(any 2 people have the same birth day) = $1 - (365! / (365^n \times (365-n)!))$

3.2.3 Comparing the Two Probabilities

Although it may be difficult to see immediately from the resulting probability equations, a plot of the two, with different values of n, shows that it is much more likely for finding a "collision" of birthdays when choosing any two people. The red dashed line shows **Pr(someone has same birth day as *x*)** and the blue solid line shows **Pr(any 2 people have the same birth day)**.



Although it doesn't directly follow, similar logic can be applied to showing that the probability of finding a collision where any attacker can choose any value of *x* and *y* (that is, similar to probability that any 2 people have birth day on same day) is much higher than the probability of finding a collision where the attacker is given *x* and must find *y* (that is, similar to probability that someone has same birth day as *x*).

3.3 Summary

In summary, it is easier for an attacker to break the strong collision resistance property (compared to the weak collision resistance property). It takes approximately $2^{n/2}$ attempts, where n is the number of bits in the hash. Therefore the brute force strength of a hash function is typically measured by this value. So, if we were to compare the effort needed to break DES is 2^{56} for a 56 bit key, then the same effort would be needed to break a hash function with a 112 bit hash.