Authentication functions

They enable a recipient to determine that a message came from the alleged sender (although he might not be able to prove this to others) and that the message has not been tampered with (inserted, deleted, changed).

Single key encryption provides some authentication, if only the sender and receiver share a key K.

Public key encryption provides no authentication unless signatures are used; if they are, there is both authentication and non-repudiation.

Encryption also provides confidentiality, often not needed. For example:

1. A significant broadcast message: should you believe it?

2. A program someone sends you. Before you run it, how can you tell that it was not modified en route?

Another type of authentication function uses a secret key K to generate a small, fixed-sized data block, called a cryptographic checksum or message authentication code (MAC), that is appended to the message.

If only the sender and recipient know K, and the recipient verifies the checksum, then he knows the message comes from the alleged sender (because no one else could compute the checksum) and the message has not been altered.

A typical implementation of a cryptographic checksum uses K in DES in CBC mode with initial register value 0 to encipher the message. Only the final block of ciphertext is saved; it is the checksum.

In the case of the significant broadcast message mentioned above, it would be better to have a "key-less" algorithm generate one "checksum" and then encipher it once for each recipient to produce authenticity.

In this situation the "key-less checksum" is called a Message Digest or Hash Function. They are a common way to provide authentication without confidentiality.

Hash Functions

A *weak hash function* is a function $h = H(M)$ of fixed length (m bits) of a message M of any length such that:

1. Given M, it is easy to compute $H(M)$,

2. Given h, it is hard to compute any M for which $H(M) = h$, and

3. Given M, it is hard to find $M' \neq M$ for which $H(M') = H(M)$.

A *strong hash function* is a weak hash function which also satisfies:

4. It is hard to find *any* two messages $M' \neq M$ for which $H(M') = H(M)$.

Usually $H(M)$ is either enciphered or transmitted separately from M.

Property 3 provides the authentication.

Property 4 protects M against "birthday attacks".

Most hash functions are built from a one-way function f which takes two values of length m bits and produces a value of length m bits. The whole message M is broken into blocks M_i of length m bits each. One computes $h_i = f(M_i, h_{i-1})$ with some standard initial value h_0. The hash value (or digest) is the final h_i.

Some examples of hash functions are SNEFRU, N-Hash, MD4, MD5, and SHA.

We discuss MD5 and SHA. MD5 produces a 128-bit digest, while SHA's digest has 160 bits.

MD5

MD5 has four "rounds" and produces a 128-bit key.

First pad the message M with a 1 and as many 0's as needed to make the total length $\equiv 448 = 512-64 \pmod{512}$. The last 64 bits hold the length of the message (in bits) before padding (modulo 2^{64}) This makes the message length a multiple of 512 bits. Call the 512-bit blocks $Y_0, \ldots, Y_{L-1}$.

MD5 uses a buffer of four 32-bit registers: a, b, c, d. These are initialized to the hexadecimal values a = `01 23 45 67`, b = `89 AB CD EF`, c = `FE DC BA 98`, d = `76 54 32 10`.

The message is hashed by computing abcd = $f(Y_i$,abcd), with the last value of abcd being the hash value.

Each round of MD5 has 16 steps.

SHA

The Secure Hash Algorithm, created by NIST and NSA, uses four rounds to produce a 160-bit message digest.

The message is first padded to a multiple of 512 bits just as in MD5.

SHA uses a buffer of five 32-bit registers: a, b, c, d, e. These are initialized to the hexadecimal values a = `67 45 23 01`, b = `EF CD AB 89`, c = `98 BA DC FE`, d = `10 32 54 76`. e = `C3 D2 E1 F0`. (The first four constants are the same as for MD5, but stored in big-endian format rather than the little-endian format used in MD5.)

Each round of SHA has 20 steps.

Comparison of MD5 and SHA

Both add a fourth round to MD4, which had only three rounds.

The four non-linear functions of MD5 are all different. SHA has the same non-linear functions in Rounds 2 and 4.

MD5 uses 64 different constants t_i. SHA uses four different constants K_t, one per round, like MD4.

MD5 has only one symmetric non-linear function. SHA has two symmetric non-linear functions.

Both MD5 and SHA add aa to a, etc., in each step to promote rapid mixing.

MD5 has variable shifts in each round. SHA has constant shifts in each round.

SHA has an expansion of each 512-bit block. MD5 does not do this.

MD5 produces a message digest of 128 bits. SHA produces a message digest of 160 bits. Thus SHA is more resistant to a birthday attack.

The design criteria of MD5 are public. The design criteria of SHA are secret.

The Birthday Problem

What is the smallest positive integer k so that the probability is > 0.5 that at least two people in a group of $k$people have the same birthday?

Ignore February 29.

Assume each birthday is equally likely.

The probability that k people all have different birthdays is

$$\frac{365}{365} \times \frac{364}{365} \times \frac{363}{365} \times \cdots \times \frac{365-k+1}{365}$$

which is

$$\frac{365!}{(365-k)! \times (365)^k}.$$

Thus the probability that at least two of k people have the same birthday is

$$P(k) = 1 - \frac{365!}{(365-k)! \times (365)^k}.$$

More generally, suppose we are given an integer-valued random variable with uniform distribution between 1 and n. Choose k instances of this random variable. What is the probability $P(n,k)$ that at least two of the k instances are the same value?

As for birthdays, we find

$$P(n,k) = 1 - \frac{n!}{(n-k)!n^k}.$$

Write this as

$$P(n,k) = 1 - (1 - \frac{1}{n})(1 - \frac{2}{n}) \times \cdots \times (1 - \frac{k-1}{n}).$$

To estimate this, note that $1 - x \leq e^{-x}$ for all $x \geq 0$ and $1 - x \approx e^{-x}$ when x is small.

This gives

$$P(n,k) > 1 - e^{-1/n} e^{-2/n} e^{-3/n} \times \cdots \times e^{-(k-1)/n}$$

$$P(n,k) > 1 - e^{-(1/n+2/n+3/n+\cdots+(k-1)/n}$$

$$P(n,k) > 1 - e^{-k(k-1)/(2n)}.$$

We will have $P(n,k) = 0.5$ when

$$0.5 = 1 - e^{-k(k-1)/(2n)}$$

or $2 = e^{k(k-1)/(2n)}$, that is, when

$$\ln 2 = k(k-1)/(2n).$$

When k is large, the percentage difference between k and $k-1$ is small, and we may approximate $k-1 \approx k$. This gives $k^2 \approx 2n \ln 2$ or

$$k \approx \sqrt{2(\ln 2)n} \approx 1.18\sqrt{n}.$$

For $n = 365$, we find

$$k \approx 1.18\sqrt{365} \approx 22.54,$$

or $k \approx 23$.

Suppose $H(M)$ is a hash function with m-bit output. There are $n = 2^m$ possible hash values.

If H is applied to k random inputs, the probability of finding a duplicate ($H(M) = H(M')$) is $P(2^m, k)$. The minimum number of k needed for a duplicate to occur with probability > 0.5 is about

$$k = 1.18\sqrt{2^m} = 1.18 \times 2^{m/2}.$$

The overlap between two sets

Given an integer random variable with uniform distribution between 1 and n, and two sets of k ($k \leq n$) instances of the random variable, what is the probability $R(n, k)$ that the two sets overlap, that is, at least one of the n values appears in both sets?

We assume k is small enough ($k < \sqrt{n}$) so that the k instances of the random variable in each set are all different. (A few duplicates won't hurt this analysis.)

The probability that one given element of the first set does not match any element of the second set is $(1 - \frac{1}{n})^k$.

The probability that the two sets are disjoint is

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

so $R(n, k) = 1 - (1 - \frac{1}{n})^{k^2}$.

Using $1 - x \leq e^{-x}$, we get

$$R(n,k) > 1 - (e^{-1/n})^{k^2} = 1 - e^{-k^2/n}.$$

We will have $R(n,k) = 0.5$ when $\frac{1}{2} = 1 - e^{-k^2/n}$ or $2 = e^{k^2/n}$ or $\ln 2 = k^2/n$ or

$$k = \sqrt{(\ln 2)n} \approx 0.83\sqrt{n}$$

Suppose a hash function H with $n = 2^m$ possible values is applied to k random inputs to produce a set X and again to k additional random inputs to produce a set Y. What is the minimum value of k so that the probability is at least 0.5 of finding at least one match between the two sets, that is, $H(x) = H(y)$, where $x \in X$ and $y \in Y$? Using the approximation above, the minimum k is about

$$k = 0.83\sqrt{2^m} = 0.83 \times 2^{m/2}.$$