Message Digest Functions

Message digest functions distill the information contained in a file, small or large, into a single large number, typically between 128 and 256 bits in length. The best message digest functions combine these mathematical properties:

- Every bit of the digest function's output is potentially influenced by every bit of the function's input
- If any given bit of the function's input is changed, every output bit has a 50 percent chance of changing.
- Given an input file and its corresponding message digest, it should be computationally infeasible to find another file with the same message digest value

Message digests are also called one-way hash functions because they produce values that are difficult to invert, resistant to attack, effectively unique, and widely distributed. Here is a sampling of more popular functions:

MD2, MD4, MD5

These were all developed by Ronald Rivest. All produce message digests and have been show to have flaws. MD5 is used in SSL and in Microsoft's Authenticode technology. It produces a 128 bit digest.

SHA, SHA-1, SHA-256,384,512

The Secure Hash Algorithm is related to MD4 and designed for use with NIST's DSS. SHA-1 is a revised SHA and incorporates minor changes. It is not publicly known if these changes make SHA-1 more secure than SHA, although many people believe that they do. SHA-1 produces a 16- bit digest. SHA-256, 384, and 512 has functions that are designed to be used with 128, 192 and 256 bit encryption algorithms, respectively. These digest were proposed by NIST for use with AES.

Message Digest Algorithms at Work

Message digest algorithms are not generally used for encryption and decryption operations. They are used to create digital signatures, message authentication codes and encryption keys from passphrases. Let's take a look at some digests:

MD5(There is $1500 in the blue bo)	=	`f80b3fde8ecbac1b515960b9058de7a1`
MD5(There is $1500 in the blue box)	=	`a4a5471a0e019a4a502134d38fb54729`
MD5(There is $1500 in the blue bob.)	=	`05f8cfc03f4e58cbee731aa4a14b3f03`
MD5(There is $1500 in the blue box!)	=	`4b36807076169572b804907735accd42`
MD5(There is $1500 in the blue box..)	=	`3a7b4e07ae316eb60b5af4a1a2345931`

The purpose of the above example is to point out the randomness of the resulting hash when the input is very similar.

Message digest functions are a powerful tool for detecting very small changes in very large files or messages. Calculate the MD5 code for your message and set it aside. If you think that the file has been changed, either accidentally or on purpose, simply recalculate the MD5 code and compare it with the MD5 code that you originally calculated. If they match, you can safely assume that the file was not modified.

In theory, two different files can have the same message digest value. This is called a *collision*. For a message digest function to be secure, it should be computationally infeasible to find or produce these collisions.

Uses of Message Digest Functions

Message digest functions are widely used today for a number of reasons: Some of those reasons are:

- Message digest functions are much faster to calculate than traditional symmetric key cryptographic functions but appear to share many of their strong cryptographic properties.

- There are no patent restrictions on any message digest functions that are currently in use.

- There are no export or import restrictions on message digest functions.

- Message digest functions appear to provide an excellent means of spreading the randomness from an input among all of the function's output bits. (diffusion)

- Using a message digest, you can easily transform a typed passphrase into an encryption key for use with a symmetric cipher.

- Message digests can be readily used for message authentication codes that use a shared secret between two parties to prove that a message is authentic. MACs are appended to the end of the message to be verified.

Because of their properties, message digest functions are also an important part of many cryptographic systems in use today:

- Message digests are the basis of most digital signature standards. Instead of signing the entire document, most digital signature standards specify that the message digest of the document to be calculated. It is the message digest rather than the entire document that is actually signed.

- MACs based on message digests provide the "cryptographic" security for most of the Internet's routing protocols.

- Programs such as PGP use message digests to transform a pass phrase provided by a user into an encryption key that is used for symmetric encryption.

HMAC

A Hash Message Authentication Code function is a technique for verifying the integrity of a message transmitted between two parties that agree on a shared secret key. Essentially, HMAC combines the original message and a key to compute a message digest function. The sender of the message computes the HMAC of the message and the key and transmits the HMAC with the original message. The recipient recalculates the HMAC using the message and the secret key, then compares the received HMAC with the calculated HMAC to see if they match. If the two HMACs match, then the recipient knows that the original message has not been modified because the message digest has not changed and that it is authentic because the sender knew the shared key which is presumed to be secret.

HMACs have an important disadvantage over digital signature systems because HMACs are based on a shared key. If either party's key is compromised, it will be possible for an attacker to create fraudulent messages.

Attacks of Message Digest Functions

There are two kinds of attacks on message digest functions. The first is finding two messages, any two messages, that have the same message digest. The second attack is significantly harder. Given a particular message, the attacker finds a second message that has the same message digest code.

MD5 is probably secure enough to be used over the next 5 to 10 years. Even if it becomes possible to find MD5 collisions at will, it will be very difficult to transform this knowledge into a general purpose attack on SSL.