HASHING ALGORITHMS

binarywarriors5@gmail.com

1. MD5 (Message-Digest algorithm 5): is a widely used cryptographic function with a 128- bit hash value. MD5 has been employed in a wide variety of security applications, and is also commonly used to check the integrity of files.An MD5 hash is typically expressed as a 32-digit hexadecimal number.

1.1 ALGORITHM:-MD5 processes a variable-length message into a fixed-length output of 128 bits.

1.2 STEPS:

1.The input message is broken up into chunks of 512-bit blocks (sixteen 32-bit little endian integers),the message is padded so that its length is divisible by 512.
2.The padding works as follows: first a single bit, 1, is appended to the end of the message.
3.This is followed by as many zeros as are required to bring the length of the message up to 64 bits fewer than a multiple of 512.
4.The remaining bits are filled up with a 64-bit integer representing the length of the original message, in bits.
5.The MD5 algorithm uses 4 state variables, each of which is a 32 bit integer (an unsigned long on most systems). These variables are sliced and diced and are (eventually) the message digest. The variables are initialized as follows:
A = 0x67452301
B = 0xEFCDAB89
C = 0x98BADCFE
D = 0x10325476.
6. Now on to the actual meat of the algorithm: the main part of the algorithm uses four functions to thoroughly goober the above state variables. Those functions are as follows:

F(X,Y,Z) = (X & Y) | (~(X) & Z)
G(X,Y,Z) = (X & Z) | (Y & ~(Z))
H(X,Y,Z) = X ^ Y ^ Z
I(X,Y,Z) = Y ^ (X | ~(Z))
Where &, |, ^, and ~ are the bit-wise AND, OR, XOR, and NOT operators
7. These functions, using the state variables and the message as input, are used to transform the state variables from their initial state into what will become the message digest. For each 512 bits of the message, the rounds performed (this is only pseudo-code, don't try to compile it)
After this step, the message digest is stored in the state variables (A, B, C, and D). To get it into the hexadecimal form you are used to seeing, output the hex values of each the state variables, least significant byte first. For example, if after the digest:
A = 0x01234567;
B = 0x89ABCDEF;

C = 0x1337D00D
D = 0xA5510101
Then the message digest would be:
67452301EFCDAB890DD03713010151A5 (required hash value of the input value).

1.3 EXAMPLE:

MD5("The quick brown fox jumps over the lazy dog")
= 9e107d9d372bb6826bd81d3542a419d6

1.4 PROS:

- It is easy to compute the hash value for any given message,
- It is infeasible to find a message that has a given hash,
- It is infeasible to modify a message without changing its hash,
- It is infeasible to find two different messages with the same hash.

1.5 CONS:

- The security of the MD5 hash function is severely compromised.
- A collision attack exists that can find collisions within seconds on a computer with a 2.6Ghz Pentium4 processor (complexity of $2^{24.1}$).
- A number of projects have published MD5 rainbow tables online, that can be used to reverse many MD5 hashes into strings that collide with the original input, usually for the purposes of password cracking.

1.6 BASIC PROPERTIES:

OUTPUT SIZE:-	128(bits)
INTERNAL STATE SIZE:_	128(bits)
BLOCK SIZE:	512(bits)
LENGTH SIZE	64(bits)
WORD SIZE	32(bits)
COLLISION ATTACKS	$2^{20.96}$

1.7 LINUX KERNEL SPACE IMPLEMENTATION:

All the functions are defined in "**md5.c".**
void **MD5_Init**(MD5_CTX *c);
void **MD5_Update**(MD5_CTX *c, const void *data, unsigned long len);
void **MD5_Final**(unsigned char *md, MD5_CTX *c);

Description

The following functions may be used if the message is not completely stored in memory:

- MD5_Init() initializes a MD5_CTX structure.
- MD5_Update() can be called repeatedly with chunks of the message to be hashed (len bytes at data).
- MD5_Final() places the message digest in md, which must have space for MD5_DIGEST_LENGTH == 16 bytes of output, and erases the MD5_CTX .

2. SHA (SECURE HASH ALGORITHM):-

Various types of SHA:-

- SHA 0
- SHA 1
- SHA 256
- SHA 512

The Secure Hash Algorithm is one of a number of cryptographic hash functions.There are currently three generations of Secure Hash Algorithm:

- SHA-1 is the original 160-bit hash function. Resembling the earlier MD5 algorithm.
- SHA-2 is a family of two similar hash functions, with different block sizes, known as SHA-256 and SHA-512. They differ in the word size; SHA-256 uses 32-bit words where SHA-512 uses 64-bit words.
- SHA-3 is a future hash function standard still in development.

2.1 BASIC PROPERTIES:

• OUTPUT SIZE:-	160(bits)
• INTERNAL STATE SIZE:_	160(bits)
• BLOCK SIZE:	512(bits)
• MAX MESSAGE SIZE	$2^{64} - 1$
• LENGTH SIZE	64(bits)
• WORD SIZE	32(bits)
• COLLISION ATTACKS	No (2^{51} attack)
• ROUNDS	80

2.2 ALGORITHM:

1-The SHA algorithm uses 5 state variables, each of which is a 32 bit integer (an unsigned long on most systems). These variables are sliced and diced and are (eventually) the message digest. The variables are initialized as follows:

h0 = 0x67452301
h1 = 0xEFCDAB89
h2 = 0x98BADCFE
h3 = 0x10325476

h4 = 0xC3D2E1F0

There are 80 rounds in SHA Algorithm
THE hash value generated by the sha hash function.

SHA1("The quick brown fox jumps over the lazy dog")

= 2fd4e1c6 7a2d28fc ed849ee1 bb76e739 1b93eb12

Even a small change in the message will, with overwhelming probability, result in a completely different hash due to the avalanche effect . For example, changing dog to cog produces a hash with different values for 81 of the 160 bits:

SHA1("The quick brown fox jumps over the lazy **c**og")
= de9f2c7f d25e1b3a fad3e85a 0bd17d9b 100db4b3

2.3 PROS:

- It is easy to compute the hash value for any given message,
- It is infeasible to find a message that has a given hash,
- It is infeasible to modify a message without changing its hash,
- It is infeasible to find two different messages with the same hash.
- The collision ratio for SHA is far less than the collision ratio MD

2.4 CONS

Cryptographers have produced collision pairs for SHA-0 and have found algorithms that should produce SHA-1 collisions in far fewer than the originally expected 2^{80} evaluations

2.5 LINUX KERNEL SPACE IMPLEMENTATION:

unsigned char *SHA1(const unsigned char *d, unsigned long n,unsigned char *md);

void SHA1_Init (SHA_CTX *c);

void SHA1_Update (SHA_CTX *c, const void *data,unsigned long len);

void SHA1_Final (unsigned char *md, SHA_CTX *c);

2.6 Description

SHA-1 (Secure Hash Algorithm) is a cryptographic hash function with a 160 bit output.)

SHA1 () computes the SHA-1 message digest of the n bytes at d and places it in md (which must have space for SHA_DIGEST_LENGTH == 20 bytes of output). If md is NULL , the digest is placed in a static array.

The following functions may be used if the message is not completely stored in memory:

SHA1_Init() initializes a SHA_CTX structure.

SHA1_Update() can be called repeatedly with chunks of the message to be hashed (len bytes at data).

SHA1_Final() places the message digest in md, which must have space for SHA_DIGEST_LENGTH == 20 bytes of output, and erases the SHA_CTX .

2.7 INFORMATION ABOUT /proc/crypto

/proc/crypto

This file lists all installed cryptographic ciphers used by the Linux kernel, including additional details for each. A sample /proc/crypto file looks like the following:

```
name         : sha1
module       : kernel
type         : digest
blocksize    : 64
digestsize   : 20


name         : md5
module       : md5
type         : digest
blocksize    : 64
digestsize   : 16.
```