# **Cryptographic Hashing Functions & HMAC**

**Definitions:**

Hash Function:

Collision Resistance:

Pigeonhole Principle:

*there does not exist an* [*injective function*](https://en.wikipedia.org/wiki/Injective_function) *whose* [*codomain*](https://en.wikipedia.org/wiki/Codomain) *is smaller than its* [*domain*](https://en.wikipedia.org/wiki/Domain_%28mathematics%29)*"*

One-Way Function:

Avalanche Effect:

**Websites:**

http://www.freeformatter.com/hmac-generator.html

<http://pythoncentral.io/hashing-strings-with-python/>

[http://www.unixwiz.net/techtips/iguide-crypto-hashes.html](http://www.unixwiz.net/tcechtips/iguide-crypto-hashes.html)

<http://www.ilovefreesoftware.com/12/windows/portable-sha1-md2-md5-hash-generator-calculate-hash-value-files.html>

http://www.tutorialspoint.com/cryptography/cryptography\_hash\_functions.htm

**Variables:**

X : a (possibly infinite) set of MESSAGES

Y : a finite set of possible TAGS (or digests)

K : a finite set of possible KEYS

H : a finite set of hash functions indexed by K so for each K ∈ K there is a function hK : X → Y

**Application of hashing functions:**

Verifying data integrity: checksum generation

Verifying data authenticity:

Password verification: password conversion and verification

Proof-of-Work:

Data identification:

Pseudorandom generation and key derivision:

**Common Hashing Functions:**

Message Digest (MD):

Versions: MD1-MD6

MD5 compromised in 2004 & no longer recommended

n-bits: 128-bit

Secure Hash Function (SHA):

Versions: SHA-0 - SHA-3

SHA-2 not yet compromised; 224-bit,256-bit,384-bit, & 512-bit

SHA-3 (2012) Keccak algorithm; better performance & resistance

RIPEMD:

Versions:

128-bit (MD4 based, not recommended), 160-bit, 256-bit, 320-bit

Whirlpool:

Versions:

Whirlpool-0, Whirlpool-T,Whirlpool

**Ideal Properties of cryptographic hash functions:**

Quick computation of hash values

Pre-image resistance (One-way function)

Second pre-image resistance

Strong collision resistance

No data modification without full hash

**UNIX/Linux Implimentation:**

md5sum program

**Mssg Auth Code (MAC):**

symmetric-key cryptography

Three Algorithms: Key generation alg; Signing alg; Verification alg

Cryptographic strength: Depends on size of secret key

Types of attacks: Brute force for collisions

Timing attack

Difference from hash: MAC uses secret key during compression

Cryptographic properties: Provides authentication

Does Not provide non-repudiation


\textit{HMAC}(K, m) = H \Bigl( (K \oplus opad) \;||\; H \bigl( (K \oplus ipad) \;||\; m \bigr) \Bigr)


where

*H* = hash function,

*K* = secret key [padded](https://en.wikipedia.org/wiki/Padding_(cryptography)) to the right with extra zeroes to the input block size of the hash function, or the hash of the original key if it is longer than that block size,

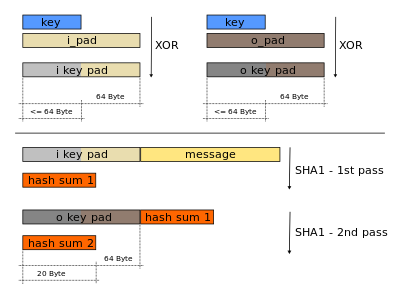
*m* = message

|| = [concatenation](https://en.wikipedia.org/wiki/Concatenation),

⊕ = (XOR),

*opad* = outer padding (0x5c5c5c…5c5c, one-block-long [hexadecimal](https://en.wikipedia.org/wiki/Hexadecimal) constant),

and *ipad* = inner padding (0x363636…3636, one-block-long [hexadecimal](https://en.wikipedia.org/wiki/Hexadecimal) constant).



Potential Vulnerabilities:

Picking specific values of message-size, tag-size, key-size. Could generalize to arbitrary but related sizes.

Multiplication /8-bit shift leads to arbitrary but relative results

it is easy to append data to the message without knowing the key and obtain another valid MAC ("[length-extension attack](https://en.wikipedia.org/wiki/Length_extension_attack)") (HMAC vulnerability)

<https://www.happybearsoftware.com/you-are-dangerously-bad-at-cryptography.html>

Timing attack:

